CS 6363.500 Homework 2

Due Tuesday February 19th on eLearning

February 6, 2019

Please answer each of the following questions.

Important advice for this homework and beyond

- This is ***not*** a homework about greedy algorithms. In particular, all the dynamic programming questions are best solved by first thinking of a backtracking approach, describing a recurrence, and then performing memoization.
- When developing any dynamic programming algorithm both in and outside this class, we ***strongly*** recommend following the steps outlined in Erickson 3.4. Do not even *start* to think about tables or for loops until you have a recursive solution to your problem, including a plain English specification of the recursive subproblems you are solving. If you do not explain what subproblems you are solving, we won't be able to tell if you're solving them correctly, and you're get very few, if any, points for your answers.

1. (a) Consider the polynomial $p(x) = x^3 + 5x^2 + 5x + 2$. Perform a discrete Fourier transform by evaluating the polynomial at the complex 4th roots of unity.

   (b) Suppose we are given two sets X and Y of non-negative integers. The Minkowski sum $X + Y$ is the set of all pairwise sums $\{x + y \mid x \in X, y \in Y\}$. Observe that $|X + Y|$ is ***not*** necessarily $|X| \cdot |Y|$.

   Describe an algorithm to compute the number of elements in $X + Y$ in $O(M \log M)$ time where M is the greatest element in $X \cup Y$. *[Hint: Perform a reduction to polynomial multiplication.]*

2. (a) Suppose we are given two sorted arrays $A[1 .. n]$ and $B[1 .. n]$. Describe an algorithm to find the median element in the union of A and B in $\Theta(\log n)$ time. You may assume that the arrays contain no duplicate elements (so the median has rank n). *[Hint: Compare $A[\lfloor n/2 \rfloor]$ and $B[\lfloor n/2 \rfloor]$. How can you reduce your search space to two sorted arrays of size $\lceil n/2 \rceil$?]*

   (b) Now suppose we are given two sorted arrays $A[1 .. m]$ and $B[1 .. n]$ with no duplicate elements and an integer k where $1 \le k \le m+n$. Describe an algorithm to find the kth smallest element in $A \cup B$ in $\Theta(\log(m) + \log(n))$ time. *[Hint: Now compare $A[\lfloor m/2 \rfloor]$ and $B[\lfloor n/2 \rfloor]$ but only reduce the search space in one of the two arrays.]*

3. In the Knapsack problem, we are given an (ordered) collection of n items represented as two arrays of non-negative integers $W[1 .. n]$ and $V[1 .. n]$ along with a non-negative integer w. Each item i has a *weight* $W[i]$ and a *value* $V[i]$. Integer w represents the *capacity* of our knapsack. Our goal is to find a collection of items of total weight at most w while attempting to maximize the total value of the items we pack. Formally, we want a set of items $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, n\}$ that maximizes $\sum_{j=1}^{k} V[i_j]$ subject to $\sum_{j=1}^{k} W[i_j] \le w$.

   (a) For any integers i and x with $0 \le i \le n$ and $0 \le x \le w$, let $MaxValue(i, x)$ denote the *maximum* total value we can achieve packing items 1 through i into a knapsack of capacity x. Give a recurrence definition for $MaxValue(i, x)$.

   (b) For any integers i and u with $0 \le i \le n$ and $0 \le u \le \sum_{i=1}^{n} V[i]$, let $MinWeight(i, u)$ denote the *minimum* total weight of any subset of items 1 through i that have total value *at least* u (or ∞ if there is no such subset). Give a recurrence definition for $MinWeight(i, u)$.

   (c) Describe a dynamic programming algorithm for the Knapsack problem based on *either* of the two specifications above. Your running time bound should be a function on one or more of n, x, and $v = \sum_{i=1}^{n} V[i]$, the total value of all n items. (An algorithm that merely computes the *value* of an optimal solution is still worth full credit.)

4. Recall the Rod Cutting problem. As stated in class, we are given an array $P[1 .. n]$ of non-negative integers where $P[i]$ represents the selling price (in dollars) of a piece of steel rod of length i. We are also given a non-negative integer n representing the total length of a steel rod we have on hand. Our goal is to compute the maximum revenue obtainable by cutting the rod of length n into smaller (integer length) pieces and selling each of the pieces.

(a) Our distributor has recently decided it is not worth selling any piece of rod worth less than \$5; however, they are willing to distribute *any* collection of rod pieces where each piece is worth at least \$5. Describe an efficient dynamic programming algorithm that computes the total number of ways we can cut our rod of length n so each piece is worth at least \$5. For this problem, the *location* of the cuts along the rod matters, so cutting a rod of length 6 into a sequence of pieces of lengths 2 then 4; lengths 3 then 3; or lengths 4 then 2 are all different ways to cut the rod.

(b) Our distributor is no longer worried about the price of individual rod pieces (so we're OK to sell a piece that costs \$4, for example), but now they want us to cut up a tungsten rod of length n as well! Let $Q[1 \,..\, n]$ be another array of non-negative integers where $Q[i]$ represents the revenue from selling a tungsten piece of length i. Our cutting machine will have less work to do if we cut both rods at the same set of locations. Describe an efficient dynamic programming algorithm that computes the maximum revenue we can obtain cutting both rods so we have identical sets of piece lengths.

5. Suppose we are given a set L of line segments in the plane, where each segment has one endpoint on the line $y = 0$ and one endpoint on the line $y = 1$, and all $2n$ endpoints are distinct.

(a) Describe a dynamic programming algorithm to compute the largest subset of L in which no pair of segments intersects. (Again, returning the size of the largest subset is worth full credit.)

(b) Describe an algorithm to compute the largest subset of L in which ***every*** pair of segments intersects. *[Hint: Perform a reduction to the algorithm from part (a).]*