CSE 551 Graded Quiz 2 Solutions

Jamison Weber

August 27, 2021

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

Question 1

In the proof of optimality for the greedy algorithm in interval scheduling, we picked r to be the maximum for which the greedy and optimal solutions were identical choices. Why was this important in the overall proof?

Correct Response

Because a contradiction about r is reached from a substitution-type argument (substituting a job from the greedy schedule into the optimal schedule).

Rationale

We first assume toward a contradiction that greedy is not optimal. Recall that the greedy solution and the optimal solution agree up to job j_r. Greedy then picks a job whose finish time is earlier than the next job optimal picks. If we have optimal pick the same job as greedy at time step $r + 1$, then the number of jobs selected by optimal remains unchanged, thus the optimal algorithm still produces an optimal schedule. Now optimal agrees with greedy through time step $r + 1$, which is a contradiction, since we defined r as the largest time step where optimal and greedy agree. Therefore, greedy is optimal.

Question 2

What is the best asymptotic run-time of the greedy algorithm for Interval Partitioning, if there are n intervals?

Correct Response

$O(nlogn(n))$

Rationale

Although the main loop clearly has a time complexity linear on the number of jobs, the preprocessing step involves sorting the jobs, and the lower bound for sorting by comparison is, of course, $O(nlog(n))$.

Question 3

Informally, in the minimizing maximum lateness problem, why does sorting jobs in ascending order of processing time not allow for a greedy-type approach to find an optimal solution?

Correct Response

Because longer jobs with earlier deadlines are possible, and delaying them by picking shorter jobs makes the solution less optimal.

Rationale

Following the counterexample from the slides: suppose job 1 has a processing time of 1 and a deadline at t=100. Job 2 has a processing time of 10 and a deadline of t=10. Since job 1 has the shortest processing time, we schedule it first, followed by job 2. Now job 2 is late by one time step. If we scheduled job 2 first, it would finish on time. We then schedule job 1, whose deadline is t=100, so job 1 finishes on time as well, yielding no lateness in the schedule. Therefore, sorting jobs in ascending order of processing time is not optimal.

Question 4

Which of the following accurately describes a high-level sketch of the proof of optimality for the farthest-in-the-future algorithm for offline caching?

Correct Response

Use induction to show that a special property holds in all cases for any given request.

Rationale

The proof relies on an inductive argument over all requests that demonstrates that the existence of an optimal reduced schedule S that makes the same eviction schedule as S_{FF} through the first $j+1$ requests remains invariant. We use this invariant to build a new schedule S' that either agrees with S (which agrees with S_{FF} to a certain point) or agrees with S_{FF}, but is no less optimal than S for any future request in the sequence (depending on the specific case). The

incorrect answers either describe an argument about a single (arbitrary) request, or misconstrue the invariant. What we really need here is an argument that considers all requests to ensure our eviction schedule S' is optimal.

Question 5

Suppose that if an item is evicted in the optimal offline caching problem, you store this eviction to a file (for logging purposes); suppose this takes $O(s)$ time for each eviction. If you run the farthest-in-future algorithm with n requests, what is the worst-case run-time for this algorithm variant?

Correct Response

$O(ns)$

Rationale

Consider the following worst-case scenario: Suppose your cache is of size one and all n incoming item requests are distinct. Obviously, there will be a total of n cache misses and $n-1$ evictions. If it takes $O(s)$ to log each eviction, then the worst case run time will be on the order of $O(ns)$.

Question 6

Do Kruskal's and Prim's algorithm find an MST where edges can have negative weight?

Correct Answer

Both do.

Rationale

Neither Kruskal's nor Prim's algorithms have any restrictions on the signs of edge weights.

Question 7

Does Dijkstra's algorithm work where edges can have negative weight?

Correct Response

It may not work.

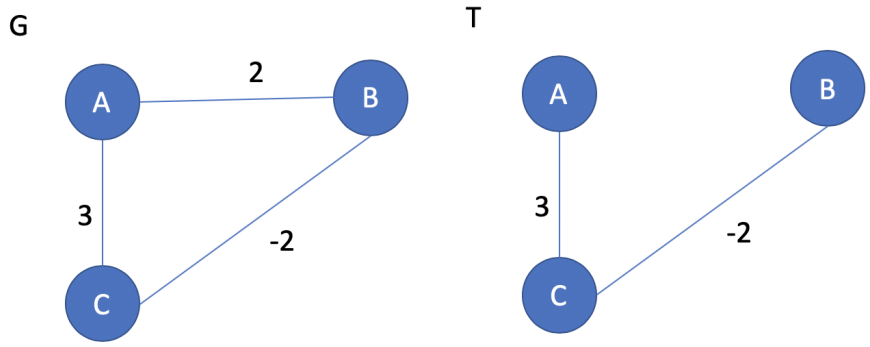

Figure 1: Figure 1: Graph G and Shortest Path Tree T

Rationale

Dijkstra's algorithm is capable of producing an optimal shortest path tree in the presence of negative edge weights, although to guarantee its optimality in all cases, all edge-weights must be non-negative. As an example of Dijkstra's algorithm finding the optimal solution with a negative edge weight present, consider the graph G, and the shortest path tree T produced by Dijkstra from G with node B as the source in figure 1. If you work out this example carefully, you will see that if your source node is B, Dijkstra will produce tree T, which is easy to see is optimal.

Question 8

In the proof of Dijkstra's algorithm's being optimal, the base case said that "$|S| = 1$ is trivial", where S is the set of explored nodes that are determined from the source node to each explored node. Why is it trivial?

Correct Response

If the size of S is 1, then S must only include the source node. The distance from the source node to itself is always 0, which must be optimal since it is the only possible path.

Rationale

In graph theory, every vertex is said to have a trivial path of length zero from itself to itself. If S consists of only one node, since Dijkstra always adds the source node first, S must only contain the source node, which as I explained, has a trivial path of length zero.

Question 9

Which of the following descriptions best embodies the paradigm of greedy algorithms?

Correct Response

Solve the given problem via a series of locally optimal decisions and never revisit a decision.

Rationale

Greedy algorithms are generally fast precisely because all decision criterion are optimized according to local criterion. For example, the greedy algorithm for interval scheduling adds jobs to its solution by selecting the next compatible job with the soonest finish time. A key feature of greedy algorithms is that these decisions are never revisited, and as such, greedy solutions are not necessarily optimal for any arbitrary problem.

Question 10

The statements below concern both Prim's and Kruskal's algorithm for obtaining minimum spanning trees from an arbitrary graph $G = (V, E)$. Mark all statements below that are true.

Correct Responses

At any given iteration of either algorithm, the partial solution obtained is necessarily a forest. The proof for Kruskal's algorithm seen in lecture applies the MST cycle property.

Rationale

Regarding the first statement, Prim's algorithm greedily grows a tree from a root node, and thus the partial solution is a forest, since a forest is simply a collection of trees (possibly one). Kruskal's algorithm adds edges one at a time, and at any given iteration, these edges and their respective nodes may not necessarily form a single component, and thus Kruskal's algorithm maintains a forest.

Regarding the second statement, see the proof in the lecture notes for the correctness of Kruskal's algorithm and how it applies the cycle property. Note that Prim's algorithm applies the cut property.