

CS302 - Problem Set 4

1. Suppose you have a line graph G on 6 vertices $((v_1, v_2, v_3, v_4, v_5, v_6)$, connected in that order) with the following vertex weights:

$$w(v_1) = 3 \quad w(v_2) = 7 \quad w(v_3) = 10 \quad w(v_4) = 5 \quad w(v_5) = 4 \quad w(v_6) = 5$$

- (a) Let G_i be the line graph on the first i vertices with the same weights as above. Let S_i be the max-weight independent set on G_i . What are S_1, S_2, \dots, S_6 ? (Note: your answers should be *sets*.)
- (b) For $i = 3, 4, 5, 6$, verify the recurrence relationship we discussed in class: that if $v_i \in S_i$, then $S_i = S_{i-2} \cup \{v_i\}$ while if $v_i \notin S_i$, then $S_i = S_{i-1}$.
2. For each of the following statements, if it is true, prove it, and if it is false, provide a counterexample and explanation. (Pre-hint hint: if you think a statement is true, before trying to prove it, think about possible proof styles - contradiction, induction, strong induction, contrapositive, direct - and consider which might be best for the situation.)
- (a) If a line graph has at least two vertices, the minimum-weight vertex is never part of the maximum-weight independent set.
- (b) As in class, consider a line graph G , with subgraphs G_i where G_i consists of the first i vertices of G . Let S_i be the MWIS of G_i . If for $j \leq i$, we have $v_j \notin S_i$ and $v_j \notin S_{i+1}$, then $v_j \notin S_n$ for $n > i + 1$.
3. Suppose you have n assignments, and assignment i takes time $t_i > 0$ and has a due date $d_i > 0$ (you can think of d_i as the number of hours from the present time until the assignment is due). Let the completion time C_i of a task be defined as in class. Let's assume that you can only work on one assignment at a time, and once you start working on an assignment, you continue until it is finished. Given an ordering σ of the tasks, we call $l_i(\sigma) = C_i(\sigma) - d_i$ the lateness of assignment i . We would like to minimize the maximum lateness: $A(\sigma) = \max_i l_i(\sigma)$.
- (a) Do you have any ethical concerns about implementing an algorithm to solve this problem? If you were to implement such an algorithm for yourself to manage your time, would you choose the same objective function, or a different one?
- (b) Propose a function f to use to order tasks, and explain why it is reasonable given the objective.
- (c) Despite your function f being reasonable, provide a counterexample to show that this choice of f is not optimal.

- (d) (Challenge) Spend some time experimenting until you think you have determined the function f that gives you an optimal greedy algorithm. Alternatively, work backwards from the greedy proof strategy to work out what condition will result in the exchange always improving the value of the objective function.
- (e) See last page for optimal f . Prove optimality of the greedy ordering using this f .
- (f) Describe your algorithm and state its runtime.

For the scheduling problem, the optimal ordering is $f(i) = d_i$. You should not assume that every assignment has a unique deadline - after all, you might have two psets due on the same day at the same time!