CS 630, Fall 2024, Homework 4
Due Wednesday, October 30, 2024, 11:59 pm EST, via Gradescope

Homework Guidelines

Collaboration policy Collaboration on homework problems, with the exception of programming assignments and reading quizzes, is permitted, but not encouraged. If you choose to collaborate on some problems, you are allowed to discuss each problem with at most 5 other students currently enrolled in the class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. Finding answers to problems on the Web or from other outside sources (including generative AI tools or anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or TA.

Typesetting Solutions should be typed and submitted as a PDF file on Gradescope. You may use any program you like to type your solutions. LaTeX, or "Latex", is commonly used for technical writing (`overleaf.com` is a free web-based platform for writing in Latex) since it handles math very well. Word, Google Docs, Markdown or other software are also fine.

Solution guidelines For problems that require you to provide an algorithm, you must provide:

1. pseudocode and, if helpful, a precise description of the algorithm in English. As always, pseudocode should include
   - A clear description of the inputs and outputs
   - Any assumptions you are making about the input (format, for example)
   - Instructions that are clear enough that a classmate who hasn't thought about the problem yet would understand how to turn them into working code. Inputs and outputs of any subroutines should be clear, data structures should be explained, etc.

   If the algorithm is not clear enough for graders to understand easily, it may not be graded.
2. a proof of correctness,
3. an analysis of running time and space.

You may use algorithms from class as subroutines. You may also use facts that we proved in class.

You should be as clear and concise as possible in your write-up of solutions. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand.

1. **Clique Counting** (7 points)

   An Erdős–Rényi graph is a random graph where each possible edge is likely to be present. For an Erdős-Rényi with n vertices and probability p, each of the $\binom{n}{2}$ edges is independently present with probability p. (This is one of two related graph models by Paul Erdős and Alfred Rényi.) In this problem, we will investigate the number of cliques in such a graph.

   (a) A triangle in a graph is a set of three distinct vertices in a graph where each pair of those vertices has an edge. What is the probability that a specific triangle is present in an Erdős-Rényi graph with parameters n and p??

   Solution. The probability that a specific triangle is present is p^3 since each of the three edges are independently present with probability p.

   (b) What is the expected number of triangles in an Erdős-Rényi graph with parameters n and p?

   Solution. First, observe that triangles in G are not independent. In case two triples share vertices, i.e. (i, j, k) and (i, j, ℓ) then the presence of a triangle on these triples both depend on the edge (i, j) being generated.

   We can define the random variable X_{ijk} to be 1 if there is a triangle on triple (i, j, k) and 0 otherwise. We have $E[X_{ijk}] = 1 \cdot p^3 + 0 \cdot (1 - p^3) = p^3$.
   Let X be the random variable that is equal to the number of triangles in G. That is, X can take any value $0, 1, \ldots \binom{n}{3}$.
   By definition $X = \sum_{(i,j,k)} X_{ijk}$. For the expected value we get $E[X] = E[\sum_{(i,j,k)} X_{ijk}] = \sum_{(i,j,k)} E[X_{ijk}]$ by linearity of expectation. Plugging in $E[X_{ijk}]$ we have $E[X] = \binom{n}{3}p^3$.

   (c) Recall that a k-clique is a set of k distinct vertices where there is an edge between each pair of vertices. What is the expected number of k-cliques in an Erdős-Rényi graph with parameters n and p?

   Solution. This is just a generalization of the expected number of triangles. The number of possible k-cliques is the number of sets of k distinct vertices which is $\binom{n}{k}$. The number of edges in a k-clique is $\binom{k}{2}$. So the overall probability is $\binom{n}{k}p^{\binom{k}{2}}$.

2. **Site Splitting** (6 points)

   After a recent hurricane, our company is forced to relocate temporarily. Unfortunately, the available buildings are not quite large enough for the whole company. At least one team will be need to be in a different building from the rest of the company, but at least only two buildings will be needed. The company is very collaborative, and would like to have as many meetings in person as possible without forcing people to move back and forth between buildings. You are given a list of meetings and their participants, and the team assignments of each person in the company, and asked to assign teams to one of two buildings to reduce the number of meetings spanning both buildings. Conveniently, each meeting only involves members of at most two teams. Show how to reduce this problem to the global min-cut problem.

Solution. We will reduce the problem of assigning teams to buildings to a global min-cut problem where each vertex is a team, and there is an edge between two teams for every meeting with participants from both teams. (So this is a multi-graph.) Each cut of this graph corresponds to an assignment to the two buildings; the first set of vertices maps corresponding teams to the first building and the second set of vertices maps corresponding teams to the second building. The cut value is the number of meetings spanning both buildings because the cut edges exactly correspond to meetings between teams assigned to different buildings.

3. **Content Resolution with Prioritization** (7 points)

You are responsible for a number of long running processes accessing a shared database. These processes access the database using the random attempt strategy discussed in class - each process attempts to access the database with probability $1/n$ at each time step where n is the number of processes, and an access attempt only succeeds if it is the only attempt at that time step. $n/10$ of these process responsible for handling financial transactions have been identified as high priority and need their performance improved. You may assume that n is a multiple of 10.

(a) The first tactic you try to improve performance is to simply halve the attempt rate of slow priority processes while keeping the attempt rate of high priority processes the same. That is, the high priority processes still make an attempt at time t with probability $1/n$ and low priority processes make an attempt at time t with probability $1/2n$.

For large n, how many time steps are necessary for every high priority process to succeed with probability at least $1 - 1/n$?

Hint: this should generally follow the proof strategy covered in class, but with adjustments for the attempt rate split. You may use $\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n = \frac{1}{e}$ *as an equality to simplify your reasoning.*

Solution. There are $n/10$ high priority processes attempting with probability $1/n$, and $9n/10$ low priority processes attempting with probability $1/2n$.

For a given high priority process i, the probability that process i succeeds at time t is

$$\begin{aligned}
\lim_{n\to\infty} P(S[i,t]) &= \lim_{n\to\infty}\left[\frac{1}{n}\left(1-\frac{1}{n}\right)^{0.1n-1}\left(1-\frac{1}{2n}\right)^{0.9n}\right] \\
&= \lim_{n\to\infty}\left[\frac{1}{n}\frac{\left(1-\frac{1}{n}\right)^{0.1n}}{1-\frac{1}{n}}\left(\left(1-\frac{1}{2n}\right)^{2n}\right)^{0.45}\right] \\
&= \lim_{n\to\infty}\left[\frac{1}{n-1}\left(\left(1-\frac{1}{n}\right)^{n}\right)^{0.10}\left(\left(1-\frac{1}{2n}\right)^{2n}\right)^{0.45}\right] \\
&= \left(\frac{1}{n}\right)\left(\frac{1}{e^{0.1}}\right)\left(\frac{1}{e^{0.45}}\right) \\
&= \frac{1}{e^{0.55}n} \approx \frac{0.58}{n}
\end{aligned}$$

The probability that process i does not succeed after t rounds is

$$\begin{aligned}
F(i,t) &= \Pi_{r=1}^{t}\left(1-P(S[i,r])\right) \\
&= \left(1-\frac{1}{e^{0.55}n}\right)^{t}
\end{aligned}$$

For all high priority processes to succeed with probability at least $1 - 1/n$, the probability that any of the $n/10$ high priority processes fail must be at most $1/n$. And via the union bound, it suffices to show that the probability of one of them failing is at most $10/n^2$.

Solving for $F(i,t) = 10/n^2$,

$$\begin{aligned}
\frac{10}{n^2} &= \left(1 - \frac{1}{e^{0.55}n}\right)^t \\
\log 10 - 2\log n &= t \log\left(1 - \frac{1}{e^{0.55}n}\right) \\
&= t \log\left(\left(1 - \frac{1}{e^{0.55}n}\right)^{(e^{0.55}n)(1/(e^{0.55}n))}\right) \\
&= t \log\left(\left(\frac{1}{e}\right)^{1/(e^{0.55}n)}\right) \\
&= t \log\left(e^{-1/(e^{0.55}n)}\right) \\
&= t\left(\frac{-1}{e^{0.55}n}\right) \\
t &= 2e^{0.55}n\log n - (\log 10)e^{0.55}n + \\
&\approx 3.47n\log n - 3.99n
\end{aligned}$$

(b) The previous tactic improved the performance of high priority processes, but was not explicitly optimized. Can better performance be achieved while maintaining the factor of two difference in attempt rates between high and low priority processes? State the best attempt rate for high priority processes as a fraction $\frac{c}{n}$ and its corresponding success rate per time step, and prove that the former maximizes the latter.

Hint: this should also follow the proof strategy covered in class, but just the part about maximizing the single step probability.

Solution. To allow the high priority attempt rate to be adjusted, we will optimize a constant c where the high priority processes attempt with probability c/n and the low priority processes attempt with probability $c/2n$. We will repeat the previous analysis of $P(S[i,t])$ with these modified formulas and then optimize the choice of c.

$$\begin{aligned}
\lim_{n\to\infty} P(S[i,t]) &= \lim_{n\to\infty} \left[\frac{c}{n}\left(1-\frac{c}{n}\right)^{0.1n-1}\left(1-\frac{c}{2n}\right)^{0.9n}\right] \\
&= \lim_{n\to\infty} \left[\frac{c}{n}\frac{\left(1-\frac{c}{n}\right)^{0.1n}}{1-\frac{c}{n}}\left(\left(1-\frac{c}{2n}\right)^{2n}\right)^{0.45}\right] \\
&= \lim_{n\to\infty} \left[\frac{c}{n-c}\left(\left(1-\frac{c}{n}\right)^{n/c}\right)^{0.10c}\left(\left(1-\frac{c}{2n}\right)^{2n/c}\right)^{0.45c}\right] \\
&= \left(\frac{c}{n}\right)\left(\frac{1}{e^{0.1c}}\right)\left(\frac{1}{e^{0.45c}}\right) \\
&= \frac{c}{e^{0.55c}}\frac{1}{n}
\end{aligned}$$

This probability will be maximized by picking c to maximize $\frac{c}{e^{0.55c}}$. Taking the derivative with respect to c,

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}c}\left(\frac{c}{e^{0.55c}}\right) &= \frac{1\,e^{0.55c} - c\,e^{0.55c}\,0.55}{e^{1.1c}} \\
&= \frac{(1-0.55c)e^{0.55c}}{e^{1.1c}} \\
&= \frac{(1-0.55c)}{e^{0.55c}}
\end{aligned}$$

and setting it to zero

$$\begin{aligned}
0 &= \frac{(1-0.55c)}{e^{0.55c}} \\
&= 1-0.55c \\
0.55c &= 1 \\
c &= \frac{1}{0.55} \approx 1.82
\end{aligned}$$

Substituting back into $P(S[i,t])$,

$$\begin{aligned}
\lim_{n\to\infty} P(S[i,t]) &= \frac{1/0.55}{e^{0.55/0.55}}\frac{1}{n} \\
&= \frac{1/0.55}{e^1}\frac{1}{n} \approx \frac{0.67}{n}
\end{aligned}$$

(c) Another measure of the performance of a system like this time is the average waiting time to get access to the shared resource. If a particular priority class has probability p of succeeding at any time step, what is the average number of time steps to get access? *Hint: this is easier to solve using geometric distributions.*

Solution.

$$
\begin{aligned}
f(p) &= \sum_{t=1}^{\infty} t\ p(1-p)^{t-1} \\
&= p + \sum_{t=2}^{\infty} t\ p(1-p)^{t-1} \\
&= p + \sum_{t=1}^{\infty} (t+1)\ p(1-p)^{t} \\
&= p + (1-p) \sum_{t=1}^{\infty} (t+1)\ p(1-p)^{t-1} \\
&= p + (1-p) \sum_{t=1}^{\infty} t\ p(1-p)^{t-1} + (1-p) \sum_{t=1}^{\infty} p(1-p)^{t-1} \\
&= p + (1-p) f(p) + (1-p) \\
&= 1 + (1-p) f(p) \\
pf(p) &= 1 \\
f(p) &= 1/p
\end{aligned}
$$