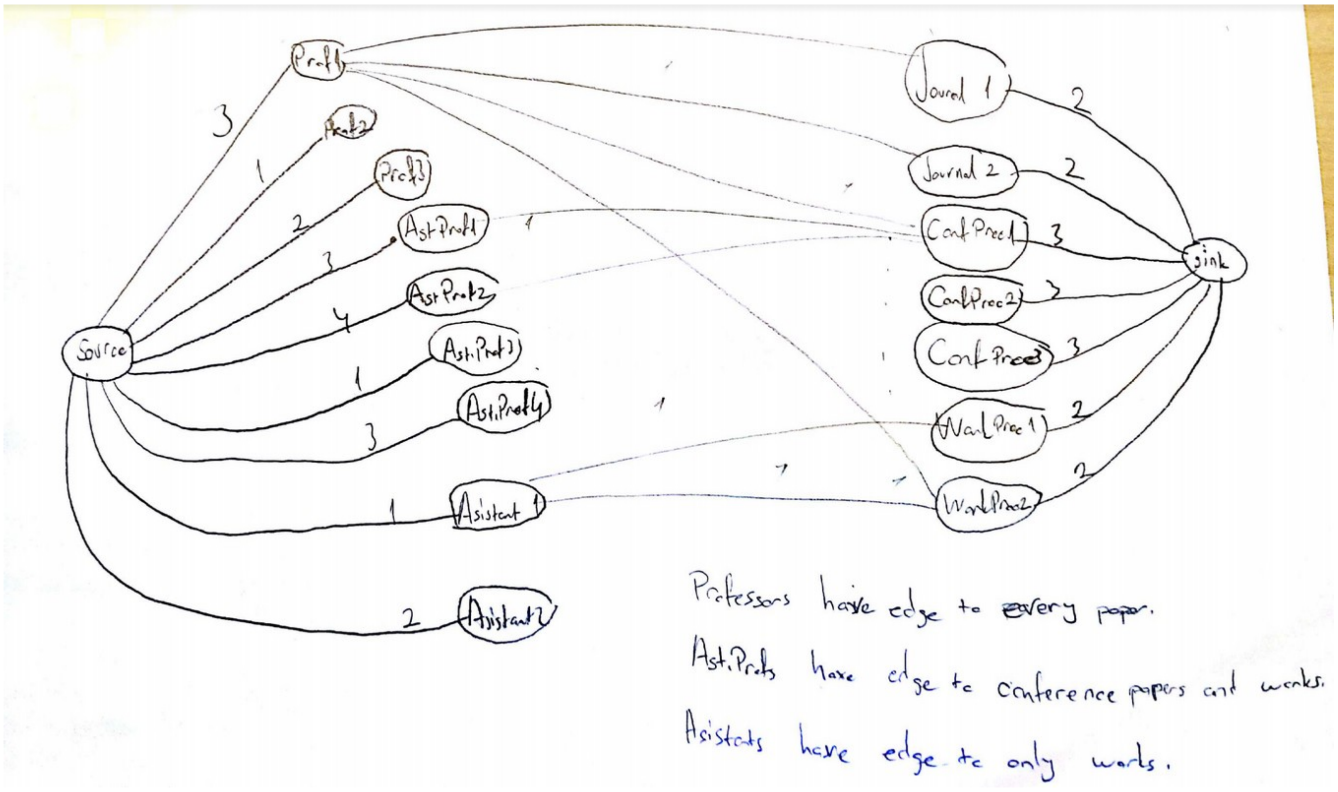
BLG336E – Analysis of Algorithms Project 3 Report

1) In my design, I put an edge between source to reviewers which has capacity equal to their time for spending to read papers. In addition, there are edges between papers to sink which has capacity equal to required readings of that paper to be accepted.(For instance, journal1 has 2 weighted edge to sink, because it needs to be read from 2 distinct reviewers)
 For making sure that each reviewer only checks a paper once, there is an edge between reviewers to the papers which has 1 weight that can read.(For instance, professor has edge to every paper, because they can review every paper, however, assistants has only edge to workshop papers)



2) I used an improved version of Ford-Fulkerson algorithm, I created an edge list to represent the graph but I only used in BFS to improve is performance. Residual graph is adjacency list which allows to back-track the path easier. However, it is not used in iterations in BFS, so it did not caused extra complexity. My algorithm is classic network flow algorithm, at initial state it finds a path from source to sink, if there is a chance to augment a path, it augments a new path and continues iterating until reaching no flow can be transferred anymore. Normally, Ford-Fulkerson algorithm has O(E * maximum_flow) complexity where E is equal to number of edges in list and assuming the weight of edges are integer and we're decrementing the capacity by one. However, finding augmenting paths is not O(1) duty, it has O(V^3) complexity with adjacency list and O(V^2) complexity with edge list. So the total complexity of algorithm O(E * V^3) complexity with adjacency list but using edge list drops the complexity to O(E * V^2) .

Şahin Olut
150140124