COMP3600/6466 — Algorithms

Convenor & lecturer: Hanna Kurniawati
E-mail: comp_3600_6466@anu.edu.au

Assignment 3
Due: Monday, 26 October 2020 23:59 Canberra Time
Grace Period Ends: Tuesday, 27 October 2020 13:00 Canberra Time
Late Penalty: 100%

Notes:

- In this assignment, you need to submit: .
  - A written/typed answer to the questions as a single .pdf file, named A3-studentID.pdf.
  - Source codes as a single file, named A3-studentID.cpp file (or suitable extension for the programming language you use). In this assignment, you can use C/C++/Java. However, support will be provided only for C/C++. Moreover, please pay attention to the following:
    - To ensure we can run your program with no problem, you need to use the template provided during tutorials and test your program on the provided test cases (the template and test cases are available in the class website).
    - We will compile and run your program from command prompt in Linux using the command:
      ```
      > g++ -std=c++11 A3-studentID.cpp -o A3-studentID
      > ./A3-studentID input1.txt > output1.txt
      ```
  - You need to compress all files (the .pdf and .cpp) into a single .zip file named A3-studentID.zip and save this .zip file as draft in Wattle before the due date .
- We provide 13 hours grace period. This means, there will be no penalty if you submit OR save as draft before the grace period ends. However, we will NOT accept assignment after the grace period ends.
- Assignment marking:
  - The total mark you can get in this assignment is 100 points.
  - Whenever explanation/argument/reasoning/derivation is requested, the explanation/derivation is worth 80% of the points you can get for the question.
  - This assignment will contribute 25% to your overall mark.
- Discussion with your colleagues are allowed and encouraged. However, you still need to work on the assignment on your own AND write the names you discussed this assignment with.
- You are allowed to use materials from textbooks, other books, websites, etc., but you do need to provide the references to these materials.

- 13 Oct: Part A Q3, universal is supposed to be uniform.
- 18 Oct: Part C Q9e, In the required time limit, M is supposed to be B.

[25 pts] Part A

1. [5 pts] In a max-heap where all elements are distinct, where is the smallest element located? Please explain.
2. [5pts] Ms RB would like to transform a red-black tree into a black-only tree by setting the children of the red nodes to become the children of the parent of the red nodes, and then removing the red nodes. What can you say about the leaves' depth of the resulting tree? Please explain.
3. [10pts] Suppose you need to represent a set of data whose keys are: 1, 2, 3, 4, 5, 6, 7, 8, 9 as an AVL tree.
   (a) [5pts] Draw an instantiation of the shallowest AVL tree that you can construct for the above data set and explain why it is indeed the shallowest AVL tree possible for the data set.
   (b) [5pts] Draw an instantiation of the deepest AVL tree that you can construct for the above data set and explain why it is indeed the deepest AVL tree possible for the data set.

4. [5 pts] Mr H argues that for a hash table with chaining and simple ~~universal~~ uniform hashing, the expected time complexity to search for an item can be made to be $O(\frac{1}{n})$. He suggested that this time complexity is achievable simply by setting the number of slots in the hash table to be n^2 (n is the number of elements in the table) because this setting will lead to a load factor $\alpha = \frac{1}{n}$. Is Mr H correct? Please explain why or why not.

[35 pts] Part B

5. [5 pts] Mr B said that he found a comparison-based method to construct a Binary Search Tree in $O(n)$, where n is the number of elements in the data set. Could this be true? Please provide a proof or counter-example to support your argument.

6. [10 pts] Consider a hash table T with m slots. Suppose T contains a single element, and this element has key k. Ms Search has been searching for r keys that are different from k in hash table T. Assuming T uses simple uniform hashing and chaining, what is the probability that at least one of the r searches collide with the m slot containing key k? Please provide the derivation.

7. [10 pts] Recall the matrix chain multiplication problem we discussed in class (week-8 Tuesday, based on [CLRS] 15.2). Mr SP says that we can compute the optimal solution to this problem faster, simply by recursively splitting the (sub-)chain of matrices $A_s \cdot A_{s+1} \cdots A_e$ at A_k, where $1 \leq s \leq e$, the size of A_i for $i \in [s, e]$ is $p_{i-1} \times p_i$, and k is selected to minimise the value $p_{i-1} \times p_k \times p_j$ (i.e., $k = \arg\min_{i \in [s,e-1]} p_i$). To make it clearer, below is the pseudo-code of the algorithm Mr SP proposed. The output of the pseudo-code is a string of the chain of matrices together with their parenthesis.

Algorithm 1 SP-MatrixChainMult(A chain of matrices $A_s, A_{s+1}, \cdots, A_e$)

```
1: if s == e then
2:     Return "A_s"
3: Let k = arg min_{i∈[s,e−1]} p_i
4: Return "(SP-MatrixChainMult(A_s, ··· A_k)) · (SP-MatrixChainMult(A_{k+1}, ··· A_e))"
```

Is Mr SP's algorithm correct? Please explain your answer by either providing a proof or a counter-example.

8. [10 pts] As a program manager at a large company, Mr PM is overseeing multiple projects. As a good program manager, he needs to foresee whether there will be delays in each of his projects. To this end, he asked your help to develop a Dynamic-Programming approach to compute an estimated longest duration to finish a project.

   To compute the above estimate, you can assume a project is divided into multiple tasks, starting from an initial task and ending at a final task. The dependencies between tasks can be represented as a weighted directed graph, called the task graph. The vertices of a task graph represent the tasks, an edge from a vertex v to another vertex v' in a task graph means the task represented by v must be completed first before the task represented by v' can start, and the weight associated with with vertex $\overline{vv'}$ indicates the time duration to complete the task represented by v. One can then compute the estimated longest duration to finish a project by finding the longest path from the vertex representing the initial task to the vertex representing the final task in the task graph.

   In this question, you need to provide the sub-problems and recursive definition of the optimal value function (i.e., step-1 and step-2 of Dynamic Programming development steps, as discussed in class in week8 Tuesday lecture based on [CLRS] 15.Intro and 15.3) to find the longest path in the task graph.

[40 pts] Part C

9. Ms C has been preparing to start her own restaurant —a TCA Burger franchise— in her city, Arrebnac. She has finalised all the necessary franchising agreement and licensing for the restaurant, as well as signed the lease and finished refurbishing her TCA Burger shop in a strategic location at the Arrebnac City Centre. Now, she needs to start hiring her staff.

   Ms C plans to fill n positions, and hires exactly one person for each position. She budgeted a total of at most $B \times \$10,000$ per year to pay the salaries of the people she will hire. In this problem, we use the term salary to include everything an employee will receive (ie, including superannuation and other benefits). For each

position-i ($i \in [1, n]$), Ms C receives k_i applicants. Each applicant stated the salary they requested. Let's denote $s_{ij} \times \$1,000$ as the yearly salary that applicant-j for position-i requested. You can assume Ms C pays the requested salary without negotiation, and that higher salary equals better performance. Ms C would like to hire her staff, such that the total yearly salary she will need to pay is the highest possible not exceeding her budget.

To simplify the problem, you can assume B and s_{ij} for all $i \in [1, n]$, $j \in [1, k_i]$ are all positive integers, with $10 \leq B \leq 300$ and $10 \leq s_{ij} \leq 100$. Furthermore, the number of positions and applicants are limited too, with $1 \leq n \leq 25$ and $1 \leq k_i \leq 150$ for $i \in [1, n]$.

Your task is to help Ms C makes the hiring decision by developing a dynamic-programming algorithm to find the applicants to hire, so that each position is filled by exactly one person, and the total salary is the highest possible not exceeding $B \times \$10,000$. To this end, please answer the following questions.

(a) [10 pts] Please provide step-1 and step-2 of the dynamic programming development steps (the steps are as discussed in class in week8 Tuesday lecture based on [CLRS] 15.Intro and 15.3).

(b) [4 pts] Please show that for the sub-problems you have defined in Q9, the sub-problem graph is a directed acyclic graph.

(c) [6 pts] Please provide the pseudo-code for top-down and bottom-up dynamic programming to help Ms C.

(d) [5] Please derive the asymptotic time complexity of your bottom-up dynamic programming solution.

(e) [10 pts] Please implement the bottom-up dynamic programming algorithm you have developed above.

Input to the Program. The program will accept a single argument, which is the name of the input file. The input is a text file containing $n + 1$ lines, where n is the number of positions to be filled. The first line consists of two numbers, separated by a white space. The first number is B and the second number is n. Each line-$(i + 1)$ in the next n lines consists of $k_i + 1$ numbers, separated by a white space. The first number represents the number applicants for position-i, while the next k_i numbers represent the requested salaries s_{ij} of each applicant, indexed from 1 to k_i.

Example:
17 2
3 100 50 70
2 95 75

The above input means that Ms C's budget is $\$170,000$ and two positions are to be filled. For the first position, 3 applicants apply, with the requested yearly salary for the first, second, and third applicants being $\$100,000$, $\$50,000$, and $\$70,000$ respectively (i.e., $s_{1,1} = 100$, $s_{1,2} = 50$, $s_{1,3} = 70$). For the second position, there's only two applicants, with their yearly requested salary being $\$95,000$ and $\$75,000$.

Output to the Program. The program should output to standard output. If the problem has one or more solution, the program should output a single line containing of $n + 1$ numbers separated by a white space. The first number is the total salary cost of the people selected to be hired. The next n numbers are the indices of the selected applicant, starting from the selected applicant for the first position, then the second, etc.. If the same total salary cost can be achieved by selecting more than one set of selected applicants, any set of them will be acceptable. If the problem has no solution, the program should output "no solution".

Output example for the example input scenario:
165000 3 1

Program Marking. If your program compiles and runs, you will get 2 points. We will then run your program on 4 test cases. For each test case, your dynamic programming algorithm (including data structure construction) will be given a total of $(0.0001 \times ~~M~~ B \times TotalNumberOfApplicants)$ ms CPU time to find a solution. You can assume your program will have access to at most 12GB RAM. It will be run as a single thread process on a computer with Intel i7 3.4GHz processor. For each test case that your program solves correctly within the given time and memory limit, you will get 2 points.
Note: The test cases for marking will be different from the example test cases.

(f) [5 pts] Please compare the empirical running time of your program and the asymptotic time complex-

ity you have derived in Q9.d. For this purpose, you need to create at least 5 scenarios with increasing parameter size, and provide a comparison between the theoretical and empirical time-complexity.