CS648 project

Team Binary Expectation
Aryan Bhardwaj | Sanat Goel | Tushaar Ranganathan
Expected Value of Weight of MST

2023-24 2nd Semester

1 Problem Description

What is the expected weight of the MST in a complete graph where each edge is assigned weight randomly uniformly in the range [0,1]?

2 Results

2.1 Result 1

Let n Independent Random variables $\mathrm{X}_1, X_2, \ldots, X_n \sim \mathrm{unif}(\alpha, 1)$

Then we will find the distribution of X_{min}

where $X_{min} = min(X_1, X_2, \ldots, X_n)$

We will find the CDF (cumulative distribution function) of X_{min}

Then we will use it to find PDF of X_{min}

After that we can use it get anything we like about $\mathrm{X}_{min} likeitsexpectedvalue$

The cumulative distribution function $F(t)$ of the random variable Y is defined as:

$$F(t) = \mathrm{Prob}(Y \leq t)$$

Proof:

$\mathrm{Prob}(\mathrm{X_{min}} > t) = \mathrm{Prob}(X_1 > t, X_2 > t, \ldots, X_n > t)$

$= \mathrm{Prob}(X_1 > t) \cdot \mathrm{Prob}(X_2 > t) \cdot \ldots \cdot \mathrm{Prob}(X_n > t)$ (Since they are Independant)

$= (\mathrm{Prob}(X_1 > t))^n = \left(\frac{1-t}{1-\alpha}\right)^n \quad t \in (\alpha, 1)$

$= \mathrm{Prob}(\mathrm{X_{min}} \leq t) = 1 - \left(\frac{1-t}{1-\alpha}\right)^n$

Let $F(t)$ be the CDF of X_{min} then

$F(t) = 1 - \left(\frac{1-t}{1-\alpha}\right)^n$

Let $f(t)$ denote the Probabilty distribution function of X_{min}

Then The probability density function $f(t)$ is defined as the derivative of the cumulative distribution function $F(t)$, denoted as $f(t) = F'(t)$.

$f(t) = \frac{n}{1-\alpha} \cdot \left(\frac{1-t}{1-\alpha}\right)^{n-1}$

The expected value $E[X_{\min}]$ of $X_{\min}$ can be found using the probability density function $f(t)$ as follows:

$$E[X_{\min}] = \int_{\alpha}^{1} t \cdot f(t)\, dt$$

$$= \int_{\alpha}^{1} t \cdot \frac{n}{1-\alpha} \cdot \left(\frac{1-t}{1-\alpha}\right)^{n-1} dt$$

$$= \frac{n}{(1-\alpha)^n} \cdot \int_{\alpha}^{1} t \cdot (1-t)^{n-1}\, dt$$

using the substitution $u = 1 - t$

$$= \frac{n}{(1-\alpha)^n} \cdot \int_{0}^{1-\alpha} (1-u) \cdot u^{n-1}\, du$$

$$= \frac{n}{(1-\alpha)^n} \cdot \int_{0}^{1-\alpha} (u^{n-1} - u^n)\, du$$

$$= \frac{n}{(1-\alpha)^n} \left(\frac{(1-\alpha)^n}{n} - \frac{(1-\alpha)^{n+1}}{n+1}\right)$$

which simplifies to

$$\frac{1 + n \cdot \alpha}{1+n}$$

Thus for the special case when $\alpha = 0$, We get the result the Expected value of minimum is $\frac{1}{n+1}$

We will use this result at various time

2.2 Result 2

Let n Independant Random variables $\mathrm{X}_1, X_2, \ldots, X_n \sim \mathrm{unif}(0, 1)$

The notation $X_{(i)}$ represents the ith smallest value among $X_1, X_2, \ldots, X_n$, denoted as $X_{(i)}$.

We will find the probability density function of $X_{(i)}$ then use integration to find the $E[X_{(i}]$

let $f(t)$ be the probability density function of $X_{(i)}$ then

$$f(t) = \binom{n}{i-1} \cdot \binom{n-i+1}{n-i} \cdot t^{i-1} \cdot (1-t)^{n-i}$$

$$f(t) = \frac{n!}{(i-1)! \cdot (n-i)!} \cdot t^{i-1} \cdot (1-t)^{n-i}$$

This result is correct since We are choosing $i-1$ Random variable and assigning them value between 0 and t and choosing $n-i$ from remaining $n-i+1$ random variables and assigning some value between t and 1

$$E[X_{(i)}] = \int_0^1 t \cdot f(t)\, dt$$

$$E[X_{(i)}] = \frac{n!}{(i-1)! \cdot (n-i)!} \cdot \int_0^1 t^i \cdot (1-t)^{n-i}\, dt$$

We can Solve this integral using Beta Function
According to which

$$\int_0^1 t^1 \cdot (1-t)^b\, dt = \frac{a! \cdot b!}{(a+b+1)!}$$

Hence

$$E[X_{(i)}] = \frac{n!}{(i-1)! \cdot (n-i)!} \cdot \frac{i! \cdot (n-i)!}{(n+1)!}$$

Therefore,

$$E[X_{(i)}] = \frac{i}{n+1}$$

2.3 Result 3 (Law of total Expecatation)

This result is called Law of Total Expectation
According to this result

$$E_X[X] = E_Y[E_{X|Y}[X|Y]]$$

This Result is useful in finding expectations when the distribution of a random variable X is dependent on some other Random Variable Y.

2.4 Corollary 1

Let X be a random variable with the expected value $E[X] = c < 1$, where c is some constant.

Let n independent random variables $Y_1, Y_2, \ldots, Y_n | X \sim \text{unif}(X, 1)$. Let $Y_{\min}$ be the random variable which denotes the distribution of the minimum value of Y_1 to Y_n. We can get an upper bound on $E[Y_{\min}]$ correctly by assuming the distribution of $Y_1, \ldots, Y_n \sim \text{unif}(X, 1)$.

Proof: By Result 1, the $E[Y_{\min} \mid X = \alpha] = \frac{1+n\alpha}{n+1}$.

Using Result 3 (Law of total expectation), we know

$$\begin{aligned} E[Y_{\min}] &= E_X[E_{Y|X}[Y_{\min} \mid X = X]] \\ &= E\left[\frac{1 + n \cdot X}{1 + n}\right] \\ &= \frac{1 + n \cdot E[x]}{1 + n} \\ &\leq \frac{1 + n \cdot c}{1 + n} \\ &\leq \frac{1 + n \cdot c}{1 + n} \end{aligned}$$

This is the same result which we would have got if we assumed the distribution of $Y_1, \ldots, Y_n \sim Unif(c, 1)$.

Hence Proved.

Note: We are not claiming that the actual distribution of Y_i is $Unif(c, 1)$. We just want to state that as long as we are interested in finding an upper bound for Expected value of Y_{min} we would get a correct result, if we assume then to be $Unif(c, 1)$.

Kruskal

We have $\binom{n}{2}$ edges, in the graph each following $unif(0, 1)$ distribution.

Suppose we sort all of these edges.

From now on we will refer to the rank in terms of smallest amongst all the edges in the graph as the "index" of the edge.

According to Kruskal's Algorithm, we go from smallest to largest edge and check whether the nodes corresponding to these edges belong to the same connected component or not, if they do not belong the same component, then we add that edge as the part of MST, and connect the component corresponding to these nodes, otherwise we simply ignore this edge, we keep on going until we have found $n - 1$ edges.

First of all, using Result 2, we have that the expected wt of edge with i^{th} index is $\frac{i}{\binom{n}{2}+1}$

Suppose the expected index of the k^{th} smallest edge in MST is e_k The Expected wt of the k^{th} smallest edge of MST is $\frac{e_k}{\binom{n}{2}+1}$ using Law of Total Expectation.

According to the notation of Result 3 here X denotes the weight of k^{th} smallest edge Y is the index of k^{th} smallest edge of MST.

$$E_X[X] = E_Y[E_{X|Y}[X|Y]]$$

In our case

$$E_X[X] = E_Y\left[\frac{Y}{\binom{n}{2}+1}\right]$$

$$E_X[X] = \frac{E_Y[Y]}{\binom{n}{2}+1}$$

$$E_X[X] = \frac{e_k}{\binom{n}{2}+1}$$

So we will partition our experiment into n stages and get an upper bound on the expected duration of each stage this will help us get an upper bound on Expected weight of each of the k^{th} smallest edge of the MST $\forall k \in \{1 \ldots, n-1\}$

We partition the experiment in terms of expected duration to go from the stage when we have found the first i edges of MST to stage where we have found i+1 smallest edges of MST.

We will find an upper bound on the expected number of edges that we have to explore for this to be possible.

NOTE: we are going to use a important fact that any of the edge of the graph with index i can be connected between any pair of vertices with equal uniform probability due to the symmetry of the situation.

First, let's find an upper bound on the probability that the edge with index i of the graph is the k^{th} smallest edge of the MST.

So We are assuming that that we have already found the first $k-1$ smallest edges of the MST and now we are testing that the nodes corresponding to edge with index i of the graph belong to different or the same connected components.

Let's call this edge **good** if the nodes corresponding to them belong to different components and hence this edge will also be part of the MST, and **bad** if they already belong to the same component and hence this edge is not part of the MST.

$Prob(i^{th}$ edge is bad given we have connected already connect k-1 edges)

$$= \frac{\sum_{j=1}^{n-k+1}\binom{sz_j}{2} - (k-1)}{\binom{n}{2} - (i-1)}$$

Where sz_j in the size of the j^{th} connected component, in total there will be $n-k+1$ connected components. For sz_j=1 we will take

$$\binom{\text{SZ}_j}{2} = 0$$

$$\text{We will now prove } \sum_{j=1}^{n-k+1}\binom{\text{SZ}_j}{2} \leq \binom{k}{2}$$

For $a_1 \geq 2$ and $a_2 \geq 2$:

$$a_1 \cdot a_2 - a_1 - a_2 + 1 \geq 0$$

$$2a_1 \cdot a_2 - 2 \cdot a_1 - 2 \cdot a_2 + 2 \geq 0$$

$$a_1^2 + a_2^2 - a_1 - a_2 + 2 \cdot a_1 \cdot a_2 - 2 \cdot a_1 - 2 \cdot a_2 + 2 \geq a_1^2 + a_2^2 - a_1 - a_2$$

$$(a_1 + a_2 - 1) \cdot (a_1 + a_2 - 2) \geq a_1 \cdot (a_1 - 1) + a_2 \cdot (a_2 - 1)$$

$$\frac{(a_1 + a_2 - 1)!}{2! \cdot (a_1 + a_2 - 3)!} \geq \frac{a_1!}{2! \cdot (a_1 - 2)!} + \frac{a_2!}{2! \cdot (a_2 - 2)!}$$

$$\binom{a_1 + a_2 - 1}{2} \geq \binom{a_1}{2} + \binom{a_2}{2}$$

Since we have only connected $(k-1)$ edges so far and assuming there are x connected components with $sz_j \geq 2$, this means that

$$\sum_{j=1}^{x} sz_j = k + x - 1$$

(as every edge added increases the sum of sz_j by 1 and every new connected component with size ≥ 2 increases the sum of sz_j by 1).
Now we proceed to solve by unfolding.

$$\sum_{i=1}^{x} \binom{\text{sz}_j}{2} = \binom{\text{sz}_1}{2} + \binom{\text{sz}_2}{2} + \ldots + \binom{\text{sz}_x}{2}$$

$$\sum_{i=1}^{x} \binom{\text{sz}_j}{2} \leq \binom{\text{sz}_1 + \text{sz}_2 - 1}{2} + \ldots + \binom{\text{sz}_x}{2}$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\sum_{i=1}^{x} \binom{\text{sz}_j}{2} \leq \binom{\text{sz}_1 + \text{sz}_2 + \ldots + \text{sz}_x - x + 1}{2}$$

Since $\sum_{j=1}^{x} \text{sz}_j$= k+x-1,

$$\sum_{i=1}^{x} \binom{\text{sz}_j}{2} \leq \binom{k}{2}$$

Hence, proven.

Ignoring linear terms where quadratic terms are present because in the limit they will get dominated anyway, We get

$Prob(i^{th}$ edge is bad | We have found first k-1 edges of MST) $\leq \frac{\binom{k}{2}}{\binom{n}{2}}$

$Prob(i^{th}$ edge is bad | We have found first k-1 edges of MST) $\leq \frac{k \cdot (k-1)}{n \cdot (n-1)}$

Therefore

$Prob(i^{th}$ edge is good | We have found first k-1 edges of MST) $\geq 1 - \frac{k \cdot (k-1)}{n \cdot (n-1)}$

We can see this as a Geometric Random variable where the probability of seeing a good edge and is lower bound by $1 - \frac{k \cdot (k-1)}{n \cdot (n-1)}$

Since the Expected value of the Geometric Random variable with probability of success p is $\frac{1}{p}$

So, the expected number of edges that we have to check to get the k^{th} smallest edges of MST, after finding the $(k-1)^{th}$ smallest edge is upper bounded by $\frac{n\cdot(n-1)}{n\cdot(n-1)-k\cdot(k-1)}$

Let the notation $E[x \to y]$ mean the expected number of edges that we have to explore to go from a stage of having x smallest edges of MST to stage of having y smallest edges of MST.

$$E[0 \to k] = \sum_{i=0}^{k-1} E[i \to i+1]$$

using linearity of expectation since the experiment $0 \to k$ can be written as the sum of the following k random variables $(0 \to 1)+(1 \to 2)+\ldots+(k-1 \to k)$

$$\leq \sum_{i=0}^{k-1} \frac{n\cdot(n-1)}{n\cdot(n-1)-i\cdot(i-1)}$$

$$= n\cdot(n-1)\cdot\sum_{i=0}^{k-1}\frac{1}{n\cdot(n-1)-i\cdot(i-1)}$$

$$= n\cdot(n-1)\cdot\sum_{i=0}^{k-1}\frac{1}{(n-i)\cdot(n+i-1)}$$

$$= \frac{n\cdot(n-1)}{2n-1}\cdot\sum_{i=0}^{k-1}\frac{2n-1}{(n-i)\cdot(n+i-1)}$$

$$= \frac{n\cdot(n-1)}{2n-1}\cdot\sum_{i=0}^{k-1}\frac{(n-i)+(n+i-1)}{(n-i)\cdot(n+i-1)}$$

$$= \frac{n\cdot(n-1)}{2n-1}\cdot\sum_{i=0}^{k-1}\frac{1}{n-i}+\frac{1}{n+i-1}$$

Here we are going to use the result $\sum_{i=1}^{n}\frac{1}{i} \approx log_e(n) + constant$
The value of the constant is approximately 0.58 but it is not relevant to our analysis.

$$E[0 \to k] \leq \frac{n\cdot(n-1)}{2n-1}\cdot\sum_{i=0}^{k-1}\frac{1}{n-i}+\frac{1}{n+i-1}$$

$$\approx \frac{n\cdot(n-1)}{2n-1}\cdot(\log_e(n)-\log_e(n-k)+\log_e(n+k-1)-\log_e(n))$$

$$= \frac{n\cdot(n-1)}{2n-1}\cdot(\log_e(n+k-1)-\log_e(n-k))$$

Suppose we plugin in $k = n - 1$ in the above expression, then what we get is an upper bound on the expected number of edges we have to see to make the MST.

So we have to explore only $\mathcal{O}(n \log_e(n))$ edges of the complete graph in expectation to get our MST.

This result is very useful since it allows us to design an efficient Las Vegas algorithm to sample from such an MST.

This is very fast compared to the brute force way to sample in which we simulate all the $\mathcal{O}(n^2)$ edges of the graph first then apply some MST algorithm.

More details about our algorithm later.

As proved earlier the expected weight of the k^{th} smallest edges of MST is $\frac{E[0 \to k]}{\binom{n}{2}+1}$

Since $E[0 \to k]$ is the expected number of edges that we have to explore to get k^{th} smallest edge of MST, So $E[0 \to k]$ is the expected index of k^{th} smallest edge of MST.

so, The weight of MST is the sum of the weight of all edges, so we can use the linearity of expectation to get this result.

$$E[\text{Weight of MST}] = \sum_{k=1}^{n} E[\text{Weight of } k^{th} \text{ smallest edge of MST}]$$

$$E[\text{Weight of MST}] = \sum_{k=1}^{n} \frac{E[0 \to k]}{\binom{n}{2} + 1}$$

$$\leq \frac{1}{\binom{n}{2} + 1} \cdot \sum_{k=1}^{n} \frac{n \cdot (n-1)}{2n-1} \cdot (\log_e(n+k-1) - \log_e(n-k))$$

$$= \frac{n \cdot (n-1)}{(\binom{n}{2} + 1) \cdot (2n-1)} \cdot \sum_{k=1}^{n} (\log_e(n+k-1) - \log_e(n-k))$$

$$= \frac{2 \cdot n \cdot (n-1)}{(n^2 - n + 2) \cdot (2n-1)} \cdot \sum_{k=1}^{n} (\log_e(n+k-1) - \log_e(n-k))$$

Let's focus on the summation $\sum_{k=1}^{n} (\log_e(n+k-1) - \log_e(n-k))$

Unfolding the summation we get

$= \log_e(n) + log_e(n+1) + \ldots + \log_e(2n-2) - (\log_e(n-1) + log_e(n-2) + \ldots + \log_e(1))$

$= \log_e(1) + log_e(2) + \ldots + \log_e(2n-2) - 2 \cdot (\log_e(n-1) + log_e(n-2) + \ldots + \log_e(1))$

$$= \log_e((2 \cdot n - 2)!) - 2 \cdot \log((n-1)!)$$

We will use Strling's Approximation here according to which $n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$

Substituting it above we get

$$= \log_e \left(\sqrt{2\pi(2n-2)} \left(\frac{2n-2}{e} \right)^{2n-2} \right) - 2 \log_e \left(\sqrt{2\pi(n-1)} \left(\frac{n-1}{e} \right)^{n-1} \right)$$

Using properties of standard properties of logarithm it simplifies to

$$= \frac{1}{2}\log(2\pi(2n-2))+(2n-2)\log\left(\frac{2n-2}{e}\right)-2\cdot\frac{1}{2}\log(2\pi(n-1))-(n-1)\log\left(\frac{n-1}{e}\right)$$

Grouping like terms together

$$= \frac{1}{2}\left(\log(2\pi(2n-2)) - 2\log(2\pi(n-1))\right)+\left((2n-2)\log\left(\frac{2n-2}{e}\right) - 2\cdot(n-1)\log\left(\frac{n-1}{e}\right)\right)$$

$$= \frac{1}{2}\left(\log(2) + \log(2\pi(n-1)) - 2\log(2\pi(n-1))\right)+(2n-2)\log(2)+(2n-2)\log\left(\frac{n-1}{e}\right)-(2n-2)\log\left(\frac{n-1}{e}\right)$$

Cancelling terms with the opposite sign we get

$$= \frac{1}{2}\left(\log(2) - \log(2\pi(n-1))\right) + (2n-2)\log(2)$$

Note: $(\log(2) - \log(2\pi(n-1))) \leq 0$
So,

$$\frac{1}{2}\left(\log(2) - \log_e(2\pi(n-1))\right) + (2n-2)\log_e(2) \leq (2n-2)\log(2)$$

Now going back to the original expression we get

$$E[\text{Weight of MST}] \leq \frac{2\cdot n\cdot(n-1)}{(n^2-n+2)\cdot(2n-1)}\cdot\sum_{k=1}^{n}\left(\log_e(n+k-1) - \log_e(n-k)\right)$$

$$\leq \frac{2\cdot n\cdot(n-1)}{(n^2-n+2)\cdot(2n-1)}\cdot(2n-2)\log_e(2)$$

$$= \frac{4\cdot n\cdot(n-1)^2\cdot\log_e(2)}{(n^2-n+2)\cdot(2n-1)}$$

using Standard limit rules

$$\lim_{n\to\infty}\frac{4\cdot n\cdot(n-1)^2\cdot\log_e(2)}{(n^2-n+2)\cdot(2n-1)} = 2\cdot log_e(2)$$

$$\approx 1.3862$$

Prim's Algorithm- $\log(n)$ Bound

Introduction

The aim of this proof is to demonstrate that the expected weight of the Minimum Spanning Tree (MST) in a complete graph, where each edge is assigned weight randomly and uniformly in the range $[0, 1]$, is bounded by $\log_e n$ using Prim's algorithm.

About the Algorithm

Prim's algorithm starts with an empty spanning tree. It maintains two sets of vertices: one set containing the vertices already included in the MST, and the other set containing the vertices not yet included. At each step, it considers all the edges connecting the two sets and selects the minimum weight edge. After selecting the edge, it moves the other endpoint of the edge to the set containing the MST.

Analysis

We in this proof we are going to constuct a spanning tree, whose Expected weight is $\mathcal{O}(\log(n))$, which is obviously a upperbound to the expected value of the minimum spanning tree.

Let S_1 denote the set of vertices in our spanning tree and S_2 denote the set of vertices not yet present in our spanning tree. Let our spanning tree currently contain i vertices. Let the last node added to S_1 be x'. There are $(n-i)$ edges going from x' to the set S_2. Since all of these $(n-i)$ edges are independent with respect to each other, and not yet seen so they distributed randomly uniform in the range $[0, 1]$, the expected weight of the minimum of these edges is $\frac{1}{(n-i+1)}$ (Using Result 1). As we will take the minimum weight edge connecting both sets in the ith step of Prim's Algorithm and x' belongs to the MST, it implies that:

Edges from x' to $S_2 \subset$ Edges from MST to S_2

$\Rightarrow$Minimum weight edge from x' to $S_2 \geq$ Minimum weight edge from MST to S_2

As we select the minimum weight edge connecting both sets at the ith step, it implies that the expected minimum weight edge from S_1 to S_2 at the ith step of Prim's Algorithm is upper bounded by the expected minimum weight edge from x' to S_2, which is $\frac{1}{(n-i+1)}$.

Therefore, the expected weight of our spanning tree is given by(Using Lin-

earity of Expectation):

$$\begin{aligned}\text{Expected weight of MST} &= \sum_{i=1}^{n-1} \text{Expected weight of } i^{\text{th}} \text{ edge of our spanning tree} \\ &\leq \sum_{i=1}^{n-1} \frac{1}{(n-i+1)} \\ &\approx \log_e n + 0.58\end{aligned}$$

Conclusion

Thus, it is proven that the expected weight of a Minimum Spanning Tree in a complete graph with uniformly random edge weights in the range $[0, 1]$ is upper bounded by $\log_e n$ using Prim's algorithm.

Prim's Algorithm- Constant Upper Bound

Introduction

The aim of this proof is to demonstrate that the expected weight of the Minimum Spanning Tree (MST) in a complete graph, where each edge is assigned weight randomly and uniformly in the range $[0, 1]$, is bounded by a constant using Prim's algorithm.

About Prim's Algorithm

Prim's algorithm starts with an empty spanning tree. It maintains two sets of vertices: one set containing the vertices already included in the MST, and the other set containing the vertices not yet included. At each step, it considers all the edges connecting the two sets and selects the minimum weight edge. After selecting the edge, it moves the other endpoint of the edge to the set containing the MST.

Our Algorithm for Constant Bound

In the analysis to get the $\log_e n$ bound using Prim's algorithm, we notice that the edges being added to our spanning tree during the algorithm where initially small in magnitude and gradually became larger and larger. The edge were added with an initial value of $\frac{1}{n}$ while the weight of the last edge added was 1. In fact the weight of the $(\frac{n}{2})^{\text{th}}$ edge added to the Spanning tree in the previous analysis is $\frac{2}{n}$. This made us realise that our analysis gave a very loose bound and it was the inspiration we required to create a new algorithm to conduct analysis for the expected value of the MST.Our algorithm to analyse the expected weight of spanning tree includes the following 3 steps-

Step 1: We construct a spanning tree for the first $\frac{n}{2}$ vertices following Prim's algorithm,with the last $(\frac{n}{2})^{\text{th}}$ edge having an expected weight of $\frac{2}{n}$.

Step 2: Among the remaining $\frac{n}{2}$ vertices,we connect $\frac{n}{3}$ vertices to our spanning tree of $\frac{n}{2}$ vertices created in Step 1, with edge weights greater than the maximum edge used to make the spanning tree in Step 1,hence following distribution U[$\frac{2}{n}$,1].

Step 3: We are left with $\frac{n}{6}$ vertices,which we connect to the $\frac{n}{3}$ vertices added to spanning tree in Step 2,each edge following distribution U[0,1] as these edges have been left unused in the entire algorithm and have remained independent and uniformly and randomly distributed from [0,1].

Analysis

Let S_1 denote the set of vertices in the MST and S_2 denote the set of vertices not yet present in the MST. Let the MST currently contain i vertices. Let the last node added to S_1 be x'. There are $(n-i)$ edges going from x' to the set S_2. Since all of these $(n-i)$ edges are independent with respect to each other and randomly uniform in the range $[0,1]$, the expected weight of the minimum of these edges is $\frac{1}{(n-i+1)}$. As we will take the minimum weight edge connecting both sets in the ith step of Prim's Algorithm and x' belongs to the MST, it implies that:

Edges from x' to $S_2 \subset$ Edges from MST to S_2

$\Rightarrow$Minimum weight edge from x' to $S_2 \geq$ Minimum weight edge from MST to S_2

Step 1: As we select the minimum weight edge connecting both sets at the ith step, it implies that the expected minimum weight edge from S_1 to S_2 at the ith step of Prim's Algorithm is upper bounded by the expected minimum weight edge from x' to S_2, which is $\frac{1}{(n-i+1)}$.

Therefore, the expected weight of Step 1 is given by:

$$\begin{aligned}\text{Weight of Step 1} &= \sum_{i=1}^{\frac{n}{2}} \text{Weight of } i^{\text{th}} \text{ edge of MST} \\ &\leq \sum_{i=1}^{\frac{n}{2}} \frac{1}{(n-i+1)}\end{aligned}$$

Let W_1 be weight of edges added in Step 1 of our algorithm.Using Linearity of Expectation,

$$E[W_1] \leq \left(\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \cdots + \frac{2}{n}\right)$$

$$E[W_1] \leq \left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{1}\right) - \left(\frac{1}{\frac{n}{2}} + \frac{1}{\frac{n}{2}-1} + \frac{1}{\frac{n}{2}-2} + \cdots + \frac{1}{1}\right) + \frac{1}{\frac{n}{2}}$$

$$E[W_1] \leq \log(n) - \log\left(\frac{n}{2}\right)$$

$$E[W_1] \leq \log(2)$$

Step 2: Now the remaining vertices have two types of edges:

1. Edges connecting them to the first $\frac{n}{2}$ vertices of connected component C.
2. Edges connecting them to the rest of the $\frac{n}{2}$ vertices not taken by the spanning tree yet.

Let's consider only the first type of edges in Step 2.These edges are no longer independent and do not belong to U[0,1] due to Step 1.However to get an upper bound on the Expected weight of MST,we can consider all of these edges

to be greater than or equal to all the edges which are currently present in the spanning tree made in Step 1.

Let $e_{n/2}$ be the last edge added to the spanning tree, which has the maximum expected value of $\frac{2}{n}$ and can be considered to be the maximum weighted edge in the spanning tree presently. Since all edges are $\geq e_{n/2}$, then by Corollary 1, we can assign their weight to belong to $\sim U\left[\frac{2}{n}, 1\right]$ to get the correct expected value of the minimum of these edges.
Now when we add the i^{th} vertex of Step 2 to our spanning set,we have $\frac{n}{2}$+i-1 vertices in our spanning set.There are $\frac{n}{2}$+i-1 edges connecting our current vertex to the spanning set, each upper bounded by a distribution of U[$\frac{2}{n}$,1].
Let $e2_i$ be the weight of the edge added for the i^{th} vertex in Step 2.Using Result 1,

$$E[e2_i] \leq \frac{(1 + (\frac{n}{2} + i - 1) * \frac{2}{n})}{\frac{n}{2} + i - 1 + 1}$$

$$E[e2_i] \leq \frac{1}{\frac{n}{2} + i - 1 + 1} + \frac{2}{n}$$

Let W_2 be weight of edges added in Step 2 of our algorithm.

$$W_2 = \sum_{i=1}^{\frac{n}{3}} e2_i$$

Using Linearity of Expectation,

$$E[W_2] = \sum_{i=1}^{\frac{n}{3}} E[e2_i]$$

$$E[W_2] \leq \sum_{i=1}^{\frac{n}{3}} (\frac{1}{\frac{n}{2} + i - 1 + 1} + \frac{2}{n})$$

$$E[W_2] \leq \frac{2}{n} + \frac{1}{\frac{n}{2} + 1} + \cdots + \frac{6}{5 * n} + \frac{2}{n} * \frac{n}{3}$$

$$E[W_2] \leq \log(\frac{5 * n}{6}) - \log\left(\frac{n}{2}\right) + 0.667$$

$$E[W_2] \leq log(5) - log(3) + 0.667$$

Step 3: In this step, we will add the remaining $\frac{n}{6}$ vertices to the spanning tree considering only edges which have not been compared with other edges throughout the algorithm and still maintain the original distribution U[0,1]. Hence we will be adding these $\frac{n}{6}$ with edges connecting them to each other as well as edges connecting these vertices to the $\frac{n}{3}$ vertices added in Step 2,that is we are using the second type of edges mentioned at the start of Step 2.

When we add the i^{th} vertex in Step 3 to our spanning set,we have $\frac{n}{3}$+i-1 vertices in our spanning set, s.t. the edges between these vertices and our current

vertex is U[0,1].Let $e3_i$ be the weight of the edge added for the i^{th} vertex in Step 3.Since all edge weight belong U[0,1], using Result 2 we get,

$$E[e3_i] \leq \frac{1}{\frac{n}{3} + i - 1 + 1}$$
$$E[e3_i] \leq \frac{1}{\frac{n}{3} + i}$$

Let W_3 be weight of edges added in Step 3 of our algorithm.

$$W_3 = \sum_{i=1}^{\frac{n}{6}} e3_i$$

Using Linearity of Expectation,

$$E[W_3] = \sum_{i=1}^{\frac{n}{6}} E[e3_i]$$
$$E[W_3] \leq \sum_{i=1}^{\frac{n}{6}} (\frac{1}{\frac{n}{3} + i})$$
$$E[W_3] \leq \frac{3}{n} + \frac{1}{\frac{n}{3} + 1} + \cdots + \frac{2}{n}$$
$$E[W_3] \leq \log(\frac{n}{2}) - \log\left(\frac{n}{3}\right)$$
$$E[W_3] \leq log(3) - log(2)$$

The expected weight of our entire algorithm=E[W_1]+E[W_2]+E[W_3]
$\leq [log(2)] + [log(5) - log(3) + 0.667] + [log(3) - log(2)]$
$\leq log(5) + 0.667$
≤ 2.2761

Conclusion

Thus, it is proven that the expected weight of a Minimum Spanning Tree in a complete graph with uniformly random edge weights in the range $[0, 1]$ is bounded by 2.2761 using Prim's algorithm.

3 Explanation of codes

3.1 Brute force Kruskal

In the Algorithm we generated $\binom{n}{2}$ edges from $unif(0,1)$ distribution and then used kruskal algorithm to find the MST of the graph.

There nothing special about this code.

The time complexity of this algorithm is $\mathcal{O}(n^2 \log(n))$

3.2 Optimised Kruskal

In this algorithm we have used a essential fact proved in the analysis of Kruskal's algorithm to sample efficiently, that on expectation we only have to explore $\mathcal{O}(n \log(n))$ edges of the graph to find the edges of MST.

To implement this idea, we had used that fact that edge with index i can be any between any two edge uniformly with equal probibilty.

Algorithm 1 Randomized Minimum Spanning Tree

Initialise: $previous\_wt = 0$, $num\_edges\_found = 0$, $Rem\_cnt = \binom{n}{2}$ (No of unexplored edges)
$Total\_wt\_of\_MST = 0$
repeat
    Sample W from the minimum of Rem_cnt random variables, each following unif$(previous_wt, 1)$ distribution.
    Assign each to a pair of nodes U and V such that the pair has not been assigned yet to any of the previous edges, uniformly randomly.
    if U and V do not belong to the same component **then**
        $Total\_wt\_of\_MST \mathrel{+}= W$
        $num\_edges\_found \mathrel{+}= 1$
        Connect the components corresponding to U and V
    end if
    $previous\_wt = W$
    $Rem\_cnt \mathrel{-}= 1$
until $num\_edges\_found < n - 1$
Output $Total\_wt\_of\_MST$

To sample from W in above algorithm we have used Inverse transform Technique.

To use this technique we require to have CDF of the random variable we wish to sample from,

In this analysis of Result 1, we calcuated the CDF of min of n random variables each following $unif(\alpha, 1)$ distribution, where α is some constant.

Thus this is $\mathcal{O}(n \log(n))$ Las vegas algorithm.

3.3 Upper bound Kruskal

In our analysis we concluded

$$Prob(\text{finding a good edge}|\text{we have found k-1 egdes of MST}) \geq 1 - \frac{\binom{k}{2}}{\binom{n}{2}}$$

And used this to get an upperbound to on the expected value of MST. In this code we have simulated the this process and we went from smallest to largest edge of graph and selected them with probabilty $1 - \frac{\binom{k}{2}}{\binom{n}{2}}$, where $k-1$ is the number of edges of MST that we have found. and we got the answer from these simulations as around 1.38 which matches our theoretical upper bound calculations, so it was useful to provide the assurance to out theoretical proof with experiment.

Algorithm 2 Randomized Spanning Tree Weight

1: Initialise: $Total\_wt\_of\_Spanning\_tree = 0$ $Edge\_cnt = 0$ $Curr\_edge\_number = 1$
2: **repeat**
3: Toss a coin with probability $1 - \frac{\binom{k}{2}}{\binom{n}{2}}$
4: **if** coin is heads **then**
5: $Total\_wt\_of\_Spanning\_tree \mathrel{+}= \frac{Curr\_edge\_number}{\binom{n}{2}+1}$
6: $Edge\_cnt \mathrel{+}= 1$
7: **end if**
8: $Curr\_edge\_number \mathrel{+}= 1$
9: **until** $Edge\_cnt < n - 1$
10: Output $Total\_wt\_of\_Spanning\_tree$