Homework 3

ECE 345 Algorithms and Data Structures
Winter Semester, 2015

Due: March 3rd at 12noon in ECE345 dropbox

- All page numbers are from 2009 3rd edition of Cormen, Leiserson, Rivest and Stein (CLRS).
- 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.
- Staple your homework properly. Use a stapler; do not use glue or other weird material to put it together. If you are missing pages, we are not responsible for it but you are!
- 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 !

---

1. **[Search Trees, 20 points]** Given the root r of a binary tree and two nodes a, b in the tree. Devise an algorithm that returns the common ancestor of a and b that is furthest from r. The optimal algorithm should run in $O(n)$ time and $O(1)$ memory. Partial credit will be received for less efficient algorithms. *Hint: think about the case when a, b have the same distance from the root.*

2. **[Search Trees, 15 points]** Given the root node of the tree, implement an algorithm to verify whether the tree is a binary search tree. Analyze the algorithm run-time.

3. **[Hashing, 15 points]** Demonstrate the insertion of keys 5, 38, 41, 22, 9, 55, 11, 21, 10, 44, 17, 31 into a doubly-hashed table where collisions are resolved with open addressing. Let the table have 12 slots, the *primary* hash function be $h_p(key) = key \bmod 12$ and the *secondary* hash function to be $h_s(key) = (key * 3) \bmod 4$. If collision cannot be resolved by open addressing, use chaining. Show the content of your hash table including the linked lists if needed. Show your work.

4. **[Greedy Algorithms, 15 points]** You are consulting a trucking company that does a large amount of business shipping packages between New York and Boston. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed limit W on the maximum amount of weight they are allowed to carry. Boxes arrive at the New York station one by one, and each package i has a weight $w_i \leq W$. The trucking station is quite small, so at most one truck can be at the station at any time. *Company policy requires that boxes are shipped in the order they arrive*; otherwise, a customer might get upset upon seeing a box that arrived after his, make it to Boston faster. At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive, and whenever the next box does not fit, they send the truck on its way.

   But they wonder if they might be using too many trucks, and they want your opinion on whether the situation can be improved. Here is how they are thinking. Maybe one could decrease the number of trucks needed by sometimes sending off a truck that was less full, and in this way allow the next few trucks to be better packed.

Prove that, for a given set of boxes with specified weights, the greedy algorithm currently in use actually minimizes the number of trucks that are needed.

Hint: Let b_i represent the ith box. Show using induction on k that for any other method of picking boxes, if $b_1, \cdots, b_i$ fits within the first k trucks using other method, and your greed algorithm fits $b_1, \cdots, b_j$ in the first k trucks then $i \leq j$.

5. [**Dynamic Programming, 20 points**] Given a string s and a dictionary of words, derive an efficient algorithm that determines if s can be segmented into a space separated sequence of one or more dictionary words. For example, given $s =$ "peanutbutter", dictionary contains ["pea" , "nut", "peanut", "butter"]. Your algorithm should return true because "peanutbutter" can be segmented as "peanut butter" or "pea nut butter". Assume that a search in the dictionary takes $O(1)$ time. Analyze your algorithm run-time and memory usage.

6. [**Dynamic Programming, 20 points**] Suppose you're consulting for a company that manufactures PC equipment and ships it to distributors all over the country. For each of the next n weeks, they have a projected *supply* s_i of equipment (measured in pounds), which has to be shipped by an air freight carrier. Each week's supply can be carried by one of two air freight companies, A or B.

   Company A charges a fixed rate r per pound (so it costs $r \cdot s_i$ to ship a week's supply s_i).

   Company B makes contracts for a fixed amount c per week, independent of the weight. However, contracts with company B must be made in blocks of four consecutive weeks at a time.

   A *schedule*, for the PC company, is a choice of air freight company (A or B) for each of the n weeks, with the restriction that company B, whenever it is chosen, must be chosen for blocks of four contiguous weeks at a time. The cost of a schedule is the total amount pair to company A and B, according to the description above.

   Give a polynomial-time algorithm that takes a sequence of supply values $s_1, s_2, ..., s_n$ and returns a schedule of minimum cost.

   For example, if $r = 1, c = 10$ and the sequence of values is $11, 9, 9, 12, 12, 12, 12, 9, 9, 11$, then the optimal schedule would be A,A,A,B,B,B,B,A,A,A.

   Hint: Let $OPT(i)$ denote the minimum cost of a solution for weeks 1 through i.