CS648 : Randomized Algorithms
Semester I, 2011-12, CSE, IIT Kanpur

Backward Analysis and Randomized Incremental Construction

1 Introduction

Backward analysis is a very powerful tool for analysing many randomized algorithms. In particular, it plays a very key role in the analysis of those algorithms which are based on randomized incremental construction. To provide a better understanding of backward analysis, we consider a simple and non-algorithmic puzzle. In the subsequent sections of this note, we shall discuss the technique of randomized incremental construction through a well known problem in computational geometry.

2 A motivational puzzle

There is a huge circle of wire hanging in the air at a huge height above the ground. The circumference of the circle is D. There are n pairs of birds. Each of these n pairs has formed a nest at any arbitrary (not random) location on this circle. See Figure 1.

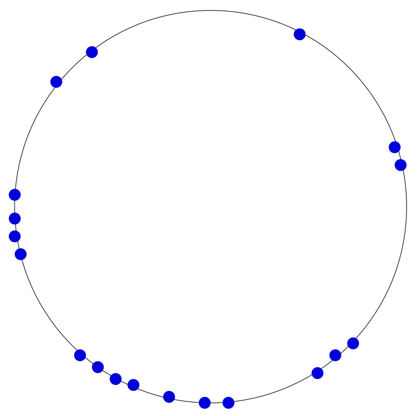

Figure 1: The location of nests (dots) on the circle is arbitrary, and not necessarily random.

On some stormy night, all pairs of birds flew away to a nearby forest. However, each pair was still together even in such moment of despair. Unfortunately, all the nests fell down in that night. In subsequent days the birds start returning to the circle to find status of their nests. Every day one pair of birds returns to the location where their nest was located. The first pair of birds arrives, they place a flag at their location of nest, and migrate to some other safer city together. Each subsequent pair of birds does the following. On returning to their earlier location of nest, they place a flag at that location. Then they start walking in the opposite directions along the circle until each of them finds a flag (see Figure 2). After this they also migrate to some other city with a heavy heart but still together. We are interested in the total distance travelled by all the birds along the circle. As a warm up, do the following exercise.

Exercise 2.1 *Construct an initial arrangement of nests on the circle and an ordering in which the pairs return so that total distance travelled is of the order of Dn.*

Now suppose the pairs return to the circle in a uniformly random order. What will be the expected total distance travelled ? This is the puzzle we wish to solve. Take a break for a few minutes to understand the complexity of this puzzle. Can you solve this puzzle effectively using the tools you have already studied ?

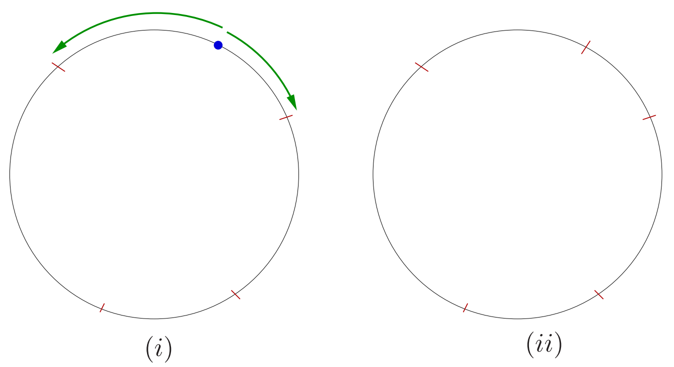

Figure 2: (i) the pair of birds (dot) travel in either directions until reaching the nearest flags, (ii) the five flags located on the circle after fifth pair of birds leaves.

Let X be the random variable for the total distance travelled by all the birds in this experiment. Let X_i be the random variable for the distance travelled by the pair of birds which arrives on ith day. Notice that $X = \sum_i X_i$. So by linearity of expectation

$$\mathbf{E}[X] = \sum_i \mathbf{E}[X_i]$$

Notice that $X_1 = 0$, $X_2 = D$. Let us focus on expected value of X_i for some $i > 2$, that is, the distance travelled by the pair of birds which arrive on ith day.

Partitioning of sample space

Let us revisit the theorem regarding the partition of sample space in the given context of finding $\mathbf{E}[X_i]$.

For any given $j > 2$, let $\mathcal{A}_j$ denotes the set of all subsets of j birds. For any subset $s \in \mathcal{A}_j$, let $\mathcal{E}_s$ be the event that s is the pairs of birds which arrive on first $i-1$ days. It can be observed that $\{\mathcal{E}_s | s \in \mathcal{A}_j\}$ constitute the partition of the sample space underlying this experiment.

2.1 Forward Analysis : a natural but ineffective approach to calculate $\mathbf{E}[X_i]$

The distance travelled on ith day depends upon the pair of birds which arrived till day $i-1$ and the pair of birds which arrives on the ith day. This motivates us to analyse $\mathbf{E}[X_i]$ by conditioning on the set of birds which arrived on first $i-1$ days. In other words, we try to evaluate the distance travelled on ith day based on the status of the circle just before the arrival of the pair of birds for ith day. This approach, called forward analysis appears to be a very natural approach. So let us pursue it. Using partition theorem, it follows that

$$\mathbf{E}[X_i] = \sum_{s \in \mathcal{A}_{i-1}} \mathbf{E}[X_i | \mathcal{E}_s] \cdot \mathbf{P}[\mathcal{E}_s]$$

The above equation is undoubtedly correct, but it can't be used to calculate $\mathbf{E}[X_i]$. This is because the conditional expectation $\mathbf{E}[X_i|\mathcal{E}_s]$ depends upon event $\mathcal{E}_s$, and therefore, there is no way to sum up the terms of right hand side of the above equation effectively (Convince yourself before proceeding further). So the forward analysis, which is a natural approach, turns out to be very ineffective to calculate $\mathbf{E}[X_i]$.

2.2 Backward Analysis : an effective approach to calculate $\mathbf{E}[X_i]$

Here is another approach to calculate $\mathbf{E}[X_i]$. We examine the status of the circle on the evening of ith day. So we know what were the birds which arrived till ith day. Now we try to analyse the expected distance travelled during ith day. This approach is called backward analysis since we are evaluating X_i in a backward way: From the status of the circle at the end of ith day we want to know what expected distance would have been travelled during the ith day. (Pause for a few minutes to see the fundamental

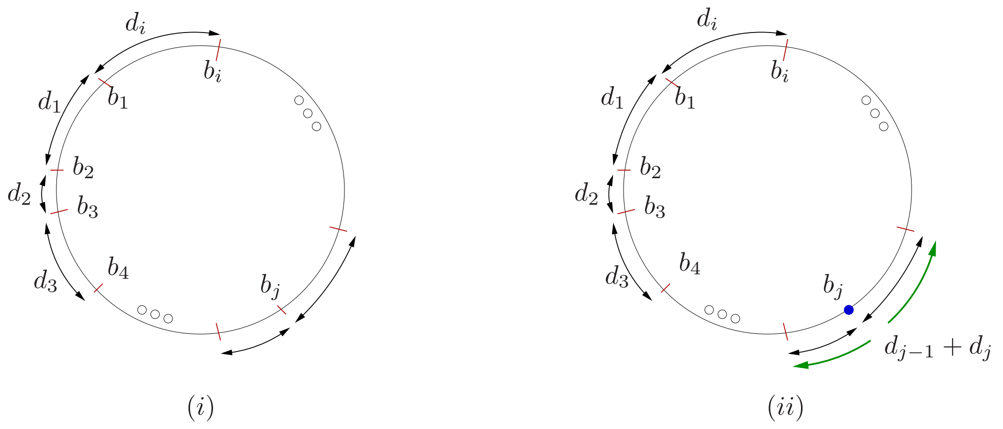

Figure 3: (i) the flags placed by set s of i pairs of birds partition the circle into i intervals, (ii) the distance travelled by bird b_j is $d_{j-1}+d_j$ if it arrives at number i.

difference between the backward approach and the forward approach). Let us see if this approach turns out to be helpful.

Firstly, observe that, along similar lines of the forward analysis, we can express $\mathbf{E}[X]$ as follows.

$$\mathbf{E}[X_i] = \sum_{s\in\mathcal{A}_i} \mathbf{E}[X_i|\mathcal{E}_s] \;\cdot\; \mathbf{P}[\mathcal{E}_s] \tag{1}$$

Let us see if we can calculate $\mathbf{E}[X_i|\mathcal{E}_s]$. (take a pause to see what it mean). The happening of event $\mathcal{E}_s$ means that we are given the locations of flags placed by the set s of i bird-pairs. Starting from any location, let us number each of these locations from b_1 to b_j in anti-clockwise order (see Figure 3(i)). Notice that all these flag positions partition the circle into i intervals numbered from d_1 to d_i, hence $\sum_j d_j = D$. Consider the flag located at location b_j.

Question 2.1 *Conditioned on the event $\mathcal{E}_s$, what is the probability that the bird-pair corresponding to the location b_j arrived on ith day ?*

The answer is $1/i$ since, given that $\mathcal{E}_s$ happened, each pair from the set s is equally likely to arrive on ith day (recall that birds arrive at the circle in a uniformly random order). So if the pair of birds corresponding to location b_j arrived on ith day, how much distance was travelled on ith day. See Figure 3(ii). This distance is $d_{j-1}+d_j$. Hence,

$$\begin{aligned}
\mathbf{E}[X_i|\mathcal{E}_s] &= \left(\frac{d_i+d_1}{i} + \frac{d_1+d_2}{i} + \cdots \frac{d_{i-1}+d_i}{i}\right)\\
&= \sum_{j=1}^{i} \frac{d_{j-1}+d_j}{i}\\
&= \frac{2}{i}\sum_{j=1}^{i} d_j \quad = \frac{2}{i}\cdot D
\end{aligned}$$

Notice that $\mathbf{E}[X_i|\mathcal{E}_s]$ turns out to be independent of $\mathcal{E}_s$. Hence plugging this value in Equation 1, we get

$$\mathbf{E}[X_i] = \sum_{s\in\mathcal{A}_i} \frac{2D}{i}\mathbf{P}[\mathcal{E}_s] = \frac{2D}{i}\sum_{s\in\mathcal{A}_i}\mathbf{P}[\mathcal{E}_s] = \frac{2D}{i}\cdot 1$$

Hence the expected distance travelled by the birds is:

$$\mathbf{E}[X] = \sum_{i=2}^{n}\mathbf{E}[X_i] = 2D\sum_{i=2}^{n}\frac{1}{i} = \Theta(D\ln n).$$

Ponder over the above solution from various perspective. Can you appreciate the simplicity and power of backward analysis ? We shall see many more nontrivial application of this tool in this course.

3 Randomized Incremental Construction (RIC)

Incremental construction is an algorithmic technique which solves a given problem incrementally. For example, insertion sort is a simple sorting algorithm based on this technique where we start from an empty set and add elements successively to it while maintaining the (increasing/decreasing) order among the elements in the set at every stage. In general an incremental algorithm works as follows : Let $o_1, o_2, \cdots, o_n$ be n input elements and objective is to compute some function $f(\{o_1, o_2, \cdots, o_n\})$. We start with empty set S and $f(\phi)$, add elements successively and maintain $f(S)$ at every stage. So there will be n stages in the algorithms and finally we shall have the result. Efficiency of an incremental algorithm depends upon the amount of time spent at every stage. In various problems it turns out that although the worst case time spent at a stage may be much larger but the probability of such cases to happen is very small. In such cases, it is useful if we increment the set S by picking elements randomly. *Randomized incremental construction*(RIC) is a technique where we add elements to set S randomly. We introduced RIC by solving two problems. The first problem is finding closest pair among n points in two dimensions. We discussed an extremely simple and expected $O(n)$ time algorithm based on RIC technique to solve this problem. The second problem is computing convex hull of a set of n points in two dimensions. We omit the latter problem from this note since it was explained with many details in the class.

3.1 Closest Pair Problem in Two Dimensions

The problem is defined as follows :

Given a set P, consisting of n points in two dimensions. Objective is to find a closest pair from the set.

We assume that each point of set P is expressed as an ordered pair of two LONG integers. This assumption hold for most of the applications of computational geometry whose input is a set of points.

We permute the sequence of points uniformly randomly. Let p_i denote the ith point in this sequence and let S_i denote the set of first i points in this sequence. Let d_i be the distance between the closest pair of points in S_i. We initialize S_2 by placing the first two points of the sequence and initialize d_2 as the distance between them. We then run a for loop from $i = 3$ to n. In the beginning of i^{th} iteration, we know the closest pair distance d_{i-1}. In i^{th} iteration, we add point p_i and compute d_i as follows.

We compute the minimum distance between p_i and any point from S_{i-1}; let it be d'. If $d' < d_{i-1}$ then $d_i = d'$ else d_i is the same as d_{i-1}.

Like in any incremental algorithm, the time required to compute d_i governs the time complexity of the algorithm. Therefore, we require a data structure which can help computing the closest neighbor of p_i in an efficient manner (better than the brute force method of $\Theta(i)$ time for computing distance between p_i and all $i-1$ points of set S_{i-1}). We now describe this data structure $D(S_i)$

At the end of i^{th} iteration, the $D(S_i)$ maintains a *virtual* gird in 2-dimensions each of whose cell has length d_i and it stores the points of S_i in respective cells of the grid. Note that a cell of grid can contain at most 4 points (otherwise it would violate the fact that d_i is the closest pair distance among the points of S_i). $D(S_i)$ will support the each of the following operations in $O(1)$ time.

- Given a point, determine the cell of the grid to which the point belongs.
- Reporting all the points (at most four in number) belonging to a given cell.
- Inserting a point in to appropriate cell in the grid.

Do the following exercise using some topic which we have covered in this course.

Exercise 3.1 *Give the design and implementation of data structure D with the above functionalities such that for a given set of i points, it occupies $O(i)$ size and achieves expected $O(i)$ construction time.*

Now we can formally describe the randomized incremental computation for computing the closest pair distance in **Algorithm 1**.

i^{th} iteration of the Above algorithm i.e. addition of p_i and computing d_i has been depicted in Figure 4.

We would like to state the following simple observation about the algorithm

Observation 3.1 *$d_i < d_{i-1}$ only if p_i is one of the two points which define the closest pair among the first i points in the permutation.*

```
Let $\langle p_1, \ldots, p_n \rangle$ be a uniformly random permutation of given points;
$d_2 \leftarrow distance(p_1, p_2)$;
for $i = 3$ to $n$ do
    Step 1 : locate the cell of the grid to which the point $p_i$ belongs;
    Step 2 : find the point $p \in S_{i-1}$ closest to $p_i$ ( $d'$ be the distance between $p$ and $p_i$);
    Step 3 :
    if $d' \geq d_{i-1}$ then
        Insert($p_i$, $D(S_{i-1})$)
    else
        $d_i \leftarrow d'$, Build $D(S_i)$
return $d_n$;
```

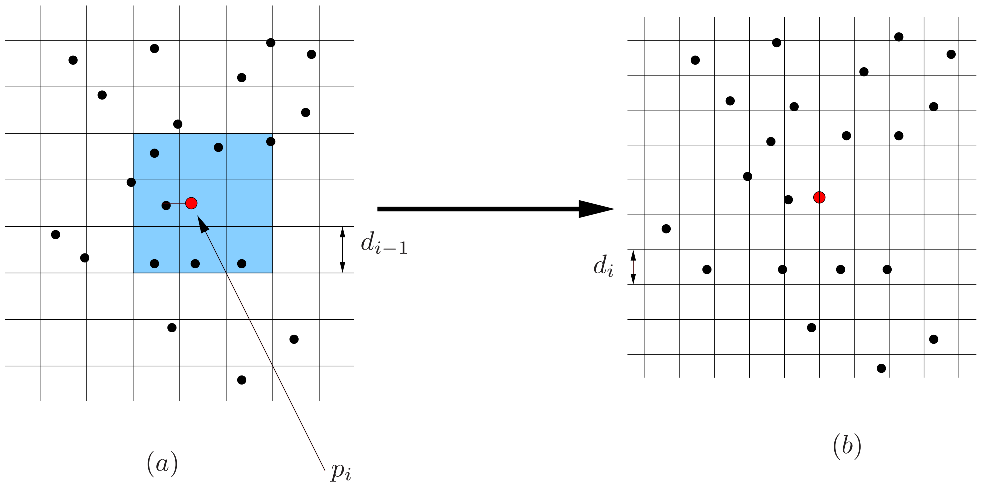

Figure 4: (a) the grid in the beginning of i^{th} iteration, with point p_i shown red, (b) the grid at the end of i^{th} iteration.

3.1.1 Time complexity of algorithm

The algorithm executes $n-2$ iterations. Consider i^{th} iteration wherein we introduce point p_i and compute d_i. Here *Step 1* and *Step 2* take deterministic $O(1)$ time. But the time spent in *Step 3* is a random variable. Let us denote it by X_i. It can be seen that X_i takes just a constant, say c_1, time if $d_i = d_{i-1}$ and takes additional $c_2 i$ time (due to building of $D(S_i)$ if $d_i < d_{i-1}$. Hence,

$$\mathbf{E}[X_i] = c_1 + \mathbf{P}[d_i < d_{i-1}] \cdot c_2 i$$

So our objective is to calculate probability of event "$d_i < d_{i-1}$". Observe that happening of event "$d_i < d_{i-1}$" is governed by the first $i-1$ points of the permutation and the point p_i we add in the i^{th} iteration. This suggests us to analyse $\mathbf{P}[d_i < d_{i-1}]$ in terms of the status of the algorithm before **the beginning** of i^{th} iteration. In other words, we are given the grid defined by first $i-1$ points (see Figure 4(a) without the red point p_i) and we calculate the probability of "$d_i < d_{i-1}$" conditioned on this grid. This is a *forward analysis* approach and appears to be very natural. Pursuing this approach, we can express "$\mathbf{P}[d_i < d_{i-1}]$" as a standard application of partition theorem as follows. Let $\mathcal{A}_j$ denote the set of all subsets of j points, and for any $s \in \mathcal{A}_j$, $\mathcal{E}_s$ denote the event that s is the set of first j points in the permutation.

$$\mathbf{P}[d_i < d_{i-1}] = \sum_{s \in \mathcal{A}_{i-1}} \mathbf{P}[d_i < d_{i-1} | \mathcal{E}_s] \cdot \mathbf{P}[\mathcal{E}_s]$$

Spend some time on the above equation and convince yourself that this equation is ineffective to calculate $\mathbf{P}[d_i < d_{i-1}]$. Also realize that the reason behind this ineffectiveness is the dependence of the conditional probability $\mathbf{P}[d_i < d_{i-1} | \mathcal{E}_s]$ on s for any $s \in \mathcal{A}_{i-1}$.

Let us now pursue backward analysis to calculate $\mathbf{P}[d_i < d_{i-1}]$. We start with the status of the algorithm at the end of i^{th} iteration. In other words, we have been given the grid defined by a set s of first i

points (as shown in Figure 4(b)) and we have to calculate the conditional probability $\mathbf{P}[d_i < d_{i-1}|\mathcal{E}_s]$. Let (q, q') be the closest pair of points in the grid. It follows from Observation 3.1 that for event "$d_i < d_{i-1}$", the point p_i must be either q or q'. Notice that each of the i points in set s is equally likely to be the point p_i. Hence

$$\mathbf{P}[d_i < d_{i-1}|\mathcal{E}_s] = 2 \cdot \frac{1}{i}$$

Notice that unlike the forward analysis, the conditional probability $\mathbf{P}[d_i < d_{i-1}|\mathcal{E}_s]$ for any $s \in \mathcal{A}_i$ turns out to be independent of s. (this is the power of backward analysis). Once again using the standard partition theorem, we get

$$\begin{aligned}\mathbf{P}[d_i < d_{i-1}] &= \sum_{s\in\mathcal{A}_i} \mathbf{P}[d_i < d_{i-1}|\mathcal{E}_s] \cdot \mathbf{P}[\mathcal{E}_s] \\ &= \sum_{s\in\mathcal{A}_i} \frac{2}{i} \cdot \mathbf{P}[\mathcal{E}_s] \\ &= \frac{2}{i} \sum_{s\in\mathcal{A}_i} \mathbf{P}[\mathcal{E}_s] = \frac{2}{i} \cdot 1\end{aligned}$$

So the expected time complexity of *Step 3* in i^{th} iteration is

$$\mathbf{E}[X_i] = c_1 + \frac{2}{i} \cdot c_2 i = c_1 + 2c_2 = O(1)$$

Hence the expected time complexity of i^{th} iteration is $O(1)$. So the expected running time of the algorithm, which basically executes $n - 2$ iterations, is $O(n)$.

Theorem 3.1 *There is an expected $O(n)$ time algorithm to compute closest pair of points from a set of n points in plane.*