Module 5 Graded Quiz

Due Oct 22, 2021 at 11:59pm **Points** 13
Questions 13
Available Oct 9, 2021 at 12am - Feb 3 at 11:59pm 4 months
Time Limit 120 Minutes

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	20 minutes	13 out of 13

Score for this quiz: **13** out of 13
Submitted Oct 22, 2021 at 10:20pm
This attempt took 20 minutes.

Question 1

1 / 1 pts

Which of the following is not an example of a famous dynamic programming algorithm?

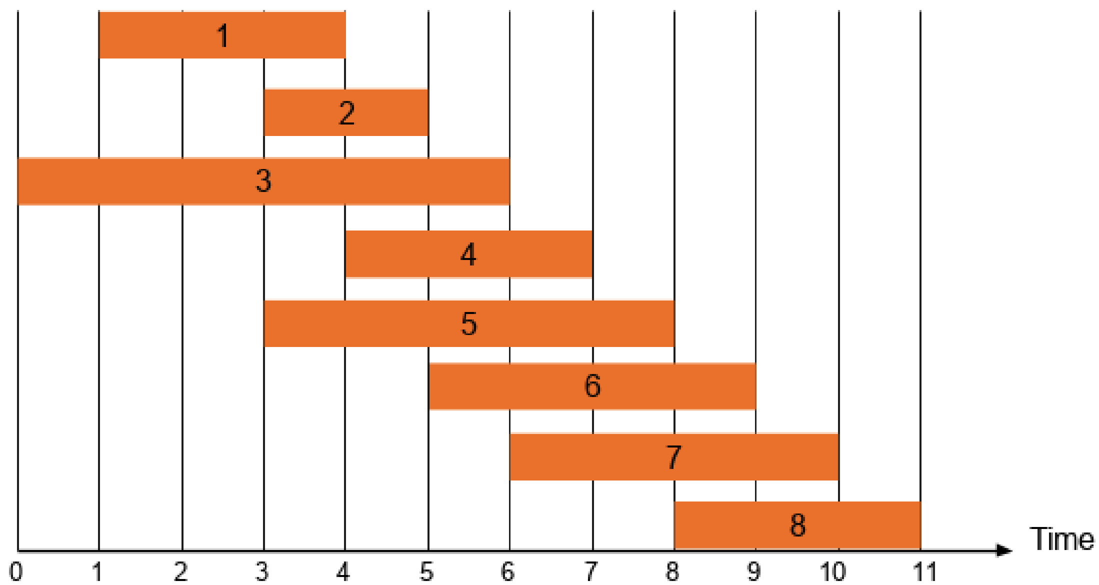
Correct!

- (selected) Dijkstra's algorithm using Fibonacci Heaps
- Viterbi for hidden Markov Models
- Unix diff for comparing two files
- Cocke-Kasami-Younger for parsing context-free grammars

Question 2

1 / 1 pts

Consider the following example for Weighted Interval Scheduling. Identify p(4)?



- 2
- 8

Correct!

- 1
- 3

Question 3

1 / 1 pts

Which of the following is the sub-problem for Weighted Interval Scheduling's dynamic programming algorithm?

- The average of v_j + OPT(p(j)) and OPT(j-1).
- The minimum of v_j + OPT(p(j)) and OPT(j-1).
- The difference of v_j + OPT(p(j)) and OPT(j-1).

Correct!

The maximum of v_j + OPT(p(j)) and OPT(j-1).

Question 4

1 / 1 pts

What is memoization?

Storing all results of all possible sub-problems in a cache and lookup as needed.

Storing the results of each sub-problem in several caches and lookup as needed.

Storing the results of some sub-problem in a cache and lookup as needed.

Correct!

Storing the results of each sub-problem in a cache and lookup as needed.

Question 5

1 / 1 pts

Why does memoization improve the exponential run-time of the original algorithm to be much faster (polynomial time)?

Correct!

Because each possible sub-problem is only computed once, and there are polynomially many of them.

- [ ] Because each possible sub-problem is computed more than once, and there are polynomially many of them.
- [ ] Because some sub-problem is only computed once, and there are polynomially many of such sub-problems.
- [ ] Because some sub-problem is computed more than once, and there are polynomially many of such sub-problems.

Question 6

1 / 1 pts

Which of the following is a recursive call of the OPT(i, w) function for the dynamic programming algorithm for Knapsack?

Correct!

- [x] The maximum of OPT(i-1, w) and v_i + OPT(i-1, w-w_i).
- [ ] The maximum of OPT(i-1, w-w_i) and v_i + OPT(i-1, w).
- [ ] The maximum of OPT(i-1, w-w_i) and v_i + OPT(i-1, w-w_i).
- [ ] The maximum of OPT(i-1, w) and v_i + OPT(i-1, w).

Question 7

1 / 1 pts

Which of the following is an informal description of OPT(i, w) for the Knapsack problem?

Correct!

- The maximum profit subset of items 1, …, i with weight limit w.
- The minimum profit subset of items 1, …, w with weight limit i.
- The maximum profit subset of items 1, …, w with weight limit i.
- The minimum profit subset of items 1, …, i with weight limit w.

Question 8

1 / 1 pts

Suppose that the weights are at most polynomial in the number of items given for the Knapsack problem. What can we say about the run-time of the dynamic programming algorithm for this problem?

- It is still pseudo-polynomial time.
- It is linear time.
- It is not efficient, since it is still an NP-complete problem.

Correct!

- It is polynomial-time.

Question 9

1 / 1 pts

Which of the following is a sub-problem for the Shortest Path problem's dynamic programming algorithm?

- The maximum of OPT(i-1, v) and the minimum of OPT(i-1, w) + l_vw over all edges vw.

The maximum of OPT(i-1, v) and the maximum of OPT(i-1, w) + l_vw over all edges vw.

The minimum of OPT(i-1, v) and the maximum of OPT(i-1, w) + l_vw over all edges vw.

Correct!

The minimum of OPT(i-1, v) and the minimum of OPT(i-1, w) + l_vw over all edges vw.

Question 10

1 / 1 pts

What does OPT(i, v) mean in the context of the dynamic programming algorithm for the Shortest Path problem?

- The length of the shortest v-t path using at least i edges.
- The length of the longest v-t path using at most i edges.
- The length of the longest v-t path using at least i edges.

Correct!

- The length of the shortest v-t path using at most i edges.

Question 11

1 / 1 pts

Given a graph *G=(V,E)* containing negative edge weights and a unique shortest path *P*, does increasing each edge weight by a constant factor such that all edge weights are positive (or zero) necessarily yield *P* when Dijkstra's algorithm is applied to it?

Yes. Dijkstra's algorithm always produces a valid shortest path when edge weights are non-negative.

No. But if we instead multiply the weight of each edge by a constant *c* such that *c* < *0*, then the shortest path solution will not change when Dijkstra's algorithm is applied.

Yes. A path with more edges will always have a total weight greater than another path that has fewer edges, so adding a constant to each edge weight will never change the shortest path solution.

Correct!

No. The total weights of paths containing more edges will vary more significantly than those with paths containing fewer edges.

Question 12

1 / 1 pts

The principle of optimality states that the solution for the tail subproblem of any optimal solution is also optimal. Which of the following examples illustrates this concept best?

Correct!

Suppose the shortest route R from Phoenix to Atlanta includes passing through Austin, Texas. Then the shortest route from Austin to Atlanta will be a sub-route of R.

None of these is an example of the principle of optimality.

Consider the shortest route R from Phoenix to Atlanta. R is simply composed of the shortest paths between each pair of consecutive cities along R.

The shortest route from Phoenix to Atlanta can be found by beginning at Phoenix and repeatedly traveling to the nearest city not yet visited.

Question 13

1 / 1 pts

Which of the following explains why the Knapsack algorithm seen in lecture only runs in pseudo-polynomial time?

- The size of W is the number of bits required to represent it, which is polynomial in the value of W.
- The size of W is the number of bits required to represent it, which is exponential in the value of W.
- The value of W may be very large.
- Correct! The size of W is the number of bits required to represent it, which is logarithmic in the value of W.

Quiz Score: **13** out of 13