1. **Randomized leader election.** (7p.) In computer networks, a frequent problem is how to choose one of the computers to be the *leader*, ie. a computer that will have a particular role in the network (eg. the root of a spanning tree can be considered to be the leader of the graph).

   We assume the network is a clique of size n where the nodes are the computers and the edges are communication links. The computers are identical to each other, but they are allowed to use randomness.

   Consider the following randomized algorithm that runs in parallel on each computer at the same time:

   - Each computer picks a value $v \in \{1, \ldots, n\}$ independently and uniformly at random and sends v to all other computers.
   - After receiving the values of all computers, if exactly one computer has chosen the value 1, it is chosen as the leader and we stop.
   - Otherwise, repeat.

   (a) (1p.) Show that the probability that the algorithm terminates in the first iteration is $p = \left(1 - \frac{1}{n}\right)^{n-1}$. Notice that $p \geq \frac{1}{e^2}$.

The condition that the algorithm terminates on the first iteration is that exactly one computer picks the value 1, while the others pick values other than 1. The probability of picking the value 1 for one computer is $\frac{1}{n}$, and the probability of picking values other than 1 is $\frac{n-1}{n} = 1 - \frac{1}{n}$. The probability that one computer chooses 1, and the others (n-1) computers choose non-1 values is $\left(1-\frac{1}{n}\right)^{n-1}\frac{1}{n}$.

Because the graph is a clique, it is symmetric with respect to all computers. Therefore, the probability that exactly one computer choose 1 and others (n-1) do not choose one is:

$$p = n\left(\left(1-\frac{1}{n}\right)^{n-1}\frac{1}{n}\right) = \left(1-\frac{1}{n}\right)^{n-1}$$ (proven)

(b) (1p.) What is the probability that no leader is picked by kth iteration?

The probability that no leader is picked is the complement of the probability found in (a): $\bar{p} = 1 - p = 1 - \left(1 - \frac{1}{n}\right)^{n-1}$. By the k-iteration, the probability that no leader is picked is:

⇨ $\bar{p}^k = \left[1 - \left(1 - \frac{1}{n}\right)^{n-1}\right]^k$ (answer)

(c) (2p.) What is the expected runtime of the algorithm when $n = 10$?
Hint: look up geometric random variables to get to the end of the computation.

Let's call X as the general runtime of the algorithm.

We have: $E(X) = 1p_{n=1} + 2p_{n=2} + 3p_{n=3} + \ldots$

- ⇨ $E(X) = 1p + 2p\bar{p} + 3p\bar{p}^2 + \ldots = 1p + 2p(1-p) + 3p(1-p)^2 + \ldots$
- ⇨ $(1-p)E(x) = p(1-p) + 2p(1-p)^2 + 3p(1-p)^3 + \ldots$
- ⇨ $(p-1)E(x) + E(x) = 1p + 1p(1-p) + 1p(1-p)^2 + \ldots$
- ⇨ $pE(x) - E(x) + E(x) = p + p(1-p) + p(1-p)^2 + \ldots$
- ⇨ $E(x) = 1 + (1-p) + (1-p)^2 + \ldots$
- ⇨ $E(x) = \dfrac{1}{1-(1-p)}$ (geometric sum)
- ⇨ $E(x) = \dfrac{1}{p}$. At $n = 10 \Rightarrow p = \left(1 - \dfrac{1}{n}\right)^{n-1} = \left(\dfrac{9}{10}\right)^9$

The expected run time is $E(x) = \dfrac{1}{p} = \left(\dfrac{10}{9}\right)^9 \approx 2.58$ (answer)

(d) (3p.) Show that the algorithm runs in $O(\log n)$ time with high probability.
Hint: use question (b).

From part (a), we have: $p \geq \frac{1}{e^2}$ => $1-p \leq 1-\frac{1}{e^2}$

From part (b), the probability of no leaders being chosen by k-iteration is:

$(1-p)^k = \left(1-\frac{1}{e^2}\right)^k = 0.8646^k \leq 1-\frac{1}{n^c}$, $n \geq 2, k \geq 2$ and c can be any value. Choose c = 2

⇨ $\left(1-\frac{1}{e^2}\right)^k \leq 1-\frac{1}{n^2}$

⇨ $\log\left(\left(1-\frac{1}{e^2}\right)^k\right) \leq \log\left(1-\frac{1}{n^2}\right)$

⇨ $k\log\left(1-\frac{1}{e^2}\right) \leq \log\left(1-\frac{1}{n^2}\right)$

⇨ $k\log\left(1-\frac{1}{e^2}\right) \leq \log\left(\frac{n^2-1}{n^2}\right)$

⇨ $k \geq \frac{\log\left(\frac{n^2-1}{n^2}\right)}{\log\left(1-\frac{1}{e^2}\right)}$ (since $\log\left(1-\frac{1}{e^2}\right) < 0$). Since $1/\log\left(1-\frac{1}{e^2}\right)$ is constant, denote it as -C

⇨ $k \geq -C\log\left(\frac{n^2-1}{n^2}\right) = -C\left(\log\left(n^2-1\right)-\log\left(n^2\right)\right) = C\left(\log\left(n^2\right)-\log\left(n^2-1\right)\right) \leq O(\log n)$

As we can see, when n grows, then the term $C\left(\log\left(n^2\right)-\log\left(n^2-1\right)\right)$ will become much smaller. The chance of k longer than $O(\log n)$ becomes greatly diminished as n grows. Therefore, the algorithm runs in $O(\log n)$ time with high probability

2. **Individual exercise: Stronger partitioning.** (3p.) In the Tutorial Exercise 2 we showed that we can partition the vertex set of a tree $T = (V, E)$ into two sets V_1 and V_2, such that each connected component of $T[V_1]$ and $T[V_2]$ has diameter $O(\log n)$ with high probability.

Suppose we wanted a similar partition, only with diameter $O(\log \log n)$. More formally, we want to partition the vertex set of a tree $T = (V, E)$ into two sets V_1 and V_2, such that each connected component of $T[V_1]$ and $T[V_2]$ has diameter $O(\log \log n)$ with high probability. If we were to use the same algorithm and the same analysis as in the Tutorial Exercise 2, where would we fail?

Because the two sets V_1 and V_2 are symmetric, the proof needs to be done for one case. Let's pick the set V_1 and call the diameter of V_1 as d_1. The probability of d_1 being $O(\log \log n)$ is:

$$P\left[d_1 = O(\log \log n)\right] = 1 - P\left[d_1 = \omega(\log \log n)\right]$$

Given an arbitrary path in V_1, the probability of this path being included in V_1 is $\left(\frac{1}{2}\right)^k$. Therefore, the probability of d_1 being $\omega(\log \log n)$ is:

$$P\left[d_1 = \omega(\log \log n)\right] \leq \left(\frac{1}{2}\right)^{\omega(\log \log n)}$$

Because diameter is the largest distance possible between any two vertices, we can deduce that

$$P\left[d_1 = \omega(\log \log n)\right] = P\left[\bigcup_{i=1}^{n} \text{path}_i = \omega(\log \log n) \in V_1\right]$$

Applying the union bound to the above probability, the identity becomes:

$$P\left[\bigcup_{i=1}^{n} \text{path}_i = \omega(\log \log n) \in V_1\right] \leq \sum_{i=1}^{n} P\left[\text{path}_i = \omega(\log \log n) \in V_1\right]$$

Since the number of paths in each set is the number of vertices pair, which is $\binom{n}{2} = O(n^2)$

$$\Rightarrow \quad P\left[d_1 = \omega(\log \log n)\right] \leq O(n^2)\left(\frac{1}{2}\right)^{\omega(\log \log n)} = \frac{O(n^2)}{2^{\omega(\log \log n)}}$$

By definition, we have: $\frac{1}{2^{\log \log n}} = \frac{1}{poly(\log n)}$, But because $\log \log n < \omega(\log \log n)$

$$\Rightarrow \quad \frac{1}{2^{\omega(\log \log n)}} < \frac{1}{poly(\log n)}$$. Multiply both sides by $O(n^2)$, we have

⇨ $\frac{O(n^2)}{2^{\omega(\log\log n)}} < \frac{O(n^2)}{poly(\log n)}$

Because the term $O(n^2)$ can outgrow much faster than $poly(\log n)$, as n increases, $\frac{O(n^2)}{2^{\omega(\log\log n)}}$ grows indefinitely and will increase the failing probability. In other words, the probability of obtaining a set with diameter $O(\log\log n)$ will decrease.

⇨ Using the same algorithm in this case does not guarantee success with high probability