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

CSE2310 Algorithm Design
Digital test – part 2 of 2

January 24, 2022

- Usage of the book, notes or a calculator during this test is not allowed.
- This digital test contains 3 questions (worth a total of 10 points) contributing 10/20 to your grade for the digital test. The other digital test contributes the other 10/20 to your grade. Note that the final mark for this course also consists for $\frac{2}{5}$ of the unrounded score for the written exam (if both ≥ 5.0).
- Please answer any questions in clear English.
- Weblab contains public tests that give an example input/output for these exercises.
- If you have questions about the exam, please use the WebLab discussions to ask them.
- The spec tests in WebLab **do not necessarily correspond with your grade**; your implementations will be manually graded.
- There is an API specification of the Java Platform available at `https://weblab.tudelft.nl/docs/java/17`.
- Please provide the **most efficient** implementation in terms of asymptotic time complexity. Providing a suboptimal implementation can lead to a subtraction of points.
- The material for this tests consists of module 3 and 4 of the course and the corresponding book chapters and lectures.
- Total number of pages (without this preamble): 1.

1. (2 points) You have previously studied the sequence alignment problem, which can be solved using the following recurrence:

$$OPT(i,j) = \begin{cases} i \cdot \delta & \text{if } j = 0 \\ j \cdot \delta & \text{if } i = 0 \\ \min\left(\alpha_{x_i y_j} + OPT(i-1, j-1), \delta + OPT(i-1, j), \delta + OPT(i, j-1)\right) & \text{else} \end{cases}$$

   For this exercise you are given sequences a and b as well as their lengths n and m. You are also given a δ and may assume that $\alpha_{x_i y_j}$ is `alphaDiff` when $x_i \neq y_j$, or 0 otherwise. Finally you are given the memory filled using the equations above. Thus `mem[i][j]` contains exactly the value for $OPT(i,j)$.

   You are tasked with returning the set of matched indices. Remember that a match is only made when the alpha value is chosen in the minimisation expression. To this end you should use the `Match` class from the solution template.

2. *You may ignore this paragraph of context if you so choose.* The Coati Queen Donna and Owl King Marty have finally managed to do "what they were also going to have to do from the very beginning. Sit down and talk!"[1] They have now decided to resolve their differences using a game. Winner takes all. Donna already has the optimal strategy figured out of course, but Marty needs your help to play optimally too.

   The game is a simple card game that features a deck of $n \geq 2$ cards. The game has both a single-player and two-player variant. In the single-player version your goal is to select a subset of cards from the deck so that the sum of their values is maximised, but you are not allowed to pick two cards that immediately follow each other in the deck.

   In the two-player variant, players take turns. In your turn you can chose to either take the top card of the deck, or the bottom card. Your goal is to *maximise* your score assuming your opponent does all in their power to *minimise* your score.

   For example, if we have a deck of 4 cards: 3, 5, 18, 7 then in the single player variant we can get to a score of at most 18+3 = 21. In the two-player variant, the starting player takes 3 (because if they take 7, the opponent would take 18 and win the game!), then the other player is forced to take either 5 or 7, meaning the starting player can take 18 and win the game. See the tests in WebLab for more examples. The tests added by the spec tests only test both methods in one, so use the visible tests to debug individual methods.

   For both versions of the game, implement **an iterative Dynamic Programming solution**.

   (a) (1½ points) Compute the maximum obtainable score in the single player version of the game.

   (b) (2 points) Given optimal play from both players, what is the maximum score the starting player can obtain? Hint: Although other solutions may exist, part of our solution is presented below in a recursive form (remember that your implementation should be iterative). i and j represent the start and end index of the (sub)sequence of cards we are considering. All the ... contain (possibly recursive) expressions of their own.

$$OPT(i,j) = \begin{cases} v_i & \text{if } i = j \\ \dots & \text{if } i + 1 = j \\ \max(\cdots + \min(\dots,\dots), \cdots + \min(\dots,\dots)) & \text{else} \end{cases}$$

   The final answer can be found by calling $OPT(1, \dots)$.

3. (4½ points) With libraries being closed all over the country thanks to Covid, the AD4LIFE (Active Demonstrators for Literacy In Friends & Enemies) foundation has decided to use this period to redistribute their $n \geq 1$ books over the $m \geq 1$ libraries in the country. Each book has exactly one author, and $b_i \geq 1$ copies available (for $1 \leq i \leq n$). Furthermore each library has a capacity for $c_j \geq 1$ books (for $1 \leq j \leq m$). AD4LIFE is looking to redistribute the books in such a way that every library has at least 3 different books from every author and no more than 1 copy of a single book.

   Implement the method `canWeRedistribute` to inform AD4LIFE about the (im)possibility of this redistribution process. Note that your implementation is graded based on the number of constraints from the text it can model correctly. Thus a solution that handles all but one correctly and ignores one is likely worth more than one that incorrectly tries to model all.

   Note the classes and methods related to network flow problems you have used before are available for you to use in WebLab. See the exercise in WebLab for a description of the different methods available.

[1] Well done if you got that reference.

End of exam questions