CS648 : Randomized Algorithms
CSE, IIT Kanpur

Assignment 4

Deadline : 6:00 PM, 10th November

Important Guidelines:

1. It is only through the assignments that one learns the most in any course on algorithms. You are advised to refrain from searching for a solution on the net or from a notebook or from other fellow students. **If you are cheating the instructor, you are cheating yourself first**. The onus of learning from a course lies first on you and then on the quality of teaching of the instructor. So act wisely while working on this assignment.
2. Handwritten submissions will not be accepted. You must type your solution using any word processor (For example : Latex, Microsoft Word, Google Doc). You will have to upload the soft copy on moodle before the deadline. You will also have to submit its printed copy before the deadline.
3. The answer of each questions must be formal, complete, and to the point. Do not waste your time writing intuition or idea. There will be penalty if you provide any such unnecessary details.
4. This assignment carries 150 marks. Each problem carries 50 marks.

1 Internalizing Min-cut Algorithm

The aim of this question is to help you fully internalize the algorithm for min-cut we discussed in the class. A basic operation in the algorithm is *contract*$(e, \mathcal{G})$ that takes a multigraph $\mathcal{G}$ and an edge e as input and merges the two endpoints of e into a single vertex. It was assumed in the $O(n^2 \log^3 n)$ time algorithm that this operation takes time of the order of number of vertices in $\mathcal{G}$. It was because of this assumption only that we realized that as the algorithm proceeds, the time complexity of *contract* operation decreases and hence we can afford to carry out multiple parallel execution of the algorithm to boost the success probability.

1. Provide complete details of the *contract*$(e, \mathcal{G})$ implementation that runs in time of the order of the number of vertices. (Hints are given on the last page of this assignment.)

2. In the $O(n^2 \log^3 n)$ time algorithm, we carry out contraction till the number of vertices get reduced to $n/\sqrt{2}$ and then invoke two independent recursive calls. What would have happened if we had carried our the contraction till the number of vertices get reduced to $n/2$. Provide complete details of the analysis.

2 A surprising problem from computational geometry

There are n line segments in a plane. None of them intersect each other and no two of them are parallel to each other. We arrange them in a list and carry out the following algorithm.

- Pick 1st line segment. Extend its left end (likewise its right end) to infinity. In doing so, many other line segments might get intersected.
- Pick 2nd line segment. Extend its left end (likewise its right end) until it either reaches infinity or hits some previous extended segment.
- Pick 3rd line segment. Extend its left end (likewise its right end) until it either reaches infinity or hits some previous extended segment.
- ... continue till nth segment ...

You have to establish the following.

1. There exists a set of n segments and a permutation for which the above algorithm will cause $\binom{n}{2}$ points of intersection.

2. For any set of n line segments, if we permute them randomly uniformly and then execute the above algorithm, the expected number of points of intersection will be $O(n \log n)$. Isn't it surprising ?

3 Internalizing Backward Analysis and Randomized Incremental Construction (RIC)

We discussed 2 computational geometry problems in the class and provided an efficient RIC based algorithm for them. One important data structure that facilitates efficient implementation of a RIC based algorithm is the conflict graph. You might like to revisit the conflict graph used in the algorithm for convex hull to see how this data structure helped in performing $(i+1)$th step of the algorithm efficiently. At a high level, the conflict graph can be described as follows.

Let $\mathcal{O} = \langle o_1, o_2, \ldots, o_n \rangle$ be the sequence of n geometric objects and $S(\mathcal{O})$ denote the computational geometry structure to be constructed. We build this geometric structure incrementally. Let S_i denote the structure built at the end of ith insertion. The conflict graph $G(i)$ consists of the objects of the geometric structure S_i (cones in the case of convex hull) and the objects $\{o_{i+1}, \ldots o_n\}$ (the points to be inserted in the case of convex hull) so as to facilitate the following operations efficiently.

1. *Locating* the object o_{i+1} efficiently in the geometric structure S_i.
2. Determining whether S_{i+1} will be different from S_i.
3. Updating the structure $S(i+1)$ if necessary.
4. Updating the graph $G(i+1)$ from $G(i)$.

The aim of this exercise is to help you fully internalize the conflict graph used in RIC technique and the backward analysis. We consider the problem of non-dominated points in a plane. Let $P = \{(x_i, y_i) | 1 \leq i \leq n\}$ be a set of n points in plane. Assume all coordinates are different. A point (x_i, y_i) is said to dominate another point (x_j, y_j) if $x_i > x_j$ and $y_i > y_j$. A point in P is said to be a non-dominated point if there is no point in P which dominates it. See Figure 1 for a better understanding of non-dominated points.

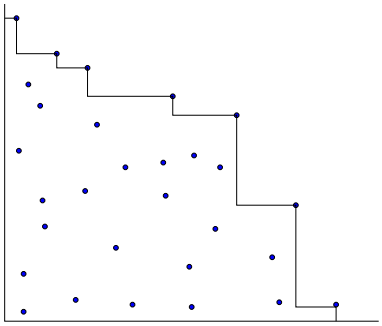

Figure 1: The non-dominated points appear in a stair-case patterns

You have to design an expected $O(n \log n)$ time randomized algorithm to compute all non-dominated points in P. The algorithm has to be based on *Randomized Incremental Construction* technique. Provide complete details of the conflict graph data structure used for this problem. Provide the detail of each incremental step and analyze its time complexity using backward analysis only.

Note: This problem has many efficient determinstic algorithms. However, the aim of this problem in the assignment is to test your skills of randomized incremental construction and conflict graph data structure. In order to realize the true power of RIC and backward analysis, try to solve the trapezoidal decomposition problem during your summer break. How will its conflict graph look like ?

Hint for Problem 1

1. Is there any need to keep each edge of the multigraph explicitly ?
2. Think of a representation of the multigraph as a simple graph with suitable weights on the edges.
3. Suppose there are wooden bars each of integral length only. Moreover, the total length of these bars is less than n^c for some fixed constant c. Our aim is to select one bar randomly out of all these bars. However, the probability of picking the bar depends linearly on their length. How will you select a bar randomly considering this constraint in $O(n)$ time ? Provide complete details. [You might like to exploit the features of the underlying word-ram model of computation.]
4. Think of a suitable augmentation of the Adjacency list of the graph. You may also invent another implementation of the Adjacency list as well.