CMPT 307 (2013)

Hints to some problems in Chapter 4 of the text.

4.4: In this problem we are given a sequence $S = \{s_1, s_2, ..., s_n\}$ and $S' = \{s'_1, s'_2, ..., s'_m\}$. We need to determine whether S' is a subsequence of S.

We can solve the problem greedily. Find the first occurrence of s'_1 in S. Starting from this location in S, look for s'_2 in S. This process is repeated. This way the search process takes $O(n)$ time.

4.5: The greedy solution is: start at one end, say west, of the road, and place a base station that is exactly 4 miles away from a house h in the west for the first time. If we go any further east without placing a base station, we won't cover h. We delete all the houses covered by this base station, and iterate this process on the remaining houses.

4.6: Let s_i, b_i, r_i denote respectively the swimming, biking and running times of contestant i. We need to order the contestants such that the completion time of the entire race is minimized. One constraint is that only one contestant can be in the swimming pool at any given time.

The greedy strategy that can be shown to be optimal is to order the contestant in decreasing $b_i + r_i$. Use the exchange argument for proving the optimality.

4.9(a): The problem was discussed in the class.

4.10: (a) When the new edge is added to T, a cycle is created. We visit the edges of the cycle (in $O(|V|)$ time), and check if the costs of the edges are no greater than the added edge. If it is true, T is the MST of the graph that includes the added edge (using the cycle property). Otherwise, MST changes. We use the adjacency list of MST T.

(b) If MST changes, the new MST can be be obtained from old MST T by removing the heaviest edge of the cycle.

4.13: An optimal algorithm is to schedule the jobs in decreasing order of w_i/t_i. Optimality property can be proved using the exchange arguments.

4.15: The generic formulation of the problem: Given a set of intervals, $I_j = [a_j, b_j]$, on a line. The problem is to select a smallest set of intervals S such that each interval not in S is overlapped by some interval in S.

Initially, all the intervals are unmarked. Look at the unmarked interval that ends the earliest. Among the intervals that intersect, it selects the interval, say s, that ends last (i.e. covers the most). Add this interval to S. Mark all the intervals covered by s. Repeat the process till there is no unmarked intervals.

4.21: Get a BFS tree in $O(n)$ time that contains $n-1$ edges. Add the remaining edges one by one. Each addition will result in a cycle. Remove the heaviest edge which can't be in any MST, by the cycle property. We need to do this nine times. Each time it takes $O(n)$ time.

4.22: It is false. Consider a graph with 4 nodes, v_1, v_2, v_3, v_4. Edges (v_1, v_2), $(v_2, v_3), (v_3, v_4), (v_4, v_1)$ have cost 2 each, and the edge (v_1, v_3) has cost 1. Every edge belongs to some MST. However the spanning tree $\{(v_1, v_2), (v_2, v_3), (v_3, v_4)\}$ is not a MST.

4.29: This was discussed in the class. A detailed description is given here.

If any of the degrees is 0, this should be an isolated node in the graph; so we can just delete that degree from the list.

Let us now sort the list so that $d_1 \geq d_2 \geq ... \geq d_n > 0$. Let $d_1 = k$. Now consider the list $L = \{d_2 - 1, ..., d_{k+1} - 1, d_{k+2}, ..., d_n\}$.

Claim: the graph we want exists iff there is a graph whose degrees are the items of L.

Note that L has one less element than the original list. So we can proceed recursively to check if G satisfies the desired property.

We prove the two directions of the if and only if separately.

(If part) If a graph G whose degrees are the elements of L exists, we can add a node v_1 and connect it to the first d_1 nodes of G in the descending order of degrees.

(Only if part) Suppose there is no graph whose degrees are the elements of L, but at the same time, there is a graph G satisfying the degree sequence $d_1 \geq d_2 \geq ... \geq d_n > 0$. We show how we can transform G

into a graph where v_1 is joined to $v_2, ..., v_{d_1}$. If this property does not hold, there exist v_i and v_j, $i < j$, so that v_1 is joined to v_j, but v_1 is not joined to v_i. (why?) Since $d_i \geq d_j$, there exist a vertex v_k, not equal to v_i, v_j and v_1 with the property that (v_i, v_k) is an edge and (v_j, v_k) is not an edge. Now we replace the edges (v_i, v_k) and (v_1, v_j) by the edges (v_1, v_i) and (v_j, v_k). This keeps the degrees of $v_1, v_i and v_j$ the same. We repeat this process till G is converted to the one with the desired property.