College of Computer & Information Science
Northeastern University
Fall 2017

CS5800
MS Algorithms
3 November 2017

Problem set 5 (Due Tuesday, November 14, 11:59 pm)

- The assignment is due at the time and date specified. Late assignments will not be accepted.
- We encourage you to attempt and work out all of the problems on your own. You are permitted to study with friends and discuss the problems; however, *you must write up your own solutions, in your own words.*
- If you do collaborate with any of the other students on any problem, please do list all your collaborators in your submission for each problem.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class (or the class staff) is strictly prohibited. If you reference any source other than the textbook or class notes, please make sure you cite them in your submission.
- We require that all homework submissions be neat, organized, and *typeset*. You may use plain text or a word processor like Microsoft Word or LaTeX for your submissions. If you need to draw any diagrams, however, you may draw them with your hand.

1. (5 points) Chokepoints

Let $G = (V, E)$ be a directed graph and let $w : E \to \mathbb{R}$ be a weight function. For a path P from u to v we define the *chokepoint* of P to be the weight of an edge with smallest weight $\min_{e \in P}\{w(e)\}$. For two vertices u and v we define the *chokepoint* from u to v to be the maximum, over all paths P from u to v, of the chokepoint of P. (If there is no path from u to v, then we can define the chokepoint from u to v to be the undefined symbol $\perp$.)

Design an algorithm that takes as input G, w, and a source s, and computes the chokepoint from s to v for every vertex $v \in V$. Prove the correctness of your algorithm and analyze its running time.

2. (5 points) Shortest paths with few tolls

We have focused our attention on shortest path problems where there is a single weight for each edge, and we need to find the shortest paths under these weights. In real life, often we have multiple criteria for determining good routes; for example, the cost of a route and the time taken. It turns out that many such problems are actually much harder than the standard shortest paths problems; but not all such problems are hard.

You are given a directed graph $G = (V, E)$ with weights on edges $w : E \to \mathbb{R}$. A subset T of the vertices in V are marked as *toll nodes*.

Design an algorithm that takes as input G, w, T, a source s, a destination t, and an integer k, and determines a path from s to t that contains at most k toll nodes and has shortest weight among all such paths. State the running time of your algorithm. In your submission, you need not prove the correctness of your algorithm; but please note that you need to be able to prove it if asked.

3. (1 + 4 = 5 points) Determining period with most rainfall deficit

Being the chief climatologist of your city, you have been given the rainfall data for each day for the last 100 years in your city. So you have an array $R[1..n]$ where $R[i]$ lists the amount of rain measured in your city on the ith day (from some starting point).

Let $\mu = (\sum_{1 \le i \le n} R[i])/n$ denote the average rainfall in R. For an interval $I = [\ell, r]$, $1 \le \ell, r \le n$ of days, we define the *deficit* of I to be $\mu(r - \ell + 1) - (\sum_{\ell \le i \le r} R[i])$. That is, the deficit of I is the difference between the total amount of rain expected in I and the total amount of rain that actually fell in I. (Note that the deficit of an interval can be negative.)

For a new climate study, you have been asked to find an interval I that has the largest deficit.

(a) Design a brute-force $\Theta(n^2)$ time algorithm for your problem.

(b) Having just completed CS5800, you think that you can do much better. Design a much more efficient $\Theta(n)$ time algorithm for your problem.

You need not prove the correctness of your algorithms, but please note that you need to be able to prove it if asked.

4. (5 points) Longest common subsequence of three sequences

Design an efficient algorithm that takes as input three sequences X, Y, and Z of length m, n, and p, respectively and returns the longest common subsequence of X, Y, and Z. Prove the correctness of your algorithm and analyze its worst-case running time.

Note that W is a common subsequence of X, Y, and Z if and only if W is a subsequence of X, W is a subsequence of Y, and W is a subsequence of Z.

5. (5 points) Taking turns

Alice and Bob play the following game: Given a sequence of numbers:

$$a_1, a_2, ..., a_n$$

each player at her/his turn, takes either the first or last number of the sequence. (So if Alice takes a_1 at the first turn, then Bob can take either a_2 or a_n at his turn. If Bob takes a_n then Alice can take either a_2 or a_{n-1} and so on)

The game ends when there are no numbers left. At the end of the game the score of each player is the sum of the numbers she/he took, minus the sum of the numbers of the opponent.

Describe a dynamic programming algorithm that computes the maximal score that the first player can guarantee. Prove the correctness of your algorithm. What is the complexity of the algorithm that you suggest?

6. (5 points) Picking entries from a matrix

Let A be an $n \times m$ matrix of positive integers; A has n rows and m columns. You are asked to pick a total of k entries from the matrix that yield the maximum sum, under the following constraint: the entries that you pick in any row have to start from the first column and be contiguous. That is, your problem is to determine k_1, k_2, ..., k_n such that

$$k_1 + k_2 + \cdots + k_n = k, \text{ and}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{k_i} A[i, j] \text{ is maximized.}$$

Design an efficient algorithm to solve the above problem. State the worst-case running time of your algorithm. You need not prove the correctness of your algorithm, but please note that you need to be able to prove it if asked.