COMP 3711H – Honors Design and Analysis of Algorithms
2016 Fall Semester – Written Assignment # 2
Distributed: October 11, 2016– Due: October 25, 2016
Revised October 24, 2016

Your solutions should contain (i) your name, (ii) your student ID #, and (iii) your email address
Some Notes:

- Please write clearly and briefly.
- Please follow the guidelines on doing your own work and avoiding plagiarism given on the class home page.
  In particular ***don't forget to acknowledge individuals who assisted you, or sources where you found solutions.*** Failure to do so will be considered plagiarism.
- Please make a *copy* of your assignment before submitting it. If we can't find your answers, we will ask you to resubmit the copy.
- The default base for logarithms will be 2, i.e., $\log n$ will mean $\log_2 n$. If another base is intended, it will be explicitly stated, e.g., $\log_3 n$.
- Each problem is worth 20 points.
- As in the previous assignment, you must submit both a hardcopy and a PDF softcopy. The hardcopyshould be submitted to the COMP3711H assignment box and the softcopy via the CASS system. The PDF can be generated by Latex, from Word or a scan of a (legible) handwritten solution.

Problem 1: Smallest k numbers in sorted order

Given a set of n numbers, we wish to find a comparison-based algorithm that finds the k smallest numbers in the set and outputs them in sorted order.

The first three items below suggest possible algorithms for solving this problem. For each algorithm, analyze its worst case running time and state it in terms of $\Theta(f(n))$ notation using *both* n and k. Note that, because of the design of the problem $k \leq n$. Thus, as an example, a running time of $\Theta(n+k)$ could be written as $\Theta(n)$. On the other hand, a runnning time of $\Theta(n+k^2)$ could not be simplified any further.

For the fourth item you must design and analyze a better algorithm than the first three.

For all parts, if you use an algorithm given in class as a subroutine you only have to state what that algorithm is and its running time; you do NOT have to analyze the subroutine from scratch.

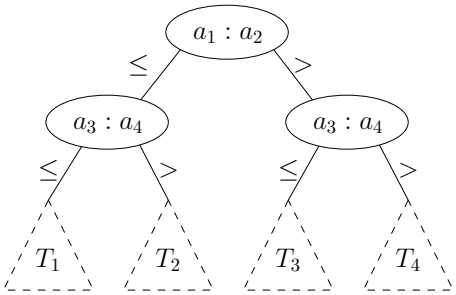
Note that all algorithms must be comparison-based and the running times must be *worst-case,* i.e., no randomized algorithms. We assume that all numbers are distinct.

1. Sort all n numbers, and output the k smallest numbers in sorted order.
2. Build a heap on the n numbers, and call Extract-Min k times.
   For this part, you may assume, as discussed in the tutorial, that you can *build* a heap on n items in $O(n)$ time.
3. Build a heap on the n numbers by repeatedly inserting them into an initially empty heap, and call Extract-Min k times.

Note: The original version of this assignment had a Part 4 which asked: "Can you design an algorithm better than all three above? Better *here means that its $\Theta(f(n))$ notation running time is never greater than any of the three algorithms above and is sometimes less than them." This part was removed in this revised version because this question is ill-defined"*

Problem 2: Decision tree

The figure below shows part of the decision tree for mergesort operating on a list of 4 numbers, a_1, a_2, a_3, a_4. Please expand subtree T_3, i.e., show all the internal (comparison) nodes and leaves in subtree T_3.



Problem 3: Greedy algorithm

Consider a long river, along which n houses are located. You can think of this river as an x-axis; the houses locations are given by their coordinates on this axis in a sorted order.

Your company wants to place cell phone base stations along the river, so that every house is within 4 kilometers of one of the base stations. Give an $O(n)$-time algorithm that minimizes the number of base stations used.

As well as giving the algorithm and showing that it runs in $O(n)$ time, you must prove that it yields an optimal solution.

Problem 4: Greedy algorithm

Give an $O(n \log n)$ algorithm for assigning location of n files on a magnetic tapes that minimizes the expected time necessary to read a file. As input you are given the size of the files and the probability that each file will be accessed.

Background: Files used to be stored on magnetic tapes rather than disks. Reading a file from tape is not like reading a file from a random access disk. On a tape, the files are laid out sequentially, one after another. To access a file on a tape we first must scan all the other files preceding the file, which takes a significant amount of time. The ordering of the files on the tape therefore defines how long it takes to read any particular file.

More explicitly, an ordering of the files will just be the ordering of how to store the files on the tape. For example, the ordering $(3, 1, 2, 4)$ means that file 1 will be placed as the 2^{nd} file, file 2 will be placed as the 3^{rd} file, file 3 will be placed as the 1^{st} file and file 4 will be placed as the 4^{th} file.

Suppose $L[i]$ is the size of file i. The time T_i it takes to read file i is the sum of the sizes of the files located before file i plus the size of file i. In the example above, if $L = [2, 6, 4, 8]$, then

$$\begin{aligned} T_3 &= L[3] = 4, \\ T_1 &= L[3] + L[1] = 4 + 2 = 6 \\ T_2 &= L[3] + L[1] + L[2] = 4 + 2 + 6 = 12 \\ T_4 &= L[3] + L[1] + L[2] + L[4] = 4 + 2 + 6 + 8 = 20 \end{aligned}$$

Suppose that you know that i will be accessed with probability p_i and the ordering in which the files are on the tape. The average time to read a file will be

$$\sum_i p_i T_i.$$

To continue the example above, if the probabilities were $p = [\frac{1}{6}, \frac{1}{12}, \frac{1}{4}, \frac{1}{2}]$ then the average read time would be

$$\frac{1}{6} \cdot 6 + \frac{1}{12} \cdot 12 + \frac{1}{4} \cdot 4 + \frac{1}{2} \cdot 20 = 13.$$

On the other hand, if we had the same probabilities but the files were in the order $(1, 3, 4, 2)$ then the reading times would be

$$\begin{aligned} T'_1 &= L[1] = 2 \\ T'_3 &= L[1] + L[3] = 2 + 4 = 6 \\ T'_4 &= L[1] + L[3] + L[4] = 2 + 4 + 8 = 14 \\ T'_2 &= L[1] + L[3] + L[4] + L[2] = 2 + 4 + 8 + 6 = 20 \end{aligned}$$

and the average read time would be

$$\frac{1}{6} \cdot 2 + \frac{1}{12} \cdot 20 + \frac{1}{4} \cdot 6 + \frac{1}{2} \cdot 14 = 10.5,$$

which is less than the time for the previous ordering.

The problem:

Input is an array of $L[1..n]$ of the n files sizes and a list $p_1, p_2, \ldots, p_n$ of the probability of accesing each file.

Give an $O(n \log n)$ algorithm that outputs an optimal ordering of the files on the tape, i.e., an ordering that minimizes the expected time to read a file. You must describe the algorithm, explain why it runs in $O(n \log n)$ time and prove that it gives the correct answer.

Hint: If all of the probabilities were the same, sorting the files and placing them on the tape in smallest to largest order would give an optimal solution. This does not work for our more general problem. Can you come up with a different ordering that would give an optimal solution?

Problem 5: Huffman Coding *Note: There was a typo in the first version of this assignment with the Fibonacci numbers being written as $F_n = \{1, 1, 3, 5, \ldots\}$. That has been corrected in this version*

(a) What is a Huffman code for the set of weights $1, 2, 3, 4, 5, 6, 7, 8$?

(b) What is a Huffman code for the set of frequencies $1, 1, 2, 3, 5, 8, 13$?

(c) Consider the family of frequency sets $F_n = \{1, 1, 2, 3, 5, \ldots, F_{n-1}\}$ where F_n is the n'th Fibonacci number defined by $F_0 = F_1 = 1$ and $\forall i > 1$, $F_i = F_{i-1} + F_{i-2}$.
Give a general structure for a Huffman code for the frequency set F_n. Prove that this structure is correct and give the cost of this Huffman code as a function of n (the cost of the code is the weighted external path length of the associated tree).
The cost should be written in the form $C(n) = \Theta(f(n))$ where you have to explictly state the function $f(n)$ in closed form. You can use the known fact that $F_n = \frac{1}{\sqrt{5}}\phi^n + O(1)$ where $\phi_n = \frac{1+\sqrt{5}}{2}$.
Note: The original version of this question asked something more complicated. More specifically it stated "The cost should be written in the form $C(n) = f(n) + O(n^2)$ where you have to explictly state the function $f(n)$ in closed form. You can use the known fact that $F_n = \frac{1}{\sqrt{5}}\phi^n + O(1)$ where $\phi_n = \frac{1+\sqrt{5}}{2}$.". This was replaced with the simplified version above. If you answer the more complicated original version you will get extra credit.