CS302 - Problem Set 7

1. Suppose you run a company with two offices, one in Washington, D.C. and the other in San Francisco, California. You always spend the whole week in one location, but each weekend, you decide whether to fly to the other office. In week i, you can make W_i dollars if you are in Washington, and C_i dollars if you are in San Francisco. Each flight from one office to the other costs \$1000. Suppose you are given the lists $W_1, W_2, \ldots, W_n$ and $C_1, C_2, \ldots, C_n$. Also suppose you are in Washington initially, and need to be back in Washington after the nth week. What schedule will maximize your profits? (Note your solution should not just give the maximum profit, but should return a schedule.)

   (a) **[6 points]** A greedy algorithm would always choose to work in the office with the larger profit that week. Give a counter example showing that this strategy is not always optimal.

   (b) **[6 points]** Think about designing a dynamic programming algorithm. In your algorithm, you should first construct an array, and then work backwards through the array. Explain what values you will put in the array, provide a recurrence relation that you can use to fill out the array, and explain why this relation is correct. (You don't need to give a formal proof.)

   (c) **[9 points]** Give psuedocode for a dynamic programming algorithm.

   (d) **[11 points]** Use a loop invariant to prove that when you work backwards through the array to construct the optimal solution is correct.

   (e) **[3 points]** What is the runtime of your algorithm?

2. For each of the following: If the algorithm need to change, please explain why- if the algorithm stays the same, you don't need to explain. If the algorithm needs to change, please briefly sketch how you would change it, if it is possible to produce a quick fix, and how that would impact the runtime.

   (a) **[6 points]** In MWIS on a line, we only looked at graphs where the vertex weights were positive integers. Would we have to change the algorithm if we used negative integers instead?

   (b) **[6 points]** In MWIS on a line, we only looked at graphs where the vertex weights were positive integers. Would we have to change the algorithm if we used positive real numbers instead?

   (c) **[6 points]** In the Knapsack problem, we only looked at items whose values were positive integers. Would we have to change the algorithm if we used negative integers instead (keeping sizes of items as positive integers)?

   (d) **[6 points]** In the Knapsack problem, we only looked at items whose values were positive integers. Would we have to change the algorithm if we used positive real numbers instead (keeping sizes of items as positive integers)?

(e) **[6 points]** In the Knapsack problem, we only looked at items whose sizes were positive integers and the capacity was a positive integer. Would we have to change the algorithm if we used negative integers for item sizes instead (keeping values of items as positive integers)?

(f) **[6 points]** In the Knapsack problem, we only looked at items whose sizes were positive integers. Would we have to change the algorithm if we used positive real numbers for item sizes instead (keeping values of items and the capacity as positive integers)?

(g) **[6 points]** (Challenge) In the Knapsack problem, we only looked at items whose sizes and values were positive integers and the capacity was a positive integer. Would we have to change the algorithm if we used both negative and positive integers for item sizes and values (keeping the capacity a positive integer)?

3. Last week, we considered two randomized search algorithms: search with and without replacement. We are interested in the random variable X which is the number of times we randomly select an item to check before we find the item we are looking for (and which gives us the runtime of the algorithm). At the end of last week's problems, we had broken up X into a sum of indicator random variables, $X = \sum_{i=1}^{\infty} X_i$, where $X_i(s) = 1$ if the loop (the loop where at each iteration we choose a new random element to check) undergoes at least i iterations on element s of the sample space, and $X_i(s) = 0$ otherwise. We previously considered the case that we were searching through an array of length 3. This week, let's consider an array of length n. For a hint about calculating probabilities, see the last page.

   (a) In random search with replacement, what is the probability that at least i iterations of the loop occur?

   (b) In random search without replacement, what is the probability that at least i iterations of the loop occur?

   (c) Using linearity of expectation, and your result from part (a), calculate the average number of iterations in random search with replacement until the item is found (assume the item you are looking for is in the array).

   (d) Using linearity of expectation, and your result from part (b), calculate the average number of iterations in random search without replacement until the item is found (assume the item you are looking for is in the array).

   (e) (Challenge) Suppose that you are doing random search with replacement, but the item you are looking for is not in the array. You stop the algorithm when you have looked at every element in the array but have not found the item you are looking for. What is the average runtime of this algorithm?

Hint: if you have a sequence of random outcomes like in an algorithm, where at each iteration of a loop you use new randomness to decide what will happen next, the probability of a sequence of outcomes is the *product* of the probabilities of each individual outcome in the sequence.