Expected distance between vertices in a complete graph

1 Problem Statement

Let $G = (V, E)$ be an undirected graph on $n = |V|$ vertices, $m = |E|$ edges with positive edge weights. The **shortest path problem** is defined as finding a path P between two vertices (s, t) in G, which minimises the sum of weights of its constituent edges.[1] The distance between (s, t) is the sum of edge weights on path P.
Find the **expected distance** between a pair of vertices in a complete graph G where each edge is assigned weight randomly uniformly and independently in the range $[0, 1]$. we are interested in finding

1. Expected distance between any (arbitrary) pair of vertices (s, t)
2. Expected eccentricity of an arbitrary vertex s (distance between s and the vertex farthest from s)
3. Expected diameter of the graph. (The pair of vertices with maximum distance over all pairs)

2 Introduction

The shortest path problem for weighted graphs, itself is a very fundamental problem. However, the most exciting thing about this problem is that the edge weights in graph G are chosen randomly. As we know many real world scenarios can be represented by graphs.

[2] solved the given problem using exponential random variables and the proof of each result is quite hard to follow.

Our Contribution: We establish each of the two cases of given problem using uniform random variables. Apart from analytical proofs, we also establish the results empirically for all three. This helps us to verify the analytical results in practice. For both analytical and empirical results, we use the Dijkstra's algorithm. We also use and establish expected value of some intermediate but useful random variables, which we believe are of independent interest.

2.1 Overview

We compute the expected distance for a fixed pair of vertices in section 5.2 and the eccentricity of a vertex in section 5.1 by analysing the Dijkstra's algorithm. Then, we give an empirical proof for expected diameter in section 5.3. We start with proving some results which we use later in our proofs.

3 Preliminaries

We denote the random variables for edge weights by v_i's and w_i's. The vertices in the graph are denoted by s, t and g_i's.
We define $X_{s,t} = dist(s, t)$ to be a random variable denoting the distance between vertices s and t in a given graph G, $Y_s = \max_{t \in V \setminus s} dist(s, t)$ as the eccentricity of vertex s and $Z = \max_{s,t} dist(s, t)$ as the diameter of G.
We define random graph to be a complete graph with edge weights as uniformly independent random variables taking values from $[0, 1]$.

We call the shortest path tree generated during a run of Dijkstra's algorithm as Dijkstra's tree. Random variable X taking value from the uniform distribution on $[0,1]$ is denoted as $X \sim U[0,1]$.

Lemma 1. *If $(n-1)$ points are sampled randomly uniformly and independently on a line segment of length 1, then the expected length of the interval between any two adjacent points is $1/n$, irrespective of which adjacent pair is chosen. This result can also be interpreted as $E[min\{v_1, ..., v_{n-1}\}] = \frac{1}{n}$ where v_i's are sampled independently from $U[0,1]$.*

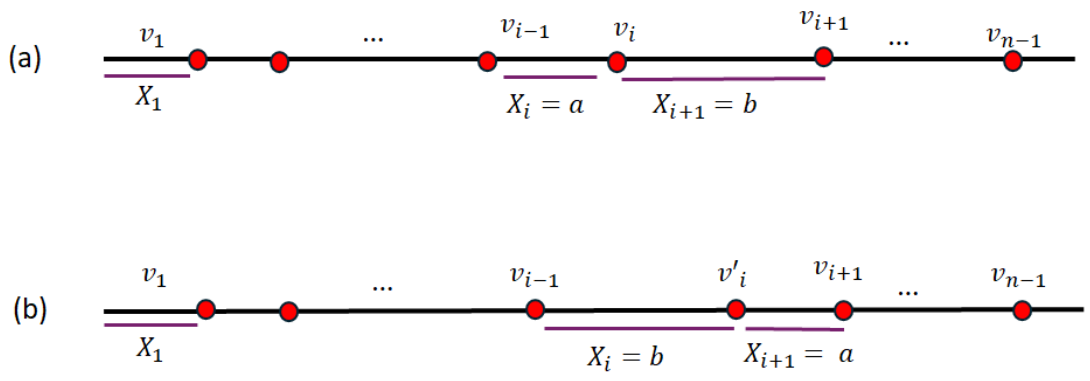

Figure 1: Sampling $n-1$ points from $[0,1]$

Proof. We sample $(n-1)$ points in the interval $[0,1]$. Say these are $\{v_1, ..., v_{n-1}\}$ when arranged in ascending order. This will lead to n intervals in $[0,1]$. The length of each interval is a random variable X_i. Thus we define n random variables $X_i = v_i - v_{i-1}$ for $i \in [n]$, where $v_0 = 0$ and $v_n = 1$.

We claim that

$$\forall i, j \; Pr[X_i = a] = Pr[X_j = a]$$

where a is any constant less than 1. It is sufficient to prove that $\forall i, Pr[X_i = a] = Pr[X_{i+1} = a]$. Figure 1 shows that for every random sample of $n-1$ points which has $X_i = a$ (Figure 1 (a)) there is another sample which has $X_{i+1} = a$ (Figure 1 (b)) . Sample 1(b) has the same points as 1(a) apart from v_i. Since each sample has the same probability ($\because$ independence and uniformity) the claim holds.
From the claim and the definition of expectation it follows that

$$E[X_1] = E[X_2] = ... = E[X_i] = ... = E[X_n] \tag{1}$$

Also, from the definition of the random variables X_i's, we have

$$\sum_{i=1}^{n} X_i = 1$$

$$\sum_{i=1}^{n} E[X_i] = 1 \quad \text{(linearity of expectation)}$$

$$E[X_i] = 1/n \quad \text{(from 1)}$$

Thus, expected length of any interval is $1/n$. In particular it is the expected length for the first interval which is also the value of v_1. □

4 Analysing Dijkstra's Algorithm

To calculate $E[X_{s,t}]$ and $E[Y_s]$ let us analyse the probabilities using a run of Dijkstra's algorithm starting from the vertex s.

The first iteration of Dijkstra's algorithm finds the edge with minimum weight starting from s. Since the graph is complete, s has exactly $(n-1)$ edges incident on it.the edge weights are independent uniform random variables in [0,1]. This corresponds to finding the minimum among $v_1, ..., v_{n-1}$ as in lemma 1 where v_i's correspond to the weights. Let the minimum weighted edge be (s, g_1) where g_1 is a place holder for the vertex closest to s with minimum weight. Let us denote its weight as a random variable D_1. Then lemma 1 gives $E[D_1] = 1/n$.

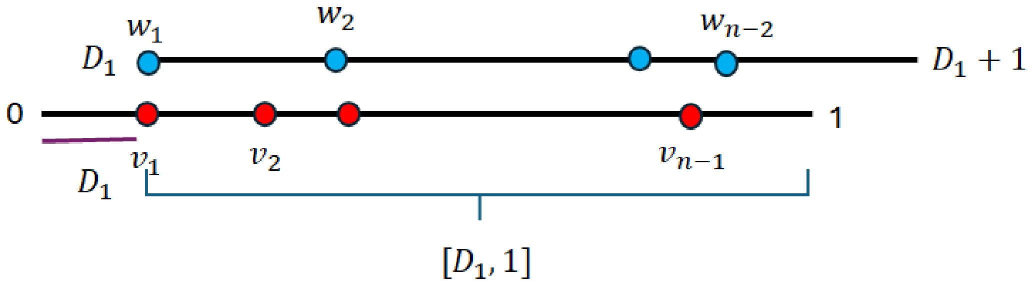

Figure 2: The set $S = \{v_2, ..., v_{n-1}, w_1, ..., w_{n-2}\}$

The next iteration of Dijkstra's algorithm considers all the remaining edges from s (let us call these $v_2, ..., v_{n-1}$ by their edge weights) as well as the edges starting from g_1 (call them $w_1, ..., w_{n-2}$ by their edge weights). It finds the minimum of the set $S := \{v_i, D_1 + w_j | i \in [n-1] \setminus \{1\}, j \in [n-2]\}$. Since $\forall i \geq 2$ $v_i > D_1$, we may as well say that v_i's in the set S take values independently from $U[D_1, 1]$ or from $D_1 + U[0, 1 - D_1]$ (see figure 2). This means $\min S = D_1 + \{v_i', w_j | i \in [n-1] \setminus \{1\}, j \in [n-2]\}$ where $v_i' \sim U[0, 1-D_1]$ and $w_j \sim U[0,1]$.

Let us allow v_i's to take values from $[0, 1]$. This might increase the minimum, giving us an upper bound on the expectation that we wish to calculate. By doing this we can reduce the problem to finding the minimum among $2(n-2)$ independent variables taking values from $U[0, 1]$. If we denote by D_2 as a random variable for this minimum, lemma 1 gives $E[D_2] = \frac{1}{2(n-2)+1}$. Thus $E[\min S] = E[D_1] + E[D_2] = \frac{1}{n} + \frac{1}{2(n-2)+1}$. Let g_2 denote the new vertex added to the Dijkstra's tree. Note by definition we have that the random variable $dist(s, g_2) = X_{s,g_2} \leq D_1 + D_2$.

This reasoning can be extended to show that in the i^{th} iteration when the Dijkstra's tree has grown to i vertices, namely $s, g_1, ..., g_{i-1}$, there are $i(n-i)$ edges whose weights are considered for calculating the minimum in this round (see Figure 3). The weight of the next edge to be added to the Dijkstra's tree has to be greater than$D_1 + D_2 + ... + D_{i-1}$. Allowing some of the edges to take slightly higher weights, like in the previous case we reduce the problem to finding minimum among $i(n-i)$ variables taking values independently from $U[0, 1]$. The expected minimum in this case is $\frac{1}{i(n-i)+1}$ and it is denoted by D_i

Since $X_{s,g_i} \leq D_1 + ... + D_i$, we have :

Lemma 2. $E[X_{s,g_i}] \leq \sum_{j=1}^{i} \dfrac{1}{j \cdot (n-j) + 1}$ *where g_i is the vertex added to the Dijkstra's tree in the i^{th} iteration.*

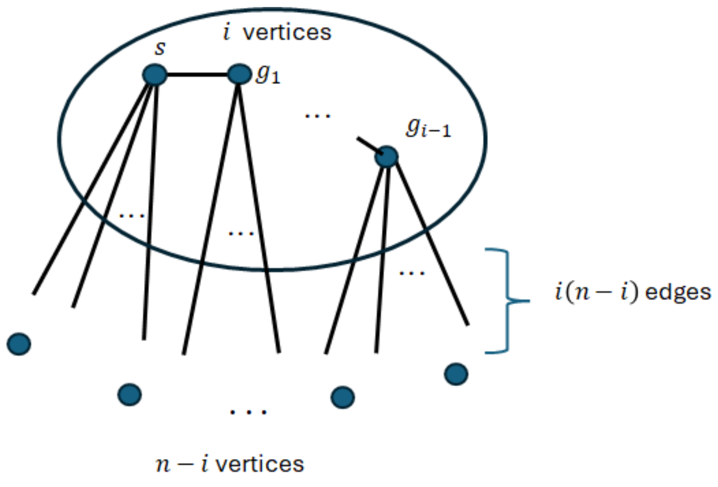

Figure 3: number of edges considered in i^{th} iteration of Dijkstra's algorithm

5 Computing Expected Distances

5.1 Expected Eccentricity of a vertex

Theorem 1. *The expected eccentricity of a vertex s in a complete graph K_n with edge weights assigned randomly uniformly and independently in the range $[0, 1]$ is $\leq \frac{2 \ln n}{n}$.*

Proof. Expected eccentricity of s is the expected distance between s and the farthest vertex from s.

Starting Dijkstra's from s, if the order in which the vertices are added to the Dijkstra's tree is $g_1, ..., g_{n-1}$, we want to find the expected distance of g_{n-1} from s i.e. $E[X_{s,g_{n-1}}]$ where the expectation is over random graphs and g_{n-1} depends on the graph. Mathematically this is $E[Y_s]$.
Following the notation of Lemma 2 the eccentricity of s in a random graph is the random variable

$$\begin{aligned}
Y_s &:= X_{s,g_{n-1}} \\
E[Y_s] &= E[X_{s,g_{n-1}}] && \text{Taking expectation} \\
&\leq \sum_{j=0}^{n-1} \frac{1}{j \cdot (n-j) + 1} && \text{Lemma 2} \\
&\leq \sum_{j=0}^{n-1} \frac{1}{j \cdot (n-j)} \\
&= \sum_{j=0}^{n-1} \frac{1}{n} \Big(\frac{1}{j} + \frac{1}{n-j} \Big) \\
&\leq \frac{2 \ln n}{n}
\end{aligned}$$

□

5.2 Expected distance between a fixed Pair

Theorem 2. *The expected distance between any fixed pair of vertices (s, t) in a complete graph K_n with edge weights assigned randomly uniformly and independently in the range $[0, 1]$ is $\leq \ln n/n$.*

Proof. Let us assume here that $t \neq s$.

Starting Dijkstra's from s, let the order in which the vertices are added to the Dijkstra's tree be $g_1, ..., g_{n-1}$. In a random graph, t is equally likely to be any of the g_i's. To understand this, consider a random graph G by fixing some edge weights. Let t be the g_k i.e. it is the k^{th} vertex added to the Dijkstra's tree. Now without changing the weights consider swapping the labels of g_1 and g_k. This new graph G' is also a valid random graph. The probability of the both the graphs, G, G' are equal.
In fact, the probability of any graph in the sample space is the same.
Thus, we have $\forall i \in \{2, ..., n-1\}$ $Pr[t = g_i] = \frac{1}{n-1}$.

$$\begin{aligned}
E[X_{s,t}] &= \sum_{i=1}^{n-1} Pr[t = g_i] E[X_{s,g_i}] \\
&= \sum_{i=1}^{n-1} \frac{1}{n-1} E[X_{s,g_i}] \\
&\leq \sum_{i=1}^{n-1} \frac{1}{n-1} \sum_{j=1}^{i} \frac{1}{j(n-j)+1} \\
&\leq \sum_{i=1}^{n-1} \frac{1}{n} \sum_{j=1}^{i} \frac{1}{j(n-j)} \\
&= \frac{1}{n} \sum_{j=1}^{n-1} \sum_{i=j}^{n-1} \frac{1}{j(n-j)} \\
&= \frac{1}{n} \sum_{j=1}^{n-1} \frac{n-j}{j(n-j)} \\
&\leq \frac{\ln n}{n}
\end{aligned}$$

□

5.3 Expected Diameter

Theorem 3. *The expected maximum distance over all pair of vertices in a complete graph K_n with edge weights assigned randomly uniformly and independently in the range $[0, 1]$ is $\leq 3 \ln n/n$.*

Proof. The radius of a graph [5] is $\min_{s \in V} Y_s$. We know expected radius is less than $2\frac{\ln n}{n}$ by Lemma 2. We also know diameter of a graph can be at most twice the radius. So, we immediately get a loose upper bound of $4\frac{\ln n}{n}$ on the expected diameter.
The bound of $3\frac{\ln n}{n}$ was shown by [2] with the help of exponential random variables. [2] makes use of the fact that the probability of Z deviating from $3\ln(n)/n$ is negligible, (i.e, $Pr(|Z - E[Z]| \geq \epsilon\frac{\ln n}{n})$ decreases significantly with n). We show the same fact through the plot in figure 4, where we show that it is very tightly bounded for $\epsilon = 0.3$ and only one out of $100,000$ values of n violates this condition.

□

6 Empirical Results

Lemma 3. *Probability that an edge with weight greater than $\frac{4\ln n}{n}$ is used in computation of a shortest path is negligible.*

Proof. The empirical data (see figure 4) shows us that the deviation of the diameter from $\frac{3\ln n}{n}$(its expected value) is very low w.h.p. This implies that in the worst case, when the diameter consists of only one edge, the edge has weight greater than $\frac{3\ln n}{n}$ with very low probability. Thus, we can safely omit edges with weights greater than $\frac{4\ln n}{n}$. □

Lemma 4. *Expected number of edges with edge weights* $\leq 4\frac{\ln n}{n}$ *is* $O(n \ln n)$.

Proof. The graph contains exactly $\binom{n}{2}$ edges. We define T_j as a Bernoulli random variable such that:

$$T_j = \begin{cases} 1 & \text{if } i^{th} \text{ edge has weight} \leq \frac{4\ln n}{n} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Since, edge weights picked randomly uniformly and independently:

$$Pr(T_j = 1) = Pr(j^{th} \text{ edge has weight } \leq \frac{4\ln n}{n}) \leq \frac{4\ln n}{n}$$

Let T denote the number of edges with edge weights less than or equal to $\frac{4\ln n}{n}$. Then,

$$T = \sum_{i=1}^{\binom{n}{2}} T_j \text{ Using linearity of expectation we get: } E[T] \leq \binom{n}{2}.\frac{4\ln n}{n} \leq 4n\ln n = O(n\ln(n))$$

As, we can see $E[T]$ is nothing but the expected number of edges we use in each of our graphs. Thus, the expected number of edges $m = O(n \ln n)$ □

6.1 Algorithm

The algorithm generates a complete graph with edge weights generated independently from $U[0, 1]$. For optimisation, it omits edges with edge weights $> 4\ln(n)/n$, rendering the graph sparse. We use an adjacency list to represent the graph. The algorithm then makes calls to the Dijkstra's algorithm by taking each vertex as source. This computes $X_{s,t}, Y_s, Z$ accordingly by comparing and taking the maximum. Then, it repeats the above procedure k times (k is any positive integer and is treated as a constant) and takes the average of each of the values.
We have varied the value of k depending on the value of n(i.e. $k = 100$ for $n < 1000$, $k = 20$ for $n \in [1000, 50000)$ and $k = 8$ for $n \in [50000, 100000]$).

6.2 Time and Space

- The expected space complexity of the algorithm is:

  Adjacency list representation takes $O(m + n)$ space. So, the expected space taken by the algorithm equals the expected number of edges in graph, which is $O(n \ln n)$.

- The algorithm makes $O(n)$ calls to the $Djikstra's$ algorithm for each vertex.The expected time complexity of the algorithm is:

$$O(k.n(n + m)\log n) = O(n.(n\ln(n))\log n) = O(n^2 \log^2 n)$$

6.3 Problems Overcome/ optimization strategies

Since for a complete graph, $m = O(n^2)$, making $O(n)$ calls would take $O(n^3 \log n)$ time. This not only takes huge time but huge space as well.

1. We only use edges with weights $\leq 4\ln(n)/n$. From Lemma 4, we know the expected number of such edges is $O(n \ln(n))$. Also, this will not make any changes to our result w.h.p..
   Pros: Without this modification, even for only $n = 35000$, memory requirement is higher than $16GB$ RAM and takes around 3 minutes for $n = 900$. After the modification, we can run the same algorithm for values higher than $n = 100,000$ with same memory requirement and the time taken for $n = 900$ is only 7.084s.

2. Observe that each of k instances are independent of each other. We use this fact to process each instance parallelly using multi-threading.
Pros: This completely bypasses the need for the CPU to wait for execution of each instance. Roughly it makes our algorithm k times faster given that there are k threads in the system.

3. Some common programming tricks like using adjacency list, Fibonacci heap in $Djikstra's$ algorithm, not storing the all pair shortest path matrix(which is $O(n^2)$) and using *float* instead of *double*.

6.4 Graphical representation of Experiments

The plots show us clearly that deviation of expected distances significantly less. As we can see in 4, only one value of $E[Z]$ out of more than 2 lakh values of n has deviated from its expectation by a considerable amount and still it is close to $2\ln(n)/n$. We see a further stronger bound on this deviation for $E[Y_s]$ in figure 5 and similarly for $E[X_{s,t}]$ in figure 6.

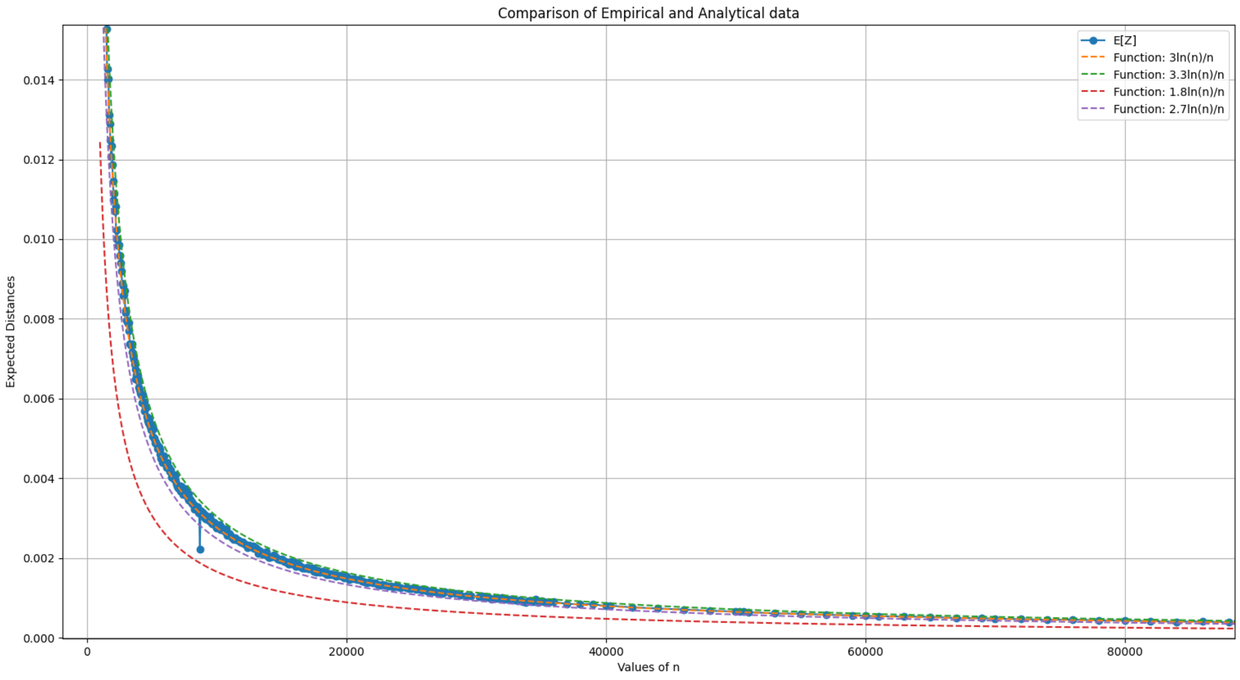

Figure 4: Deviation of $E[Z]$

Conclusion

We observe that most vertices are at an average distance of $\frac{\ln n}{n}$. So, We can divide the set of vertices into two disjoint set. One of them (developed villages) contains the vertices from which it takes almost $\frac{\ln n}{n}$ to reach most of other vertices. Another one (remote villages) contains the villages which are not developed villages.

1. It takes $\frac{\ln n}{n}$ to travel across any two developed villages.

2. It takes $\frac{2\ln n}{n}$ to travel from developed to remote villages.

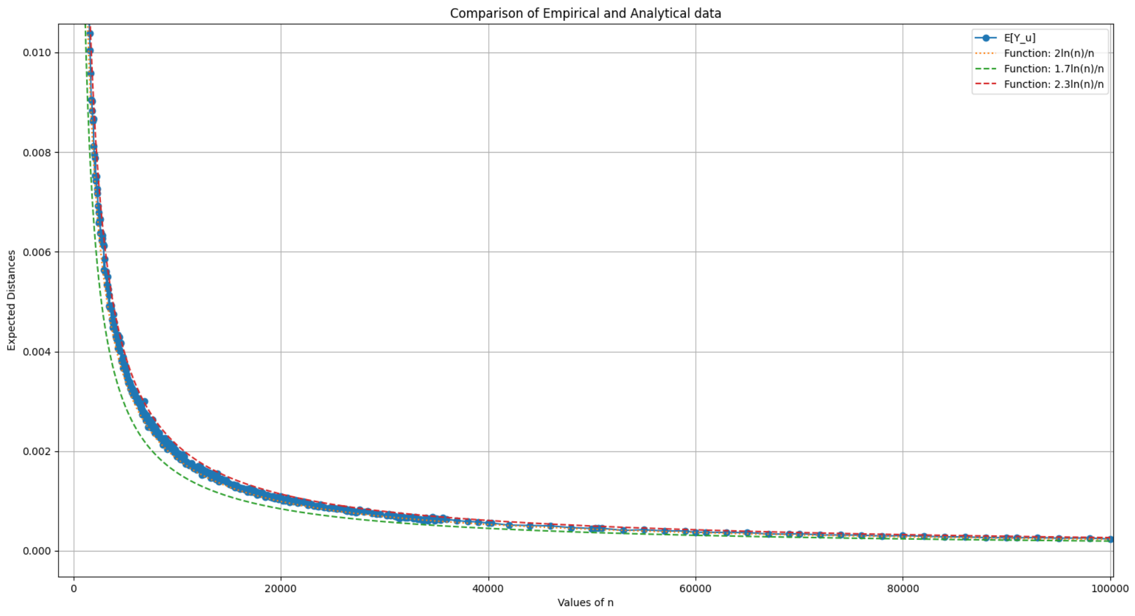

Figure 5: Deviation of $E[Y_s]$

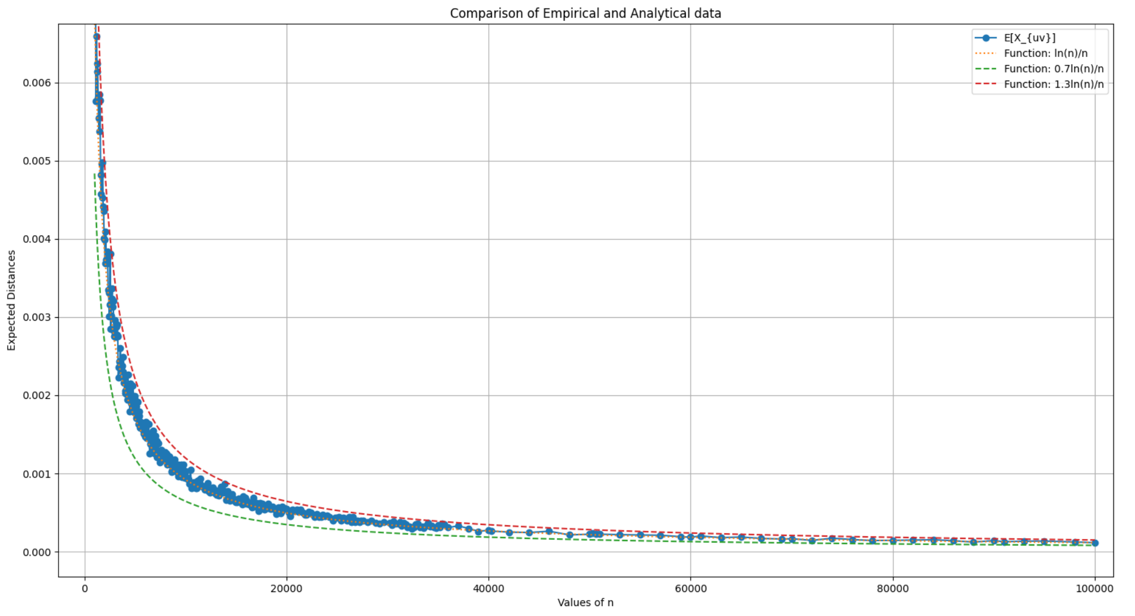

Figure 6: Deviation of $E[X_{s,t}]$

3. It takes $\frac{3\ln n}{n}$ to travel across two remote villages.

References

[1] https://en.wikipedia.org/wiki/Shortest_path_problem

[2] https://www.stat.berkeley.edu/~aldous/260-FMIE/Papers/janson_123.pdf

[3] https://moodle.cse.iitk.ac.in/mod/resource/view.php?id=4423

[4] https://en.wikipedia.org/wiki/McDiarmid%27s_inequality

[5] https://en.wikipedia.org/wiki/Distance_(graph_theory)