University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2023

February 17, 2023

Homework 7, Due Friday, February 24, 2023

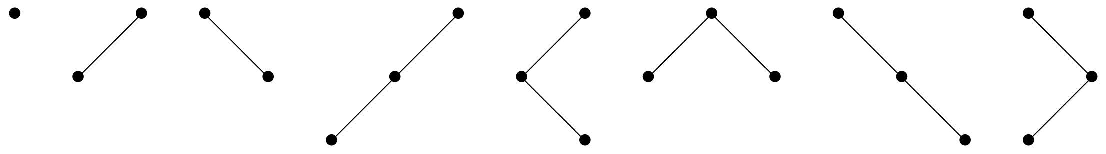
On problems one to three provide justification of your answers. Provide a clear explanation of why your algorithm solves the problem, as well as a justification of the run time. Since this assignment is from the dynamic programming section - your algorithms should use dynamic programming!

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 this problem, we restrict attention to a graph that is a path. Suppose P 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 P with weights and returns an independent set of maximum total weight. The run time of the algorithm should be polynomial in n (i.e., $O(n^k)$ for some k).

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. (You may assume that the task payoffs are all greater than zero.) 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 (i.e., $O(n^k)$ for some k).

Problem 3 (10 points) Counting Trees:
A binary tree is a rooted tree where each node has at most two children. There is one binary tree with one node, two binary trees of two nodes, and five binary trees of three nodes as shown here:



The number of binary trees $T(k)$ with k vertices is given by the formula:

$$T(k) = \begin{cases} 1 & k = 0 \\ 1 & k = 1 \\ \sum_{j=0}^{k-1} T(j)T(k-j-1) & k > 1 \end{cases}$$

Give an algorithm to compute the number of binary trees with n nodes. Your algorithm should run in time $O(n^2)$. (While there is a formula for this that you can Google, you should not just compute the answer based on the formula.)

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

This programming problem and the next looks at the weighted interval scheduling problem with the objective function of maximizing the weight of selected intervals: The input for a weighted 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 a value v_k and the output is a set of non-overlapping intervals that has the maximum possible sum of values.

Implement routines for the following:

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

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 maximum value first order.

d) Define the *value density* as the value divided by the length, i.e, $v_k/(f_k - s_k)$. A greedy algorithm for interval scheduling based on maximum value density first.

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

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

Implement a dynamic programming algorithm that optimally solves the Interval Scheduling problem to maximize the value of a set of non-overlapping intervals. You should base the algorithm on the one presented in class in Lecture 18.

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 $n = 10,000$, $L = 1,000,000$, $r = 2,000$, and $v = 100$ for submission. (You will likely want to experiment with smaller values during debugging. You may also want to modify the generator during debugging, and test inputs where you can easily evaluate the results, such as setting all values to 1.)

For this problem, submit your code for the dynamic programming problem along with the output from a series of 10 runs on all four algorithms. Each run should compare the four algorithms on the same set of intervals. You should give the number of intervals found, as well as the sum of the values (which is what you want to maximize).