AAD Problem Set - 4

October 2025

Flow Algorithms

Problem 1

Show that splitting an edge in a flow network yields an equivalent network. More formally, suppose that flow network G contains edge (u, v) and define a new flow network G' by creating a new vertex x and replacing (u, v) by (u, x) and (x, v) with $c(u, x) = c(x, v) = c(u, v)$. Show that maximum flow in G' has the same value as a maximum flow in G.

Problem 2

Try to extend the flow properties and defitiions to the multiple-source, multiple-sink problem and show that any flow in a multiple-source, multiple-sink flow network corresponds to a flow of identical value in the single-source, single-sink network obtained by adding a supersource, supersink. And vice-versa.

Problem 3

Suppose that in addition to edge capacities, a flow network has *vertex capacities*. That is each vertex v has a limiit $l(v)$ on how much flow can pass through v. Show how to transform a flow network $G = (V, E)$ with vertex capacities into an equivalent flow network $G' = (V', E')$ without vertex capacities, such that a maximum flow in G' has the same value as a maximum flow in G. How many vertices and edges does G' have.

Problem 4

Show the execution of Ford-Fulkerson algorithm on the flow network of Fig-4.1

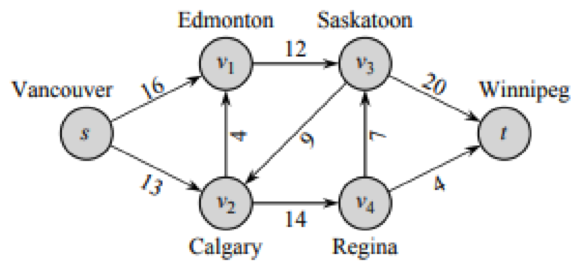

Fig-4.1

Problem 5

Show how to find a max flow in a flow network $G = (V, E)$ by a sequence of atmost $|E|$ augmenting paths. (*Hint:* Determine the paths *after* finding the max flow.)

Problem 6

The *edge connectivity* of an undirected graph is defined as the minimum number k of edges that must be removed to disconnect the graph. For example, the edge connectivity of a tree is 1, and the edge connectivity of a cyclic chain of vertices is 2. Show how to determine the edge connectivity of an undirected graph $G = (V, E)$ by running a maximum-flow algorithm on at most $|V|$ flow networks, each having $O(V + E)$ vertices and $O(E)$ edges.

Problem 7

Let $G = (V, E)$ be a flow network with source s, sink t, and integer capacities. Suppose that we are given a maximum flow in G.

(a) Suppose that we increase the capacity of a single edge $(u, v) \in E$ by 1. Give an $O(V + E)$-time algorithm to update the maximum flow.

(b) Suppose that we decrease the capacity of a single edge $(u, v) \in E$ by 1. Give an $O(V + E)$-time algorithm to update the maximum flow.

Approximation Algorithms

Problem 8

Give an example of a graph for which 2−approximation algorithm for vertex cover problem yields a suboptimal solution. Also give an example scenario for which it yields an optimal solution.

Problem 9

Prove that set of edges picked in 2−approximation algorithm for vertex cover forms a maximal matching in the graph G

Problem 10

Give an efficient greedy algorithm that finds an optimal vertex cover for a tree. Can you find a linear time algorithm for the same?

Problem 11

Both min vertex-cover and clique problem are NP-complete. Infact, they are complementary in the sense that an optimal-vertex cover is the complement of a maximum-size clique in the complement graph. Does this imply that there is a poly-time approximation algorithm with constant approximation ratio for the clique problem? Justy your answer.

Problem 12

Let a graph $G = (V, E)$ be a complete undirected graph containing atleast 3 vertices, and let c be a cost function that satisfies the triangle inequality. Prove that $c(u, v) \geq 0$ for all $u, v \in V$.

Problem 13

Assuming $P \neq NP$ show how in polynomial time to transform one instance of the traveling-salesperson problem into another instance whose cost function satisfies the triangle inequality. The two instances must have the same set of optimal tours. Explain why such a polynomial time transformation does not contradict the fact that for any constant $\rho \geq 1$, there is no poly-time approximation algorithm with approximation ratio ρ for the general traveling-salesperson problem.

NP-completeness and Reductions

Pitfalls

Make sure that you don't get the reduction backwards. That is, in trying to show that a problem Y is NP-complete, you might take a known NP-complete problem X and give a polynomial-time reduction from Y to X. That is the wrong direction. The reduction should be from X to Y, so that a solution to Y gives a solution to X.

Additionally remember that reducing a known NP-complete problem X to a problem Y does not in itself prove that Y is NP-complete. It proves that Y is NP-hard. In order to show that Y is NP-complete, you additionally need to prove that it's in NP by showing how to verify a certificate for Y in a polynomial time.

Problem 14

Prove that for a language L, $L \leq_p \overline{L}$ if and only if $\overline{L} \leq_p L$.

Problem 15

Show that the problem of determining the satisfiability of boolean formulas in disjunctive normal form is polynomial-time solveable.

Problem 16

Formally state the complexity classes P, NP, co-NP, NP-hard and NP-complete. Give an example of problem for each of the classes.

Problem 17

Is the dynamic-programming algorithm for 0-1 knapsack problem a polynomial-time algorithm? Prove it **rigorously** for a full credit. What happens if we restrict the knapsack size to a polynomial in n?

Problem 18

DOUBLE-SAT $= \{\langle \varphi \rangle \mid \varphi$ is a Boolean formula that has at least two distinct satisfying assignments$\}$.

Show that DOUBLE-SAT is NP-complete.

Problem 19

3SAT is polynomial time reducible to CLIQUE.

$$\text{CLIQUE} = \{\langle G, k\rangle \mid G \text{ is an undirected graph with a } k\text{-clique}\}.$$

A clique in an undirected graph is a subgraph, wherein every two nodes are connected by an edge. A k-clique is a clique that contains k nodes.

Problem 20

If G is an undirected graph, a vertex cover of G is a subset of the nodes where every edge of G touches one of those nodes. The vertex cover problem asks whether a graph contains a vertex cover of a specified size.

$$\text{VERTEX-COVER} = \{\langle G, k\rangle \mid G \text{ is an undirected graph that has a } k\text{-node vertex cover}\}.$$

Show that 3SAT is polynomial time reducible to VERTEX-COVER.

Problem 21

Show that the hamiltonian-path problem is NP-complete.

Problem 22

$$\text{LPATH} = \{\langle G, a, b, k \rangle \mid G \text{ contains a simple path of length at least } k \text{ from } a \text{ to } b\}.$$

Show that LPATH is NP-complete.

Problem 23

Show that the decision version of the set-covering problem is NP-complete by reducing the vertex-cover problem to it.