CS256: Algorithm Design and Analysis **Spring 2021**

Assignment 6 (due 04/14/2021)

Instructor: Shikha Singh *Solution template:* Overleaf

Dynamic Programmming and Shortest Paths

Problem 1. (From Steve Skiena's *Algorithm Design Manual*)

Consider the problem of storing n books on shelves in a library. The order of the books is fixed by the cataloging system and so cannot be rearraged. Let book b_i have thickness t_i and height h_i, for $1 \leq i \leq n$. Let the length of each bookshelf at this library be L. Suppose we have the freedom to adjust the height of each shelf to fit the tallest book on it. The cost of a particular layout is the sum, over each shelf, of the height of the largest book on that shelf. (So if shelf 1 has books with heights $(1, 5, 3)$ and shelf 2 has books with heights $(2, 4)$, the total cost is $5 + 4 = 9$.)

(a) Give an example to show that the greedy algorithm of stuffing each shelf as full as possible (that is, fill the first shelf with as many books as possible until book b_i does not fit, and then repeat the same process on subsequent shelves) does not always give the minimum overall height. *Solution.* □

(b) Give a dynamic programming algorithm that computes the height of the optimal arrangement, and analyze its time and space complexity. Follow the same recipe and instructions for your dynamic program as Assignment 5. (*Hint. We have done a similar example in class with a different cost function and constraints.*)

Solution. □

Problem 2. (KT 6.13)

Consider a firm that trades shares in n different companies. For each pair $i \neq j$, they maintain a trading rate r_{ij}, meaning that one share of i trades for r_{ij} shares of j. Here we allow the rate r to be fractional; that is, $r_{ij} = \frac{2}{3}$ means that you can trade three shares of i to get two shares of j.

A trading cycle for a sequence of shares $i_1, i_2, \ldots, i_k, i_1$ consists of successively trading shares in company i_1 for shares in company i_2, then shares in company i_2 for shares i_3, and so on, finally trading shares in i_k back to shares in company i_1.[1]

After such a sequence of trades, one ends up with shares in the same company i_1 that one starts with. Trading around a cycle, in practice, 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 particular, a sequence of trades $i_1, i_2, \ldots, i_k, i_1$ form an opportunity cycle if $r_{i_1,i_2} \cdot r_{i_2,i_3} \cdots r_{i_k,i_1} > 1$.

In analyzing the state of the market, a firm engaged in trading would like to know if there are any opportunity cycles. Give an efficient algorithm that determines whether or not such an opportunity cycle exists.

Hint. Reduce this problem to a problem we have discussed in class. A reduction from a problem A to a problem B means you take an input instance of A and turn it into an input instance of problem B. To show correctness, you can rely on the correctness of the algorithm for B from class, and use that to show that you get a correct solution to problem A.

Solution.

□

[1] If it helps, you can think about trading different curriences: suppose that 1 U.S. (item i) dollar buys 0.7 British pound (item j), then $r_{ij} = 1/0.7 = 10/7$.

Understanding Flows and Cuts

Problem 3. (KT 7.5) Is the following statement true or false? If true, you must give a justification.; if false, you must give a counterexample.

Let G be an arbitrary flow network, with a source s, a sink t, and a positive integer capacity c_e on every edge e. Let (A, B) be a minimum s-t cut with respect to the capacities $\{c_e : e \in E\}$. Now suppose we add 1 to every capacity; then (A, B) is still a minimum s-t cut with respect to the new capacities $\{1 + c_e : e \in e\}$.

Solution.

□

Problem 4. (Modified KT 7.23 and 7.24) Suppose you're looking at a flow network G with source s and sink t, and you want to be able to express something like the following intuitive notion: *Some nodes are clearly on the "source side" of the main bottlenecks; some nodes are clearly on the "sink side" of the main bottlenecks; and some nodes are in the middle.* However, G can have many minimum cuts, so we have to be careful in how we try making this idea precise. Here's one way to divide the nodes of G into three categories of this sort.

- We say a node v is *upstream* if, for all minimum s-t cuts (A, B), we have $v \in A$—that is, v lies on the source side of every minimum cut.
- We say a node v is *downstream* if, for all minimum s-t cuts (A, B), we have $v \in B$—that is, v lies on the sink side of every minimum cut.
- We say a node v is *central* if it is neither upstream nor downstream; there is at least one minimum s-t cut (A, B) for every $v \in A$, and at least one minimum s-t cut (A', B') for which $v \in B'$.

In this question, we design an algorithm to classify vertices of G into these categories and use the classification to characterize graphs that have a unique minimum cut. Let f be the maximum flow in G. Consider the cut (A^*, B^*), where $A^* = \{u \mid u \text{ is reachable from } s \text{ in } G_f\}$ and let $B^* = V - A^*$. Thus, $v(f) = cap(A^*, B^*)$ and (A^*, B^*) is a minimum cut of G.

(a) Show that the set A^* is the set of upstream vertices of G, that is, v is upstream if and only if $v \in A^*$.

Solution.

□

(b) Using part (a), describe an efficient algorithm to find the downstream vertices in G. You must justify why the algorithm is correct. (*Hint.* Consider the graph G^R, with all direction of edges in G reversed. G^R now has source t and sink s.)

Solution.

□

(c) Show that G has a unique minimum cut if and only if G has no central vertices, that is, the union of upstream and downstream vertices is the set V.

Solution.

□