CS302 - Problem Set 8

1. `Selection` is an algorithm that is similar to QuickSort. Instead of sorting the array, it finds the k^{th} smallest element of an unsorted array. (Also called the kth order statistic). Note that if the pivot is the rth smallest element of the array, Partition puts the pivot into rth position in the array. Here is pseudocode:

Algorithm 1: `Selection`(A, s, f, k)

Input : An array A of n integers (not necessarily unique), start index s, final index f, and an integer $k \in \{1, \ldots, n\}$, with the promise that the kth smallest element of A is between indices s and f inclusive

Output: The kth smallest element of $A[1:n]$.

```
1  piv = ChoosePivot(s, f);
   // Chooses a pivot between s and f inclusive using some procedure
2  Run Partition on elements of A from s to f (inclusive) using piv as pivot;
3  Let p be the index of the pivot in the array A[1 : n] after Partition;
4  if p = k then
5  |  return A[p];
6  end
7  if p < k then
8  |  return Selection(A, p + 1, f, k);
9  else
10 |  return Selection(A, s, p − 1, k);
11 end
```

(a) **[6 points]** Using the master method, what is the runtime of this algorithm if `ChoosePivot` somehow always chooses the pivot to be the middle order statistic (i.e. the median value of the array), and the array is originally size n.

(b) **[6 points]** Create and analyze a recurrence relation for the runtime of this algorithm if `ChoosePivot` somehow always chooses the pivot to be the first order statistic (i.e. the smallest element the array), and the array is originally size n.

(c) Now consider the case that `ChoosePivot` chooses a pivot uniformly at randomly from s to f inclusive. Consider the random variable X_{ij}, for $i < j$, which is the number of times the i^{th} and j^{th} smallest elements of A are compared at some point in the algorithm.

i. **[3 points]** What is the sample space of this problem?

ii. **[6 points]** Explain why X_{ij} is an indicator random variable.

iii. **[6 points]** What is the probability of z_i and z_j being compared if we are trying to find the kth order statistic, and $k < i.j$?

iv. **[6 points]** What is the probability of z_i and z_j being compared if we are trying to find the kth order statistic, and $i, j < k$?

v. **[6 points]** What is the probability of z_i and z_j being compared if we are trying to find the kth order statistic, and $i \leq k \leq j$?

vi. **[6 points]** Use linearity of expectation, and properties of the expectation value of indicator random variables to create an explicit expression involving sums that gives the average number of comparisons done over the course of the algorithm. (See hint on last page.)

vii. **[6 points]** (Challenge) Analyze the sums from the previous part to get. $O(n)$.

2. You run a plant that produces sheets of aluminum alloy, and then you cut them to size for customers. Your machine produces sheets of dimension $A \times B$, and you can cut any sheet into two smaller sheets by making a vertical or horizontal cut. So for example, if you have a 2×4 sized piece, you could cut it into

   - Two 1×4 pieces.
   - Two 2×2 pieces.
   - A 2×1 and a 2×3 pieces.

   You can also rotate pieces (so a 2×3 piece is the same as a 3×2 piece).

   Suppose you produce n different sized products, where product i has dimensions $x_i \times y_i$, and you can sell product i for v_i dollars. (You can sell multiple copies of the ith product if you can produce multiple pieces of that size.) Assume x_i, y_i, and v_i for $i \in \{1, 2, \ldots, n\}$ are positive integers. You will design an algorithm that figures out what your maximum profit is for a sheet initially of size A and B. (You can assume A, B are also positive integers.) See hint on last page if you are stuck getting started.

   (a) **[9 points]** Please provide pseudocode for a dynamic programming algorithm that outputs your maximum profit. (Note: your code doesn't have to output how you should actually divide the piece, just the profit.) You can assume you are given 3 arrays of size n: An array x containing the values of x_i, an array y containing the values of y_i and an array v containing the values of v_i.

   (b) **[11 points]** Prove your algorithm is correct.

   (c) **[6 points]** What is the runtime of your algorithm?

   (d) **[6 points]** Explain briefly how you would modify your algorithm to tell you whether you should divide the current sheet, and if so, where you should make the cut.

   (e) **[6 points]** (Challenge) The most straightforward algorithm (the one you probably came up with), does not have optimal time complexity. Describe how you could modify your algorithm to run in $O(\max\{n, AB \times \max\{A, B\}\})$ time.

- Hint on problem 1.c.vi: For some function $f(i,j,k)$,

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} f(i,j,k) = \sum_{i=1}^{k-1}\sum_{j=i+1}^{k} f(i,j,k) + \sum_{i=1}^{k-1}\sum_{j=k+1}^{n} f(i,j,k) + \sum_{i=k}^{n-1}\sum_{j=i+1}^{n} f(i,j,k). \quad (1)$$

- Hint on problem 2. Previous dynamic programming algorithms we have considered have only had two options for the optimal solution. For example, either item n is in the optimal solution or it is not. In this problem, there are many more options for the optimal solution. The options correspond to all of the places where we could make a horizontal or vertical cut (or we also have the option of not making a cut and selling our current sheet). Before, we just had to consider a maximum of two options. Now we will have to consider a maximum of many options.