

## Problem Set #3

[Help](#)

The **due date** for this quiz is **Mon 28 Jul 2014 2:59 PM CST**.

☐ In accordance with the Coursera Honor Code, I (朱顺) certify that the answers here are my own work.

### Question 1

Which of the following is true for our dynamic programming algorithm for computing a maximum-weight independent set of a path graph? (Assume there are no ties.)

☐

As long as the input graph has at least two vertices, the algorithm never selects the minimum-weight vertex.

☐

If a vertex is excluded from the optimal solution of two consecutive subproblems, then it is excluded from the optimal solutions of all bigger subproblems.

☐

If a vertex is excluded from the optimal solution of a subproblem, then it is excluded from the optimal solutions of all bigger subproblems.

☐

The algorithm always selects the maximum-weight vertex.

### Question 2

Consider a variation of the Knapsack problem where we have two knapsacks, with integer capacities  $W_1$  and  $W_2$ . As usual, we are given  $n$  items with positive values and positive integer weights. We want to pick subsets  $S_1, S_2$  with maximum total value (i.e.,  $\sum_{i \in S_1} v_i + \sum_{i \in S_2} v_i$ ) such that the total weights of  $S_1$  and  $S_2$  are at most  $W_1$  and  $W_2$ , respectively. Assume that every item fits in either knapsack (i.e.,  $w_i \leq \min\{W_1, W_2\}$  for every item  $i$ ). Consider the following two algorithmic approaches. (1) Use the algorithm from lecture to pick a max-value feasible solution  $S_1$  for the first knapsack, and then run it again on the remaining items to pick a

max-value feasible solution  $S_2$  for the second knapsack. (2) Use the algorithm from lecture to pick a max-value feasible solution for a knapsack with capacity  $W_1 + W_2$ , and then split the chosen items into two sets  $S_1, S_2$  that have size at most  $W_1$  and  $W_2$ , respectively. Which of the following statements is true?

- ☐ Algorithm (1) is guaranteed to produce an optimal feasible solution to the original problem provided  $W_1 = W_2$ .
- ☐ Algorithm (1) is guaranteed to produce an optimal feasible solution to the original problem but algorithm (2) is not.
- ☐ Neither algorithm is guaranteed to produce an optimal feasible solution to the original problem.
- ☐ Algorithm (2) is guaranteed to produce an optimal feasible solution to the original problem but algorithm (1) is not.

### Question 3

Recall our dynamic programming algorithm for computing the maximum-weight independent set of a path graph. Consider the following proposed extension to more general graphs. Consider an undirected graph with positive vertex weights. For a vertex  $v$ , obtain the graph  $G'(v)$  by deleting  $v$  and its incident edges from  $G$ , and obtain the graph  $G''(v)$  from  $G$  by deleting  $v$ , its neighbors, and all of the corresponding incident edges from  $G$ . Let  $OPT(H)$  denote the value of a maximum-weight independent set of a graph  $H$ . Consider the formula  $OPT(G) = \max\{OPT(G'(v)), w_v + OPT(G''(v))\}$ , where  $v$  is an arbitrary vertex of  $G$  of weight  $w_v$ . Which of the following statements is true?

- ☐ The formula is always correct in trees, and it leads to an efficient dynamic programming algorithm.
- ☐ The formula is always correct in general graphs, and it leads to an efficient dynamic programming algorithm.
- ☐

The formula is always correct in trees, but does not lead to an efficient dynamic programming algorithm.

- ☐ The formula is correct in path graphs but is not always correct in trees.

## Question 4

Recall the dynamic programming algorithms from lecture for the Knapsack and sequence alignment problems. Both fill in a two-dimensional table using a double-for loop. Suppose we reverse the order of the two for loops. (I.e., cut and paste the second for loop in front of the first for loop, without otherwise changing the text in any way.) Are the resulting algorithms still well defined and correct?

- ☐ Both algorithms remain well defined and correct after reversing the order of the for loops.
- ☐ The Knapsack algorithm remains well defined and correct after reversing the order of the for loops, but the sequence alignment algorithm does not.
- ☐ Neither algorithm remains well defined and correct after reversing the order of the for loops.
- ☐ The sequence alignment algorithm remains well defined and correct after reversing the order of the for loops, but the Knapsack algorithm does not.

## Question 5

Consider an instance of the optimal binary search tree problem with 7 keys (say 1,2,3,4,5,6,7 in sorted order) and frequencies

$w_1 = .05, w_2 = .4, w_3 = .08, w_4 = .04, w_5 = .1, w_6 = .1, w_7 = .23$ . What is the minimum-possible average search time of a binary search tree with these keys?

- ☐ 2.08
- ☐ 2.18
- ☐ 2.42
- ☐ 2.9

- ☐ In accordance with the Coursera Honor Code, I (朱顺) certify that the answers here are my own work.

Submit Answers

Save Answers

You cannot submit your work until you agree to the Honor Code. Thanks!