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

Expected distances in a complete graph with random edge weights

Let K_n denote a complete graph on a set V of n vertices. So there is an edge between any two vertices and hence a total of $\binom{n}{2}$ edges. Suppose each edge is assigned a random weight which is uniformly and independently distributed in the interval $[0, 1]$. Let $\delta_{u,v}$ denote the distance between vertices u and v in the graph. Clearly $\delta_{0,1}$ is a random variable with value in the interval $[0, 1]$. What can we say about the expected value of $\delta_{u,v}$?

Spend a few minutes on just understanding the nontriviality of this problem. What answer would you expect ? And more importantly, how simple or complex would be the way to arrive at that answer ? Without much delay, we would like to state that

$$\mathbf{E}[\delta_{u,v}] = \frac{\ln n}{n}$$

where $\ln n$ is the natural logarithm (with base e).

The overall analysis to prove the above equality is quite insightful and calculation-free. It uses the following three tools.

1. portioning of experiment into stages.
2. simple observations about a uniformly distributed random variable ?
3. principle of deferred decision.

The last tool, principle of deferred decision is a very useful tool in analysing complex randomized experiment. The reader will understand its applicability and importance through the elegant solution for the problem described above. There is a problem in assignment 4 where you would like to apply this tool as well.

Henceforth, we shall focus on a fixed vertex $s \in V$ called source vertex henceforth. Let $s(= v_0), v_1, \cdots, v_i$ be i nearest vertices from s. Let X_i denote the distance from s to v_i. Obviously $X_0 = 0$. We shall calculate $\mathbf{E}[X_i]$ for each $i < n$.

Question 0.1 *How is* $\mathbf{E}[\delta_{u,v}]$ *related to* $\mathbf{E}[X_i]$*'s ?*

1 Notations and elementary observations

Let $U_{a,b}$ be a continuous random variable uniformly distributed in the range $[a, b]$. You need not know the theory of continuous random variable to understand $U_{a,b}$? We just need to be aware of the following basic and intuitively obvious description of $U_{a,b}$.

$$\text{For any interval } \Delta \subseteq [a, b], \qquad \mathbf{P}[U_{a,b} \in \Delta] = \frac{|\Delta|}{b - a}$$

We now state a couple of very simple but useful observations about the uniform probability distribution. The following observation can be easily proved using direct application of the definition of conditional probability (do it as an exercise).

Observation 1.1 *If it is given that the random variable $U_{a,b}$ is greater than some constant c, $a < c < b$, then it is uniformly distributed in the interval $[c, b]$. In other words,*

$$\mathbf{P}[U_{a,b} \in \Delta | U_{a,b} > c] = \frac{|\Delta|}{b-c}$$

The following observation suggests that we just need to study random variables $U_{0,b}$.

Observation 1.2 *The random variable $U_{a,b}$ has same probability distribution as the random variable $(a + U_{0,b-a})$.*

We shall also use the following lemma which we earlier proved in the class (see the Lecture notes on the problem of estimating the size of transitive closure of a directed graph).

Lemma 1.1 *If there are ℓ random variables distributed uniformly and independently in the interval $[0, 1]$, the expected value of the smallest of them is $\frac{1}{\ell+1}$.*

We shall use the following corollary of Lemma 1.1 in our analysis.

Corollary 1.1 *If there are ℓ random variables, where jth random variable is distributed uniformly and independently in the interval $[0, 1 - \Delta_j]$, the expected value of the smallest of them is $\frac{1}{\ell+1}$.*

As a warm up, what can we say about X_1, that is, the distance between s and its nearest vertex in K_n? The path from s to the nearest vertex of s must be an edge between the two (give reason). Hence $\mathbf{E}[X_1]$ is the expected weight of the least weight edge incident on s which using Lemma 1.1 is equal to $\frac{1}{(n-1)+1}$. What can we say about $\mathbf{E}[X_2]$, the expected distance between s and second nearest vertex ? In general, what is $\mathbf{E}[X_i]$ for some $2 \leq i < n$? To calculate it, it makes sense to explore relationship between X_i and X_j's for $j < i$. So we revisit the Dijkstra's algorithm for single source shortest paths problem in the following section.

2 Dijkstra's algorithm

The structure of shortest paths and distances from s are captured quite precisely by Dijkstra's algorithm. In particular, it shows that there is a tree rooted at s such that the paths from s to each vertex in this tree is the shortest path to that vertex. This tree is called shortest path tree rooted at s. The Dijkstra's algorithm performs $n - 1$ steps. In the beginning of $(i + 1)$th step, we have shortest path tree T_i storing i nearest vertices and the algorithm computes $(i + 1)$th nearest vertex. T_0 is the singleton vertex s. For computing v_{i+1} and its distance, the algorithm makes use of the following lemma (prove it if you are seeing it for the first time).

Lemma 2.1 *The shortest path from s to v_{i+1} is of the form : the shortest path $P(s, v_j)$ for some $j \leq i$ followed by edge (v_j, v_{i+1}). See Figure 1.*

So for a given instance of tree T_i, the value of X_{i+1} is

$$\min_{j \leq i, x \in V \setminus T_i} \delta(s, v_j) + \omega(v_j, x)$$

and the vertex x minimizing the above quantity is v_{i+1}.

The overview of Dijkstra's algorithm described above prompts us to use the technique of partitioning an experiment into stages to calculate $\mathbf{E}[X_i]$'s. The transition from ith stage to $(i + 1)$th stage is the $(i+1)$th step of the algorithm as described above. Pursuing this approach we aim to calculate expectation of X_{i+1} conditioned on the values of $X_1, \ldots, X_i$. However, the naive way of analysing Dijkstra's algorithm will fail. The reader is strongly advised to ponder over this claim and proceed only after getting fully convinced.

To calculate the expected value of X_i, we shall analyse Dijkstra's algorithm with the help of the principle of deferred decision.

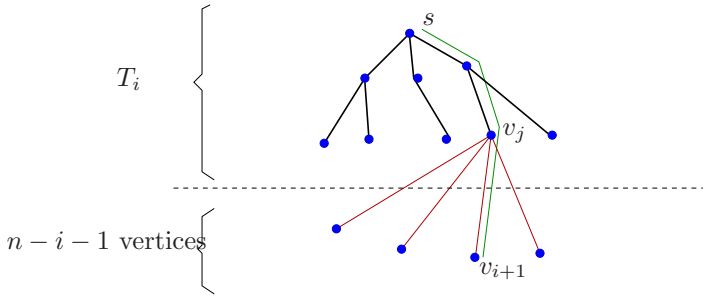

Figure 1: The shortest path tree storing i nearest vertices from s.

3 Dijkstra's algorithm + principle of deferred decision

We present a new implementation of Dijkstra's algorithm. This implementation performs $n-1$ steps such that during $(i+1)$st step it finds the value of X_{i+1}. The key point of this implementation where it differs from the usual implementation of Dijkstra's algorithm is the following. It exposes the weights of edges of the graph in a *lazy* fashion. Interestingly, there is a much simpler way to incorporate this feature in our implementation. Instead of the usual data structures, like adjacency list or adjacency matrix, we employ the following really simple structure, which can be called *stick* structure.

For each vertex x, visualize a stick of length 1 unit where we have placed x on the left end and all the $n-1$ edges are placed on this stick according to their length from the left end. See Figure 2.

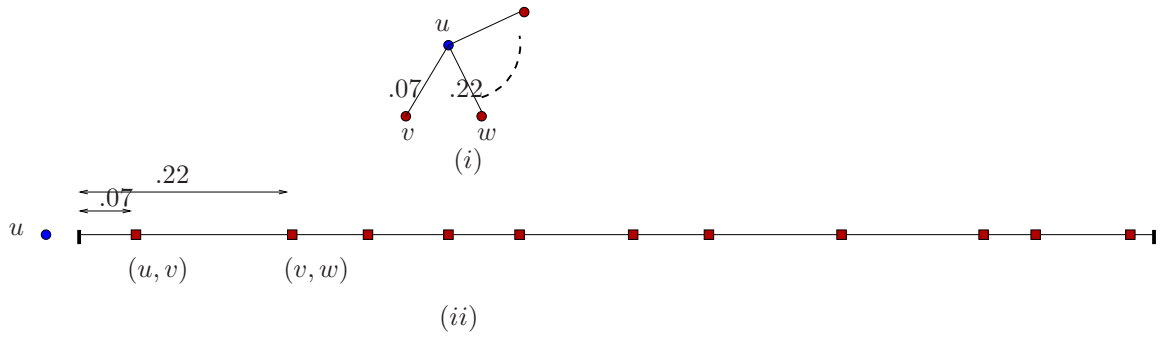

Figure 2: (i) the edges incident on a vertex u, (ii) the stick storing the edges incident on u according to their length

We now describe the first step of the algorithm. We have the stick associated with source s and we are on the left end of it. We are not aware of the positions on the stick of the edges incident on s (hence their length) at this stage. We start traversing the stick starting from the left end and stop as soon as we find the first edge. What is the other end point of this edge ? Well, it is exactly v_1. Now let us describe the second step to compute v_2. Before that, what can we say about v_2? It follows from Lemma 2.1 that the shortest path to v_2 is either an edge from s or it is the path from s to v_1 concatenated with an edge from v_1. In particular, if $\omega(s,v_2) < \delta(s,v_1) + \omega(v_1,v_2)$ then the shortest path to v_2 is direct edge from s, otherwise it is the shortest path to v_1 concatenated with (v_1,v_2). In order to find v_2 and its shortest path from s (i.e., distinguish between the two cases mentioned above), here is a very simple method. Introduce the stick of v_1 into the picture. Place it parallel to that of s but shifted by distance $\delta(s,v_1)$ to the right of s. Now imagine a vertical line passing through v_1. Move this line to the right until it finds some edge on either of the two sticks. This defines v_2. See Figure 3.

Let us describe $(i+1)$st step of this implementation of Dijkstra's algorithm. Before the beginning of this step, we have the following situation. We have computed distance to i nearest vertices: $v_1,\ldots,v_i$. We have $i+1$ sticks associated with $v_0,...,v_i$ placed parallel to each others with their left end aligned as follows. For each $j<i$, the left end of stick of v_{j+1} is shifted by distance $\delta(s,v_{j+1}) - \delta(v,j)$ to the right

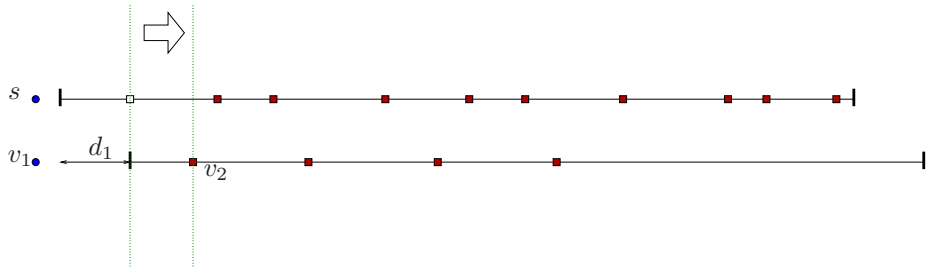

Figure 3: The step of computing v_2

of the stick of v_j. See Figure 4 for details.

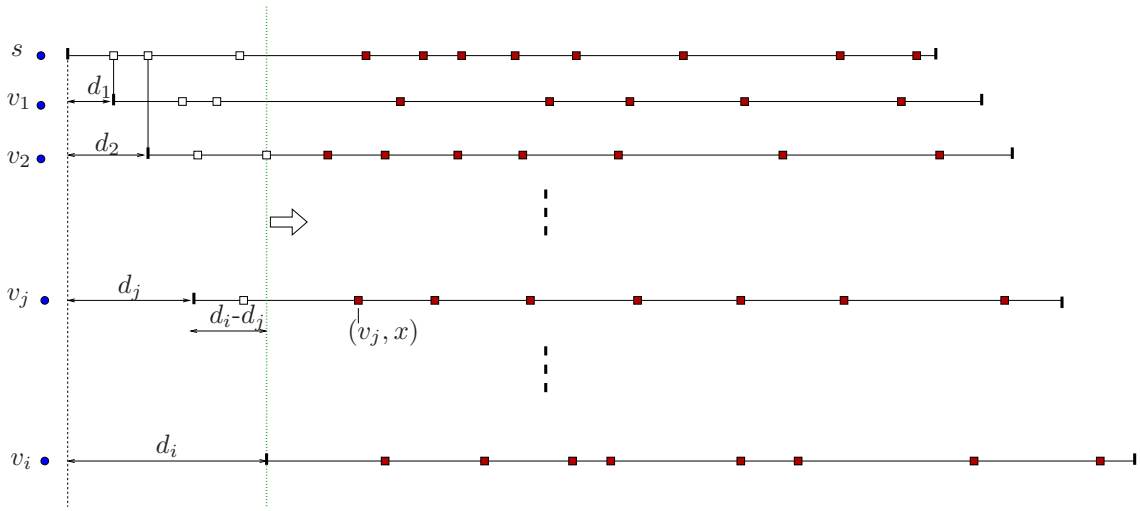

Figure 4: The snapshot of the algorithm when the vertical line (shown green) reaches v_i. The edges (shown white) to the left of the line are exposed whereas the edges (shown red) to the right of the line are still unexposed.

Recalling Lemma 2.1, the shortest path from s to $i+1$st nearest vertex must be a path from s to v_j concatenated with edge (v_j, x) for some $j \leq i$ and $x \in V \backslash V_i$. So there are potentially $(i+1)(n-i-1)$ edges which can define v_{i+1} and X_{i+1}. Which of these will be v_{i+1} and how do we find it out? For this objective, the set of $i+1$ sticks aligned as described above makes the task of finding v_{i+1} quite easy as follows. The vertical line passing through v_i is moved gradually to the right until it hits an edge joining some v_j and $x \in V \backslash T_i$. This edge defines v_{i+1} and let Y_{i+1} denote the random variable for the distance traversed by the vertical line from v_i to v_{i+1}. The following equality is obvious.

$$X_{i+1} = X_i + Y_{i+1} \tag{1}$$

Let us calculate the conditional expectation of Y_{i+1}. At the moment the vertical bar was passing through v_i, each $X_j = d_j$ for some arbitrary but fixed values d_j. But none of the $(i+1)(n-i-1)$ edges between T_i and $V \backslash T_i$ have been exposed till now. That is, we do not know their values yet. Therefore, in order to calculate $\mathbf{E}[Y_{i+1} | X_1 = d_1, \ldots, X_i = d_i]$, we need to know the probability distribution of the edges $\{(v_j, x) | j \leq i, x \in V \backslash V_i\}$ conditioned on $X_1 = d_1, \ldots, X_i = d_i$. For an edge (v_j, x), this conditioning implies that (v_j, x) has weight greater than $d_i - d_j$. Originally its weight was a random variable distributed uniformly in $[0, 1]$. So using Observation 1.1, the following lemma holds.

Lemma 3.1 *Conditioned on $X_1 = d_1, \ldots, X_i = d_i$, the length of edge (v_j, x) for any $j \leq i, x \in V \backslash T_i$ is a random variable distributed uniformly in the interval $[d_i - d_j, 1]$.*

To determine which of the $(i+1)(n-i-1)$ edges defines the random variable Y_{i+1}, we need to compare their distances from the vertical bar passing through v_i. Notice that the vertical bar is at distance $d_i - d_j$ to the right of the left-end of the stick associated with v_j. Hence, it follows from Lemma 3.1 that the distance of (v_j, x) from the vertical bar passing through v_i is a random variable distributed uniformly in the interval $[0, 1 - d_i + d_j]$. So conditioned on $X_1 = d_1, \ldots, X_i = d_i$, Y_{i+1} is the smallest of the $(i+1)(n-i-1)$ random variables each of which is distributed randomly uniformly and independently of each other in the range $[0, 1-\Delta]$ for some $\Delta \geq 0$. Hence, using Corollary 1.1,

$$\begin{aligned}
\mathbf{E}[Y_{i+1} | X_1 = d_1, \ldots, X_i = d_i] &\leq \frac{1}{(i+1)(n-i-1)+1} \\
&< \frac{1}{(i+1)(n-i-1)} \\
&= \frac{1}{n}\left(\frac{1}{i+1} + \frac{1}{n-i-1}\right)
\end{aligned}$$

An important point to be noted here is the following. We have got an upper bound on the conditional expectation of Y_{i+1} which is independent of the event we condition. Hence,

$$\mathbf{E}[Y_{i+1}] \leq \frac{1}{n}\left(\frac{1}{i+1} + \frac{1}{n-i-1}\right)$$

Using linearity of expectation, it follows that

$$\mathbf{E}[X_{i+1}] = \sum_{j=1}^{i+1} \mathbf{E}[Y_j] \leq \frac{1}{n}\sum_{j=1}^{i+1}\left(\frac{1}{j} + \frac{1}{n-j}\right)$$

We can thus conclude the following theorem.

Theorem 3.1 *Let K_n be a complete graph on $n = |V|$ vertices where each edge weight is distributed randomly uniformly in $[0, 1]$. For any vertex $s \in V$, expected distance to $(i+1)$th nearest vertex is*

$$\sum_{j=1}^{i+1} \frac{1}{n}\left(\frac{1}{j} + \frac{1}{n-j}\right)$$

An immediate corollary of the above theorem is that expected distance from s to the farthest vertex in K_n is at most $2\frac{\ln n}{n}$. As an exercise, use Theorem 3.1 to prove that the expected distance between two randomly selected vertices u, v in K_n is $\ln n/n$.

4 Summary of the principle of deferred decision

Let f be a function of n random variables. These n random variables may be initially chosen to be independent and assumed to have certain probability distribution in the beginning. Sometimes there is an algorithm which computes f. In such scenario, the naive and usual way of calculating the expected value of f will be to compute value of f for each possible input (assigning all possible values to the random variables), and then taking an average of these values. This approach usually fails to compute expected value of f. On the contrary, there is a lazy way of exposing the random variables during the execution where we defer revealing the exact value of a random variable until it is absolutely needed. Taking this approach turns out to be very effective in calculating the expected value of f. The reason why it is sometimes helpful is that this approach imposes the *least dependency* on the values of the unexposed random variables in terms of the values taken by the already exposed random variables. As a result, it is much easier to analyse a particular step of the algorithm in order to calculate the final value taken by f.

We would like to conclude with the following optional problem. The first group of students who solve this problem will get 10 marks.

Question 4.1 *Let $K_{n,n}$ be a complete bipartite graph on sets U, V of vertices where $n = |U| = |V|$. Each edge of the graph is assigned a weight which is selected randomly uniformly and independently from $[0, 1]$. It can be seen that there are $n!$ perfect matchings in this graph. Let M denote the perfect matching with the least total weight and let X be the random variable for the weight of M. Prove that $\mathbf{E}[X] = O(1)$.*