Assignment 3 solutions

You have **five problems**, marked out of a total of 100 marks.
NOTE: Your solutions must be typed, machine readable .pdf files. **All submissions will be checked for plagiarism!**

1. After the success of your latest research project in mythical DNA, you have gained the attention of a most diabolical creature: Medusa. Medusa has snakes instead of hair. Each of her snakes' DNA is represented by an uppercase string of letters. Each letter is one of S, N, A, K or E. Your extensive research shows that a snake's venom level depends on its DNA. A snake has venom level x if its DNA:

   - has exactly $5x$ letters
   - begins with x copies of the letter S
   - then has x copies of the letter N
   - then has x copies of the letter A
   - then has x copies of the letter K
   - ends with x copies of the letter E.

   For example, a snake with venom level 1 has DNA SNAKE, while a snake that has venom level 3 has DNA SSSNNNAAAKKKEEE. If a snake's DNA does not fit the format described above, it has a venom level of 0. Medusa would like your help making her snakes venomous, by deleting zero or more letters from their DNA. Given a snake's DNA, can you work out the maximum venom level this snake could have?Your algorithm should run in time $O(n \log n)$

   Hint: Combine binary search with greedy.

   Solution: Observe that if you can make a snake with venom level x after some deletions, then you can make snakes with any venom level from 1 to x, by deleting some more letters. This implies that there is some cut-off point k where it is possible to make every snake with venom 1 to k, but it is impossible to make snakes with venom larger than k. Therefore, k will be the maximum venom level. Let $m = \min\{n(S), n(N), n(A), n(K), n(E)\}$, where $n(X)$ is the number of occurrences of letter X, $X \in \{S, N, A, K, E\}$. We can find the maximum venom level through a binary search over venom levels between 0 and m. For a given venom level x, it is trivial to check whether we can make a snake with that venom level by looping over the string, and finding x earliest occurrences of S, followed by x occurrences of N, and so on. If it is possible to make a snake with venom level x, then we know that $k \geq x$. Otherwise, we know that $k < x$. Thus, we would first try to see if it is possible to make a venom level m and if this fails, then level $\lfloor m/2 \rfloor$ and so forth. The complexity of this is $O(n \log n)$.

2. Rock. Paper. Scissors. The rules are simple. The game is contested by two people over N rounds. During each round, you and your opponent simultaneously throw either Rock, Paper or Scissors. Rock beats Scissors, Scissors beats Paper, and Paper beats Rock. If your throw beats your opponent's, you gain one point. Conversely, if their throw beats yours, you lose one point. Your opponent is very predictable. You know that they will throw Rock in the first R_a rounds, throw Paper in the next P_a rounds, then finally throw Scissors in the last S_a rounds, where $R_a + P_a + S_a = N$. You have to throw Rock in R_b rounds, Paper in P_b rounds, and Scissors in S_b rounds, where $R_b + P_b + S_b = N$. However, as you are an experienced player, you may throw these in any order you like. At the beginning of the game, you start with 0 points. How should you play to maximise the number of points you can finish with?

   Solution: Try to win as many games as possible first, then tie as many games as possible after that, and finally admit defeat on the rest. Thus, you will reserve $mRP = \min(R_a, P_b)$ Paper throws for the first mRP Rock throws of your opponent. This leaves you with $P_B = P_b - mRP$ Paper throws and leaves your opponent with $R_A = R_a - mRP$ Rock throws. Similarly, you reserve $mPS = \min(P_a, S_b)$ throws of Scissors for the first mPS throws of Paper of your opponent. This leaves you with $S_B = S_b - mPS$ throws of Scissors and leaves your opponent with $P_A = P_a - mPS$ throws of Paper. Finally, you reserve $mSR = \min(S_a, R_b)$ throws of Rock for as many throws of Scissors of your opponent. In this way you maximised the number of wins. Next, you reserve $\min(P_A, P_B)$ throws of Paper for (some of) the remaining throws of Paper by your opponent, $\min(R_A, R_B)$ throws of Rock for the remaining throws of Rock of your opponent, and similarly for your remaining throws of Scissors. This will maximise your ties. The leftover of your throws will produce losses, but minimal number of them.

3. You are given a time schedule of arrivals a_i and departures d_i of n trains, so $1 \leq i \leq n$, during each 24 hour period (note: a train can arrive before the midnight and leave after midnight; each train arrives and departs at the same time every day). You need to find the minimum number of platforms so that each train can stay at a platform without interfering with other arrivals and departures.

   Solution: Sort the arrivals and departures into one big list by increasing times. First, we count the number of trains which arrive before or at the midnight and leave after the midnight and set a counter to this starting value. We now go through the list of arrival and departure times; an arrival increments the counter, and a departure decrements the counter. The number of platforms you need will be the maximum value the counter reaches during the algorithm runtime. This is $O(n \log n)$

4. You are given a set of n jobs where each job i has a deadline $d_i \geq 1$ and profit $p_i > 0$. Only one job can be scheduled at a time. Each job takes 1 unit of time to complete. We earn the profit if and only if the job is completed by its deadline. The task is to find the subset of jobs that maximises profit.

   Solution: This is an example of an important logistic principle, called "Just in Time". Sort all jobs in a decreasing order of profits. You now go through the list, scheduling each job in the slot which empty and as close to the deadline of that job as possible. Thus, you are scheduling each job as late as possible. Optimality is easy to prove: assume a schedule is optimal but not greedy; look at the first job which was scheduled before the greedy choice. It is easy to see that you can swap it with the job which is placed in the slot which should be chosen by the greedy choice.

5. You have to produce and deliver N chemicals. You need to deliver W_i kilograms of chemical C_i. Production of each chemical takes one day, and your factory can produce only one chemical at any time. However, all of the chemicals evaporate at a rate of p percent a day, so you need to produce more than what you need to deliver. Schedule the production of the chemicals so that the total extra weight of all chemicals needed to produce to compensate for the evaporation loss is as small as possible.

   Hint: First determine how much of a chemical is left after k many days and then compute how much of it has been lost, and then apply greedy.

   Solution: Sort all chemicals in order of increasing weight, and on each day produce the next chemical in that order, only producing just enough so that there remains exactly W_i at the end of the N days.

   To prove this works, let's say we start off with S_i kg of a chemical C_i, produced k_i days before delivery. Then after k_i days, we have $S_i(1-p)^{k_i} = W_i$ kg left, so $S_i = W_i(\frac{1}{1-p})^{k_i} = W_i q^{k_i}$, where $q = \frac{1}{1-p}$ is constant. The total amount lost is $\sum_i W_i q^{k_i} - W_i$. As the sum of W_i is constant, we simply need to minimise $\sum_i W_i q^{k_i}$.

   Consider two chemicals C_i and C_j, produced days k_i and k_j away from the delivery date respectively. Observe that if $k_i > k_j$

$$\begin{aligned}
&W_i q^{k_i} + W_j q^{k_j} \leq W_i q^{k_j} + W_j q^{k_i} \\
&\iff W_i q^{k_i} - W_i q^{k_j} \leq W_j q^{k_i} - W_j q^{k_j} \\
&\iff W_i (q^{k_i} - q^{k_j}) \leq W_j (q^{k_i} - q^{k_j}) \\
&\iff W_i \leq W_j, \quad \text{as } q^{k_i} - q^{k_j} > 0
\end{aligned}$$

Therefore given two days from the delivery date k_i and k_j, and two chemicals, it is always optimal to assign the one with the smaller W_i to the larger date. In other words, it is always optimal to produce the chemical with the smaller W_i first.