CS483 - Practice Problems 3

Jana Košecká

1. Chapter 3, Problem 7
2. Chapter 3, Problem 12
3. Chapter 4, Problem 3
4. Chapter 4, Problem 4
5. Chapter 4, Problem 13

Solutions:

1. Chapter 3, Problem 7. The claim is true. Suppose by the way of contradiction that the graph is not connected. Then let S be the smallest connected component. Since there ar least two connected components we have $|S| \leq n/2$. Now consider any node $u \in S$. Its neighbours use lie all in S, so its degree can be at most $n/2 - 1 < n/2$. But that contradicts the assumption that every node has agree at lead $n/2$.

2. Chapter 3, Problem 12 Hint construct a graph where for each person you will have two node b_i and d_i representing the unknown birth and death dates respectively and the fact that birth precedes the death. At the beginning we include these edges for all nodes i. If person P_i died before P_j we include edge from d_i to b_j. If P_i and P_j life spans overlap partially we include edges (b_i, d_j) and (b_j, d_i).

   No if G has a cycle, this means that each event precedes another even and there is no event (node), which can be put first. If G has no cycle then we can topologically order the nodes and get a consistent ordering.

3. Chapter 4, Problem 3 We want to prove that greedy alg. uses as few trucks as possible, by showing that it stays ahead of any other solution. If greedy alg. fits boxes $b_1, \ldots b_j$ and the other solution fits boxes $b_1, \ldots b_i$ then $i \leq j$ into first k trucks. Now we will show by induction on k that the geed alg. always stays ahead. Case $k = 1$ is clear; greedy alg. fits as many boxes as possible in the first truck. Assume that the hypothesis holds for $k - 1$; the greedy lag fits j' boxes in $k - 1$ trucks and the other solutions fits i' boxes, where $i' \leq j'$. Now for the kth truck the alternate solution packs $b_{i'+1}, ..., b_i$ boxes. Since $j' \geq i'$ the greed alg. is able to fit at least all the boxes $b_{i'+1}, ..., b_i$ into kth truck and possible more. That concludes the induction step.

4. Chapter 4, Problem 4
   Suppose $S = s_1 \ldots s_n$ and $S' = s'_1 \ldots s'_m$ To find the subsequence in the same order,we design a greedy algorithm which will find the first event in S that is the same as s'_1, then finds the first event after that that is is same as s'_2 and so on. Keep track of matches founds so far $k_1, k_2, \ldots$.

   Initially $i = j = 1$
   while $i \leq n$ and $j \leq m$
   if $s_i = s'_j$ then
   $k_j = i$
   $i = i + 1$
   $j = j + 1$
   otherwise let $i = i + 1$
   end
   if $j = m + 1$ return subsequence $k_1, \ldots k_m$
   else return "S' i snot a subsequence of S".

   The running time of the algorithm is $O(n)$ as the while takes n steps and $O(1)$ time and each iteration increments i so there can be at most n iterations. There is straightforward to see that the lag. find the correct match if it finds anytime.

5. Chapter 4, Problem 13.
   An optimal algorithm is to schedule the jobs in decreasing order of w_i/t_i. We prove the optimality of this algorithm by an exchange argument.
   Consider another schedule, then the schedule must have an inversion - a pair of jobs i and j for which i comes before j in this alternate solution. For this pair $w_j/t_j \geq w_i/t_i$ by the definition of the greedy schedule. If we can show that swapping this pair does not increase the weighted sum of completion times, then we can iteratively do it until there are no more inversions, arriving at the greedy schedule without having increase the function we're trying to minimize. From that it will follow that the greedy

algorithm is optimal.
Consider swap i and j. The completion times of all other jobs remain the same. Suppose that the completion time of before i and j is C then get same the contribution of i and j to the total sum was $w_i(C + t_i) + w_j(C + t_i + t_j)$ where the after the swap sum is $w_j(C + t_j) + w_i(C + t_i + t_j)$. The difference between the values after the swap, compared to the value before the swap is (canceling terms in common between the two expressions $w_i t_j - w_i t_j$. Since $w_j/t_j \geq w_i/t_i$ this different is bounds above by 0 and so the total weighted sim of compassion times does n to increase due to the sam as desired.