Introduction to Algorithms	**Problem Set 5**
CS 4820, Spring 2019	**Due 11:59pm Thursday, March 7**

Hand in your solutions electronically using CMS. Each solution should be submitted as a separate file. Collaboration is encouraged while solving the problems, but:

1. **list the names of those with whom you collaborated;**
2. **you must write up the solutions in your own words;**
3. **you must write your own code.**

Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.

(1) *(15 points)* Let $X_1, X_2, \ldots, X_k$ be a independent random variables taking values in the range $\{0, 1, \ldots, n-1\}$. Their sum $X = X_1 + \cdots + X_k$ takes values in the range $\{0, \ldots, k(n-1)\}$. Suppose we are given an input that specifies:

1. **the distribution of each** X_i, expressed in the form of a two-dimensional array P such that $P[i, j]$ denotes the probability that $X_i = j$.
2. **an interval** $[a, b]$, such that $0 \leq a \leq b \leq k(n-1)$.

Given this data, we are interested in computing the probability that $a \leq X \leq b$.

(1a) *(5 points)* Design a dynamic programming algorithm that computes, for every pair i, j, the quantity $q_{ij} = \Pr(X_1 + \cdots + X_i = j)$, and then outputs the sum $\sum_{j=a}^{b} q_{kj}$.

You may omit the proof of correctness, but you should analyze the running time of this dynamic programming algorithm.

(1b) *(10 points)* Design an algorithm that computes $\Pr(a \leq X \leq b)$ in time $O(k\, n \log(k) \log(kn))$.

For this part of the problem, include both the running time analysis and the proof of correctness.

HINT: If Y and Z are independent random variables taking values in $\{0, 1, \ldots, n-1\}$, show that the probability distribution of their sum $Y + Z$ can be computed as the convolution of two vectors representing the probability distributions of Y and of Z.

(2) *(15 points)* Given as input a list of n points $L = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ on the real plane, your task is to compute the largest rectangle (in terms of area) that can be formed by selecting two points from L, one representing the bottom-left vertex of the rectangle and the other representing the top-right vertex. For simplicity, assume that all the a_i's and b_i's are distinct real numbers.

(2a) *(3 points)* Define two lists of points BL and TR in the following way:

$$BL = \{(a_i, b_i) \in L : \text{for any } j \neq i, \text{ either } a_i < a_j \text{ or } b_i < b_j\}$$

and

$$TR = \{(a_i, b_i) \in L : \text{for any } j \neq i, \text{ either } a_i > a_j \text{ or } b_i > b_j\}.$$

Prove that there exists a rectangle with largest area (using points from L) that has its bottom-left vertex in BL and top-right vertex in TR. Provide an $O(n \log n)$ time algorithm to compute BL and TR. You must output each of the two lists BL and TR by sorting them according to the x-coordinates of the points (in increasing order). **You don't have to provide proof of correctness of your algorithms. You do have to analyze run-time of the algorithms you provide.**

(2b) *(2 points)* Let (a_i, b_i) and (a_j, b_j) be points in BL such that $a_i < a_j$. Further, let (a_k, b_k) and (a_ℓ, b_ℓ) be points in TR such that $a_k < a_\ell$. Define $\Delta_{e,f}$ to be the area of the rectangle using (a_e, b_e) as the bottom-left vertex and (a_f, b_f) as the top-right vertex, where $e \in \{i, j\}$ and $f \in \{k, l\}$. Prove that $\Delta_{i,k} + \Delta_{j,\ell} > \Delta_{i,\ell} + \Delta_{j,k}$.

(2c) *(10 points)* Design an algorithm that runs in time $O(n \log n)$ to compute the largest rectangle (in terms of area) that can be formed by selecting the bottom-left vertex from BL and the top-right vertex from TR. The output of the algorithm should be the area of the largest rectangle.