CS 401: Homework 5

Due on April 25, 2024 at 11:59pm

Professor Sidiropoulos 9:30am

Ryan Magdaleno
rmagd2@uic.edu

Problem 1: Evacuating a building.

In an emergency, a building has to be evacuated in a hurry. When a disaster strikes an evacuation plan should be in place and there are many aspects that such a plan takes into consideration: https://www.osha.gov/SLTC/etools/evacuation/evac.html. Let us consider a much simplified version. Let $G = (V, E)$ be a directed graph, representing a building. Every vertex $v \in V$ represents a room, and every edge $(u, v) \in E$ represents a corridor that leads from room u to room v (there may be also an edge (v, u)). For every corridor $(u, v) \in E$, at most $c_{u,v}$ people can traverse it simultaneously in the direction from u to v, where $c_{u,v}$ is a positive integer. Traversing a corridor takes one time step, and traversing a room takes zero time.

Suppose that there are k people in the building, all initially located in some room $s \in V$, and they all want to move to some room $t \in V$ (e.g., exit). Give an algorithm that computes the fastest possible way for all people to move from room s to room t. The running time of your algorithm should be polynomial in both n, and k (for example, $O\left(|V|^{10} k^{10}\right)$.

Hint: Clearly, this problem seems related to the max-flow problem. However, an additional complication in building evacuation is that the algorithm has to account for time. One possible approach for overcoming this obstacle is as follows: First, argue that the maximum time necessary to evacuate the building it at most $O(kn)$ (why?). It is enough to solve the following simpler "decision" version of the problem: Given some integer T , find a way to evacuate the building in at most T steps, if possible. Given an algorithm for the decision problem we can solve the original problem by trying all $O(kn)$ possible values for T. Finally, show that the decision problem can be reduced to a max-flow computation; that is, construct a flow network N_T, such that the building can be evacuated in at most T steps, if and only if the max-flow in N_T has value at least k.

Solution

kn time complexity

The reason the time to evacuate the building is $O(kn)$ is that for each person in the building (k), we need to find room t which in the worst case would require us to traverse n rooms, therefore for everyone to exit the building, the time complexity is $O(kn)$.

High Level Algorithm Explanation

1. Create a flow network N with n vertices representing the rooms, and edges representing each corridor (u, v) with capacity $c_{u,v}$.
2. Connect a source node A to the starting room s with an edge capacity of k.
3. Connct a sink node Z to the exit room t with an edge capacity of ∞.
4. Construct a flow network N_T based on our building graph and some given T time constraint, check all values ranging from $T = 1$ to $T = kn$ until N_T's max flow is at least k, this minimum T value is the shortest amount of time. This can be sped up with binary search rather than a linear check. Adjust the capacities of the edges to account for time, for edges between rooms, if T is $<$ than the $c_{u,v}$ of that room, set it to T, else keep $c_{u,v}$, this prevents the flow from exceeding the time constraint.

Problem 1 continued on next page...

Time Complexity Justification

Constructing the flow network requires checking each corridor, resulting in $O(n^2)$, where n is the number of vertices/rooms in the graph.

The binary search to find the T value takes $O(\log(kn))$.

The capacity adjustments and maximum flow calculations is done against each binary search, the capacity adjustments take $O(n^2)$ time due to us modifying/checking each corridor. Computing the maximum flow using Ford-Fulkerson's algorithm takes $O(maxflow \cdot n^2)$ where n^2 is the number of edges in the graph, the maxflow can be at most k so we get $O(n^2 \cdot k)$.

Putting everything together we get the following time complexity:

$$T(n,k) = O(n^2 + \log(kn) \cdot n^2 \cdot k)$$
$$T(n,k) = \boxed{O(\log(kn) \cdot n^2 \cdot k)}$$

where n is the number of rooms and k is the number of people.

Problem 2: Paper reviewing.

Computer science is "the only scientific community that considers conference publication as the primary means of publishing our research results" (rather than journals): https://cacm.acm.org/opinion/conferences-vs-journals-in-computing-research/. In other disciplines, a "publication" means a paper has appeared in what is known a "refereed" and indexed (by an officially recognized body, such as the Web of Science) journal. So in computer science, the conference publications are reviewed (by volunteer researchers, such as your professors, for the so-called honor of having a line in your resume that says they served as a member of the conference Program Committee). Suppose a conference receives n paper submissions and there are m members of the Program Committee. The Program Committee members then bid on which papers they can and are willing to review. After that, reviewers are assigned to papers (from among those they bid on) so that each paper has, lets say, at least 3 reviewers and each reviewer reviews at most, lets say, 5 papers. (The reviewers collective scores then decide whether a paper is accepted for the conference or not. Typically, for a good conference, only about 20% of the papers are accepted.)

Design and analyze a polynomial-time algorithm to make the assignment of reviewers to papers (so that each reviewer reviews only the papers they bid on and the min and max constraints are respected) or decide that no feasible assignment exists.

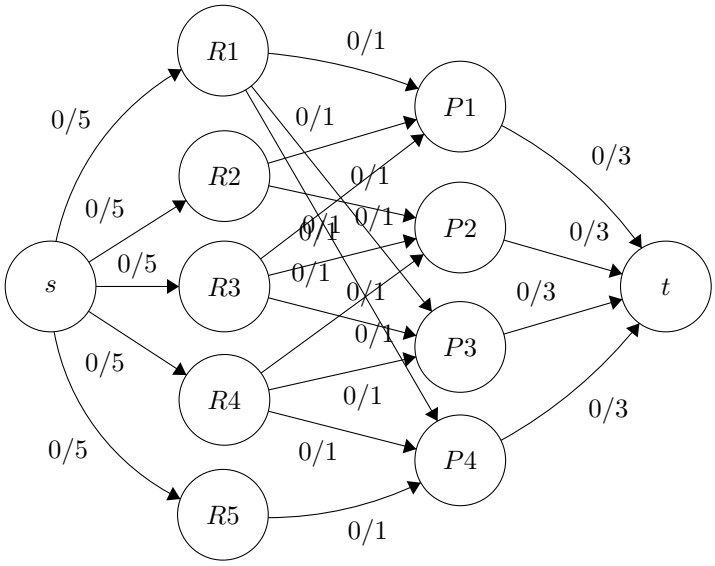
Solution

High Level Algorithm Explanation

1. We need to construct a directed bipartite graph flow network as follows:
   (a) We create nodes representing the reviewers R_1 to R_m.
   (b) We create nodes representing the papers P_1 to P_n.
   (c) A source node s and a sink node t.
   (d) We shall assign capacities from the source node s to each reviewer, each reviewer can review at most 5 papers, so a capacity of 5 is needed.
   (e) We shall assign capacities from each reviewer to each of their desired papers, a capacity of 1 is needed because that reviewer can review that paper only once.
   (f) We shall assign capacities from each paper to the sink node t, each paper will have a capacity of 3 representing the minimum reviewers needed for that paper.
2. We can now use Ford-Fulkerson's algorithm to find the maximum flow in our newly constructed network.
3. We must ensure that each paper has at least 3 reviewers and each reviewer has at most 5 papers assigned, we shall do the following to check this:
   (a) For each paper node P_1 to P_n, sum up the flow coming into it. This represents the number of reviewers assigned to the current paper, if the flow sum is < 3 then reject, else continue.
   (b) For each reviewer node R_1 to R_m, sum up the outgoing flow, each outgoing edge representing a desired paper, the total flow is the number papers assigned, if the flow for the current reviewer node is > 5 then reject, else accept.

Problem 2 continued on next page...

Here's an example of my proposed flow network.



Time Complexity Justification

Creating the flow network graph involves iterating over all papers and reviewers to create capacity 1 edges, this takes $O(n \cdot m)$ time.

Adding the source and sink capacity connections can be done during the previous step on each node giving a $O(n \cdot m)$ time complexity.

Computing the maximum flow using Ford-Fulkerson's algorithm takes $O(maxflow \cdot (nm + n + m))$, note the $nm + n + m$ term represents the total edges in the graph. nm = reviewers to papers, n = papers to sink, and m = source to reviewer connections. This is determined by the number of augmenting paths needed to reach the maximum flow.

All together our combined time complexity is as follows:

$$T(n, m, maxflow) = \boxed{O(nm + maxflow \cdot (nm + n + m))}$$

where n is the number of papers and m is the number of researchers.
