Homework 3

Due: Fri, Sep 19, 11.59 PT

1. Suppose you want to drive from USC to Santa Monica. Your gas tank, when full, holds enough gas to drive p miles. Suppose there are n gas stations along the route at distances $d_1 \leq d_2 \leq \cdots \leq d_n$ from USC. Assume that the distance between any neighboring gas stations, and the distance between USC and the first gas station, as well as the distance between the last gas station and Santa Monica, are all at most p miles. Assume you start from USC with the tank full. Your goal is to make as few gas stops as possible along the way. Design an efficient algorithm for determining the minimum number of gas stations you must stop at to drive from USC to Santa Monica. Prove that your algorithm is correct. Analyze the time complexity of your algorithm. (25 points)

2. Suppose you are given two sequences, A and B, of n positive integers each. You are allowed to permute the numbers in A and B (rearrange the numbers within each sequence, but not swap numbers between sequences). After permuting, let a_i be the i-th number in A, and let b_i be the i-th number in B. Given this arrangement, you receive a score of $\prod_{i=1}^{n} a_i^{b_i}$

   Design an efficient algorithm for permuting the numbers so as to maximize the resulting score. Prove its correctness and analyze its running time. (25 points).

3. You have n heavy iron chains of weights $W_1, W_2, \ldots, W_n$ respectively. You want to connect them together and form one long chain. You decide that everyday you will connect two existing chains into one (so that you have one less chain than the previous day), and repeat this for n-1 days. To connect two chains of weights W and W' into one chain of weight $W + W'$, you need to first carry the chains to the wielding station and then bring back the bigger chain to the pile of chains - all this heavy lifting makes you burn 100 * ($W + W'$) calories. You want to finish the whole chain-connecting task (two at a time as described above) while exerting the least effort, i.e., burning the fewest calories in total. Design an efficient algorithm to decide the order in which to connect all the chains and determine the minimum calories burnt as a result. Analyze the time complexity of your algorithm. You do not need to prove that your algorithm is correct. For full credit, your algorithm should run asymptotically faster than $\Theta(n^2)$. (20 points)

Example: Suppose you have 3 chains with weights 2, 5, 6. Combining 5 and 2 first leads to 700 calories burnt, leaving us with chains of weight 7 and 6. Combining these two leads to 1300 more calories, 2000 in total. This turns out to be the lowest possible among all possible orders.

Ungraded Problems:

1. You plan to have an awesome long weekend. Suppose that there are n activities with where each activity has degree of enjoyment $e_1, e_2, \dots, e_n$ and it takes you t_k time to complete the k-th activity. If you finish an activity you will receive its full degree of enjoyment, and if you give up on an activity without finishing it you will still get enjoyment from it proportional to the time you spent on it. Assuming you have T time during your long weekend find a way to maximize your enjoyment. (10 points)
   (a) Design a greedy algorithm to decide on the optimal set of activities to complete. (4 points)
   (b) Prove the correctness of your algorithm. (4 points)
   (c) Justify the time complexity of your algorithm (2 points)

2. Given a positive integer n we would like to find non-negative integers x, y, z, w such that $n = 15x + 10y + 5z + w$ and $x + y + z + w$ is minimum possible. (10 points)

   (a) Show that in an optimal solution where $x + y + z + w$ is minimum possible, we must have $w < 5$, $z < 2$, and $y < 3$. (4 points)

   (b) Give an efficient algorithm that finds an optimal solution, and justify its running time (i.e., the number of arithmetic operations (additions, multiplications, and divisions)). (6 points)

3. There are N tasks that need to be completed by 2 computers A and B. Each task "i" has 2 parts that take time: a_i (first part) and b_i (second part) to be completed. The first part must be completed before starting the second part. Computer A does the first part of all the tasks while computer B does the second part of all the tasks. Computer A can only do one task at a time, while computer B can do any amount of tasks at the same time. Find an O(nlogn) algorithm that minimizes the time to complete all the tasks, and give a proof of why the solution is optimal. (15 points)

4. The United States Commission of Southern California Universities (USCSCU) is researching the impact of class rank on student performance. For this research, they want to find a list of students ordered by GPA containing every student in California. However, each school only has an ordered list of its own students by GPA and the commission needs an algorithm to combine all the lists. Find an efficient algorithm for yielding the combined **sorted** list and give its runtime in terms of the total number of students (m) and the number of colleges (n). For full credit, your algorithm must be asymptotically faster than the brute-force algorithm that will sort all m students. (15 points)

5. The array A below holds a max-heap. What will be the order of elements in array A after a new entry with value 19 is inserted into this heap? Show all your work.
   A = {26, 14, 10, 8, 11, 9, 3, 2, 4, 1} (10 points)