Homework 3

ECE 345 Algorithms and Data Structures
Fall Semester, 2020

Due: Friday, October 30, 5PM

- All page numbers are from 2009 edition of Cormen, Leiserson, Rivest and Stein.
- For each algorithm you asked to design you should give a detailed *description* of the idea, proof of algorithm correctness, termination, analysis of time and space complexity. If not, your answer will be incomplete and you will miss credit. You are allowed to refer to pages in the textbook.
- Do not write C code! When asked to describe an algorithm give analytical pseudocode.
- **Read the handout with the instructions on how to submit your Homework online. Failure to adhere to those instructions may disqualify you and you may receive a mark of zero on your HW.**
- Write *clearly*, if we cannot understand what you write you may not get credit for the question. Be as formal as possible in your answers. Don't forget to include your name(s) and student number(s) on the front page!
- No Junk Clause: For any question, if you don't know the answer, you may write "I DON'T KNOW" in order to receive 20% of the marks.

---

1. **[Search Trees, 15 points]**

   Suppose we have a list of n fixed length strings that are sorted lexicographically. This list is stored in a balanced binary search tree for easy access. Let k be the number of strings with prefix x. Devise an algorithm to output all strings with prefix x in $O(k + \lg(n))$ time. *Hint:* First, find the first node in the tree with prefix x.

   Example: A sorted list with strings of length three may look like: *ABC, ABD, BCC, BDC, CAB, CAD, DAB.* If x"CA", then your algorithm should print out CAB, CAD.

2. **[Search Trees, 15 points]**

   A full binary tree is a binary tree where all non-leaf nodes have two children, and all leaf nodes are at the same height. Answer the following questions about full binary trees.

   (a) Give a *recurrence* for the number of nodes in a binary tree of height n. (note that a tree of height 1 represents a single node).

   (b) Use your solution from the previous question to argue that there are more leaf nodes than non-leaf nodes in a full binary tree.

3. **[Hashing, 10 points]**

   Demonstrate the insertion of keys 1, 22, 54, 13, 12, 3, and 27 (in that order) into a doubly-hashed table where collisions are resolved with open addressing. Let the table have 11 slots, let the *primary* hash function be $h_p(k) = k \bmod 11$, and let the *secondary* hash function be $h_s(k) = (3k \bmod 4 + 1)$.

4. **[Greedy Algorithms, 15 points]**

   You have n younger siblings who are each playing in a soccer tournament. Every sibling wants you to watch them play at least once, but you have other things to do, so you want to go to the tournament as few times as possible while still seeing all of your siblings play. Their schedule looks like $S = [(s_1, f_1), (s_2, f_2), ..., (s_n, f_n)]$ where s_i and f_i are times where sibling i will start and finish their game respectively and $s_i \leq f_i$. The instant you go to the tournament you see all players who are playing, and leave immediately (*i.e.* ignore time spent at the tournament, you go for only an instant in time). So if you go to the tournament at time t you will see sibling i play if $s_i \leq t \leq f_i$

   (a) Devise a greedy algorithm to compute an optimal solution to the problem.

   (b) After making your greedy choice, show that the problem is reduced to a smaller subproblem.

   (c) Prove the greedy choice property by showing that there is an optimal solution that agrees with the first greedy choice your algorithm makes.

   (d) Prove the problem's optimal substructure by showing that the optimal solution to the subproblem (described in (b)) combined with the greedy choice leads to an optimal solution.

5. **[Dynamic Programming, 20 points]**

   You want to go for a long, multiday hike and you need to decide how many stops to make and where you will rest overnight. More precisely, you are hiking between points x_1 and x_n and through locations $x_2, x_3, \ldots, x_{n-1}$. There are campsites where you must rest at x_1 and x_n, but the problem is to decide whether to camp at x_i, $i = 2, 3, \ldots, n-1$. If you camp at site x_i there is an associated cost b_i (you have to rent the site, set up the tent, pay for food, etc.), and if you hike between x_i and x_j—without stopping inbetween—there is an associated cost cost $c_{i,j}$ (from the wear and tear on your body). The total cost is the sum of the camping costs and the corresponding hiking costs. The goal is to decide the optimal location of rests so that the total cost for the trip is minimized. Suggest a general algorithm that you could use to plan your trip (without any assumptions about the costs). The algorithm must return the locations of rest, not just the total cost of the trip. As usual describe it clearly (pseudcode is not mandatory), argue that it is correct and analyze its running time. Try to make it as efficient as you can.