Module 5 Graded Assignment and Quiz

Due Oct 22, 2021 at 11:59pm **Points** 6 **Questions** 6
Available Oct 9, 2021 at 12am - Feb 2 at 11:59pm 4 months
Time Limit None

Instructions

Question 1

Run the weighted interval schedule algorithm on the example below.

Question 2

Let G = (V, E) be a directed graph with nodes v1,..., vn. We say that G is an ordered graph if it has the following properties.

1. Each edge goes from a node with a lower index to a node with a higher index. That is, every directed edge has the form (vi, vj) with i < j.
2. Each node except vn has at least one edge leaving it. That is, for every node vi, i = 1, 2,..., n-1, there is at least one edge of the form $$(vi, vj).

The length of a path is the number of edges in it. Consider the following problem: given an ordered graph G, find the length of the longest path that begins at v1 and ends at vn.

Answer the following two questions:

1. Show that the following algorithm does not correctly solve this problem, by giving an example of an ordered graph on which it does not return the correct answer. Initially set L = 0, and w = v1. While there is an edge out of w, choose such an edge (w, vj) for the smallest j possible. Then set w = vj, and increase L by 1. Then output L.
2. Give an efficient algorithm for the considered problem.

Question 3

As some of you know well, and others of you may be interested to learn, a number of languages (including Chinese and Japanese) are written without spaces between the words. Consequently, software that works with text written in these languages must address the word segmentation problem—inferring likely boundaries between consecutive words in the text. If English were written without spaces, the analogous problem would consist of taking a string like “meetateight” and deciding that the best segmentation is “meet at eight” (and not “me et at eight,” or “meet ate ight,” or any of a huge number of even less plausible alternatives). How could we automate this process?

A simple approach that is at least reasonably effective is to find a segmentation that simply maximizes the cumulative “quality” of its individual constituent words. Thus, suppose you are given a black box that, for any string of letters x = xi ... xk, will return a number quality(x). This number can be either positive or negative; larger numbers correspond to more plausible English words. (So quality(me’’) would be positive, while quality(ght’’) would be negative.)

Given a long string of letters y = y1 ... yn, a segmentation of y is a partition of its letters into contiguous blocks of letters; each block corresponds to a word in the segmentation. The total quality of segmentation is determined by adding up the qualities of each of its blocks. (So we’d get the right answer above provided that quality(meet’’) + quality(at’’) + quality(eight’’) was greater than the total quality of any other segmentation of the string.)

Give an efficient algorithm that takes a string y and computes a segmentation of maximum total quality. (You can treat a single call to the black box computing quality(x) as a single computational step.)

(**A final note, not necessary for solving the problem:** To achieve better performance, word segmentation software in practice works with a more complex formulation of the problem—for example, incorporating the notion that solutions should not only be reasonable at the word level, but also form coherent phrases and sentences. If we consider the example “theyouthevent,” there are at least three valid ways to segment this into common English words, but one constitutes a much more coherent phrase than the other two. If we think of this in the terminology of formal languages, this broader problem is like searching for a segmentation that also can be parsed well according to a grammar for the underlying language. But even with these additional criteria and constraints, dynamic programming approaches lie at the heart of a number of successful segmentation systems.)

Question 4

The problem of searching for cycles in graphs arises naturally in financial trading applications. Consider a firm that trades shares in n different companies. For each pair i =/= j, they maintain a trade ratio rij, meaning that one share of i trades for rij shares of j. Here we allow the rate r to be fractional; that is, r=2/3 means that you can trade three shares of i to get two shares of j.

A trading cycle for a sequence of shares i1,..., ik consists of successively trading shares in company i1 for shares in company i2, then shares in company i2 for shares i3, and so on, finally trading shares in ik back to shares in company i1. After such a sequence of trades, one ends up with shares in the same company i1 that one starts with. Trading around a cycle is usually a bad idea, as you tend to end up with fewer shares than you started with. But occasionally, for short periods of time, there are opportunities to increase shares. We will call such a cycle an opportunity cycle, if trading along the cycle increases the number of shares. This happens exactly if the product of the ratios along the cycle is above 1. In analyzing the state of the market, a firm engaged in trading would like to know if there are any opportunity cycles.

Give a polynomial-time algorithm that finds such an opportunity cycle, if one exists.

Question 5

Suppose it’s nearing the end of the semester and you’re taking n courses, each with a final project that still has to be done. Each project will be graded on the following scale: It will be assigned an integer number on a scale of 1 to g > 1, higher numbers being better grades. Your goal, of course, is to maximize your average grade on the n projects.

You have a total of H > n hours in which to work on the n projects cumulatively, and you want to decide how to divide up this time. For simplicity, assume H is a positive integer, and you’ll spend an integer number of hours on each project. To figure out how best to divide up your time, you’ve come up with a set of functions {fi : i = 1,..., n} (rough estimates, of course) for each of your n courses; if you spend h <= H hours on the project for course i, you’ll get a grade of fi(h). (You may assume that the functions fi are nondecreasing: if h < h’, then fi(h) <= fi(h’).)

So the problem is this: Given these functions {fi}, decide how many hours to spend on each project (in integer values only) so that your average grade, as computed according to the fi, is as large as possible. In order to be efficient, the running time of your algorithm should be polynomial in n, g, and H; none of these quantities should appear as an exponent in your running time.

Question 6

To assess how “well-connected” two nodes in a directed graph are, one can not only look at the length of the shortest path between them but also count the number of shortest paths.

This turns out to be a problem that can be solved efficiently, subject to some restrictions on the edge costs. Suppose we are given a directed graph G = (V,E), with costs on the edges; the costs may be positive or negative, but every cycle in the graph has strictly positive cost. We are also given two nodes v, w ∈ V.

Give an efficient algorithm that computes the number of shortest v-w paths in G. (The algorithm should not list all the paths, just the number suffices.)

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	5 minutes	6 out of 6

Score for this quiz: **6** out of 6
Submitted Oct 22, 2021 at 10:03pm
This attempt took 5 minutes.

Question 1 — 1 / 1 pts

Regarding question 1, what is the optimal weight of the given schedule?

- 7
- 6
- 8 (Correct!)
- 5

Question 2

1 / 1 pts

Regarding question 2, what is a subproblem for a dynamic programming framework in sub-question b (call it OPT(i))?

Correct!

- (selected) Compute the length of the longest path from v_1 to v_i.
- Compute the path that has longest length from v_1 to v_i.
- Compute the path that has shortest length from v_1 to v_i.
- Compute the length of the shortest path from v_1 to v_i.

Question 3

1 / 1 pts

Regarding question 3, let OPT(i) be the score of the best segmentation of the prefix consisting of the first i characters of y. Which of the following is a correct top-down recurrence relation for an optimal solution?

- $OPT(i)=\max_{i\leq j}\{OPT(j)+quality(1\cdots j)\}$
- None of these

Correct!

- (selected) $OPT(i)=\max_{j\leq i}\{OPT(j-1)+quality(j\cdots i)\}$
- $OPT(i)=\max_{j\leq i}\{OPT(n)+quality(j\cdots n)\}$

Question 4

1 / 1 pts

Regarding question 4, construct a directed graph G(V,E) where for all v∈V node $\mathbf{v}$ represents a stock. For any directed edge e=(i,j), which of the following choices of edge-weight representations would render a correct polynomial-time algorithm for detecting negative-weight cycles?

- $\log r_{ij}$
- $-r_{ij}$
- r_{ij}
- $-\log r_{ij}$

Correct!

Question 5

1 / 1 pts

Regarding Question 5, which of the following is an appropriate recurrence relation on a subproblem for this problem, A[i,h], which corresponds to the maximum grade one can receive on the first i courses, using at most h hours?

- $A[i,h]=\max_{k\le h}\sum_{j\le i} f_j(k)+A[i-1,h-k]$.
- $A[i,h]=\max_{k\ge h} f_i(k)+A[i-1,h-k]$.
- $A[i,h]=\max_{k\le h}\sum_{j\le i} f_j(k)+A[i-k,h-1]$.
- $A[i,h]=\max_{k\le h} f_i(k)+A[i-1,h-k]$.

Correct!

Question 6

1 / 1 pts

Regarding question 6, why can’t an algorithm for this problem be adopted to output the paths themselves, instead of just the number of paths?

- Because the number of such paths can be much smaller than the size of the graph itself, and so can be computed in polynomial time.
- Because different paths have different costs, and so one would need to take this into account.

Correct!

- (selected) Because there may be exponentially many such paths.
- Because there are polynomially many such paths, but it takes exponential time to find them.

Quiz Score: **6** out of 6