University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2020

February 18, 2020

Homework 7, Due Wednesday, February 26, 2020

On all problems provide justification of your answers. Provide a clear explanation of why your algorithm solves the problem, as well as a justification of the run time. Since this assignment is from the dynamic programming section - your algorithms should use dynamic programming!

Problem 1 (10 points) Weighted Independent Set on a Path:

The weighted independent set problem is: Given an undirected graph $G = (V, E)$ with weights on the vertices, find an independent set of maximum weight. A set of vertices I is independent if there are no edges between vertices in I. This problem is known to be NP-Complete.

For a simpler problem, consider a graph that is a path, where the vertices are $v_1, v_2, \ldots, v_n$, with edges between v_i and v_{i+1}. Suppose that each node v_i has an associated weight w_i. Give an algorithm that takes an n vertex path with weights and returns an independent set of maximum total weight. The run time of the algorithm should be polynomial in n.

Problem 2 (10 points) Task Choice:

Suppose that each week you have the choice of a high stress task, a low stress task, or no task. If you take a high stress task in week i, you are not allowed to take any task in week $i+1$. For n weeks, the high stress tasks have payoff $h_1, \ldots, h_n$, and the low stress tasks have payoff $l_1, \ldots, l_n$, and not doing a task has payoff 0. Give an algorithm which given the two lists of payoffs, maximizes the value of tasks that are performed over n weeks. The run time of the algorithm should be polynomial in n.

Problem 3 (10 points) Word segmentation:

(This problem is based on problem 5 on Page 316 of the text without the excessive verbiage.) The word segmentation problem is: given a string of characters $Y = y_1 y_2 \ldots y_n$, optimally divide the string into consecutive characters that form words. (The motivation is that you are given at text string without spaces and have to figure out what the words are. For example, *"meetateight"* could be *"meet ate ight"*, *"me et at eight"* or *"meet at eight"*.) The problem is to find best possible segmentation. We assume we have a function *Quality* which returns an integer value of the goodness of a word, with strings that correspond to words getting a high score and strings that do not correspond to words getting a low score. The overall quality of a segmentation is the sum of the qualities of the individual.

Give an dynamic programming algorithm to compute the optimal segmentation of a string. You can assume that calls to the function *Quality* take constant time and return an integer value.

Programming Problem 4 (10 points) Greedy Algorithms for Interval Scheduling:

This programming problem and the next will look at the interval scheduling problem with the objective function of maximizing the sum of the lengths of selected intervals: The input for an interval scheduling problem is a set of intervals $I = \{i_1, \ldots, i_n\}$ where i_k has start time s_k, and finish time f_k and the output is a set of non-overlapping intervals that has the maximum possible sum of lengths.

Implement routines for the following:

a) A random interval generator. Given integer parameters n, L, and r, generate n intervals, where each interval has a starting position uniformly chosen from $[1, L]$ and length uniformly chosen from $[1, r]$.

b) A greedy algorithm for interval scheduling which selects intervals in earliest starting time first order.

c) A greedy algorithm for interval scheduling which selects intervals in longest length first order.

For this problem, submit your code for the three routines.

Programming Problem 5 (10 points) Dynamic Programming for Interval Scheduling:

Implement a dynamic programming algorithm that optimally solves the Interval Scheduling problem to maximize the sum of the lengths of non-overlapping intervals.

Evaluate the performance of the dynamic programming algorithm compared with the two greedy algorithms from Problem 4 on randomly generated intervals. In your test generator use $n = 10,000$, $L = 1,000,000$ and $r = 2,000$.

For this problem, submit your code for the dynamic programming problem along with the output from a series of test runs on all three algorithms.