College of Computer & Information Science
Northeastern University
Fall 2017

CS5800
Algorithms
November 14, 2017

Some Practice Problems for Midterm 2

1. Depth-First Search

Let $G = (V, E)$ be a directed graph, and let $u, v \in V$ be two vertices such that there exists a path P_1 from u to v and a path P_2 from v to u. Show that in any execution of the DFS algorithm the vertices u and v are in the same component of the DFS forest.

2. MSTs and Shortest Paths

Let G be an undirected connected graph with positive weights on edges. For each of the following claims, indicate whether it is true or false. Briefly justify your answers.

(a) If T is a minimum spanning tree of a weighted connected undirected graph G, then a minimum-weight edge of T is also a minimum-weight edge of G.

(b) Let G be a weighted directed graph and s be a vertex in G. Let p denote a shortest path from s to t. If we increase the weight of every edge in the graph by 1, then p remains a shortest path from s to t.

3. Subsequences

Recall that we say that S is a *subsequence* of T if S can be obtained from T by deleting some characters from T. For instance, the sequence A, T, C, T, G is a subsequence of $G, \underline{A}, A, A, \underline{T}, \underline{C}, G, G, \underline{T}, T, \underline{G}$. (Positions indicating the subsequence property are underlined.)

Give an algorithm that takes as input two sequences S and T and determines whether S is a subsequence of T. State the worst-case running time of your algorithm. The more efficient your algorithm is in terms of its worst-case running time, the more credit you will get.

4. Directed cycle of minimum weight

Given a directed graph $G = (V, E)$ with positive weights on each edge, design an efficient algorithm to determine a directed cycle in G of minimum total weight. If no directed cycle exists in G, then your algorithm must indicate so.

Analyze the worst-case running time of your algorithm. The more efficient your algorithm is in terms of its worst-case running time, the more credit you will get.

5. Maximum weight independent set for a path

A path graph G is simply an undirected graph consisting of the vertex set $V = \{v_1, v_2, \ldots, v_n\}$ and the edge set

$$E = \{(v_i, v_{i+1}) : 1 \leq i \leq n-1\}.$$

An *independent set* of G is a subset I of V such that there is no edge in E between any pair of vertices of I. We also associate a positive weight w_i with vertex v_i. The weight of any subset S of vertices is simply the sum of the weights of the vertices in S.

For example, consider a path graph G with five vertices v_1, v_2, v_3, v_4, v_5, with weights 1, 9, 6, 3, and 7, respectively and four edges $(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5)$. The set $\{v_2, v_5\}$ is an independent set with weight 16, while the set $\{v_1, v_3, v_4\}$ is not an independent set.

Give an efficient algorithm to determine a maximum weight independent set of a given weighted path graph G.

6. 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.

7. Alternating red-blue paths

Let $G = (V, E)$ be a directed graph in which each vertex has been assigned a color, either red or blue. A directed path in G is called an *alternating red-blue path* if and only if no two consecutive vertices on the path have the same color. Give an efficient algorithm that determines for *all* pairs of vertices u, v in V whether v is reachable from u via an alternating red-blue path.

Analyze the worst-case running time of your algorithm. The more efficient your algorithm is in terms of its worst-case running time, the more credit you will get.

8. Longest Non-Increasing Subsequence

Design an algorithm that gets a sequence of distinct integers $a_1, a_2, \ldots, a_n$ and finds a longest non-increasing subsequence.