MCS/CS 401, Spring 2020, Midterm 2

Wednesday, April 22, 11:00-12:30

Solutions

1. Omitted.

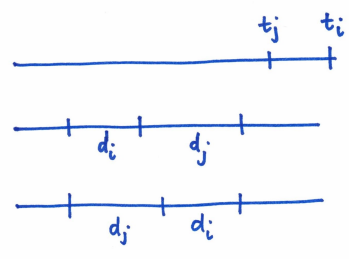
2. Let s_i be the length of the longest super-quadratic subsequence having a_i as its last element. Then it holds that

$$s_i = 1 + \max\{s_j \,:\, j < i,\ a_j^2 \le a_i)\}.$$

   If there is no such s_j then max $= 0$. This formula holds because a super-quadratic sequence ending with a_i can have any element a_j preceding a_i such that $a_j^2 \le a_i$, and the maximal length of any such sequence is $1+s_j$. Adding 1 accounts for a_i itself. The s values can be computed by two **for** loops of length at most n, and inside the loops $O(1)$ steps are done, so the complexity is $O(n^2)$.

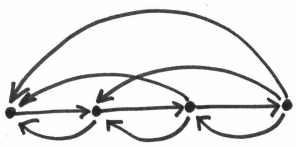
3. Algorithm: order the activities according to non-decreasing target times (using mergesort). If in this work plan every activity finishes in time then return it, otherwise return "no solution". The running time is $O(n \log n)$.

   For correctness we claim that if there is a solution to the problem then the ordering produced by the algorithm is a solution as well. Consider a solution with a different ordering. Then there are two activities i and j such that i comes before j and $t_j < t_i$.



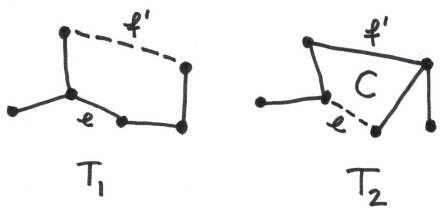
   Following the hint, let us exchange the two activities (shown on the picture above). Then j finishes earlier than before, so it is OK. Activity i finishes at the same time as j finished in the original work plan, so at most at time t_j, but as $t_j < t_i$, it is also OK. Finishing times for other activities remain the same, so those are OK as well. Thus the modified work plan is also a solution. Using this exchange operation several times (like in bubble-sort), we can transform the work plan into the one given by the algorithm, so that work plan is also a solution, as claimed.

4. Consider a directed path with n vertices with every back edge included. So the edges are $(i, i+1)$ and (i, j) for every $i > j$. If the edges in the path come first in the adjacency lists then DFS returns the ordering $1, \ldots, n$ with $\binom{n}{2} = \Theta(n^2)$ "bad" edges. In the reverse ordering only the $n-1$ path edges are "bad".



5. Following the hint, consider the cycle C formed by adding e to T_2. Every edge on the path $P = C - e$ has weight at most $w(f)$.

   Deleting e from T_1 two disjoint subtrees are formed. The endpoints of the path P are in different subtrees, so some edge f' on P connects the two subtrees. Thus $T_1 - e + f'$ is again a spanning tree. As $w(f') \leq w(f) < w(e)$, the weight of this tree is *less* than the weight of T_1, contradiction.



Note: it is allowed to use the textbook, your notes and the course website. It is forbidden to use any other resource, including any form of collaboration. Exams can be scanned or photographed, please double-check readability. Submission will be closed at 12:30 so please make sure to upload your exam in time. Emailed exams are not accepted.

Problems

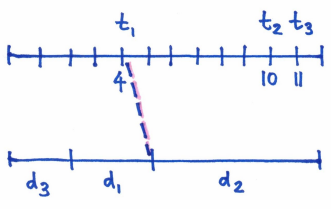
1. Find a longest common subsequence of the strings *abbaa* and *babab* using the algorithm learned in class. (No credit is given for finding an LCS without using the algorithm.)

2. Consider a sequence of positive numbers $a_1, \ldots, a_n$. A subsequence is *super-quadratic* if each number is at least the square of the previous one. Give a $O(n^2)$ algorithm to find the length of a longest super-quadratic subsequence.

   For example, in the sequence 3, 2, 6, 4, 5, 20 the subsequence 2, 4, 20 is a longest super-quadratic subsequence.

3. There are n activities, having durations $d_1, \ldots, d_n$. Each activity has a target time $t_1, \ldots, t_n$. A *work plan* is an ordering of the jobs. Give a $O(n \log n)$ greedy algorithm to determine whether there is a work plan where every activity is finished by its target time. Justify the correctness of your algorithm and its running time.

   For example, consider 3 activities with durations and target times $(3, 4)$, $(6, 10)$ and $(2, 11)$. Here $d_1 = 3, t_1 = 4$, etc. If the work plan is 3, 1, 2 then activity 3 is finished at time 2, which is at most its target time 11. However, activity 1 is finished at time $2 + 3 = 5$, which is past its target time 4. (See also the figure below.) So this ordering does not give a solution. Does some other ordering work? What is the "best" ordering?



   Hint: if your algorithm produces an ordering then it has to be shown that either this ordering works, or there is no solution. For this, it can be useful to show that from any *other* solution we can get a solution which is "more similar" to this ordering by changing the order of two neighboring activities.

4. Consider applying topological sort to a directed graph which contains directed cycles. For every $n \geq 4$ describe a directed graph on n vertices such that topological sort on that graph produces an ordering with $\Omega(n^2)$ "bad"

edges, but there is an ordering which only has $O(n)$ "bad" edges. (A "bad" edge is an edge which is inconsistent with the ordering produced, i.e., it goes backwards.)

Hint: consider a directed graph with "many" back edges. Note that an n-vertex path has $n-1$ edges, and an n-vertex complete graph (i.e., a graph with an edge between any two vertices) has $\Theta(n^2)$ edges.

5. Consider a connected graph $G = (V, E)$ with edge weights w. Let T_1 be a minimum spanning tree and T_2 be any other spanning tree. Let e be a largest weight edge in T_1, and f be a largest weight edge in T_2. Show that $w(e) \leq w(f)$.

   Hint: By contradiction, assume that $w(e) > w(f)$. Adding e to T_2 we get a cycle in T_2. Show that some edge of this cycle can be used to replace e in T_1 to get a better spanning tree.