

MAX-FLOW, MIN-CUT, FORD-FULKERSON'S Analysis

V1.2: Steven Halim

October 01, 2018

Preliminaries

We will start T04 by asking some quick questions from Lecture 05+06 combo.

- Write MAX-FLOW as a Linear Program. Again, are you going to solve MAX-FLOW that way?
- What is the MAX-FLOW of the (random, but reasonably small) Flow Network drawn in class (using <https://visualgo.net/en/maxflow>)? Run one Ford Fulkerson's based algorithm manually (Tutor will randomly select either basic Ford Fulkerson's method/simple DFS, Edmonds Karp's algorithm/BFS, or Dinic's algorithm/BFS/layer graphs) before executing the animation to show your tutor/classmates that you have fully understand the algorithm.

Discussion Points

Q1: Show how to use (standard) Ford-Fulkerson's algorithm to find a maximum sized matching on *Bipartite* Graph, or formally known as the MAX-CARDINALITY-BIPARTITE-MATCHING (MCBM). Prove that the result is a matching, and that it is the maximum-sized matching. Analyze the running time of your algorithm. Note that we have seen the topic of Graph Matching briefly in Lecture 1 (Deterministic Vertex Cover-2), Lecture 4 (part of Christofides's algorithm), and will properly come back to this topic on Lecture 8.

Q2a: Assume that you have an undirected graph $G = (V, E)$ with a source vertex s and a target vertex t (the graph is *unweighted*). Give an algorithm that finds the maximum number of *edge-disjoint paths* from s to t . Two paths P_1 and P_2 are called edge-disjoint if they do not share any edges—but they may share a vertex. (Hint: Use (standard) Ford-Fulkerson's algorithm.) What is the running time of your algorithm?

Q2b: What if you want to find the maximum number of *vertex-disjoint paths* from s to t instead? Two paths are called vertex-disjoint if they do not share any vertex.

Q3: Please read <https://uva.onlinejudge.org/external/113/11380.pdf> and give the correct Flow Graph to solve this problem.

Q4: Please read <https://uva.onlinejudge.org/external/117/11757.pdf> and try to reduce this problem into a max flow problem, solve it using $O(n^2 \times m)$ Dinic's algorithm (assuming that you have such implementation ready), and analyze its time complexity.

Q5a: Can we write the runtime of Ford-Fulkerson's algorithm as $O(m \times F)$ where F is the eventual max flow value of the input graph? This is an output-sensitive analysis, whereby the runtime speed of the algorithm depends on the size of the output.

Post Tutorial

Students can discuss PS2 solutions with the lecturer/tutor if you still have doubts (PS2 version V maybe requires too much time to explain if you have not taken CS3233 or involved in ICPC/IOI before). The visualization page <https://visualgo.net/en/steinertree> will be made better over time.