COMP3600/6466 — Algorithms

Convenor: Hanna Kurniawati (hanna.kurniawati@anu.edu.au)

Assignment 3

Due: Tuesday, 22 October 2019 23:59 GMT+10

Notes:

- This assignment consists of two parts. In the second part, you need to write computer programs. You should write your programs in C/C++/Java. No other programming language is accepted. We will support only C++.
- Please submit both the report and the source codes, following the format below, with [studentID] replaced by your student ID, e.g., U1234567.
  - Please write your report in a single .pdf file, named A3[studentID].pdf .
  - Please name the source code that contains your main function for Part B Q1 and Q3 as stated in those questions. If you use other language than C++, please use the suitable extension.
  - Please put all your source codes and test cases (for Part B Q4) in a single directory, named A3[studentID].
  - Please zip your report and the directory containing your source codes (please exclude the object files and executables) into a single file, named A3[studentID].zip .
  - Please submit the .zip file via Wattle. Please note that the maximum file size you can upload is 200MB.
  - Please **save your submission as draft**, so that you can re-upload it until the last minute. Once the grace period is over, the last saved draft in wattle will automatically be locked as your submission.
- You can submit a scanned copy of a handwritten report. However, the handwriting must be neat enough for the teaching staff to read them. If we can't read them, we will not mark them and therefore, you will get a 0 mark for the report component. Our preference is you type your report. This is a good time to learn how to use TeX/LaTeX.
- We provide 13 hours grace period. This means, there will be no penalty if you submit before Wednesday, 23 October 2019 13:00 Canberra time. However, we will NOT accept assignment submission beyond this time.
- Assignment marking:
  - The total mark you can get in this assignment is 100 points.
  - This assignment will contribute 20% to your overall mark.
  - This assignment is redeemable. However, we strongly suggest that you do this assignment.
- 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 books, slides, etc. as reference. You still need to write the solution yourself and put a reference to the source. A reference need to be a full citation and specific, e.g., T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. Introduction to Algorithms 3rd Ed. MIT Press. 2009. Sec. 2.2.

Clarifications:

- 6 October: Colored blue in:
  - Part A Q1: Clarify G does have at least 1 Minimum Spanning Tree.
  - Part A Q3b: Clarify that we are still referring to the representation of Huffman code stated in the paragraph that proceed the two questions.
  - Part B, description, par. 3: We fix a typo on price limit —the correct one is: "at least one item in each furniture type is a natural number with at most \$$M/T$". We also add clarifications of the bounds.
  - Part B Q3d: Time limit is changed to $\max(20, 0.1(M+1))$ms. We also add a note to remind that the bounds you use should be based on description of the problem and NOT based on extrapolation from test cases.

[40 points] Part A.

1. [10 pts] Let $G(V, E, w)$ be a weighted undirected graph, where V is the set of vertices, E is the set of edges, and $w : E \to \mathbb{R}^+$ is the weight of the edges ($\mathbb{R}^+$ is the set of real positive numbers). Suppose $\mathcal{T}(G)$ is the set of all

minimum spanning trees of G and is non-empty. If we know that the weight function w is a injection, i.e., no two edges in G have the same weight, then:

(a) [3pts] How many elements would $\mathcal{T}(G)$ has?

(b) [7pts] Please support your answer to the above question with a proof.

2. [20 pts] A string is a palindrome whenever it is symmetric, in the sense that it is identical when read from left to right and from right to left. For instance, the string "noon" is a palindrome. Now, it is known that any string can be made into a palindrome by adding extra characters. For instance, the string "abcabc" is not a palindrome. But, by adding 3 characters, it can be made into a palindrome: "abcabacba". However, we cannot construct a palindrome by adding less than 3 characters to the string "abcabc". Your task is to design an algorithm that takes a string S as its input and finds the smallest number of extra characters that need to be added to S in order to convert S into a palindrome. The algorithm should have the running-time of $O(n^2)$, where n is the length of the input string. In particular, please:

   (a) [3 pts] Write the pseudo-code of your algorithm.

   (b) [7 pts] Explain why your algorithm is correct.

   (c) [5 pts] Derive the time-complexity of your algorithm, to show that your algorithm takes $O(n^2)$ time.

   (d) [5 pts] Derive the memory-complexity of your algorithm.

   Note: A string in this context is a sequence of any character, including white space, which means that in our context, the string "my gym" is not a palindrome but "mygym" is a palindrome, for instance.

3. [10 pts] Huffman code is a variable-length encoding of characters that utilises the relative frequency of the corresponding character.

   Encoding of a character is a mapping that assigns a unique binary string to each character. If we use fixed-length encoding, then we will need $\lceil \log(n) \rceil$ bits to encode each character in a set of n characters. When our data contains certain characters much more than others, fixed-length encoding is inefficient. For instance, suppose we have a file made up of a total of 1,000 occurrences of 5 different characters, $\{a, b, c, d, e\}$, and the appearance frequency of each character is $\{a : 500; b : 400; c : 50; d : 30; e : 20\}$. Then, fixed-length encoding will require $3,000$ bits to code the file. But, the following variable-length encoding (in fact, the Huffman code) $a = 1, b = 01, c = 001, d = 0000, e = 0001$ would require only $1,650$ bits to code the entire file, which is only 55% of the required storage for fixed-length encoding.

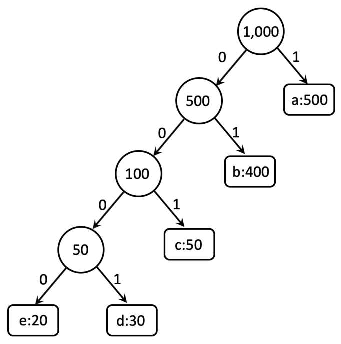

Figure 1: An illustration of a Huffman tree for the example provided, i.e., a data with the following 5 characters and frequency $\{a : 500; b : 400; c : 50; d : 30; e : 20\}$. The Huffman code is then $a = 1, b = 01, c = 001, d = 0000, e = 0001$.

   The Huffman code is represented as a full binary tree, where each encoded character is located at a leaf node and each edge of the Huffman tree is augmented with a binary number. This tree is called the Huffman tree. The Huffman code that encodes a character C is then the binary string formed by concatenating the binary numbers along the path from the root to the leaf node that represents C. Figure 1 illustrates this tree.
   Please answer the following questions:

   (a) [5 pts] What would be the characteristic of the Huffman tree when the efficiency of Huffman code is no better than fixed-length encoding? Please explain your answer

   (b) [5 pts] Mr T says that an AVL tree representation of the Huffman code would result in a more efficient encoding than a full binary tree, where efficient means less bits are required to store the same file. Is Mr T correct? Please explain why or why not.

[60 points] Part B.

Congratulations!!! Your startup company has just gotten its first major customer: AFR (Australian Furnitures & Rental) Inc.. AFR is a furniture company that has just expanded its business to rental property management. The landlord will usually provide a certain amount of budget for AFR to purchase various types of necessary furnitures. Your company has been hired to develop a program that will allow AFR to purchase all the necessary types of furnitures, such that the total purchase price is the highest that is still within the customer's budget (yup, the highest...).

In particular, given a landlord's budget of \$$M$, T types of furnitures (e.g., sets of bed, dining tables, sets of dining chairs, sets of desks, sofa, etc.), and K_i with $1 \le i \le T$ different models for furniture type-i, the program should output the model (one model) for each type of furniture that AFR should purchase, so that the total price of the purchased furnitures is the highest possible not exceeding the client's budget. If there are more than one possible solutions, any one of the solutions would be acceptable.

You can assume K_i for $i \in [1, T]$, M and T are all natural numbers with possible values $K_i \in [1; 100]$, $M \in [1; 100, 000]$, $T \in [1; 100]$. Moreover, you can assume that $M \ge T$ and the price of at least one item in each furniture type is a natural number with at most M/T, which means a solution always exists. You can also assume that the price of each other item will not exceed \$1, 000, 000.

Upon hearing about the major contract from AFR, your company's VC and advisor got excited and suggested three approaches for solving this problem:

1. Complete Search.
2. Dynamic Programming.
3. Greedy, in the sense that the most expensive model within the budget is always chosen first.

Your tasks

1. [13 pts] Please design the algorithm using complete search and provide:
   (a) [5 pts] The pseudo code of the algorithm.
   (b) [5 pts] Derivation of the time complexity of your algorithm.
   (c) [3 pts] Implementation of your algorithm. Please named the program A3-complete-[studentID].cpp.
   Program Marking: If your program compiles and runs, you will get 1 point. We will then run your program on 2 test cases of size $M \in [1; 100]$, $T \in [1, 10]$, $K_i \in [1, 10]$. For each test case, your program will be given a total of $(M+1)$ms CPU time to find a solution. This time limit includes the time for reading the file, finding the solution, and printing the solution. The time limit will be rounded up to 2 decimal digit. 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 1 point.

2. [10 pts] The company's VC and advisor are not exactly correct. Only one of dynamic programming or greedy can be used to generate correct results to the aforementioned problem. The question is:
   (a) [3 pts] Which one is it?
   (b) [7 pts] Please show that the two requirements for your selected approach are satisfied.

3. [27 pts] For the approach that could generate correct results (i.e., answer to Question B2), please provide:
   (a) [10 pts] The pseudo code of the algorithm.
   (b) [5 pts] Correctness proof.
   (c) [5 pts] Derivation of the time complexity of your algorithm.
   (d) [7 pts] Implementation of your algorithm. Please named the program A3-approach2-[studentID].cpp.
   Program Marking: If your program compiles and runs, you will get 1 point. We will then run your program on 6 test cases: 2 cases would have $M \in [1; 1, 000]$, 2 cases would have $M \in [1, 001; 10, 000]$, and 2 cases would have $M \in [10, 001; 100, 000]$. For each test case, your program will be given a total of $\max(20, 0.1(M + 1))$ms CPU time to find a solution. This time limit includes the time for reading the file, finding the solution, and printing the solution. The time limit will be rounded up to 2 decimal digit. 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 1 point.

Note: We do provide example test cases but, it is not an exhaustive test cases. The limit in your program should be based on the description, rather than on extrapolation of problem properties from the test cases.

4. [10 pts] Empirical comparison of the time complexity of the algorithms you developed in Question B1 and B3. For this purpose, you will need to develop your own test cases. In this comparison, please also analyse if the results are in-line with the theoretical results of the two algorithms. Hint: You might want to develop the test cases with varying size, from very small problem size to very large. If your complete search implementation cannot solve the large cases, that's fine. You can still include them in your analysis.

Input to the Program

The program will accept a single argument, which is the name of the input file. The input file contains $T + 1$ lines, where T is the number of types of furnitures to be purchased.
The first line consists of two numbers, separated by a white space. The first number is M and the second number is T. Each line in the next T lines consists of $K_i + 1$ numbers, separated by a white space. The first number represents the number of models for furniture type-i, while the next K_i numbers represent the price of each model of the particular furniture type.

Example:
500 2
3 100 200 200
2 240 100

The above input means that the landlord's budget is \$500 and two types of furnitures are to be purchased. For the first type, 3 models are available, with the first, second, and third model priced at \$100, \$200, and \$200. For the second type, only two models are available, the first model costs \$240 and the second one costs \$100.

Output of the Program

The output is a line containing $T + 1$ numbers, separated by white spaces. The first number is the amount of money spent, while the j^{th} number for $2 \leq j \leq T + 1$ represents the index of the model purchased for furniture type-$(j - 1)$.

Example on the output of the above input example:
440 2 1