CSE 551: Quiz 5 Solutions

Jamison Weber

October 8, 2021

Note that some of the solutions provided in this document may apply to questions that did not appear in your particular quiz.

1 Problem 1

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

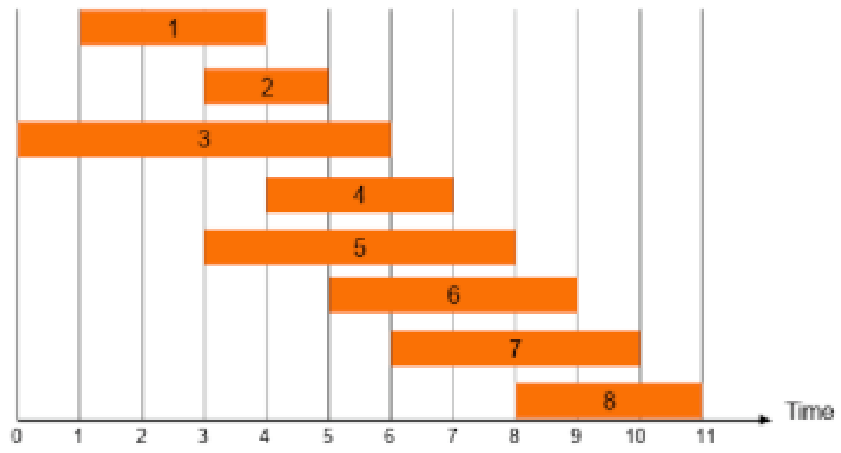
- [✓ Dijkstras algorithm using Fibonacci Heaps]
- Cocke-Kasami-Younger for parsing context-free grammars
- Viterbi for hidden Markov Models
- Unix diff for comparing two files

1.1 Rationale

Dijkstra's algorithm uses greedy as its paradigm. The other three algorithms are all classic examples of dynamic programming.

2 Problem 2

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



- 1
- 2
- 3
- 8

2.1 Rationale

Job 1 ends just before job 4 begins, and thus fits the definition of p(4) since it is the latest finishing job that does not overlap with job 4.

3 Problem 3

Which of the following is the sub-problem for Weighted Interval Schedulings dynamic programming algorithm?

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

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

3.1 Rationale

The highlighted answer is the correct dynamic programming framework for the weighted interval scheduling problem. The other three choices are incorrect.

4 Problem 4

What is memoization?

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

4.1 Rationale

We must store the result of each subproblem exactly once in order to reduce a brute force exponential time solution to a polynomial time solution.

5 Problem 5

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

- **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.

5.1 Rationale

A problem for which dynamic programming is applicable will satisfy that it has an optimal substructure and overlapping subproblems. Typically, the dynamic programming framework used to solve these problems will produce an exponential number of computations on the input due to these overlapping subproblems. If we store the solution to each subproblem in memory, then can access them as needed and therefore, we do not perform wasteful, repetitive computations. This has the effect of reducing an exponential time algorithm to a polynomial time algorithm since the number of problems we need to compute in total will be equal to the height of the recursive tree, which will be logarithmic on an exponential number of subproblems, i.e. polynomial.

6 Problem 6

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

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

6.1 Rationale

The highlighted answer is precisely the dynamic programming framework for the knapsack problem. The other three choices are incorrect.

7 Problem 7

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

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

7.1 Rationale

The highlighted answer correctly maps the parameter labels to the solution given by an execution of the knapsack algorithm. The other three answers are incorrect.

8 Problem 8

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

8.1 Rationale

The algorithm for solving the knapsack problem runs in pseudopolynomial time on the input. Since there are n items, each with their own weight, we require at least one bit to represent the weight of each of these n items. To represent W, we only need $log(W)$ bits, thus to relate the value of W to the size of the input, I end up with the equation $n \leq log(W)$, or $W \geq 2^n$. But if we can guarantee that the value of W is at most some polynomial function on the size of the input, then we can say that the algorithm will terminate in a polynomial number of computations.

9 Problem 9

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

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

9.1 Rationale

The highlighted answer is the correct dynamic programming framework for the shortest path problem. The other three answer choices are incorrect.

10 Problem 10

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

10.1 Rationale

The highlighted answer choice is precisely how Opt(i,v) was defined in lecture. The other three choices are incorrect.

11 Problem 11

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?

- **No. The total weights of paths containing more edges will vary more significantly than those with paths containing fewer edges.**
- 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.
- 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. Dijkstra's algorithm always produces a valid shortest path when edge weights are non-negative.

11.1 Rationale

As seen in the lecture and in the midterm, the only operation you can perform on all edge weights and ensure that the shortest path solution found by Dijkstra remains the same is to multiply each edge weight by some constant. Adding a constant to each edge weight can change the solution found by Dijkstra since paths with more edges (not necessarily more weight) can have their total weight disproportionately increased relative to other paths. In any case, multiplying each edge by a constant will not solve the negative edge weight problem Dijkstra faces (since signs will still be opposite each other after the multiplication), therefore, we require a dynamic programming solution that will effectively test all possible paths and return the smallest weight.

12 Problem 12

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?

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

12.1 Rationale

The first option demonstrates the principle of optimality. Finding the shortest route from Austin to Atlanta is a tail subproblem of finding the shortest route from Phoenix to Atlanta. Option two captures more of a divide-and-conquer approach, where the whole path is broken into non-overlapping smaller paths that are computed optimally. Option 3 reflects a greedy strategy, as it optimizes some local criterion.

13 Problem 13

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

13.1 Rationale

The key concept here is that in order for an algorithm to be considered polynomial-time, it must be guaranteed to terminate in time polynomial in the size of its inputs. One of the inputs of the Knapsack algorithm is an integer W. But recall the size of an integer is given by the number of bits required to represent that integer, which we know to be $\log W$ bits. Recall that the logarithmic identity $a^{\log b} = b^{\log a}$ for all real $a, b > 0$. The Knapsack algorithm runs in time proportional to $W = W^{\log_2 2} = 2^{\log_2 W}$, and thus runs in time exponential in the size of W.