College of Computer & Information Science
Northeastern University
Spring 2014

CS4800
Algorithms and Data
14 March 2014

A Few Practice Problems for Midterm 2

1. Planning a road trip

You are planning a road trip that starts at location 0 and ends at location n, going through locations 1, 2, ..., $n-1$ in that order. Due to health and logistical reasons, you are able to visit at most k locations a day. If at the end of a day, you are at a location i, then you need to shell out d_i dollars to stay at a hotel in location i. (If you reach a location i on a particular day and stay overnight, then this location is counted toward the limit of k for the day you reach i, not for the following day. Also, location 0 does not count toward the limit for the first day.)

Given the significant differences among the hotel costs, you would like to find a schedule that minimizes the total hotel cost, while maintaining the constraint that you visit at most k locations in a day.

Give an $O(nk)$ time algorithm to determine the minimum total hotel cost you will incur for your road trip. It is sufficient to give a recurrence, and describe your algorithm briefly in words. For partial credit, you may give an algorithm that is less efficient.

2. Finding an end within an infinite array

You are given an infinite array A in which the first n cells contain integers in sorted order and the rest of the cells are filled with ∞. You are not given the value of n. Describe an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in $O(logn)$ time.

3. Intercepting encoded communication

You have intercepted some communication – a sequence σ of m bits – that you suspect is between your rivals Alice and Bob. You know that every communication between them is a sequence of codewords drawn from a set S of n binary words that they share. And you know the set S. You would like to determine whether σ is indeed a concatenation of words from S.

Give an algorithm that takes σ and S as input and determines whether σ is a concatenation of words from S. Your algorithm must run in time polynomial in m and n. Analyze the worst-case running time of your algorithm.

Example: Suppose S is the set $\{1, 101, 111, 00, 100, 10, 110\}$. If σ is 10111100, then your algorithm should return "yes" since σ can be written as 101+111+00. Note that there are multiple ways that σ can be written as a concatenation of words in S; for instance, 10+111+100 and 10+1+1+1+1+00.

If σ is 00000, then your algorithm should return "no" since σ cannot be written as any concatenation of the words in S.

4. MST and lightest edge in graph

Let G be an undirected connected graph with weights on edges. Assume that the edge weights are all distinct. Prove or disprove: Then, the edge with the smallest weight is always in the minimum spanning tree.

5. Shortest paths in directed graphs with both vertex and edge weights

Suppose you are given a directed graph with positive weights on both edges and vertices. The length of a path in the graph is the sum of the weights on the edges and the vertices along the path. Give a polynomial-time algorithm to determine the shortest path from a given source s to all other vertices in the graph.

6. Impact of scaling weights on shortest paths

Let G be a directed graph with positive edge weights Let P be a shortest path from s to t in G. True or False: If the weight of every edge in G is doubled (i.e., multiplied by 2), then P remains a shortest path from s to t.

7. Assigning programmers to software projects

Nubert is a high-level manager in a software firm and is managing n software projects. He is asked to assign m of the programmers in the firm among these n projects. Assume that all of the programmers are equally (in)competent.

After some careful thought, Nubert has figured out how much benefit i programmers will bring to project j. View this benefit as a number. Formally put, for each project j, he has computed an array $A_j[0..m]$ where $A_j[i]$ is the benefit obtained by assigning i programmers to project j. Assume that $A_j[i]$ is nondecreasing with increasing i. Further make the economically-seemingly-sound assumption that the marginal benefit obtained by assigning an ith programmer to a project is nonincreasing as i increases. Thus, for all j and $i \geq 1$, $A_j[i+1] - A_j[i] \leq A_j[i] - A_j[i-1]$.

Help Nubert design a greedy algorithm to determine how many programmers to assign to each project such that the total benefit obtained over all projects is maximized.