MACHAZINE

Volume 23 - Issue 3
May 2019

Wiskunde Informatica Studievereniging 'Christiaan Huygens'

THE CHILLING ADVENTURES OF:

The ski trip

DREAMTEAM AS A MATH STUDENT:

Solar Boat

STUDY IN THE USA, STOCKHOLM AND SINGAPORE:

Minor Abroad

DELFT STUDENT E-SPORTS ASSOCIATION:

Esports in Delft

CONTAINING:

Current Affairs | Association | Computer Science | Mathematics | Miscellaneous

Prodrive Technologies puzzle - Win a drone!

Puzzle

Suppose you have a list of n integers, each integer is a number from 1 to *n*, and each number appears exactly once in the list.
Take the value of the first value of the list, let's call it *k*, and reverse the first *k* elements in the list. Continue this procedure until the first value of the list is equal to 1.
The goal is to maximize the number of reversals you will do.

For example, assume the list is arranged as follows (first value of the list is to the left):
2, 5, 4, 1, 3

After swapping the first two elements, continue, resulting in a total of 6 reversals.
5, 2, 4, 1, 3
3, 1, 4, 2, 5
4, 1, 3, 2, 5
2, 3, 1, 4, 5
3, 2, 1, 4, 5
1, 2, 3, 4, 5

For this puzzle the values of *n* are 5, 10, .., 125.
For each value of *n*, submit the order of the list gives the highest number of reversals.

Scoring

For each *n*, you will get a subscore between 0 and 1. The subscore is calculated by dividing your best score by the best score of any contestant.

Assume you are the first to submit a solution for *n*=10 with a score of 30. Since this is the only submission for *n*=10, it is also the best score currently submitted for *n*=10, resulting in a subscore of 1.0 points.

Another contestant submits a solution for *n*=10 with a score of 34. Now your subscore is reduced to 0.88 (=30 / 34).

Your total score is the sum of all of your subscores. This means your total score is between 0 and 25. The goal is to maximize your total score. Note that your total score is not fixed! As other people submit solutions your total score might drop, so keep an eye on the submission page.

Submitting

You can submit your list as a comma separated list of the integers.

You can submit multiple solution at once, by separating them with a new line. A newer solution for a value of *n* replaces the older solution for that *n*, but only when it is better, i.e. when the number of reversals is larger. This means that in a new submission you only need to submit the solutions for n where your solution is better than before.

To see the current standing, deadline and to submit your solution,
go to: https://prodrive-technologies.com/christiaanhuygens

Prize

The two best submissions win a drone, among the other participants a third drone will be raffled!

MACHAZINE
is a publication of
W.I.S.V. 'Christiaan Huygens'

Chief Editor
Eva Slingerland, Marjolein Leegwater
Editorial staff
Daniël van Gelder, Kilian Buis, Willemijn Tutuarima, Wouter Versteegh
Art Directors
Louise Leibbrandt, Jeroen Nelen
Contact address
Mekelweg 4, 2628 CD Delft
E: machazine@ch.tudelft.nl
T: 015-2782532
Concept and design
G_2O Kesteren
Publisher
BladNL
Cover
https://www.pexels.com/photo/beautiful-beauty-bloom-blooming-531299/
Cooperating on this issue:
Willemijn Tutuarima, Jari de Keijzer, Fred Vermolen, Wouter Versteegh, Jan-Willem van Leeuwen, Karel Bouwmeester, Jolijn van Delft, Pravesha Ramsundersingh, Gijs de Jong, Timo van Asten, Joey van der Krogt, Vera Martens, Louise Leibbrandt, Kilian Buis, Daniël van Gelder, Eva Slingerland, Marjolein Leegwater

Advertiser index

Prodrive	Inside front cover
ASML	8
Technolution	16-17
TNO	28

Willemijn Tutuarima

The end of quarter three is probably one of the most important periods for students at the TU Delft. People start planning what they want to do next year and in the upcoming summer. It is especially for freshmen a very stressful but also hopeful time, because they are working hard to get a positive binding recommendation (BSA). I remember being really stressed at the end of my first year, but I can tell you that the stress levels do get better after you get your positive BSA. This year my calendar is more filled with social events than with studying all day every day. This is one of the things I really like about studying in Delft: one moment you can focus on the future and on getting more educated in the field of your choice and another moment you can choose to relax more and to enjoy being a student.

Another change that the end of quarter three brings, is the change in the seasons. Winter is officially over and the sun has come out, which leads to crowded streets. Also, the campus of the TU Delft is filled with students who like to enjoy their lunch in de sun. But it is still spring and not summer, which you realize every time you almost get blown of your bike on your way to the university and in some of these situations gloves are still necessary.

At CH there is still a lot of things happening towards the end of the year. The Dies week took place, where the Dies committee put together an entire week with activities from a pub quiz to a lunch lecture from a magician. This week is very special because all members can get to know each other better and have a little break from all the studying. I think these sorts of weeks really make studying in the Netherlands different from studying in other places in the world because everyone is trying to get people together.

A big part of the MaCHazine consists of articles about technology, whether it is mathematics or computer science related. This is logical, because the MaCHazine is meant for students, teachers and alumni of these studies. One article, the science trends, is not only about mathematics and computer science related topics, but can be about any new technology that is in the news currently. What I noticed is that, although these separate topics are not always directly related to our own faculty, the mathematics and computer science are always used in some way. Mathematics and computer science is used everywhere and in all studies, even if it is just to calculate statistics or remaking some molecules. This really shows that a computer science or mathematics student can go into any field of research.

I would like to wish all freshmen good luck with their last courses and getting their positive BSA. Hopefully the MaCHazine can bring you some enthusiasm and motivation again by showing you the possibilities of your future! If you have any questions or suggestions for the MaCHazine, I encourage you to send a mail to machazine@ch.tudelft.nl. We're always glad to receive input from our readers.

Table of Contents

MACHAZINE

From the Board

Jari de Keijzer

Public Relations

Dear reader,

Most computers science students have heard of it: the halting problem.

The halting problem is the problem of determining whether an arbitrary program with a certain input will stop running over time or not. In other words, halt. Unfortunately, I did not solve the halting problem, but what I did solve, or learn this year, is how to cope with tasks that never seem to be finished.

In Board 62, I fulfill the position of Public Relations Affairs and on top of that I also fulfill the position of IT affairs at 'De Delftse Bedrijvendagen' or DDB for short. My main task within CH is leading the Board of Acquisitioners or BoA for short. Together with a team of four board members we form BoA and with BoA we have a lot of contact with companies with whom we cooperate to make all our events possible (think of lunch lectures, data sets for HackDelft, the company diner, etc.). As IT affairs, my main task is to keep track of all the developments regarding the platform (website) of DDB. The platform has been an ongoing project for about 2 years now and has changed a lot in these 2 years. If you participated in DDB you have used the platform too, so feedback is always welcome!

In both of my main tasks in both the Boards I got some good ideas from my predecessor to start working on. Many of those ideas were ideas that he thought of during his year as a Board member. For instance, with BoA we wanted to have a lot more graduation projects on the CH website. This is an idea that BoA is executing right now. For the DDB platform, I got a whole list of ideas that could be implemented. All these ideas were already implemented in 2018. However, right now I have a list that is twice the size of the original list of features. How did this happen?

While going through a year as a Board member, you will go through a transition of first executing all the tasks at hand to start trying to change something in the association. Many new ideas to change something will come when working on a certain topic a lot. That is the reason I now have an even longer list of ideas I would like to change this year, when the year is already coming to an end.

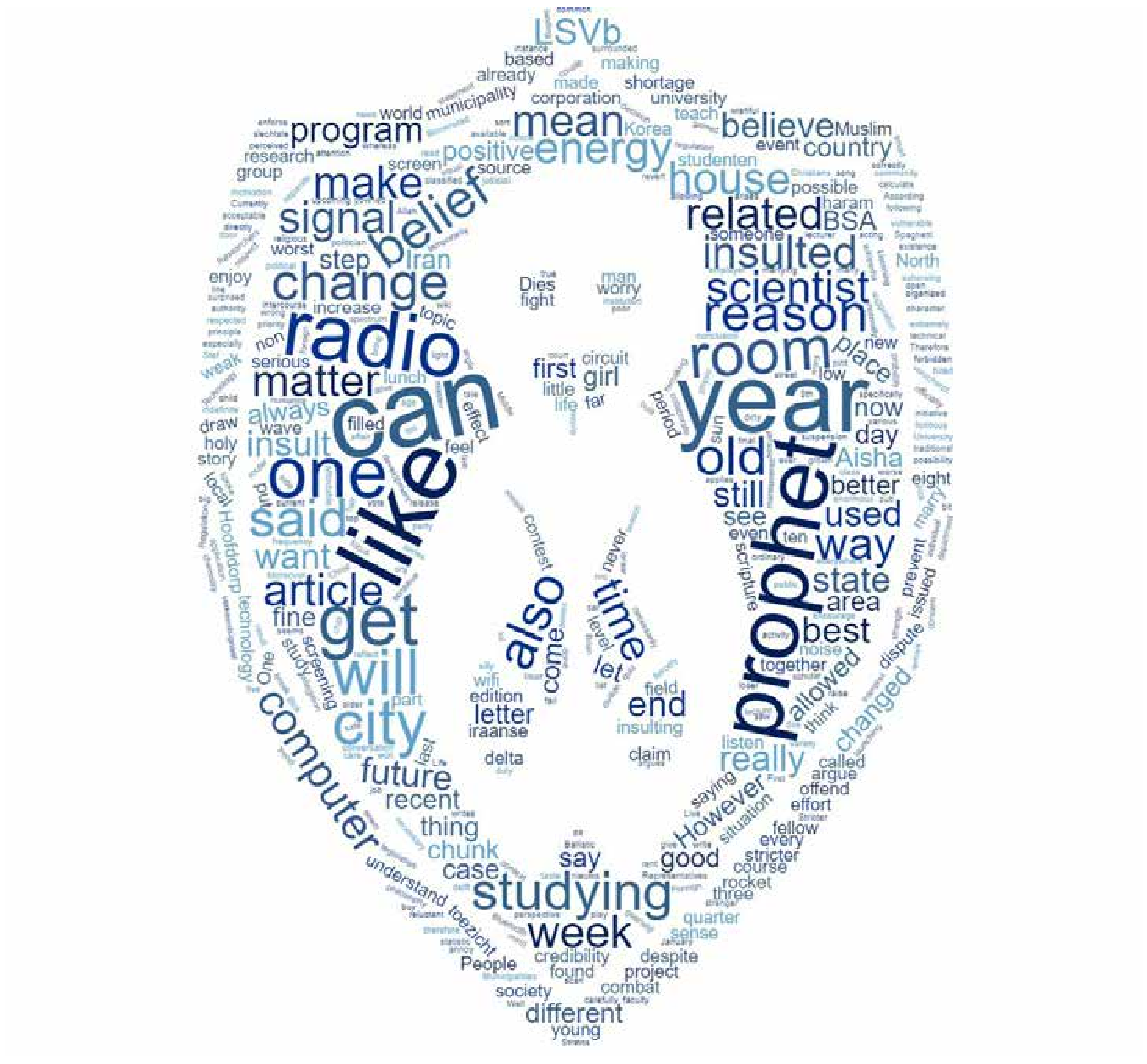
The problem that arises from all these ideas is that you have limited time to execute them and that you will get stressed trying to execute them all. The point I am trying to make is that prioritizing your ideas is important. It is better to execute a couple of ideas in a very neat manner than to execute all your ideas while cutting corners. This may seem logical but choosing between one idea or another can be difficult.

To conclude, halfway through the year I came to realize that prioritizing your goals can be difficult but when it is done it will take a load of your shoulders. My message to you is that it is important to keep thinking of new ideas and to write those down. However, is it also important to understand that you can't execute an infinite amount of ideas. Prioritizing them and maybe handing over very good ideas to your successor is a good way to handle your workload. I can now conclude that CH is a none halting association, it keeps changing in a positive way every year!

HuygsCHe!

Board 62

Current Affairs



Volume 23 • Issue 3 • May 2019

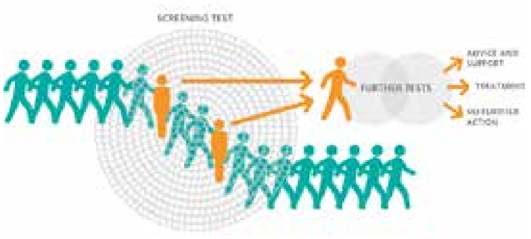
TU Delft News

Wouter Versteegh, Editorial Staff MaCHazine

The Delft University of Technology is the biggest and oldest public technical university in the Netherlands, established by King Willem II on January 8th, 1842. But what is currently happening in and around the TU Delft? This article will list the most important events of the recent months.

Stricter screening of Iranian students and scientists at TU Delft

The Dutch cabinet has decided to screen Iranian students and scientists in a stricter manner with immediate effect. This decision is made to prevent that specific knowledge gained in the Netherlands is used for the development of weapons in Iran. In a letter written by Minister of Foreign Affairs Stef Blok to the House of Representatives (Tweede Kamer), he writes that a "recent case" was the reason to screen students and scientists from Iran with immediate effect. Sources have reported to the NOS that this recent case is an Iranian student who could be studying here to learn about making and launching missiles that could benefit the Iranian Ballistic Rocket Program.



The letter also contained in broad lines what this means for Iranian students, scientists and staff at the TU Delft. A taskforce will write a test that will check Iranian people who are active in "sensitive science departments" if they are related to Iranian rocket programs. Currently there are already foreign students being screened whether they could be related with North Korea, which is recorded in the Sanction Regulation North Korea 2017. This regulation applies to five TU Delft programs, and the Delft rocket project "Stratos". The cabinet now wants to extend this screening policy to more countries, in the first place Iran, but perhaps also other countries who are perceived as risky.It has already been tried in the past to reject Iranian students from particular programs, but the judicial court prevented that. This applied to master programs in nuclear physics and chemistry. However, according to the cabinet there are now enough reasons to enforce a stricter policy regarding Iranian students and scientists.

Listening to quantum radio

Researchers at the TU Delft have created a quantum circuit that allows them to listen to the weakest radio signal allowed by quantum mechanics. This could possibly open the door to future applications such as radio astronomy and quantum MRI scans.

We have all been annoyed by weak radio signals before: your favorite song in the car turning to noise, being too far from the wifi router to watch a series or too far from your Bluetooth speaker to play your music. Our usual solution is to increase the signal strength, such as fine tuning your radio or moving within the room to pick up a better signal. However what if we could listen more carefully?

In quantum mechanics, energy comes in little chunks called "quanta". This means the following: if I want to add energy to something, I can only increase the energy one "quantum step" at the time. Quantum mechanics states that I cannot raise the energy with half a "quantum step", so it is the smallest energy chunk possible. These quantum steps are usually too small to notice. Until recently this was also true for radio waves, but the circuit that the TU Delft built can actually detect these chunks in radio frequency signals. This means that we can sense radio waves at the quantum level. If this gets developed further, it could mean that you won't have to worry about weak wifi ever again!

Delft worst city for student rooms

Where Delft was the best city for student rooms in 2017, in this year's ranking made by the National Student Union Delft (LSVb) it is ranked the worst. Enschede is the best city. Enschede was awarded last week in the second edition of the city student housing contest. This contest is held to draw attention to the shortage of rooms for students. Municipalities who combat this problem with smart policy are rewarded. The thirteen cities with the most students are judged based on their student housing policy. Do they collaborate with housing corporations, and are individuals able to rent out their house to students? Are students being provided enough information about housing, projects and new initiatives? Based on questions like these, the LSVb makes a ranking. Enschede came out as the best because "despite the low need for rooms, the city still makes an effort to make housing as affordable and safe as possible". They work closely with housing corporations and give all students an equal chance of a room.

Delft being ranked as the lowest comes a bit as a surprise; in the first edition Delft was ranked the best, so what changed? The main reason is that the policy regarding the houses itself changed. It is way harder now to buy a house and make it into a house for students. This used to be a good way for students to get a room in Delft. The municipality doesn't understand the ranking. They state that student housing is a top priority for the city. They understand that the low ranking arises from the policy change, but they state that the LSVb didn't interpret the reason for that correctly. The LSVb stated in their research that noise and nuisance are reasons for the policy change, to prevent students from buying a house in non-student areas and annoy other civilians. However, the municipality states that they changed the policy to combat an imbalance in the local housing market, since particular areas were getting more crowded than intended. The areas specifically mentioned are the Binnenstad, Wippolder and Westerkwartier. The local student union VSSD is making an effort to revert this policy change.

References:

[1] https://nos.nl/artikel/2275965-kabinet-verscherpt-toezicht-op-iraanse-studenten.html

[2]https://www.delta.tudelft.nl/article/strenger-toezicht-op-iraanse-studenten-en-wetenschappers

[3] https://www.delta.tudelft.nl/article/delft-dit-jaar-slechtste-studentenkamerstad

[4] https://www.tudelft.nl/en/2019/tnw/listening-to-quantum-radio/

Volume 23 • Issue 3 • May 2019

Do not insult the 'prophet'!

Fred Vermolen, Department Numerical Analysis

In February 2019, a teacher at a VMBO-school was suspended because of allegations that he insulted 'prophet'[1] Muhammad. Is this a wishful situation in a secular country like the Netherlands?

February 2019, a teacher at the Hoofdvaart College in Hoofddorp has been suspended for an indefinite period because he would have insulted 'prophet' Muhammad [1]. His students issued a complaint about him insulting 'prophet' Muhammad to the director. The teacher said that he heard one of his students say that her mother found him (the teacher) 'haram' ('forbidden by Islamic law', or in the current context 'dirty'). The student said this to one of her fellow students. The teacher spoke to her about this matter and suddenly the teacher found himself surrounded by a group of students, who were shouting that he had insulted the 'prophet'. Therefore, a number of students issued a complaint to the director. The director decided to 'temporarily release the teacher from duties', which in ordinary English reads as that the teacher was suspended. The 49 years old teacher received a letter from his employer saying that 'you have said that Muhammad married an eight-year-old girl and that if a grown up man marries an eight-year-old girl in the Netherlands, then he is classified as a pedophile'. The teacher denies all accusations. He claims that he has never said this and that he did not insult the 'prophet'. He said that he had even never mentioned the 'prophet' during his conversations with his students. The teacher fights his suspension at the disputes committee of educational affairs. The school is not available for any comments.

In my opinion, this is a very serious matter from different angles. First of all, in my opinion, everyone's religious beliefs should be respected. Hence, if the lecturer has insulted the 'prophet' then his students may feel offended. In this sense, it is a matter of very bad taste to offend the students by saying that Muhammad was a pedophile.

There is an enormous variety of sources about Muhammad's life. One of these sources is the Quran, which is regarded as the holy scriptures by the Muslim community. According to [2], the traditional scriptures say that Muhammad 'consummated' the marriage with Aisha when she was nine or ten years old. Modern Islamic scholars estimate Aisha much older than this, which changes the story. To reflect on this, one should bear in mind that during the time when Muhammad was alive, marriages were based on different principles in the Middle East from what is considered as common in nowadays Western societies. Marriage was not only aimed at allowing sexual intercourse or at getting children. A man could also marry a (young) woman because he wanted to take care of her, for instance if she was an orphan or if she was divorced and therefore had no other means to sustain her life. Sex was not necessarily a part of marriage in these days. In this light, despite Muhammad marrying young Aisha, it would be inappropriate to draw the conclusion that Muhammad was a pedophile. Of course, currently, it is not allowed to marry a girl of around ten years old in the Netherlands, see [3], and there are good reasons for this!

Another serious matter concerns the credibility of the students. The students are at a VMBO institution, which implies that their age is normally in the range of 12–16 years old (if they did not fail any classes). To what extent should a school director believe a group of hysterically shouting students (which I saw in a footage, see [4])? Should not the director have had a chat with his teacher? Was not the teacher hired for his credibility? If I would have a dispute with a policeman, then, in general the judge would believe the policeman, and not me. Should not the director have called the students and teacher and should he not have listened to their stories simultaneously? In my opinion, the director should have done this. By acting the way he did, the teacher has been maneuvered into a vulnerable position. If the students do not like a teacher, then they can issue various false accusations to the board or to the director. As a result of these claims, a teacher can be suspended or sacked.

On the other hand, the VMBO school in Hoofddorp is not an Islamic school, hence 'prophet' Muhammad is not a holy figure according to the school's philosophy. In that case, why should a school defend a symbol that is hypothetical to them, like Muhammad, so fiercely? The student said to a fellow student that the teacher was 'haram' according to the student's mother. Is not this insulting to the teacher? Does the teacher not have the right to feel offended?

Another thing that worries me in this matter is that this incident feeds the powers at the extreme sides of the political spectrum in the Netherlands. Some politicians will see a confirmation of the statement that the Netherlands are heading towards a more Islamic society. People who vote for such parties will be strengthened in their belief that Islam is taking over and that we are heading towards Sharia legislation. Moreover, many other people argue that Jesus Christ has also been insulted by comedians, movie-makers (think of Monty Python's Life of Brian), so why is it wrong to insult a fictitious character like Muhammad? In these people's perspective, one argues why is it not allowed to insult Muslims, whereas it is acceptable to offend Christians?

As a final remark, I would like to argue that there seems to be a shortage on teachers in the Netherlands, in particular, nowadays teachers seem to be reluctant to teach at VMBO-schools. If teachers are treated like this, then their authority is jeopardized, and then I am not surprised that nobody wants to teach at VMBO-schools. This will be worse in the future.

This tale will only have losers because of the extremely poor job of the director. The director should have believed his teacher and if the circumstances are such that he cannot believe the teacher, then, he should have organized a meeting where the students, their parents and the teacher are invited. It is a silly world. I would like to say that we are all humans with our own beliefs and non-beliefs, so let us not fight about these beliefs, but let us respect each other. If someone believes in Allah and Muhammad, it is fine with me, if someone does not believe in the existence of God, or in a Spaghetti-monster, it is fine with me too. Live and let live.

Well, I have bored you with all this. It is time for a couple of pints. *Skål!*

[1] I have put the word *prophet* between quotes because I am not a Muslim.

References

[1] https://www.nrc.nl/nieuws/2019/03/01/schorsing-leraar-om-profeet-mohammed-is-verkeerd-signaal-a3877353

[2] https://en.wikipedia.org/wiki/Muhammad

[3] http://www.wetrecht.nl/seks-met-minderjarigen/

[4] https://www.geenstijl.nl/5146469/powned-over-pedo-mohammed-in-hoofddorp/

Volume 23 • Issue 3 • May 2019

Knowing how to code is not enough for career success

ASML

Software development skills are in demand, as any quick scan of online job boards will confirm. But the people doing the hiring have an important piece of advice: knowing how to code isn't enough for long-term career success. The developer skill set is changing.

"Software engineering is about abstraction and structure," says Jan Friso Groote, professor in Computer Science at the Eindhoven University of Technology (TU/e). "The real problem of software is that it is so immensely complex that if it is not well structured, it becomes unmaintainable." As a result, the most important skill of a software developer isn't writing code and testing it until the bugs are quashed. It is understanding the essence of a problem and building a structured, reliable, extendable and maintainable approach for solving it.

For development teams who have taken this approach to its logical conclusion, it means that software engineers write very little traditional code. They spend most of their time working in abstract modelling languages, specifying the behavior of a system. Formal verification tools allow those teams to be confident that their solution is complete and error-free, and the code itself is then automatically generated.

With such a model-driven engineering (MDE) approach, a team at ASML recently replaced half a million lines of code that had been built the conventional way. "When we made this change to our software, it was a challenging period and a lot of energy was needed from our software engineers," said David van Beek, who leads a group of software engineers at ASML. "We really grew as a group and as a department. We continue to grow now, and we need developers with this energy and drive to ensure we continue to produce a clean and extendable design in the years ahead."

It's not surprising that companies like ASML are embracing model-driven software development. ASML makes equipment for chip manufacturing. All of the world's leading makers of processors and memory chips are using ASML's lithography systems to create the nanometer-sized electric circuits found on modern chips. These are some of the most sophisticated machines ever built, so the demands for the software that runs them are high.

Rogier Wester, manager of the Lithography Systems Software Architecture group at ASML, said he looks for candidates who demonstrate abstraction skills, who understand the essence of the problem and are still able to create simple solutions. This is because complex solutions do not usually work and even if issues do not crop up immediately, bugs will still appear when customers start to use the product.

This requires developers to think in a very different way. "Think about what will go wrong. Divide and conquer. Use models for abstraction and conciseness. Use appropriate tools, that allow you to refactor and change with confidence," Wester said. "We need very skilled software designers and, in my honest opinion, we see the challenge for the universities to offer an integral computer science education on software architecture and design, abstract modeling, and formal specification and verification," he added.

Know more about Software at ASML: workingatasml.com/sw

ASML

Be part of progress

Advertorial

Association

Volume 23 • Issue 3 • May 2019

Department Symposium

Jan-Willem van Leeuwen, Secretary of VerdiepCie committee

The VerdiepCie is a committee at Christiaan Huygens which aims to provide in-depth information about a multitude of things. It consists of five students and one board member of CH. The five students take up the roles of Chairman, Secretary, Treasurer, Mathematics Affairs and Computer Science Affairs.

Firstly, we organized a Department Symposium for Computer Science (CS) and Applied Mathematics (AM), about both the bachelor thesis as well as the master. Initially we wanted to schedule these both on one day, so with the AM and the CS schedules parallel. Then we would have the morning for talks about the bachelor thesis, and the afternoon would be filled with master information. However, scheduling this was impossible, which is why we resorted to spreading out our schedule over multiple days.

Everything was scheduled in quarter 3 of the academic year, so we held the AM bachelor thesis talks on Tuesday the 12th of February in the morning. Two days later, on Thursday the 14th , we had scheduled both the AM and the CS master talks to start in the afternoon. Before that, during the lunchbreak, there was a special presentation for students who did the double bachelor Applied Mathematics-Applied Physics programme, on what they had to do for their bachelor thesis. One week later, on Friday the 22nd of February, the CS bachelor thesis talks, which were held in the afternoon, concluded the Department Symposium.

However, before we could even start thinking of what kind of talks we wanted and how we wanted to schedule them, we had to look at what these study programs had to offer. We looked at all the different departments of these studies and tried to figure out how many speakers we needed. We also drew up a plan on what we wanted to offer during these talks. Did we want to offer coffee and tea? Did we maybe also want some cookies, or fruit? Were we going to provide lunch for all the participants? All of these decisions had to be made before continuing.

After a lot of talking back and forth, budgeting, and planning, we figured out we could offer lunches before the afternoon talks, which were the AM master and the CS master parts as well as the CS bachelor thesis part. We also decided on fruit, cookies, and drinks in the breaks between talks.

Students had to enrol for the parts they were interested in, so we could see how much lunch and drinks and such we had to provide, as well as knowing just how many people would even show up. Initially the enrolment went quite slowly, thus we had to promote more than we already had. We had already used a banner which was strategically placed at the entrance of EEMCS, we had sent texts to our fellow students, we had sent Brightspace reminders, and so on. After increasing the amount of promotion we put out, more people started enrolling, as we had hoped.

Eventually we had a fairly high enrolment, about 70 for the AM bachelor thesis talks, 30 for the double bachelor thesis, 50 and 70 for the AM and CS master information talks respectively, and 80 for the CS bachelor thesis talks. And this turned out to be quite a good metric to check how many people would show up, except for the CS bachelor thesis talk. With an enrolment of about 80, we decided to order 100 sandwiches, where the extra sandwiches were for ourselves and the CH board, but only about 20 people showed up. We ended up having to throw out quite a lot of sandwiches, which is, of course, very sad.

But this is all reflection. In order to plan the day, we had to mail several lecturers and students. From each department we asked one lecturer to talk about what their department did, what kind of research was done there, and, depending on whether it was a bachelor thesis or master talk, we asked them to explain what a bachelor project would entail, or what a master programme within their department would look like.

Luckily for us, this all worked like clockwork. All of the departments had someone represent them, and, nearly, all of the talks were very informative as well.

All in all, we can happily conclude that the department symposium was a huge success and we thoroughly enjoyed preparing it all. At first it seemed a very monumental task, but as we tackled all the issues in a very strategic order, we ended up having quite a lot of fun. The next thing on our agenda is preparing a lunch lecture, and we hope it will be even better than the symposium!

Winetasting by the MeisCie

Jolijn van Delft, Chairman of MeisCie committee

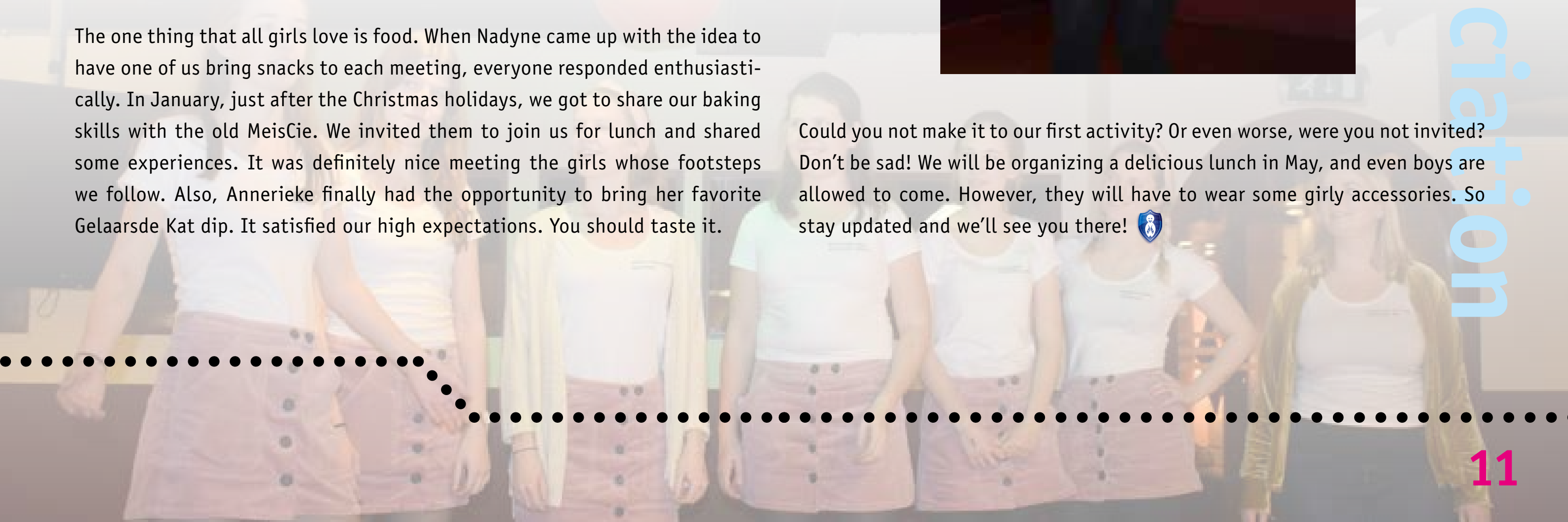
In February, the first MeisCie activity took place. On a Wednesday evening just after dinner, around 30 girls gathered to attend a wine tasting in the /Pub. Some were completely refreshed after the holiday, others were still recovering from the skiing trip. As the chairman of the MeisCie 2018-2019, I've been asked to write an article to tell something about this year's MeisCie committee.

Before you'll get to read about how great of a success our first activity was, let us first shortly introduce ourselves. Just as previous year's committee, the MeisCie committee consists of girls only. Our names are Helena, Nadyne, Pauline, Elise, Annerieke and me, Jolijn and we're computer science and mathematics freshmen. After we all confirmed to join the MeisCie, we first met in November. The communication so far had been in English, but none of us necessarily expected an international student to become part of the MeisCie. This was new for committees at CH in general. We can conclude that communication in English is no problem for us, since we even managed to explain Helena what the 'Gelaarsde Kat' dip is (ask Annerieke if you want to know more). So Helena, if you read this, we're more than happy to have you as our secretary!

The one thing that all girls love is food. When Nadyne came up with the idea to have one of us bring snacks to each meeting, everyone responded enthusiastically. In January, just after the Christmas holidays, we got to share our baking skills with the old MeisCie. We invited them to join us for lunch and shared some experiences. It was definitely nice meeting the girls whose footsteps we follow. Also, Annerieke finally had the opportunity to bring her favorite Gelaarsde Kat dip. It satisfied our high expectations. You should taste it.

As the year passed, we enjoyed the activities that were organized by other freshmen committees. On the 13th of February, it was finally our time to shine. We'd spent our time well during the meetings, so we were more than ready to host the girls that bought tickets for the wine tasting. After we put on our pastel pink skirts, we made sure the /Pub was fully decorated in pink as well (once in a lifetime experience). Around 8 o'clock the activity started. The first wine was served: a medium fresh Spanish Cava. As we had carefully selected and reviewed all types of wine we would be serving, we first told something about the origin of the wines before people actually got to taste. Also, the wines were served in combination with snacks that perfectly went well with the type of wine. About 20 minutes after the Cava, an Italian chardonnay followed. After that the classic pinot grigio, followed by a South-African mix between a sauvignon and a shiraz. The first (and only) red wine served was a French merlot. We ended off with a sparkling dessert wine, served in combination with ice cream and delicious fruits. Around 10 o'clock, the activity was officially finished. However, since there were about twelve full bottles of wine left, most people decided to stay. Who on Earth denies free wine? In the end, all bottles were empty. Enough reason to conclude that the activity was a great success!

Could you not make it to our first activity? Or even worse, were you not invited? Don't be sad! We will be organizing a delicious lunch in May, and even boys are allowed to come. However, they will have to wear some girly accessories. So stay updated and we'll see you there!



The CHilling adventures of the ski trip

Karel Bouwmeester, Participant of the ski trip

As the roosters cuckoo'd their merry way into the morning, and as the dusk had begun to subside, nothing was more clear. Earth's orbit around the sun had once more birthed February onto us. And boy was February excited to wreak its havoc onto, as of yet, 40 unrelated youngsters. The road ahead was a long one, an exciting one, and in retrospect some may even say a joyous one. As the hours passed on this inaugural day of February, the excitement escalated for every single one of these 40 persons. 18.00 on the 1st of February, 2019. The Wispo had begun.

Chapter 1: A new beginning

One after another, people began to take place in the bus destined to take them on the ride of a lifetime. Kilometer after kilometer was being traversed by the bus, the distance eclipsing country after country. Not before long, only a few hours after the ride had begun, the passengers were starting to feel a bit tired.

Falling like dominoes into a deep slumber, most had awakened to find themselves in France at the foot of the mountain. The mountain on which they had been promised a splendid week of skiing, snowboarding, and shots. The trifecta of S's was indeed what they had been promised, however this trip brought another s with it: the s for stall. For they had another 6 hours to wait until the climb to the top of the mountain began. Finally, around 16.00 on the succeeding day from when their journey had started, they had arrived at the location that was supposed to be granted to them. In true unsupervised student fashion this meant it was time for the holy grail of beverages, the ale to end all ales, the brew with an unparalleled level of passion and party: beer. As our party of 40 partied their way into yet another few hours of darkness a new challenge awaited.

Chapter 2: The roads untraveled

Sunday morning. Awakening from their deep slumber, it was time for the second obstacle -- going to everyone's respective skiing or snowboarding class. As the narrator does not know what happened during any of the snowboarding- or advanced skiing classes, this part of the story will focus on the beginner skiing classes. It all started halfway on a green slope. Should be easy, right? Right? It's only green, right? WRONG! Following a solid 10 minutes of trying to put on their skis, our great heroes Kilian and Karel had to make their way to the bottom of the slope in order to reach their class. Their lovely, momentary teacher Noortje tried to tell them a multitude of times how to 'pizza punt' with their skis as to not fall on whichever limb had to be so unlucky. All her effort remained in vain. Fall after fall, drop after drop, descension after descension. Kilian and Karel kept on falling into the snow at high, uncontrolled speeds. In the wake of their traumatizing path to glory, Kilian and Karel had finally reached the bottom of the slope and it was time to go to their classes. Here they found out that they were to meet up with 2 more leCHends. Claire and Lynn had been the chosen ones to join our 2 heroes on their path. The Fab Four had been formed. Longing for technical development, they listened carefully to whatever their instructor, Thomas, told them. The first few exercises went well. Enthusiasm increased. Our foursome had finally learned how to 'pizza punt'. No more traumatizing descensions for them! The exercises after that went well as well. Enthusiasm increased once more. Balance was no longer an issue for our talented bunch. Everything seemed to be a walk in the park on this first day. Emphasis on the 'seemed'. The class went further without a hitch and before long they had completed their first day. Time to replenish their energy levels with something satiating. Satiated, our group of 4 decided to split up. Claire went on with her day skiing with her boyfriend, Lynn joined Merel and Julian and Kilian and Karel opted on going with their friends Mandy, Noortje, Jules, Jolijn and Janne. Remember how everything seemed to have gone well? Surprise! Not anymore. Remember how they had been traumatized? Well not like ever before. It was time for the slope of a lifetime. Their first official, fully crossed slope. Having reached the top of the slope, Kilian and Karel's blood sugar spiked to levels comparable to only the most adrenaline inducing activities. A rush overcame them, and on they went. Kilometer per hour after kilometer per hour, their speed increased to heights never reached before by themselves. White -- that was all they saw when their heads peered up from the snow bank they had landed into. Perhaps a careful approach was more fitting for their current level of skill. After this new approach the descent went a tad smoother and they reached the bottom after some time. Hours passed and our friends procured themselves a grand amount of intoxicants to ingest during fanciful activities. One inebriated sleep later and the sun had come anew.

Association

Chapter 3: A new day has come

Cheerful mornings aside, day two on ski's provided new options to our festive friends. A new instructor had come across their path to greatness: miss Nathalie. Nathalie strived for success, and was determined to extract the hidden talents out of the Fab Four and make them the best skiers they could be. She had not realized, however, that our Fab Four had been practicing for hours after the instruction provided by Thomas, and had developed themselves into quite competent skiers for the level they were supposed to be at. The challenge for this day had been the slope they had gone off of the day prior. Today this proved to be so easy, that they had taken it upon themselves to go off of the highest green slope to which they had access. At a height of 2.2km, this slope far surpassed the challenges of what seemed like yesteryear. As that slope only had a height of 1.5km. Turn after turn Kilian and Karel went downwards, the slope seemed to go on forever until they had finally reached the bottom. Not having fallen, they felt extremely accomplished. Not wanting to ruin their high, our musketeers went home. An unexciting night was ahead, as the Yeti bar had been closed for our friends from CH. Thus, and because they were very tired from the early awakening from that morning, they went to bed to maximize their skiing hours on Tuesday.

Chapter 4: The night to end them all

Feeling refreshed from their long rest, Karel and Kilian were very excited for a new day of adventures. Their class proved to be a bit of a letdown in when it came to this however, and they had to go down the same slope they practiced the day before. An unexciting class later and they went off of the massive 2.2km slope a couple of times to practice some more skills, this time including parallel turns. Longing to prove themselves to their peers, the students of CH played a game known, and loved, by students across the globe: Beer pong. The teams were divided by sleeping rooms, so Jules and Karel were in the same team. Kilian was part of the team that organized the entire trip. Janne, Mandy, Noortje and Jolijn were on the same team. And lastly Lynn, and Merel were joined by their companions. It was a true spectacle. Tears had been shed, blood had been drawn, but there could only be one winner. The previous board had won.

Chapter 5: Into the darkness

After the excitement of the beer pong tournament, the days went on without much differentiation. On Thursday it was Karel's birthday and that was magnificently celebrated in the Yeti bar. So much so, that he'd been denied re-entrance after exiting for some cooling down. On Friday we made a group photo with the entire gang. To commemorate a, so far, very fulfilling trip. It was also the last day of the skiing classes and after a week of hard work Kilian, Karel, Lynn and Claire had finally earned their long yearned after certificate. Saturday was used by most to turn in their ski's and wait on the bus which would take them back to EWI, but Jules decided it was much more fun to use his last day of skiing for overstretching his medial collateral ligament. This was obviously much more fun than a calm day of waiting, because this red devil craved the excitement of recklessness. Once this was all taken care of it was time to go to the buses and start the journey of going back to EWI.

Hours passed, alongside a detour to the McDonalds, and the end started to approach. The gang arrived early on Sunday morning, and departed their ways. This was the ending of an unforgettable ski trip. The second semester of the year had begun.

Party by the SjaarCie

Pravesha Ramsundersingh, Secretary of SjaarCie

Since the beginning of my journey in Delft, I was always interested in the study association Christiaan Huygens (CH) and its ability to engage students within Computer Science and Applied Mathematics. This academic year, I am honored to be given the role of Secretary within the SjaarCie. This is a first year committee, commonly known for its organization of a party in February and a barbeque in June. This Committee consists of 12 lively students within the study association, each assigned with a specific role dedicated to the organization of the aforementioned events.

The first event on the list was the party! On February 20th, the SjaarCie of 18'-19' organized their first event: Hakuna MaSjaarCie. The theme for the party was inspired by Disney's The Lion King, which allowed us to cleverly combine our Committee name with the popular saying 'Hakuna Matata'. The main goal of the event was to give students the opportunity to unwind and forget their worries for a night.

Although I entered the committee at a relatively later stage than the other members, I was still able to witness and participate in the incredible amount of effort the committee had put into the preparation of the event. My role as Secretary included the taking of minutes for each weekly meeting, producing a schedule to assign roles for the evening and any extra promotional affairs such as posters or the selling of tickets.

A week before the event, the first and second floor of EWI were covered in vibrant SjaarCie posters. You may have also noticed a bunch of enthusiastic students strolling around the faculty building, actively promoting the event.

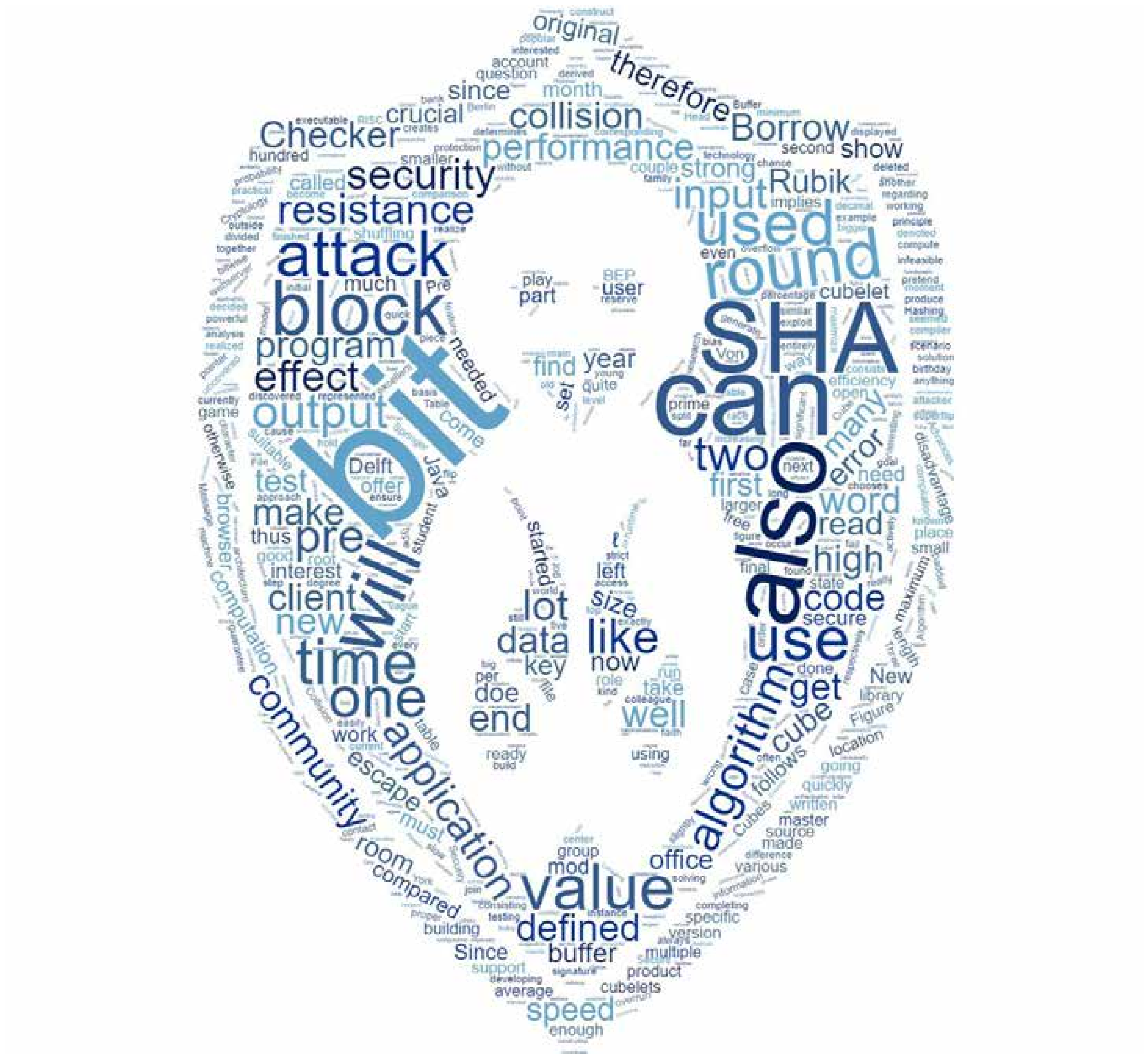
The day of the party had finally arrived and the SjaarCie was feeling a mixture of emotions, ranging from anxious to thrilled. We still had much to do. From last minute stops at SoLow to decorating the Bierfabriek with balloon animals, the SjaarCie was constantly on the move. To be easily recognizable and blend with the playful theme of the party, the SjaarCie also decided to dress up as a variety of animals. With our white shirts, green suspenders and animal costumes, the party was ready to commence!

Following the great success from last year, the SjaarCie decided to hold the party in Bierfabriek again, which is located perfectly in the center of Delft. Despite the fact that we had sold a small number of tickets in the beginning, that evening the Bierfabriek reached its maximum capacity at a very fast pace. The combination of the upbeat music, beer pong and the high-spirited people, generated the perfect problem-free vibes.

Alongside Alessandra, responsible for the Clothing Affairs of SjaarCie, I was in charge of serving the welcome drinks of the event. Matching the theme of the party, the welcome drink was Safari, allowing our guests to be welcomed by a sea of golden shots. Not only did I get the opportunity to greet everyone, but I was even able to take a selfie with DJ PJ. And what's a party without a selfie with DJ PJ? Throughout the night we had insanely talented DJs, including Hidden Dancefloor, DJ PJ and Struikrover. Even as the party came to its end and eyes were beginning to close, the SjaarCie Committee enjoyed cleaning up the decorations whilst reflecting on the highly successful night.

Although not everyone could make it to the early morning lectures the next day, the reviews of the party were all positive. I am proud to say that the SjaarCie, once again, held a sensational party. The effort and dedication of the team was accurately displayed in the event and we hoped that each and every one of you enjoyed it as much as we did. If you were not able to attend this party, be sure to join us at our upcoming barbeque in June! Not only will there be delicious food, but the sun will also be shining. Till then you can often spot the SjaarCie in our dark green sweaters!

Computer Science



New programming language ‘Rust’ provides well developed memory management

Technolution

The young programming language Rust can do anything C and C++ can, so what is the difference? Rust guarantees protection against a large number of memory problems. Rust therefore solves a problem with which programmers have been struggling for decades.

Von Neumann and memory management

The basis for computers and computer programs is the Von Neumann model which is over seventy years old. This architecture has one memory that holds both the instructions as well as program data. Programmers must thus always spend a lot of time on the memory management of their code.

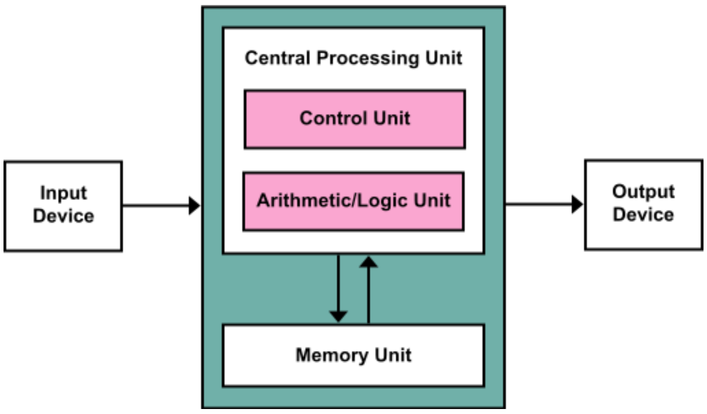


Figure: The Von Neumann model.

The commonly used programming languages C and C++ offer no support for automatic memory management. The programmer must deal with this himself. An error is easily made. When the programmer reserves a hundred memory locations and the program writes something to location 101, we call it a ‘buffer overflow’.

During a buffer overflow, it is often unclear what is going on at that particular moment. The best case scenario is that a piece of memory is being written that was not being used anyway. But the consequences could be much bigger. What if a crucial piece of data, a password for instance, is being overwritten or deleted? Will the deleted data be read by a different application down the line or by the operating system? In that case, sensitive information can end up in the wrong place. Whatever happens, it will occur outside the programmer’s control.

Buffer overruns – Heartbleed

Buffer overruns are the cause of many computer problems, such as crashing and vulnerability in terms of attacks. The past years have revealed multiple exploits: vulnerabilities that are the result of these types of errors and which could have significant consequences.

Recently, a vulnerability was discovered that became known as Heartbleed. The total costs for a world-wide update of the systems is estimated at a minimum of five hundred million dollars. The error was included in a component, OpenSSL, that is being used, among other things, in network routers and webservers. This vulnerability allowed encryption keys to be read remotely from these systems. With captured encryption keys, an attacker can read traffic from and to the systems and pretend to be the original owner of the key. When the key of a bank’s webserver is discovered, an attacker can pretend to be the webserver of the bank without the user being able to tell the difference in their browser. This vulnerability would not have occurred if OpenSSL had been written in Rust.

Secure programming in Rust

Rust is a relatively new programming language. Like C and C++, it is a system language that is suitable to construct system software, such as operating systems, drivers and compilers. Rust programs compile directly to machine code, enabling the executables to remain small and making maximum performance achievable.

There is one thing that sets Rust apart from most of the other mainstream languages: when it comes to memory management, Rust is very strict. It is an automatic process. Even during compilation, the Borrow Checker (see frame) is checking the memory management. Does the program code run the risk of memory problems during runtime? Borrow Checker will then abort the compilation. The same happens when the code causes data races: parallel processes that access the same memory location simultaneously. Secure programming is thus enforced. For hardcore programmers, programming with Rust is therefore an entirely new experience, despite the similarities to C and C++. They need to get used to the strict Borrow Checker.

Rust is not the only language that offers a solution for memory management. A language like Java does the same but during runtime which results in serious consequences for the performance. A significant disadvantage when software needs to respond as quickly as possible, for instance in medical or industrial applications. Rust does not have this disadvantage. It is also possible to program unsecured in Rust, but this can only be done when the programmer explicitly chooses to do so.

Borrow Checker – automatic memory management
The secret to Rust's automatic memory management, without loss of performance, lies within the Borrow Checker. This bit of software, which is part of the compiler, carefully records how the programmer is using the data in the computer's memory. The Borrow Checker refuses to translate the program code into an executable file when it includes memory errors. The Borrow Checker detects many common errors such as double

The clients' interests

Technolution uses various programming languages and chooses the one that is most suitable for the solution. Most of our clients do not have a preference when it comes to language. They are mostly interested in the finished product and expect an application that is powerful, can be maintained well and does what it is expected to do. The selection of the underlying technology is generally left up to us. The decision regarding which language to use is therefore often made by us.

When do we decide to use Rust? This strongly depends on the context. Various questions play crucial roles. What are the security demands? How important is high performance? Rust and the corresponding ecosystem are still in the developing phase, but the language will certainly be considered for the development of new applications that require high performance, robustness and protection against exploits. The interests of the client are, obviously, the deciding factor amongst these deliberations.

Ecosystem and community

Compared to other programming languages, Rust is still very young. The language started out as a private project of a Mozilla employee; Mozilla being the manufacturers of the open source Internet browser Firefox. Mozilla published the first stable version in 2015. Just for comparison: Java dates from 1996 and the first version of C hit the market in 1972. New programming languages become available on an almost daily basis. Most of these are quick to disappear and fade into oblivion. It takes time for a new language to gain enough credit amongst developers and become a fixture. Nevertheless, there are enough reasons to have faith in Rust's future.

What makes a programming language stand out? And when is a language labeled 'successful'? Firstly, it is important that the language has a proper syntax and can be easily read. Performance is also crucial in many fields of application, just like support for data structures and various operating systems. Rust meets all of these demands. However, a good programming language is not the same as a successful programming language.

In order to be successful in the world of developers it is especially important that a programming language comes with an extensive ecosystem and a strong community. Java is a good example of a language that made it big due to a proper ecosystem with many libraries, supported by a lively and well-organized community. The community is also one of Rust's strong points.

Mozilla has an excellent reputation when it comes to supporting open source. Their Firefox browser has a large community that actively contributes to the safety of the browser, the development of new features and solving bugs. From the very beginning of Rust, Mozilla opted for an insightful update scheme, a flexible and elaborate support for libraries, and a powerful and quickly increasing community. A developer that starts to use Rust will find that excellent documentation is already waiting, a large set of ready-to-go libraries is available and there are many others at hand to help whenever questions or problems occur.

Technolution and Rust

Our introduction to Rust took place in a typical Technolution manner. Most of Technolution's developers have experience with multiple programming languages. One of them needed a well performing language for the development of a videowall application. Java did not offer the desired performance and it had been a long time ago since our colleague had used programming language C. He therefore decided to go with Rust and was pleasantly surprised by the stability, the ecosystem and the ease with which he could work productively in Rust. Other colleagues were soon infected by his enthusiasm. This was enough incentive for us to further explore Rust.

A year has passed since. Rust has proven itself in various projects. The language has gained a lot of momentum and we have started organizing internal and public workshops for developers. The reactions have been promising. There is a lot of interest, both within as well as outside of Technolution.

Trust and investment

We have faith in Rust and are therefore contributing to the Rust community. We are actively involved in several Rust user groups in the Netherlands. We are also working on increasing the applicability of Rust from a content perspective. In a collaboration with the TU Delft, we are, for example, developing a tool that should recognize all possible error conditions in software that have been written in Rust and perform corrective measurements accordingly. The software will no longer collapse and is thus continuously available to execute the required functionality.

In addition, we are also looking into the possibilities for the use of Rust in the open source processor architecture RISC-V. This is currently entirely coded in C. Rust compiles to a native executable in machine code, offers automatic memory management and is therefore particularly suitable for this application. Obviously, we also take any disadvantages into account. Is it even possible with Rust? Are there any consequences regarding performance? What limitations do we encounter? We use a softcore version of RISC-V in our security platform PrimeLink which creates quick secured connections based on encryption that is embedded in electronics. We aim for maximum security and believe that Rust can contribute to it.

Would you like to join our next Rust Gouda Meetup – please join us at meetup.com/Rust-Gouda

Advertorial

Volume 23 • Issue 3 • May 2019

RMD: Design of cryptographic hash functions using Rubik's Cubes

Gijs de Jong, Honours Student Computer Science

The Merkle–Damgård construction is a method of building cryptographic hash functions from one-way compression functions [1]. It is used in many popular hash algorithms such as MD5, SHA-1 and SHA-2. Can a secure hash algorithm also be created by applying this construction to Rubik's Cubes?

Design of RMD (Rubik's Message Digest)

Constructing a secure hash algorithm from represented bits in a Rubik's Cube can be done in infinite creative ways. RMD will use an approach based on solving a $3 \times 3 \times 3$ Rubik's Cube from a superflip state. This state requires the maximum minimum moves of all configurations to solve it.
Blocks of bits will be represented as cubelets, smaller cubes that make up the bigger cube. Consequently, RMD will use 26 blocks. Note that there is no center cubelet. For efficiency and affordable hash (output) sizes, RMD has a block size of 32 bits. This results in a final output of 832 bits.

Description

The hash function RMD can be described in the following four steps:

1. Pre-process the message (input).
2. Initialize a 832-bit buffer $(H_0^{(0)}, H_0^{(1)}, \dots, H_0^{(26)})$ of 26 32-bit variables.
3. Compress each block of 832 bits, split into 26 32-bit words, extended to 28 words.
4. Calculate the 832-bit message digest, or hash.

Pre-processing

To ensure the compression function of RMD has 832 bits available for the hash computation, the input message needs to be padded.

Padding The padding guarantees the message length ℓ is congruent to $768 \bmod 832$. This process can be described as follows: append a single bit "1", followed by k zero bits and the binary 64-bit representation of ℓ. This results in the following definition for k:

$$\begin{aligned} k &\equiv 832 - 64 - 1 - \ell \\ &= 768 - (\ell + 1) \bmod 832 \end{aligned}$$

Dividing the padded message The compression function of RMD compresses fixed sized blocks of 832 bits. To generate these blocks, the message is split into n 832-bit blocks $x_0, x_1, \dots, x_n$. Each of these blocks is then divided into 26 sub-blocks of 32 bits, called words. Message block j is divided as follows:

$$x_j = (x_j^{(0)} \mid\mid x_j^{(1)} \mid\mid \dots \mid\mid x_j^{(25)})$$

Initial value H_0 An initial value for H_0 is needed to start with the hash computation. A 832-bit buffer consisting of 26 32-bit values is used to hold this initial hash value. The values for RMD are generated by taking the first 32 bits of the decimals of the inverse of the decimals of the root of a unique prime number p_n. Variable p is defined to be the series of prime numbers, with n being the n^{th} prime. This results in the following definition for H_0:

$$H_0^{(n-1)} = \frac{1}{\sqrt{p_n} - \lfloor \sqrt{p_n} \rfloor}$$

Hash computation

The hash computation consists of 28 rounds, each executing a specific transformation and pre-defined round operations in successive order. The pre-defined round operations make use of one word W_i and constant K_i per round i. Since the message block consists of only 26 words, it needs to be expanded.

Expand message block The message block is expanded by defining two new words, consisting of previously defined words rotated and XOR'ed together. This results in the following definition for W_i for the message block j:

$$W_i = \begin{cases} x_j^{(i)} & 0 \leq i \leq 25 \\ W0_{\lll_3} \oplus W1_{\lll_{19}} & i = 26 \\ W2_{\lll_7} \oplus W3_{\lll_{21}} & i = 27 \end{cases}$$

This article defines $\lll_n$ to be a circular left shift of n bits. The rotation constants were chosen so as to maximize the shuffling of bits.

Transformations In each round i, a specific transformation $t_i \in T$ is applied to the cube (and consequently to the current hash value H_i). This transformation further resolves the superflip state, which RMD does in 28 transformations (quarter turns). The transformations T recorded in Singmaster notation [2] are as follows:

$$T \equiv \begin{array}{llll} U, & R, & R, & F, \\ B, & R, & B, & B, \\ R, & U, & U, & L, \\ B, & B, & R, & U', \\ D', & R, & R, & F, \\ R', & L, & B, & B, \\ U, & U, & F, & F \end{array}$$

Round operations At the end of each round, a set of the same pre-defined operations is also executed. These operations ensure that the values of the cubelets are mixed sufficiently, which is crucial for achieving security requirements [3]. The round operations use two bitwise functions and 28 32-bit constants. The intermediate value of a cubelet at round i at coordinates x, y and z is denoted by $h_i^{x,y,z}$. The value of the closest top left cubelet is stored at the first index of $(0, 0, 0)$.

RMD makes use of two bitwise functions f_1 and f_2, which are called majority and parity functions respectively. The functions are derived to be efficient and free of bit-based bias. The function definitions are as follows:

$$\begin{aligned} f_1(A,B,C,D,E) =& (A \wedge B \wedge C) \vee (A \wedge B \wedge D) \vee (A \wedge B \wedge E) \vee \\ & (A \wedge C \wedge D) \vee (A \wedge C \wedge E) \vee (A \wedge D \wedge E) \vee \\ & (B \wedge C \wedge D) \vee (B \wedge C \wedge E) \vee (B \wedge D \wedge E) \vee \\ & (C \wedge D \wedge E) \\ f_2(A,B,C,D,E) =& A \oplus B \oplus C \oplus D \oplus E \end{aligned}$$

In each round i, these functions are used to calculate intermediate values of five cubelets located at the edges and center at the top and bottom of the cube. This results in the following definitions for $f_{1,i}$ and $f_{2,i}$:

$$\begin{aligned} f_{1,i} &= f_1(h_i^{0,0,1}, h_i^{1,0,0}, h_i^{1,0,1}, h_i^{1,0,2}, h_i^{2,0,1}) \\ f_{2,i} &= f_2(h_i^{0,2,1}, h_i^{1,2,0}, h_i^{1,2,1}, h_i^{1,2,2}, h_i^{2,2,1}) \end{aligned}$$

Each round i also uses a constant K_i. These constants are similarly generated to the buffer H_0, but uses a cube root in the process instead of a square root.

The process of the end-round operations that uses of these definitions can now be described.
Two operations are used in the end-round operations, which are defined below.

$$\begin{aligned} x^+ &= x + 1 \text{ mod } 2 \\ x^- &= x - 1 \text{ mod } 2 \end{aligned}$$

The two operations are used in two extra defined values:

$$h_{i-1}^{x',y',z'} = \begin{cases} h_i^{z^-,x,y} & x = y = z^+ = 1 \\ h_i^{z^+,x,y} & otherwise \end{cases}$$

$$h_i^{x',y',z'} = \begin{cases} h_i^{x^+,y^+,z^-} & x^- = y^- = z^- = 1 \\ h_i^{x^-,y^-,z^-} & otherwise \end{cases}$$

For every cubelet at the end of round i, a new value $h_i^{x,y,z}$ is obtained through the following modification function:

$$h_0^{x,y,z} = H_0^{(x,y,z)}$$

$$h_i^{x,y,z} = h_{i-1}^{x',y',z'} + \begin{cases} h_i^{x,y,z}{}_{\lll_8} + h_i^{x',y',z'}{}_{\lll_{16}} & x = y = z = 0 \\ +f_{1,i} + f_{2,i} + W_i + K_i & \\ h_i^{x^-,y,z} + \begin{cases} h_i^{x,y,z}{}_{\lll_3} & x = 1, y = z = 0 \\ +h_i^{x',y',z'}{}_{\lll_{31}} & \\ h_i^{x,y,z}{}_{\lll_{21}} & x = 2, y = z = 0 \\ +h_i^{x',y',z'}{}_{\lll_5} & \end{cases} & \\ h_i^{x',y',z'} & otherwise \end{cases}$$

Applying this compression to every block and adding the output together results in the final output of 832 bits, denoted as 208 hexadecimal characters.

Analysis of RMD

The security of a cryptographic hash function is crucial, since it must withstand all known types of cryptanalytic attacks. The speed of a hash function is also important, since it should be sufficiently "slow" to prevent brute-force attacks, but remain able to compute signatures of large files. Both the security level and speed of RMD will be measured and compared to popular hash functions.

Security requirements

Three distinct properties have been found to determine the security level of a hash function [4]. An overview of these properties is displayed in figure 1.

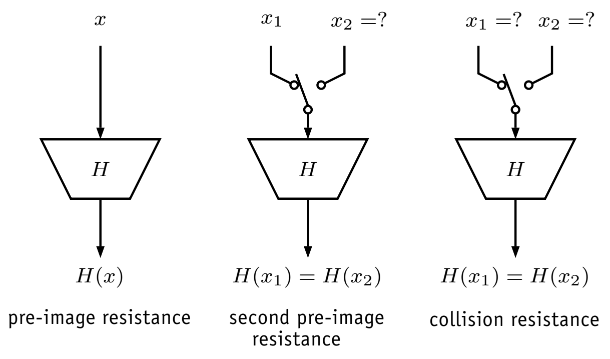


Figure 1: Three main security requirements for a hash function

Pre-image resistance determines the "one-wayness" of a hash function. Given a cryptographic hash function H and a corresponding output (hash) h, it is infeasible to find an input (message) x, such that:

$$H(x) = h$$

Second pre-image resistance determines the difficulty of finding a message x_2 which results in the same hash as a different message x_1:

$$H(x_1) = H(x_2)$$

According to the pigeonhole principle [5], a large domain of such cases, called collisions, must exist for hash functions. Cryptographic hash functions use shuffling and transformations to minimize the chance of a collision occurring. Having a strong second pre-image resistance implies that this chance is sufficiently low.

Collision resistance implies that it is infeasible to find any two different messages x_1 and x_2 that generate a collision:

$$H(x_1) = H(x_2)$$

Security analysis

Preventing collision attacks (and also second pre-image attacks) can be achieved by ensuring the amount of bits in the output of a cryptographic hash function is sufficiently high. Collision attacks are based on the birthday attack, which tries to find a collision with high probability [6]. With a probability of 50%, this results in an attack complexity of $2^{\frac{832}{2}}$. Since this will take hundreds of years to compute, even on a quantum computer, it is currently far from feasible.

Ensuring strong pre-image resistance means to be free of any bit-based bias, and show a strong avalanche effect. This effect implies that modifying an original input slightly should result in a completely different output [3]. Since the degree of this effect can not be derived from the properties of the hash function, we will compare it to the SHA-2 family. Compromised hash functions, such as MD5 and SHA-1, are left out of this comparison.

To test the avalanche effect the following steps are performed. First 10^5 random messages of length $[1, 32]$ are generated. This amount is sufficient to average out outliers. Next, for each message, triplets are generated that consist of the original message, and two where one bit and two bits are flipped respectively. The hash of the original message is then compared with the latter ones, to find the percentage of bits changed. The average percentage of all the messages is displayed in figure 2.

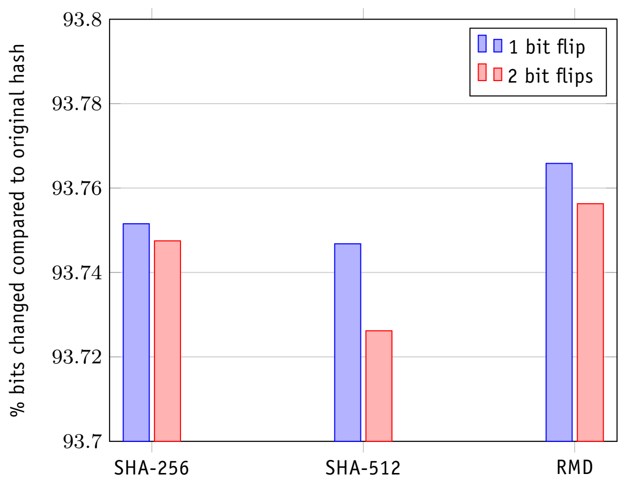


Figure 2: Comparison of avalanche effect between SHA-2 and RMD

The results show that RMD exhibits the avalanche effect to a stronger degree.

Performance analysis

Comparing the execution time of RMD is also important. To do so, the average execution time of 10^5 runs of hashing different file sizes is measured. The results of MD5, SHA-1, SHA-256, SHA-512 and RMD-832 are all taken into account. The hash functions are implemented in C++ and run on an Intel Core i7-4790K with 16 GB of RAM on Windows 10. The results of the experiments are shown in table 1.

Table 1: Hashing speed of hash functions

Algorithm time (ms)	File size		
	1 B	1 KB	1 MB
MD5 (128)	0.0006096	0.005724	5.9290
SHA-1 (160)	0.0007570	0.007921	8.0820
SHA-256	0.0011074	0.013670	13.4830
SHA-512	0.0024840	0.015598	15.4372
RMD-832	0.0025554	0.020423	19.8795

The results show that RMD is slower than the other algorithms. As RMD produces the largest output (and uses a similar construction), this is the expected result. A lower execution time could indicate insufficient shuffling. RMD performs very similar to SHA-512 for smaller inputs, but does not speed up as much when the input gets larger. The older algorithms seem to slow down for larger input.

Since RMD produces a much larger output, comparing the speed per bit is also important. The results for this are given in table 2.

Table 2: Hashing speed per bit of hash functions

Algorithm time (μs)	File size		
	1 B	1 KB	1 MB
MD5 (128)	0.00476250	0.04471875	46.32031
SHA-1 (160)	0.00473125	0.04950625	52.66796
SHA-256	0.00432578	0.05339840	57.12890
SHA-512	0.00485156	0.03046484	30.15078
RMD-832	0.00307139	0.02454688	23.89362

The execution times show that RMD hashes its bits more efficiently compared to the other algorithms. This is not necessarily obvious, as the absolute execution time in table 1 is expected to be higher.

Although RMD seems to be collision resistant, using the Merkle–Damgård construction introduces a vulnerability to length extension attacks [7]. The high bit efficiency of RMD could be used to protect it against these attacks. By truncating the hash to 512 bits, RMD will reserve $832 - 512$ secure bits against these attacks.

Conclusion

If you have read and understood everything so far, you now understand the basics of (constructing) a cryptographic hash function! I really enjoyed designing this algorithm and am happy with the results. RMD delivers comparable execution time to SHA-512 for smaller inputs, a higher avalanche effect and high bit efficiency. The reader with a keen eye may even have spotted that I slightly adapted the used Merkle–Damgård construction to maximize bit propagation. So to answer our main question: yes, we can construct a secure hash algorithm from Rubik's Cubes! And now so can you!

Managed to break RMD? Or have any suggestions for improvements? Do not hesitate to contact me: `g.p.dejong@student.tudelft.nl`

References

[1] I. B. Damgård, "A design principle for hash functions", in *Advances in Cryptology — CRYPTO' 89 Proceedings*, G. Brassard, Ed., New York, NY: Springer New York, 1990, pp. 416–427, ISBN: 978-0-387-34805-6.

[2] D. Singmaster, *Notes on Rubiks magic cube*. Enslow Publishers, 1981.

[3] S. Ramanujam and M. Karuppiah, "Designing an algorithm with high avalanche effect", Dec. 2018.

[4] R. C. Merkle, "Secrecy, authentication, and public key systems", PhD thesis, Stanford University, Jun. 1979.

[5] I. N. Herstein, *Topics in algebra*. Xerox College Publishing, 1964.

[6] M. Tuba and N. Stanarevic, "Relation between successfulness of birthday attack on digital signature and hash function irregularity", *WSEAS Transactions on Information Science and Applications*, vol. 7, Feb. 2010.

[7] Y. Dodis, T. Ristenpart, and T. Shrimpton, "Salvaging merkle-damgård for practical applications", in *Advances in Cryptology - EUROCRYPT 2009*, A. Joux, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 371–388, ISBN: 978-3-642-01001-9.

What a BEP really teaches you: Discussion, Failure and Success

Timo van Asten, Student Computer Science

Have you ever been to an escape room? If you did, you probably noticed that you are being monitored by cameras as you play. The person monitoring you is called the game master. The game master can give you hints when you get stuck or can trigger certain events in the escape room to progress the storyline.

Last year, I completed my Bachelor End Project (BEP for short) at Popup-Escape. Popup-Escape is a small company started by an old TU Delft student. The company creates and hosts personalized escape rooms. Our goal was to create a system to automate the role of the game master.

We came in contact with Popup-Escape through BEPSys. They posted an opportunity to research ways to enhance their escape room experience with AI characters. This seemed like an interesting and fun research opportunity, so halfway through March me and 3 other interested students set up a meeting to discuss the specifics of the project. After this meeting, we were all excited about the project and thus contacted a TU Delft supervisor. We found a supervisor on BEPSys who had showed interest in supervising the project and scheduled a meeting with him as well. With a project and a supervisor in place, we were now ready to get started.

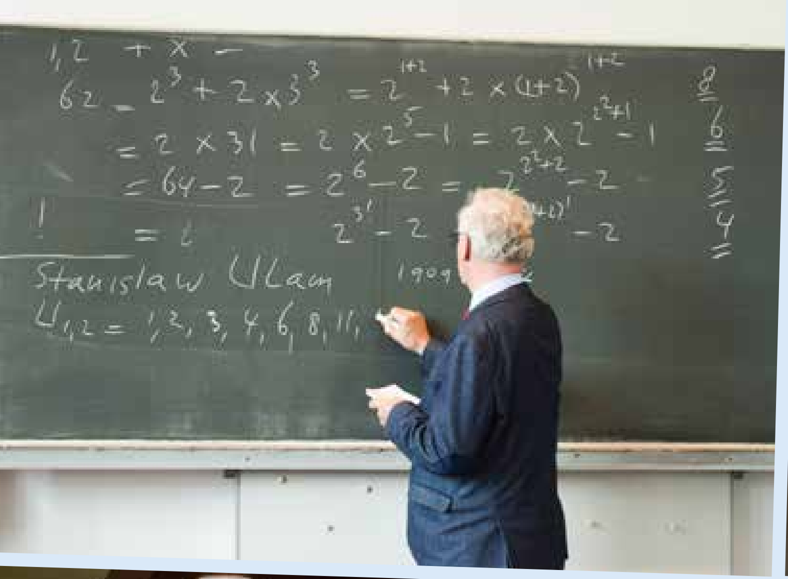
On the 25th of April, we had our first day in the office. Our office was on the 19th floor of the Torenhove, one of the tallest buildings in Delft. We shared that office with another group who were also completing their BEP at Popup-Escape. Walking into that office space you also realize that this project is the only thing you have to focus on for the next couple of months. No lectures, no assignments. We were in charge now, and I liked that feeling of independence. The start of the project was an interesting part of the project. It kind of felt like setting up a small company. Every team member got assigned a role with accompanying responsibilities like Team Leader, Lead Programmer, Head of Communications and Head of Management. You have a vague idea of your goal, but computers do not like vague. You have to choose what technologies to use, what your system will look like and what features it will have. We had multiple meetings with our client discussing our current thoughts on the projects. It became more and more clear what we would be building the next couple months.

In this part of the project I realized our team had quite a diverse mix of personalities. We also had quite a different style on how to tackle these kinds on projects. This led to a lot of fierce discussions. At first, I thought this was a bad thing. The other team in our office did not have any of these discussions, so it seemed like they had everything figured out. Later, I realized that these discussions were needed to get everyone on the same page on what you are going to build and how you are going to build it. The other team ran into trouble with that later on. If there is anything you should remember from reading this, it should be that silence is not always a good sign and it is okay to disagree (a lot), as long as you agree on an approach in the end. It will only make your final product better.

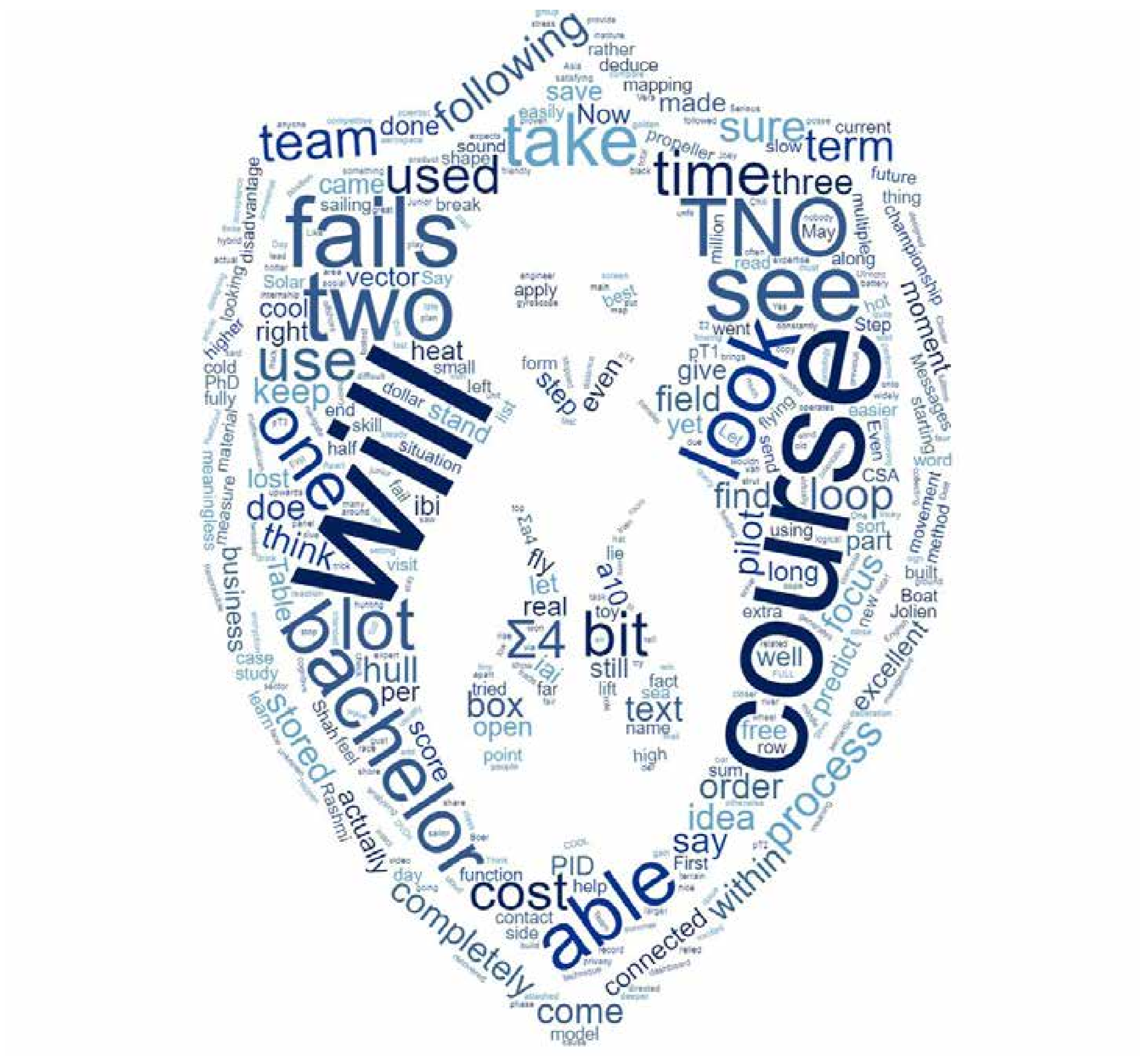
After two months of work, our system began to take shape and we got the opportunity to do something really cool. A group of 27 people would play an escape room powered by our system. We were quite afraid that our system would fail, since we would not get an opportunity to test the system on this scale again. And of course, a lot of things went wrong. A few hours before the test, a service we relied on decided to do maintenance on their servers which was not a scenario we were prepared for. We quickly changed the code to at least get the other parts of our system ready for testing. During our test, that partial system also did not work as expected. We uncovered a lot of user behavior we did not account for in our system. While my team members thought the test was a failure, I thought the opposite. The system did not work as we wanted to, but we gained so much information on how our system could fail. That is the whole point of testing in the end. The remainder of the project we worked on fixing all the mistakes we had uncovered and did another user test. This time, everything worked exactly as expected and our client was very enthusiastic about our product.

Once the project was done, I was the one to present the finished project to our client, supervisor, family and friends. It is nice to show everybody what you have been working on for the last couple of months. After the presentation, our client asked us if we wanted to further develop the system for an upcoming project, which I did during the summer.

After completing your BEP, you realize you have learned a lot during your bachelor. Not only do you have the theoretical knowledge, but also the practical skills needed to make such a big project a success. That is something you can and should be proud of.


Stanislaw Ulam 1909
U1,2 = 1, 2, 3, 4, 6, 8, 11,

Mathematics



Erasure Coding for Storage of Data in the Cloud

Joey van der Krogt, Student Applied Mathematics

I know that anyone reading this article has data, stored somewhere in the cloud. May it be on Google Photos, Dropbox or even Facebook. This data is stored on servers across the world. The problem is, when such a server fails, your data will be forever lost. How do all major IT-companies make sure this doesn't happen to their, and therefore your, data? And how can they do this without spending millions of dollars?

In today's world, the collecting and saving of data plays a tremendous role. A countless amount of companies try to collect data in order to predict future behavior of anything, e.g. customers, terrorist attacks, the weather. These companies store their data in servers which could all be located in the same building, but these servers can also be spread out over different data centers all over the world. Today, Google possesses 15 different data centers, 4 of which are located across Northern Europe, 8 in the United States of America, 1 in Chili and 2 in Eastern Asia. All of these data centers are of course connected with each other to store your photos, your email, your browser and search history, your location and what not. For multiple reasons, sometimes a server node is unavailable. However, these nodes are connected in such a clever way that when one fails, the data can be recovered from the other servers.

All major IT-companies nowadays make use of the same code which secures their data. However, there has always been one great disadvantage to this code: its node repair scheme cost a lot of computing power, and therefore money and time. The authors of [1] found that a tremendous amount (median of 180TB) data was downloaded per day to repair failed nodes. Moreover, this was a data center where only *a part* of Facebook's cold data[1] is stored. This costs these companies millions of dollars per year, so every percent of handling data failure more efficiently would mean the saving of a fair amount of money. This was an issue, until recently a group of researchers from India found a solution [2]. Their code performs very well: it saves not just a percent, it can save 25%-50% of data download! Before we will discuss this, we must look at what causes the problem of the currently used code. For the reader with no knowledge about erasure coding, we start off with two simple examples by which we introduce the two parameters that define the quality of a code. The reader that already has this knowledge can move straight on to the Reed Solomon Code section.

Introduction to erasure coding

An erasure code is a mathematical way of adding "meaningless" data to a system of original data in such a way that when a node fails, the data can be restored from the others. We say that the original data consists of *message symbols* and the meaningless data consists of *parity symbols*. We will look at two basic ways of storing 16 message symbols. For simplicity the symbols here are bits, and we will see how efficient these simple codes perform by looking at the two most important parameters: *storage overhead* and the *repair bandwidth*. Both definitions speak for themselves when we discuss the following

[1]Cold data is data that is rarely used or accessed.

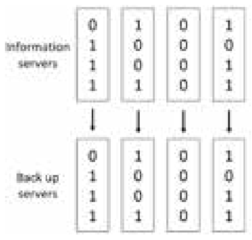


Figure 1: A direct copy of every information server (often called 2-Replication).

two examples. The code above stores 32 symbols where the original data was only 16 symbols, which makes the *storage overhead* = 32/16 = 2. For the repair bandwidth we look at what happens when we want to repair 1 failed node. When the first information server fails, we have to download the 4 symbols of the first back-up server in order to restore the lost information, making the *repair bandwidth for 1 failure* = 4/16 = 25%. Let's compare this to the following code.

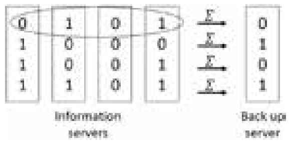


Figure 2: One parity server based on the sum of each row of bits of the information servers.

This code stores 20 symbols, so the storage overhead here is only 20/16 = 1.25. This means that the second code scores better in terms of storage overhead. However, when node 1 fails we need to download all remaining 16 symbols (and take the sum of each row) which makes the repair bandwidth for 1 failure = 16/16 = 100%. This means that the first code scores better in terms of the repair bandwidth (when only 1 server fails), but the second code scores better in terms of storage overhead.

The two codes I studied also had this trade-off. You could say that a code that has low storage overhead costs but a slow repair method is more useful for cold data, since it does not matter how long it takes to repair the data. Where a code with a faster repair method and higher storage overhead costs, is more useful for hot data, since that data needs to be repaired as quickly as possible. Now let's dive somewhat deeper into the currently frequently used code: the Reed Solomon Code.

Reed Solomon Code

From what you have read in the preface, you probably think that the Reed Solomon Code is a rather useless code and that the world was waiting for it to be replaced. I want to stress that this is certainly not the case! The Reed Solomon Code is a very beautiful and elegant code which made all sorts of communication (from reading DVDs and bar codes to receiving pictures from the Voyager Spacecraft) possible. It is far too complex to explain in this amount of text, however a lot is written about them for those who are interested. We will unfortunately only look at its disadvantage, the repair bandwidth, so that we can see how this problem was recently solved. We will see in a bit that for the Reed Solomon Code, the repair bandwidth is always 100%, which makes node repair extremely expensive and slow. It not only costs money, but when a server fails it also takes a lot of time to repair, making it unfit for the storage of hot data.

Facebook makes use of the (14,10)-RS code, meaning that to 10 servers with message symbols, 4 servers with parity symbols are added. Again, we will not look at how the Reed Solomon code works exactly, we will only discuss a superficial way of looking at them, in order to understand their problem. The only real important thing you have to know up front is that symbols are not bits anymore: we now have more than 2 options per symbol. This makes multiplication possible. When for example we multiply a bit by 2, it is either 0*2 = 0 or 1*2 = 0 and you can see the problem with that. This is now not the case anymore, because we work over the finite field $GF(p^r)$ with p prime and r a positive integer. When for instance we choose $GF(2^4)$, a symbol consists of 4 bits, making 2^4 = 16 possible symbols. The only thing you need to remember for now is working with non-binary symbols makes multiplication "possible".

Say we have a message **a** consisting of 10 symbols $a_1, ..., a_{10}$. The way a parity symbol is created, is by multiplying **a** to a coding vector **p**. All 4 coding vectors are linearly independent. This gives a situation like this:

Node 1	a_1
⋮	⋮
Node 10	a_{10}
Node 11	$\mathbf{p}_1^T\mathbf{a}$
Node 12	$\mathbf{p}_2^T\mathbf{a}$
Node 13	$\mathbf{p}_3^T\mathbf{a}$
Node 14	$\mathbf{p}_4^T\mathbf{a}$

*Table 1: The result of coding a message **a** of 10 symbols, according to a (14,10)-RS encoding scheme.*

You can see that the storage overhead is 14/10 = 1.4, which is excellent. However, when node 1 fails, the optimal repair scheme is to download the parity symbol of node 11 ($\mathbf{p}_1^T\mathbf{a}$), along with the message symbols $a_2, ..., a_{10}$ (encode them) and subtract these from the parity symbol. This means that the repair bandwidth for 1 failed node is 10/10 = 100%.

When 2 nodes fail, say nodes 1 and 2, we download the message symbols of nodes 3,...,10 along with the parity symbols of nodes 11 and 12, we subtract the message symbols, and we have two linearly independent equations, with two unknowns, which is solvable. Again a repair bandwidth of 100%. This way you can deduce that it can repair up to 4 node failures, which is also excellent, but the repair scheme is not that impressing. The researchers that came up with the following code saw the same problem. They discovered a way to keep the excellent parameters intact, whilst improving the repair bandwidth. They came up with the brilliant idea of Piggybacking.

Piggybacking

The real problem with the Reed Solomon Code is that each parity symbol consist of a mapping of *all* message symbols. This way we always have to download all remaining message symbols to subtract them from the parity symbol(s). The saving of the repair bandwidth in Piggybacking lies in the fact that the researchers tried to save downloading message symbols. They do this by (here comes a tricky sentence) "piggybacking" parts of message vectors onto other parity symbols. We will illustrate with the following toy example what we mean by this. Say we use a (6,4)-code, not necessarily RS, which stores two messages **a** and **b** next to each other. We have the following situation:

Node 1	a_1	b_1
Node 2	a_2	b_2
Node 3	a_3	b_3
Node 4	a_4	b_4
Node 5	$\sum_{i=1}^4 a_i$	$\sum_{i=1}^4 b_i$
Node 6	$\sum_{i=1}^4 ia_i$	$\sum_{i=1}^4 ib_i$

*Table 2: Messages **a** and **b**, both of length 4, encoded according to some (6,4)-code scheme, not necessarily RS and stored next to each other.*

Same as with the Reed Solomon Code, we see that when a node fails, say node 1, we need to download the parity symbols of node 5, and subtract the message symbols of nodes 2, 3 and 4, to restore the lost symbols. The repair bandwidth again is 8/8 = 100%. What we now mean by piggybacking is done in two steps. **Step 1.** add half of the summation of node 6, instance 1, to the parity symbol of node 6, instance 2. **Step 2.** subtract that result from the parity symbol of node 6, instance 1. This looks as follows:

Node 1	a_1	b_1
Node 2	a_2	b_2
Node 3	a_3	b_3
Node 4	a_4	b_4
Node 5	$\sum_{i=1}^4 a_i$	$\sum_{i=1}^4 b_i$
Node 6	$\sum_{i=3}^4 ia_i - \sum_{i=1}^4 ib_i$	$\sum_{i=1}^4 ib_i + \sum_{i=1}^2 ia_i$

*Table 3: Messages **a** and **b** first encoded by some (6,4)-code scheme and then piggybacked*

Now the magic happens, because when now node 1 fails, we download all 5 remaining symbols from instance 2, together with the symbol a_2. First we deduce b_1 by subtracting b_2, b_3 and b_4 from $\sum_{i=1}^4 b_i$. Then we subtract all four message symbols b_i together with a_2 from the parity symbol in node 6, instance 2, and we know a_1. We did this by downloading only 6 symbols, making the repair bandwidth 6/8 = 75%! We also see that this did not cost us anything extra. There is no extra data saved, the storage overhead did not increase, and it can still repair up to 2 failures, which is equal to the amount of parity nodes. It simply decreased the repair bandwidth.

Like I said, the example above is just a toy example. The real Piggybacking encoding scheme is a bit more complex, and in practice we mostly use more nodes with longer messages, making calculations more complex. And secondly, what happens when a parity node fails? And what happens when 2 nodes fail? Or 3 or 4? The Piggybacking scheme is a complex scheme, and it does not help for any amount of failures, however, it increases the performance of the Reed Solomon Code drastically. During my thesis I found that for 1 failed node the repair bandwidth of the Piggybacked (14,10)-RS code, which Facebook uses, is 67.21%.

Currently the Piggybacking codes are being tested in multiple IT-companies. I am very curious if the beautiful Reed Solomon Code will disappear from the face of the earth. May it be because of Piggybacking, or because of an even faster code that is yet to be discovered.

References

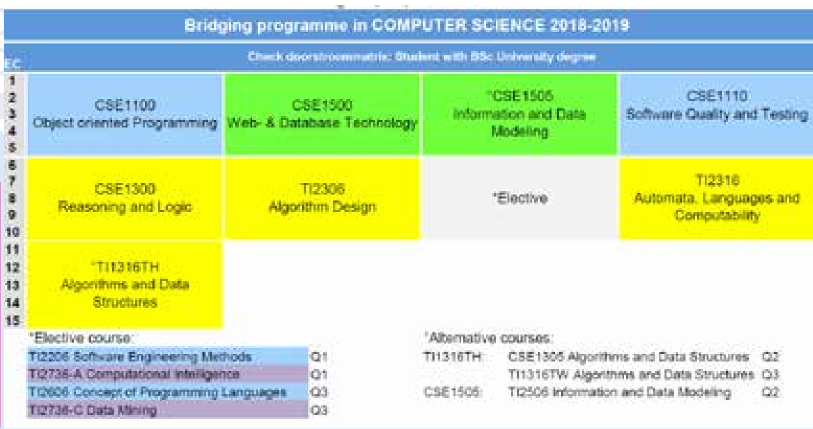
[1] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, K. Ramchandran *A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster*, 2013.

[2] K. V. Rashmi, Nihar B. Shah, Kannan Ramchandran *A Piggybacking Design Framework for Read-and Download-efficient Distributed Storage Codes*, 2013.

Volume 23 • Issue 3 • May 2019

Bridging minor to Computer Science

Vera Martens, Student Applied Mathematics



Bridging programme in COMPUTER SCIENCE 2018-2019

Check doorstroommatrix: Student with BSc University degree

EC				
1–5	CSE1100 Object oriented Programming	CSE1500 Web- & Database Technology	*CSE1505 Information and Data Modeling	CSE1110 Software Quality and Testing
6–10	CSE1300 Reasoning and Logic	TI2306 Algorithm Design	*Elective	TI2316 Automata, Languages and Computability
11–15	*TI1316TH Algorithms and Data Structures			

*Elective course:

TI2206 Software Engineering Methods	Q1
TI2736-A Computational Intelligence	Q1
TI2606 Concept of Programming Languages	Q3
TI2736-C Data Mining	Q3

*Alternative courses:

TI1316TH:	CSE1305 Algorithms and Data Structures	Q2
	TI1316TW Algorithms and Data Structures	Q3
CSE1505:	TI2506 Information and Data Modeling	Q2

Are you an Applied Mathematics bachelor student, but not sure if you want to do a Mathematics master afterwards? Do you like programming and algorithms? Then maybe it is an idea to make the switch in your minor!

First, let me introduce myself: my name is Vera, I'm 21 years old and I'm a third year Applied Mathematics bachelor student. In September, I started with the bridging program to Computer Science, which I do as my Free Minor. I came up with this idea myself and so far it has worked out really well! What it exactly is, how I made it work and why, you can all read down here!

Even though I like mathematics, I do not have a particular area in mathematics that I really like. The course that I like the most is optimization, which is the most Computer Science-ish course we have in the AM bachelor. I had already done the CS elective course in my first year and really liked it. This made me think about doing a CS minor.

I checked out all the minors within EWI, but they were all still too mathematical in my opinion. I went to the Master Orientation Day to find out what track within CS I would like the most and if it would even be possible for me to do a CS master after my bachelor. Yes, it is possible, but only if I would do the bridging program to CS.

The bridging program to CS contains a total of 45 EC of courses that cover the knowledge that you necessarily need to be able to do a master in Computer Science. You can see the courses down here, although it possibly differs for the year 2019-2020. Normally, you would do this after your bachelor, and after the bridging program you would then be allowed to do the master. However, I decided to do this switch during my bachelor instead of afterwards.

This was possible for me, because I will do my bachelor in 3.5 years, and so I had the time and space to do the CS courses next to my AM courses. Furthermore, I did not have to do all the courses. Because of my AM bachelor, I did not have to do Reasoning and Logic and OOP. The course material would already have been covered in the AM courses Mathematical Structures and Introduction to Programming. As already mentioned, I also chose Algorithm and Data Structures as elective course in my first year, and thus I did not have to do that course either. For me only 30 EC were left, which fits in a minor.

Even if you did not do Algorithm and Data Structures in your first year, it is still possible to do the switch. For example, it is allowed to finish your program after starting your master.

Either way, it all sounds straightforward, but the process actually took quite some time. This is because I had to follow two procedures: the one of the bridging program and the one of the free minor. After I had found out what I wanted to do, I went to Jolien Kooijman, the academic counsellor for bridging students, among others. She explained to me everything about the program and answered my questions. She also gave me the schedule of the courses, and explained the process, which you can also find on the website[1].

It is important that you visit her before starting any step of the process, so that everything will go smoothly and she knows about you when applying for the program. This is namely the next step: I had to apply for the program through mail, where I also motivated why I wanted to do the program and what courses were left for me after talking to Jolien.

After a while, I received an email with the good news that I was accepted, which had as attachment a form with the right courses on it, which I had to sign and send to the CSA. I also visited the CSA several times to make sure that my registration in StudieLink would work out well.

The process for the free minor is the same as usual. However, it is important to know that it is better to wait for the confirmation of acceptance before you start this second procedure. Furthermore, it is also good to know that you need the signature of the director of Computer Science, which is Hans Tonino now. However, another person will take over his place next year, so make sure that you visit the new director of Computer Science for the signature.

At the moment I'm half way through the program and I really like it! Compared to AM, I like how practical CS is and it feels like solving some sort of puzzles and challenges. I also like that during the programming, you get feedback on what works and what doesn't, while in mathematics you do not get feedback about your proof. The way of finding out what you are doing by getting feedback during the process is very nice. I'm glad that I could have this experience, also to find out what I like within CS. Besides, now I also have the opportunity to do a CS master if I want to. Therefore I'm happy that it was possible to do the bridging program during my minor.

I hope you enjoyed reading this, and that I could give you a good idea of how the whole procedure works.

If you have any questions, feel free to contact me!

References:

[1] https://www.tudelft.nl/studenten/faculteiten/ewi-studentenportal/onderwijs/studeren-bij-ewi/schakelprogrammas-ewi/

Dreamteam as a Math student

Mariette Schonfeld, Member of Solar Boat Team

Hello, my name is Mariette and I am a proud member of the TU Delft Solar Boat Team! I am a fulltime member, which means that I completely put my studies - namely my bachelor Applied Mathematics- on hold for an entire year.

Within one year, we design, produce and race a hydrofoiling solar boat with a team of 28 students. Hydrofoiling means that the boat has wings underneath the hull attached with a strut. When the boat reaches about 30 km/h, due to the hydrofoils, the hull completely lifts out the water and the only contact point is the wings. This means our boat actually flies!

When I tell people that I am in this team, I usually get the same reaction. It starts with: “Oh, that sounds really cool!” followed by: “But you are a mathematician, how is that useful for making a boat?” But I have learned this year that a team such as ours needs a lot of different skills to reach the best product. When you think of a boat, you probably think of the shape of the hull. But this is just a small part of what makes a boat. You also need a propeller to move, a steering system to navigate, a way to store energy and get that energy to the propeller, a dashboard for the pilot to control the boat and a way to store all the information and to show this on a screen for the pilot. These last few functions are where the main focus of my department lies on: the electronics.

The electronics department focuses on harvesting and storing the energy from the sun using solar panels, batteries and the electrical controls within the boat. This means that the pilot can see on his steering wheel where exactly he is on a map, how fast he is sailing and the motor temperature. Then he is also able to control the motor according to that information. Apart from that, he also has control over how high the boat is flying. This is where my function comes in: I am responsible for programming the boat to fly.

I am of course not the only one working on the flying functionalities of the boat, there are several aerospace engineers working on the shape of the wings and the materials used for it. But to make it easier for the boat to fly, we use a height control system. This is a system that is able to move the wings to change the angle of attack. If the wing is parallel to the flowing of the water, the boat will not be able to lift so easily. If you point it upwards though, the boat will rise out of the water a lot easier. The height control system moves the wing to help the boat take off and keep a steady height above the water. This system operates fully autonomously, apart from the pilot setting the desired height.

The height control system has several sensors around the boat, namely three height sensors to measure the distance to the water and one gyroscope to measure the orientation and acceleration in the three dimensions. These sensors are connected to control boxes: these are like tiny computers that I designed and built myself. The control boxes take in the sensor data, see the difference between the desired height and the current height and then calculate how the wing needs to change position and send this information to small electrical motors that change the wing angle.

While the designing of the control box and programming the sensor and the motors relied more on the skills that I gathered during my minor in computer science, the actual calculation of the wing angle is a lot more mathematical. The calculation makes use of a PID loop: this is a proven effective way to control systems using sensors. Think of it as a thermostat with a heater and an air-conditioning. If the desired temperature is higher than the current, the thermostat will switch the heater on. But when does the thermostat know when to switch the heat off? If you do so when you reach the desired temperature, you are already too late. The heater needs time to cool down, so the residual heat will make the room hotter than first intended. You’d want to switch the heater off just before you reach the right temperature, so the residual heat will be just enough to reach the right temperature.

This is where the PID loop comes in. P stands for proportional gain, as in, the motor goes faster the larger the angle is. I stands for integral: the loop looks at a past movement to predict what the next movement will result in. D stands for derivative: the loop also looks at the future to predict when to stop. The PID loop takes in several parameters that are dependent on the system it is used in. These parameters can be tweaked while testing and seeing the result, but they can also be predicted by modelling our boat in MATLAB.

In the end, it is impossible to completely model the behavior of the water and our boat. This is why testing is a very important phase. In the end, we are going to compete in the world championship for sustainable boats: the Monaco Solar and Energy Boat Challenge! For the first time, we are participating in the offshore class, which means that we will be sailing on open sea rather than shore waters and rivers. The design of our boat is completely different from previous years in order to cope with the high waves and heavy gusts of wind on the open sea. This year’s boat will take the form of a trimaran. This is a boat with three hulls, so the boat can be built widely to provide the stability that is needed. With this boat, we plan to win not only the world championship for solar boats in Monaco this summer but also to break a world record by crossing the English Channel as the fastest solar boat in the world. We are the only student team, so we are very excited! If you are interested in joining the next generation of solar sailors, be sure to check us out on social media or come to one of our interest drinks!

“Top of my list of employers since my Bachelor’s”

TNO

Maaike de Boer is a Junior Scientist Innovator at Data Science. She sees TNO as the golden middle ground between university and business, focused on independent research in the long term, yet close to the customer.

“At Data Science we focus on sharing and analysing structured and unstructured data. We do this in three ways, thereby dividing up our fields of work. Explainable Data Science looks at how you can make data analysis explainable: how can you break open a black-box algorithm, for example? Responsible Data Science is about the privacy-friendly side of data sharing and analysis. An example of this is encryption. And Interoperable Data Science investigates how to share data in a structured and meaningful way, for example through blockchain technology and ontologies. We actually work continuously with all units, for solutions for the medical world, defence, logistics and the agricultural sector, among others. In this context ‘hybrid AI’, a combination of knowledge-driven and data-driven artificial intelligence, is emerging strongly. At the moment I’m working on the question of how you can build a structured knowledge model from unstructured text. In other words: which text leads to which information?”

COGNITIVE ARTIFICIAL INTELLIGENCE

“I studied cognitive artificial intelligence at Utrecht University. After that I was able to do a PhD via TNO and Radboud University. My dissertation was about semantic mapping in retrieving video images based on a search question. I was able to apply a combination of machine learning and text mining. A difficult task, because there were no examples of how we could train the system. I used the birthday party as an example: which words are related to the query – for example decorations or a party hat – that you can use to let the system find images that look like a birthday party? After my PhD I immediately started working at TNO. For me it was logical, because I had already done an internship here during my Bachelor studies. From that moment on TNO was already at the top of my list: since then I tried my best to get to TNO!”

“TNO does very cool research that you wouldn’t easily be able to do at university or in business.”

COOL RESEARCH

“TNO does very cool research that you wouldn’t easily be able to do at university or in business. At university you are constantly hunting for research funding, in a very competitive environment. In the business world, research can be stopped just like that or be directed in a different direction because it generates more money. As an independent institute, TNO can really focus on the medium to long term. Serious research, and yet you are closer to the customer than you are at the university. Moreover, TNO has so many experts in virtually every field of expertise. Let them get together in a multidisciplinary team and you get results that nobody expects. Moreover, you learn a lot from each other. In fact, you keep learning here instead of just repeating the same trick you once learned.”

AI IN FULL DEVELOPMENT

“TNO gives me the freedom to choose my own projects. I get the most energy from being able to solve a problem for a customer. By programming. Or by brainstorming together with the customer or in a multidisciplinary team. My day is also satisfying when I learn something new, for example about the world of the customer or about an unexpected application of a technique. I am now a junior and I would like to continue on the scientist side: continue to do a lot of research! Now and then I do some project management and I would like to continue doing that for the variation it brings. But the terrain of AI is just developing so much. You have to stay fully focused on this, otherwise you just won’t be able to keep up with this challenging field…”

Advertorial

Miscellaneous

Science Trends

Willemijn Tutuarima, Editorial Staff MaCHazine

The older you get the faster time goes

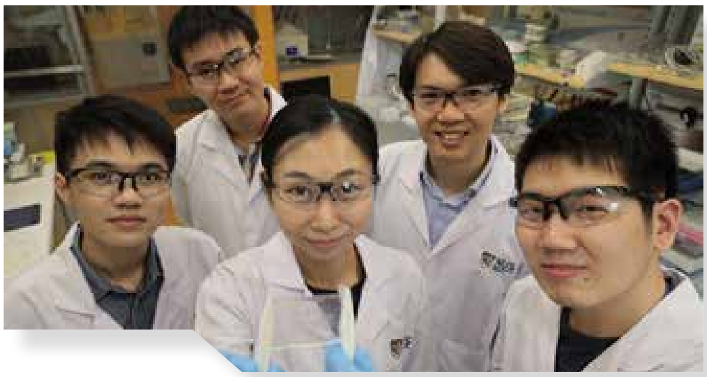
Perception of time isn't the same for everyone. People who are older perceive their current time to go faster in comparison to time when they were young. They experienced their younger days to be longer while they were able to remember everything. J.A. Jones, Professor of Mechanical Engineering at Duke University, started a research into this phenomenon and came up with an interesting theory. It is actually quite simple but explains a lot. The brain makes new neural connections every day so when someone is younger they have a lot less connections than when they are older. In the brain electrical signals run through these connections, which means that these signals will get to their target slower if there are more connections. On top of this, the older the brain gets the less stable the connections are and this leads to more time required for every signal to be processed. The last piece to the theory is that the older someone gets, the less images they process due to a slower brain. Because the passing of time is highly related to the changing of the images in your head, time seems to go by faster than when a lot of images are processed.

Apple Watch used as Cardiac Consultation

The brand Apple Inc. is well known for their elegant and expensive products, from phones to computers and since April 2015 the Apple Watch entered their collection. This watch has a lot of different features like: checking texts, making photographs and keeping track of your heart rate. This last feature is very interesting because of course the watch is worn on the wrist where an artery is located right under the skin. Now Apple has started funding a research which looks for a way to make more use out of this feature. The researchers believe that the Apple Watch could detect heart problems and started a large experiment with 419.000 applicants who wear the watch to see if this could actually be a possibility. The watch could apparently see if the person wearing it could have a heart condition called atrial fibrillation which is a heart problem that could lead to a stroke or heart failure. A part of the people who wore the watch got a message that some irregularities were found and a third of the participants that got a consultation afterwards actually had problems with their heart. This shows that a lot of research still needs to be done but Apple believes that it is already a step into the right direction.

Self-Healing Touch Screen

Almost everyone has had the problem of using a touch screen outside when it is raining and finding out that they actually don't work that well. A team of scientists from the National University of Singapore (NUS) started a year long research into making touch screens that would have at least three aspects: they had to be easy to use when wet, they had to be transparent and they had to self-heal. The team got inspiration from one of the most interesting sea creatures, the jellyfish, because they are transparent and they sense the environment although it is wet. They created a gel which consists of a polymer and ionic liquid which, when combined, allows to self-heal because the polymer network interacts with the ionic liquid via highly reversible ion-dipole interactions. The skin is created by printing novel material into electronic circuits and turns it into a soft and stretchable material. It's electrical properties change when being touched which can be measured and converted into readable electrical signals. There are two large advantages of using this technology: Firstly, it makes a machine more mechanical compliant in human-machine interaction because it mimics biological tissue and secondly, because it would create less waste than other touch screens because of its self-healing properties. Who wouldn't want to have a cell phone where the cracks in their screen fix themselves?



Miscellaneous

Historical Person: David Hilbert

Daniël van Gelder, Editorial Staff MaCHazine

Early years and personal life

When a child is born into a successful and wealthy family, it often carries high expectations for the child to do well. Such was the case for David Hilbert, born in 1862. His father was a judge and his mother, although she didn't work, came from a wealthy family and had a deep fascination for mathematics. David had a very strict upbringing and this was exaggerated by the fact that he was his parents' only son. Surprisingly, for a successful future mathematician, David didn't do well in school. He described himself as a 'dull and silly boy' in his school years, this already shows that he didn't enjoy it that much. However, when David changed schools for his final year and was educated with a different approach, he showed great talent for mathematics. Hilbert himself reflected about this the following way:

'I didn't work especially hard at mathematics at school, because I knew that's what I'd be doing later.'

Mathematical Works

Hilbert's list of 23 problems

Due to his very broad and deep understanding of mathematics, Hilbert contributed in many areas of mathematics. One of the works he is very well known for is his list of 23 problems. This was a list of problems varying greatly in topic and nature and were unsolved at the time. Hilbert argued that if these problems were to be solved, then it would raise the whole of mathematics to a new level. Many academics worked avidly on the problems and many were solved within several years after the publication. However many remain to be solved. The first problem that still has to be solved was problem number three which was formulated as follows:

"Given any two polyhedra of equal volume, is it always possible to cut the first into finitely many polyhedral pieces that can be reassembled to yield the second?"

It turned out that this was not true and was proved to be wrong as early as 1900, in the same year as the publication. One of the few unsolved problem is for example the Riemann hypothesis.

Hilbert's Paradox of the Grand Hotel

Another famous work by Hilbert is the paradox of the Grand Hotel. It is a thought experiment showing the counter-intuitiveness of infinite sets. This thought experiment was introduced in a lecture in 1924. The paradox is about a fully occupied hotel with an infinite amount of rooms. Suppose that a new guest arrives at the hotel, how would the owner fit the person in if all rooms are occupied? The owner decides to move every person up one room, so the guest in room 1 goes to room 2, and the guest in room n goes to room n + 1. This procedure makes room 1 available for the new guest even though no one left the hotel.

You can do this for any finite amount of new guests that arrive. In fact, there are also ways to do this for an infinite amount of new guests, but the approach is slightly different. The thought experiment shows that the two statements 'there is a guest in every room' and 'no more guests can be accommodated' are not necessarily equivalent. This paradox is a very interesting introduction to the theory of infinite sets.

Hilbert's Program

Another famous work by Hilbert is a proposed solution to the so-called 'foundational crisis in mathematics', also referred to as Hilbert's program. Mathematics suffered from a lack of a formalised foundation. Many previous attempts suffered from inconsistencies and paradoxes. Hilbert believed that there was a way to ground all mathematical theories in a finite set of axioms, all of which were to be proven to be consistent. This would finally provide a consistent basis for mathematics from which all theories could follow. Hilbert had strong philosophical reasons for finding this foundation, saying:

"We are not speaking here of arbitrariness in any sense. Mathematics is not like a game whose tasks are determined by arbitrarily stipulated rules. Rather, it is a conceptual system possessing internal necessity that can only be so and by no means otherwise."

Unfortunately, Gödels incompleteness theorems from 1931 has shown that Hilbert's program is unattainable in some key areas of mathematics. He showed that one could always construct a statement which is true that does not follow from the established set of axioms. In addition, he showed that the system could not be able to prove its own consistency. Therefore the system can't exist in a formalised way.

Nevertheless, Hilbert sparked a debate that would continue to rage on for many decades about the foundations of mathematics. Watered down versions of his program have already been adopted and shown to be consisted. This debate also put down the groundwork for a theoretical basis of computer science, as worked on by Alonzo Church and Alan Turing. Therefore Hilbert is still well-known and appreciated for his work.

Later years and Death

David Hilbert worked for many years at the University of Göttingen. In his late life and near his retirement the Nazi's took over Germany and purged the university of many of its Jewish scientists, many of which were his friends. It is a sad and lonely ending for a man with such great achievements. He is most well-known for the following famous words, which were also inscribed on his tombstone:

"Wir müssen wissen. Wir werden wissen."

"We must know. We will know."

References:

[1] https://www.britannica.com/biography/David-Hilbert
[2] https://www.famousscientists.org/david-hilbert/
[3] https://en.wikipedia.org/wiki/David_Hilbert

Minor abroad

Marjolein Leegwater, Eva Slingerland, Daniël van Gelder, Third year Bachelor Students

During the first semester of the third year of your bachelor, every student needs to do a minor. This can be done at your own faculty, another faculty in Delft, another faculty at another university or abroad. Three of our editors chose the last option and want to tell you about their experience.

Marjolein Leegwater

University of Pittsburgh

For my minor I decided to go abroad for a semester to study at The University of Pittsburgh, in Pittsburgh, Pennsylvania, in the East of the United States. The university has almost 30.000 students, an American Football team with cheerleaders, an orchestra, a mascot, pep rallies and all the other clichés. Living on campus has been an amazing experience for me: The American "meal plan" food, gyms in every dorm, a gigantic cathedral on campus, the way lectures work and all the foreign people that I met. I have seen so many things!

Apart from the campus, the city has a lot to offer. I have been to an NFL Football game of the Pittsburgh Steelers, to an Ice Hockey game of the Pittsburgh Penguins, the overlook where you can see the skyline. I visited museums, theaters, cinemas and a lot of restaurants in the city. And that's just Pittsburgh. I have made trips to many places, including New York City, Washington D.C., Boston, Chicago and Toronto.

I had the opportunity to spend Thanksgiving with an American family in Buffalo where we ate the traditional turkey and pumpkin pie. It's hard to summarize everything I did and everyone I met in an article of only half a page. I am really happy I got the opportunity to go abroad. Some adjustments are hard and not every day is as amazing as others but I have so many unforgettable memories about my time there. If you are considering to study abroad, I would definitely recommend going.

Eva Slingerland

Kungliga Tekniska Högskolan

For my minor abroad, I have spent a semester at the Kungliga Tekniska högskolan Royal Institute of Technology in Stockholm. Most of the courses are 7.5 ECTS, so for my minor I had to choose 4 courses. I ended up taking a Swedish language course, two machine learning courses and a course about the theory and methodology of science. Pretty soon, I noticed that the courses didn't demand as much work as what I am used to in Delft and that the TU Delft is very highly respected. When I told people that I study there, they were really impressed and a lot of students I talked to told me they applied there but didn't get in.

I joined Osqstämman, which is a choir, and THS MAIN, which is an association for international students. With MAIN, I organized and participated in pub nights, a beer pong tournament, a karaoke evening, a pub crawl and a big Christmas dinner and party. It's really exciting to meet people from all over the world, but I wanted to get to know some Swedes as well, which was one of the reasons for me to join Osqstämman. About half of the choir was Swedish and the other half international, but we sang both English and Swedish songs which was pretty hard, especially since we needed to know quite a lot of songs by heart for some performances. We had a fall concert together with an orchestra and ballet group, a Christmas concert in a beautiful church and Lucia performances at companies throughout the city, so it was really awesome to be part of all this. Doing your minor abroad is really a challenge: you have to arrange a lot of stuff and being on your own in a completely new environment can be hard sometimes, but in my opinion it's definitely worth it!

Daniël van Gelder

Nanyang Technological University Singapore

For my minor, I decided to take a leap of faith and go study abroad. During the last semester I spent my time studying at Nanyang Technological University in Singapore (NTU). I had an unbelievably great time and an experience of a lifetime! Not only did I get to meet some awesome people, but I also got to travel around Asia and see many great things. However, studying was of course the most important part of my time there. For my minor I took five courses: three computer science courses, an economics course and a Chinese language course. It was a lot of fun studying Chinese and although I only learned the bare basics, the course was really interesting. As for the other courses, they were less interesting as the level wasn't that high and the content was not that interesting. The workload was in general lower than in Delft. This gave me a lot of opportunities to travel and do cool stuff in Singapore itself. Singapore is a really interesting country and is a kind of hub for Southeast Asia, this means that you can do a lot of cultural stuff there. On the other hand, it is a very modern country where you can do lots of shopping and partying.

I got to do a lot of travelling around Asia and got to visit amazing countries. The countries I visited were: Malaysia, Vietnam, Thailand, Taiwan and Hong Kong. There are, however, many other countries that I would like to have visited but didn't get the chance to travel to. I would recommend anyone to study abroad for their minor, it is an amazing experience where you learn a lot. I'm very glad I took the opportunity to go to Singapore and would do it all over again.

Computer Science puzzle

Louise Leibbrandt, Editorial Staff MaCHazine

Extreme logical thinking: card sorting, well sort of

a) One day you are shown the four cards below. You are told that each card has a number on one side and a letter on the other. You are also told that every card that has a vowel on one side has an even number on its opposite side: a nice and simple fact, but is it true? Which card or cards must you turn over to prove whether or not the vowel/even rule is true? Explain why.

Card A	Card B	Card C	Card D
E	G	2	3

b) On another day, you are shown a different set of four cards as below. This time each card has the details of a person in a shop on it. On one side is their age and on the other is what they are buying. If a person is buying fireworks then they must be over 18. Which cards should you turn over to check everyone is shopping legally? Explain why.

Card A	Card B	Card C	Card D
Age 25	Buying Rockets	Age 16	Buying Milk

Solution to last issue's puzzle

The message was:

THE YOUNG ADA DREAMT OF BEING ABLE TO FLY AND APPROACHED THE PROBLEM OF HER EARTH BOUND STATE WITH TYPICAL GUSTO AND A SCIENTIFIC EYE The following key was used to encrypt it, though not all letters appear in the message. Therefore, the full key can't be reconstructed from this message.

K^{1}	D^{2}	M^{3}	V^{4}	N^{5}	T^{6}	A^{7}	O^{8}	U^{9}	B^{10}	J^{11}	I^{12}	S^{13}
X^{14}	Z^{15}	L^{16}	C^{17}	W^{18}	F^{19}	Y^{20}	G^{21}	P^{22}	R^{23}	E^{24}	H^{25}	Q^{26}

Volume 23 • Issue 3 • May 2019

Mathematical Puzzle

Kilian Buis, Editorial Staff MaCHazine

Problem 1
Suppose you are the cargo director of a ship and you have agreed to transport several tanks containing rare fish that you need to bring from A to B. Unfortunately, as you are passing through shark-infested waters, the boat is battered by a fierce storm throwing your precious cargo overboard. And to make it worse, no one seems certain just how many fish tanks are missing.

Fortunately, you have a rescue sub at your disposal, but only enough fuel for one trip to the ocean floor. You need to know where the tanks are, so you can gather them all in one quick pass. Not a single fish can be lost. You find out there are 3 sectors where the cargo could have landed, sectors Alpha, Beta and Gamma. Thermal imaging shows 50 organisms in the area, including both your fish and sharks. Furthermore, you find out that sector Alpha contains 4 fish tanks and 2 sharks and that sector Beta has 2 fish tanks and 4 sharks. About sector Gamma nothing is known.

You check the shipping notes but all you learn is that each tank had the same number of fish inside. The cargo hold had space for anywhere from 1 tot 13 tanks. Finally, the old captain tells you that this area has the odd property that no two sectors can have the same number of sharks, but every sector has at least one and no more than seven sharks.
Now here is your challenge;
How many fish tanks do you need to find in sector Gamma to rescue all the fish?

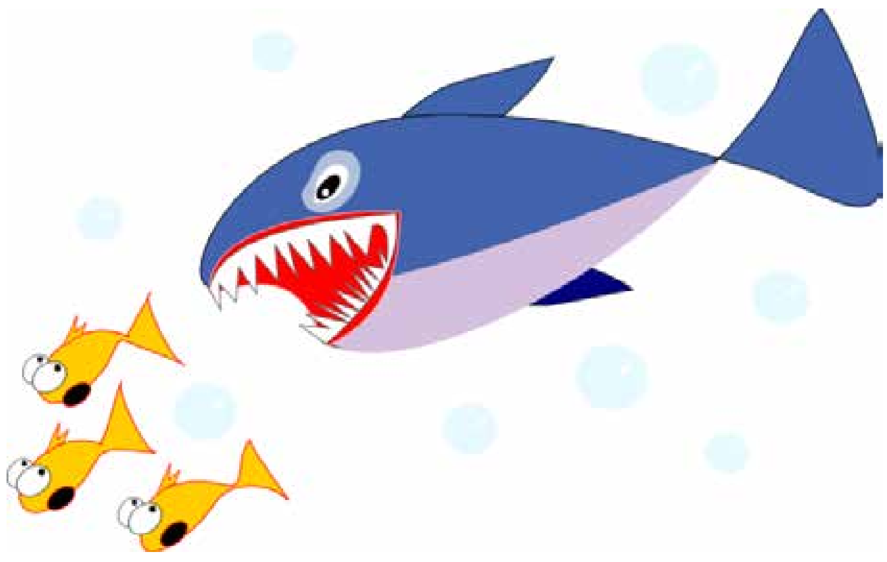
Problem 2 Find the answers of the following two infinite square roots:

$$1: 10\sqrt{10\sqrt{10\sqrt{10\sqrt{\dots}}}}$$

$$2: 10\sqrt[n]{10\sqrt[n]{10\sqrt[n]{10\sqrt[n]{\dots}}}}$$

Answer Problem 1 N2
Divide the 8 batteries in 3 groups; group A of 3 batteries, group B of 3 batteries and group C of 2 batteries. Test the three possible combinations of group A. If all the three combinations fail, then you know this group has either one good battery, or none at all. Do the same with group B. If one of the combinations in group A works, you found the two working batteries in 3 tries or less. If not, but one of the combinations in group B works, you found the two working ones in 6 tries or less. In the worst case scenario, both the combinations in group A and group B fail. Then you will know that group C contains 2 working batteries and you have found them in 7 tries.

Answer Problem 2 N2
14.5



Source: https://www.wpclipart.com/cartoon/animals/fish/toothy_fish/shark_chasing_fish.png.html

DSEA - Esports in Delft

Floris Doolaard, Secretary of Delft Student E-sports Association

Esports, also known as competitive gaming, has been on the rise for decades. Over the last few years it has seen exponential growth with the arrival of Twitch.tv. Esports involves playing, watching and enjoying your game. This is what DSEA is all about.

We make new, compete with, and learn from friends.

DSEA

Delft Student E-sports Association (DSEA) has been founded in 2015 and is the first student electronic sports association in The Netherlands. It was the idea of three Applied Physics students to build a community for esports in Delft. The first DSEA event was a humble gathering of gamers in the Sport & Culture cafe, now known as cafe X, where we watched LCS on the big screen and played games on our laptops. Now, four years later, we have become a dazzling community with more than 100 proud members.

Weekly LAN gathering

Being a DSEA member is about the freedom to game in any way you like. For instance, you may wish to join our weekly LAN party at cafe X. On this evening, starting 18:30 and ending around 00:30, we drain all of the available energy from the TU Delft campus into our battle stations.

We play any kind of game you can imagine including the most popular ones: League of Legends, CS:GO, Super Smash Bros. (Melee & Ultimate), Overwatch, Hearthstone, StarCraft II, Apex Legends and many more! If you want to find people playing your game, you can simply head over to our Discord channel.

Alongside gaming we also play offline board games during our LAN evenings, in combination with some drinks, to improve our bantering skills! To make the evening even more interesting and help you meet other members, we host a weekly tournament with a random party game.

Competing in esports

Someone has to be the best. Those who choose to vie for this position will find a home in our competitive section. We build and support teams by providing them with regular coaching and practice hours, but also with competitions and tournaments for them to play at. We love seeing those blue DSEA flames make quick work of their opposition. The same structure is in place for one-on-one minded esports athletes and is tailored to their specific games and needs.

To learn from the very best in the world we host viewing parties where we watch Fnatic almost win a League of Legends world finals in South-Korea, Faze goup or down in a CS:GO major, and follow Leffen taking down the gods of melee one by one.

In real life

The most beautiful thing about DSEA is that we are a bunch of people that all love the same thing: gaming! That is why we like to host social 'IRL' events such as trips to Gamescom, volunteering for Riot Games' LEC finals at Ahoy and also more regular events such as bowling, karting, paintball, and you name it.

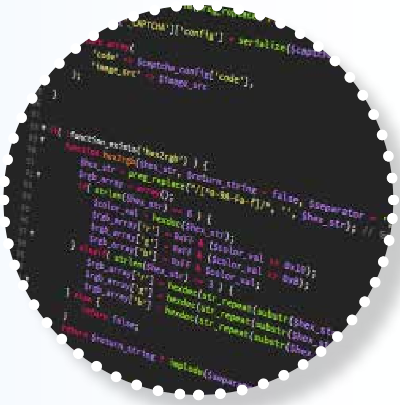
If you are in any way interested in (board)gaming, you owe it to yourself to check us out.
Game On!

Esportsdelft.nl
info@esportsdelft.nl

Calendar

HackDelft

HackDelft is the third hackathon of the HackDelft Committee. Join 150 other students interested in Mathematics, Computer Science or design for 24 hours of learning, building, and having fun. We're excited to work with our premium partners to provide HackDelft-exclusive tools and data for you to hack with this year!



Freshmen Algorithm Programming Contest

This programming contest is specially for freshmen. Join other freshmen in this afternoon of programming and you may win something!

Studytrip to Manchester

This year, the iCom will organize a studytrip from 11 - 16 June to Manchester. 38 students and two teachers from EEMCS will visit several companies, bring a visit to the University and have a look at the cultural aspects of the city and country.

May

1	Freshmen Algorithm Programming Contest
2	CH vs VvTP by the MaPhyA
3	Pubquiz for Master students by ComMa
7	In-Depth Lecture by the VerdiepCie
8	Pub Crawl by the AkCie
11-12	HackDelft - “It’s Hackening”
14	T.U.E.S.Day Lecture: Technolution
14	Yearbook Presentation Drink
25-26	Weekend activity by the WiFi
27	Members Lunch by the MeisCie

June

3	Teacher of the Year award ceremony
4	T.U.E.S.Day lecture with drinks by Shell
5	BBQ by the SjaarCie
6	Career College 4.2
11-16	Studytrip to Manchester by iCom
14-16	EIWEIW
18	T.U.E.S.Day Lecture CS & AM
20	Committee thank you activity