DELFT UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering, Mathematics and Computer Science

Resit TI2306 Algorithm Design

April 23, 2018, 18.30-21.30

- Usage of the book, notes or a calculator during this test is not allowed.
- This exam contains 6 open questions (worth a total 43 points) contributing 6.5/10 to your grade, and 15 multiple choice questions contributing 3.5/10 towards your grade. Note that the final mark for this course also consists for $\frac{1}{3}$ of the unrounded score for the lab course (if both ≥ 5.0).
- The open questions require about two hours of your time:
  - Please answer in clear English or Dutch and write legibly (first use pencil or scrap paper).
  - Do not give any irrelevant information. This might lead to subtraction of points. The indication of the number of lines is appropriate for typed lines (extra space caused by your handwriting and use of white space will not be held against you).
  - If an algorithm is asked, please provide the most efficient algorithm. Providing a suboptimal algorithm can lead to a subtraction of points.
  - If asked for pseudocode, you may call algorithms from the book unless specifically asked for the respective pseudocode.
  - Before handing in your solutions, please verify that your name and student number are on every page. Provide the course code and the number of pages (at least) on the first page.
- Regarding the multiple choice questions:
  - Each question has only one correct answer.
  - All questions count equally towards the grade. In computing the grade we will correct for the effects of random guessing. If you do not provide an answer, this is treated as a wrong answer.
  - Write your answers on scrap paper before copying them to the answer form.
  - Fill in your student number using digits as well as using the blocks.
  - Preferably use a dark pen or pencil for filling in the form and limit the number of corrections.
  - Try to spend no more than 1 hour on the multiple choice questions.
- The exam material consists of Chapters 4–7 from Algorithm Design by Kleinberg and Tardos (except sections 4.9*, 5.6, 6.10*, 7.4*, en 7.13*), as well as the notes on the Master Method and Proving techniques (guide), also applied to simple graph algorithms (from Chapter 3).
- Total number of pages (without this preamble): 7.

Open questions

1. Your friend has become co-owner and chef of a large restaurant. During the first months she is the only chef in the restaurant kitchen. To keep up with the growing number of customers, she needs your advice on how to efficiently use the n workstations in the kitchen. Each workstation can be used to prepare one type of dish. However, with some extra time for cleaning and getting the right utensils, a workstation can be setup for preparing a different dish.

   Suppose that a sequence S of m dishes is given that need to be completed in order, can you determine the minimum number of times a workstation needs to be setup for a different dish?

   (a) (6 points) Give pseudocode of a greedy algorithm that determines the minimum number of times the n workstations need to change to prepare the sequence of orders. Your algorithm should return this minimal number of changes.

   Briefly explain your pseudocode.

   (b) (2 points) Give a tight upper bound on the run time of your algorithm and explain.

2. (6 points) Dr. N. Flix wants to watch all of the Oscar nominated movies in one long session. To this end each movie needs to be "obtained" in some way [1] after which the doctor can watch the movie. Obtaining movie with name m_i takes d_i time and watching a movie takes w_i time for all $1 \leq i \leq n$ movies. Obtaining the movies can be done in parallel (the doctor has unlimited Internet bandwith, but the sources of the movies may not). Dr. Flix uses the following algorithm to determine the minimal length of the whole session:

   function OPTIMALORDERING($d_1, \ldots, d_n, w_1, \ldots, w_n, m_1, \ldots, m_n$)
   Sort $d_1, \ldots, d_n$ ascendingly and relabel d, w, m accordingly
   $t \leftarrow 0$
   while $i \leq n$ **do**
   $t \leftarrow \max(t, d_i) + w_i$
   print m_i
   end while
   return t
   end function

   Prove that the order of movies returned by this algorithm has an endtime no later than the optimal end time (in at most 15 lines). Use an exchange argument for this proof.

3. (6 points) Give an asymptotic tight upper bound for $T(n)$ in the following recurrence relations using the master method. Indicate the steps you use to get to your solution (in 3 lines each).[2]

   (a) $T(n) = 4T(n/3) + n$

   (b) $T(n) = 4T(n/2) + n^2$

4. (6 points) Mark is handing out flyers for a political party at a railway station. The railway station has two entrances, A and B. Mark has obtained (perfectly) predicted numbers of people coming through each of the entrances for each slot of 15 minutes that he is at the station, numbered from 1 to T. Let us denote these numbers by a_t and b_t for $1 \leq t \leq T$, for entrance A and B respectively. Mark obviously wants to maximize the number of people that he can offer his flyer to, but he cannot be in both spots at the same time, so he needs to decide where he is going to be. However, moving his stack of flyers from one entrance to the other takes about 15 minutes, during which he has his hands full and cannot hand out any flyers.

   (a) (5 points) Give a recursive function in *math notation* that defines the *maximum number of people* that can be offered a flyer during this visit of length T (times 15 minutes). Also indicate with what function call the solution to the complete problem can be found. Explain your answer and notation briefly.

   (b) (2 points) Give pseudocode of an iterative dynamic programming algorithm for this problem.

   (c) (1 point) Give a tight upper bound on the runtime of your algorithm and briefly explain your answer.

[1] A system by the name of "Tribler" may or may not be involved.
[2] It is okay to have a $\log$ in your answer.

5. Having made millions with the 0.1 bitcoin you recieved for your birthday[3] you now want to invest this in a sustainable way. To help you out a nice salesman by the name of Frank Sahwit has provided you with a portfolio of n projects to invest in. Obviously each project has some investment cost c_i and will result in some profit b_i after its completion. Furthermore some of these projects depend on each other, but you need to select what projects to invest in right now. For instance in order to invest in a space shuttle, you first need to invest in a Tesla car. You have sufficient money to invest in all projects if you want to.

   The question you now ask yourself is: how can you grow your fortune most? I.e. what projects should you invest in?

   (a) (4 points) Model the problem as a network flow problem. Clearly indicate what nodes your model has and what edges are connecting them. Also indicate clearly the weights of the edges. Also provide a drawing of an example with at least 4 projects and 2 dependencies.

   (b) (5 points) Explain how the maximum flow can be used to derive the maximum profit and prove that if the flow is maximal, the profit is also maximal.

[3] see Algorithm Design exam 2017-2018

Multiple-choice Questions

1. Given are a list of tasks and some known dependencies between these tasks expressing that some tasks cannot start before others have been completed. (These dependencies do not form a cycle.) Which algorithm provides a feasible order of executing these tasks with the least worst-case runtime?

   A. An algorithm using topological sort.

   B. An algorithm using divide and conquer.

   C. An algorithm using a network flow model.

   D. An algorithm using dynamic programming.

2. Suppose you have n jobs that all require processing. For each job we are given the start and finishing time. We need to determine the minimum number of machines required to run all jobs. We are given the following algorithm.

   ```
   1: Sort all jobs into a sequence J
   2: m ← 0
   3: for Job j_i in J in this order do
   4:     if j_i can be scheduled on one of the m allocated machines k then
   5:         schedule it on k
   6:     else
   7:         m ← m + 1
   8:         schedule it on m
   9:     end if
   10: end for
   11: return m
   ```

   In which order should the jobs be sorted in line 1 to obtain the minimum number of machines m?

   A. earliest deadline first

   B. earliest start time first

   C. earliest finishing time first

   D. shortest processing time first

3. Consider the following claims about algorithms that can be used to find a minimum spanning tree. Which of the following claims is **true**?

   A. Prim's algorithm is similar to applying Dijkstra without a target vertex.

   B. Kruskal's algorithm can be terminated prematurely to get a suboptimal MST.

   C. Kruskal's algorithm has better worst-case performance than Prim's for graphs in which $m > n$.

   D. The Reverse-Delete algorithm starts by sorting the nodes descendingly on the number of outgoing edges.

4. Consider the following algorithm for determining the k-clustering of a graph $G = (V, E)$ with maximal spacing that uses the Minimum Spanning Tree of the Graph.

```
function K-CLUSTERING(G, k)
    E' ← MST(G)                                   ▷ E' ⊆ E contains all edges of the MST
    Sort E' by decreasing cost and label them e_1 through e_{n-1}.   ▷ n = |V|
    c ← 1
    while ... do
        remove e_c from E'
        c ← c + 1
    end while
    return E'
end function
```

Which of the following claims is **true**?

A. This algorithm is correct if we use $c \leq k$ as the condition of the while-loop.

B. This algorithm is correct if we use $c < k$ as the condition of the while-loop.

C. This algorithm is incorrect, it should sort the edges by *increasing* cost instead.

D. This algorithm is incorrect, it is not possible to create a k-clustering of maximal spacing from an MST.

5. The "merge" sub-routine of the mergesort algorithm combines the results of two recursive calls of length $\frac{1}{2}n$ into one. What is a tight upper bound on the runtime of this "merge" sub-routine (excluding the runtime for the recursive calls)?

A. $O(1)$

B. $O(\log n)$

C. $O(n)$

D. $O(n \log n)$

6. Consider the algorithm that finds the closest pair of points in a given set of points S. Which of the following claims is **true**?

A. This algorithm would benefit from memoization.

B. This algorithm does not work if all points have the same x coordinate.

C. This algorithm does not work if all points have the same y coordinate.

D. The optimal recurrence relation for this algorithm is: $T(n) = 2T(n/2) + O(n)$.

7. Suppose you have a set J with n jobs, and for each job $1 \leq j \leq n$ we are given the start time s_j, finishing time f_j, and value w_j. The value of a set of jobs is simply the sum of values of jobs in the set. We aim to find the maximum value of a set of non-overlapping jobs. Please indicate which recursive function returns this maximum value for $W(n)$, setting $W(0) = 0$ and using the predecessor function $p(i) = \arg\max_k \{f_k \leq s_i\}$.

A. Sort J on increasing start time and $W(i) = \max_{1 \leq j \leq p(i)} \{W(j) + w_i\}$

B. Sort J on decreasing start time and $W(i) = \max_{1 \leq j \leq p(i)} \{W(j-1) + w_j\}$

C. Sort J on increasing finish time and $W(i) = \max\{W(p(i)), W(i-1) + w_i\}$

D. Sort J on decreasing finish time and $W(i) = \max\{W(i-1), W(p(i)) + w_i\}$

8. We have a set of n items, with for each item $1 \leq i \leq n$ a value v_i and a weight w_i. The following recursive function expresses the maximum total value of a subset of items from $\{1, 2, \ldots, i\}$ within a weight limit w.

$$\mathsf{Opt}(i, w) = \begin{cases} 0 & \text{if } i \leq 0 \\ \max\{\mathsf{Opt}(i-1, w), \mathsf{Opt}(i-1, w-w_i) + v_i\} & \text{if } w_i \leq w \\ \mathsf{Opt}(i-1, w) & \text{otherwise} \end{cases}$$

Please consider running an efficient iterative dynamic programming implementation of this recursive function on an input consisting of n items and a weight limit of W.

1. Is this a polynomial algorithm, and

2. what is a tight upper bound on the worst-case runtime

of this implementation for this input?

A. No, its runtime is $O(2^n)$.

B. No, its runtime is $\Theta(Wn)$.

C. Yes, its runtime is $\Theta(n^2)$.

D. Yes, its runtime is $\Theta(nW)$.

9. Given strings $X = x_1, x_2, \ldots, x_m$ and $Y = y_1, y_2, \ldots, y_n$, we say an alignment M is a set of non-crossing pairs (i, j) such that $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. The cost of an alignment is the sum of:

   1. a penalty $\delta \geq 0$ for each i and j that do not occur in M, and
   2. a penalty $\alpha_{i,j} \geq 0$ for every pair $(i, j) \in M$ where x_i is different from character y_j.

   The goal is to find an alignment with minimum costs.

   We are given a function $\mathsf{OPT}(X, Y)$ that returns an array of length $m + 1$ with for every $0 \leq i \leq m$ the minimal alignment costs of the (prefix) substring of x of length i, i.e., $x_1, x_2, \ldots, x_i$ and $y_1, y_2, \ldots, y_n$ in linear space and runtime $\Theta(i \cdot n)$, without giving the alignment itself.

   Consider the following pseudocode for an algorithm that uses linear space to print the minimum cost alignment of X and Y.

```
1:  function ALIGN-D&C(m, n, x_1, x_2, ..., x_m, y_1, y_2, ..., y_n)
2:      if m ≤ 2 or n ≤ 2 then
3:          Try all possible combinations.
4:      else
5:          h = ½n
6:          f = OPT(x_1, x_2, ..., x_m, y_1, y_2, ..., y_h)
7:          g = OPT(x_m, x_{m-1}, ..., x_1, y_n, y_{n-1}, ..., y_{h+1})
8:          Determine i
9:          Align-D&C(i − 1, h − 1, x_1, x_2, ..., x_{i-1}, y_1, y_2, ..., y_{h-1})
10:         Print (i, h)
11:         Align-D&C(m − i, h, x_{i+1}, x_{i+2}, ..., x_m, y_{h+1}, y_{h+2}, ..., y_n)
12:     end if
13: end function
```

   How should i be determined in line 8? Select i with $0 \leq i \leq m$ which minimizes...

   A. $f[i] + g[m - i]$

   B. $f[i] - g[m - i]$

   C. $\min\{\alpha_{i,h} + \mathsf{OPT}(m - i, h - 1),\ \delta + \mathsf{OPT}(m - i, h),\ \delta + \mathsf{OPT}(i - 1, h - 1)\}$

   D. $\min\{\alpha_{i,h} + \mathsf{OPT}(i - 1, h - 1),\ \delta + \mathsf{OPT}(i - 1, h),\ \delta + \mathsf{OPT}(i, h - 1)\}$

10. Consider the following recursive function that Prof. X.I. Ting wants to implement using dynamic programming.

$$\mathsf{Opt}(n,m) = \begin{cases} 0 & \text{if } n = 0 \\ n & \text{if } m = 0 \\ \min(\mathsf{Opt}(n-1,m-1), c_n + \mathsf{Opt}(n-1,m) - 10, \mathsf{Opt}(n,m-1) + d_m - 20) & \text{else} \end{cases}$$

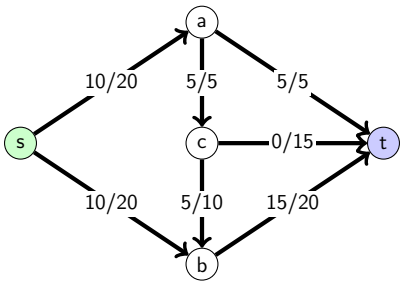
Prof P. Pol has now asked for a proof of correctness for whatever it is this algorithm is computing. Which of the following should **not** be part of a proof by induction.

A. A base case which considers the case where $n = m = 0$.

B. A base case which considers the case where $n > 0$ and $m = 0$.

C. An induction step which proves that $Opt(s,t)$ provides the correct output, given that $Opt(s',t')$ provides the right output for all pairs $s', t' \geq 0$ such that $s' + t' > s + t$.

D. An induction step which proves that $Opt(s,t)$ provides the correct output, given that $Opt(s',t')$ provides the right output for all pairs $s', t' \geq 0$ such that $s' + t' < s + t$.



Figuur 1: The graph used in MC question 11

11. Consider the graph depicted in Figure 1. The numbers on an edge represent the flow and capacity respectively. I.e. if an edge e has "5/10" as its annotation, then this means $f(e) = 5$ and $c(e) = 10$. By how much can the flow maximally be increased?

A. 5

B. 10

C. 15

D. 20

12. The flow value lemma states that:
    A. The value of a flow f is maximal iff there is no augmenting path.
    B. The value of a flow f is at most equal to the capacity of any s-t cut (A, B).
    C. The value of the max flow is equal to the capacity of the minimum s-t cut (A, B).
    D. The value of a flow f is equal to the difference in flow entering and exiting any s-t cut (A, B).

13. Given is an s-t flow network G and any s-t cut (A, B). Can the value of the flow through the network be more than the capacity of this cut (A, B)?
    A. Yes, this follows directly from the definition of weak duality.
    B. Yes, the maximum flow could be more by taking another route.
    C. No, this would imply that the maximum-flow can be higher than the minimum cut, which contradicts the max-flow min-cut theorem.
    D. No, the value of the flow is never more than the flow from A to B, and this is limited by the capacity of edges (u, v) with $u \in A$ and $v \in B$.

14. Consider the following problem:

    There are a number of wireless sensor nodes that all need to communicate their data to one central base station. Since all nodes have a limited transmission range, they rely on each other to forward the messages. We want to know the minimum required number of interrupted connections (selected by an adversary) before a certain special node x can no longer contact the base station.

    Which of the network flow problems can this most easily be mapped to?
    A. The maximum bipartite matching problem.
    B. The maximum edge-disjoint paths problem.
    C. The image segmentation problem.
    D. The project selection problem.

15. Consider the (known) image segmentation problem: for each pixel i we are given a likelihood of it belonging to the foreground (a_i) as well as to the background (b_i). Additionally, for each pair of neighboring pixels i and j (i.e., $(i, j) \in E$), we are given a penalty p_{ij} if one of them is considered to belong to the foreground F and the other to the background G. The question is to find a partition (F, G) of the pixels that maximizes the quality $q(F, G) = \sum_{i \in F} a_i + \sum_{j \in G} b_j - \sum_{i \in F, j \in G, (i,j) \in E} p_{ij}$.

    Consider the following statements about a network flow model for this problem where each pixel is represented by a pixel node.

    1. Every pixel node is connected to both source and sink.
    2. Every pixel node is connected to its neighbors with directed edges in both directions.
    3. There are always some pixel nodes connected by edges with infinite capacity.

    Which of the above three statements is *not* true?
    A. Statement 1 is not true.
    B. Statement 2 is not true.
    C. Statement 3 is not true.
    D. All of the three statements are true.

End of exam questions