Class 28

MAXIMUM FLOW PROBLEM

Maximum Flow Problem

Maximizing the flow through a network is an important problem:

- Traffic flow
- Network flow
- Flow of electricity

Let us assume that we can represent such a problem by a connected weighted digraph with *n* vertices.

Maximum Flow Problem

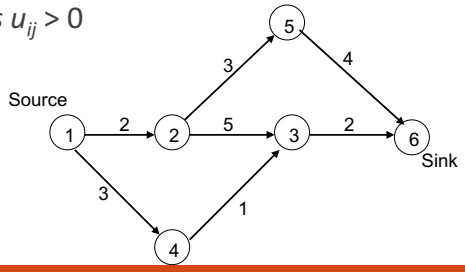
Maximum flow problem: maximize flow from source to sink while staying under edge capacities

A single *source* vertex v_1 with no entering edges.

A single *sink* vertex v_n with no leaving edges.

Edges (i,j) have *capacities* $u_{ij} > 0$

Example:



Flow-Conservation Requirement

Flow-conservation requirement: For all intermediate vertices:

total inflow = total outflow

$$\sum_{j:(j,i)\in E} x_{ji} = \sum_{j:(i,j)\in E} x_{ij} \quad \text{for } i = 2, 3, \ldots, n-1,$$

Nothing gets added, nothing gets removed.

This implies that the total going into the network at the source must end up at the sink:

$$\sum_{j:(1,j)\in E} x_{1j} = \sum_{j:(j,n)\in E} x_{jn}.$$

Formal Problem Statement

Let the *capacity constraints* be such that:

$$0 \leq x_{ij} \leq u_{ij} \quad \text{for every edge } (i,\ j) \in E.$$

A *flow* is an assignment of flow values x_{ij} to edges (*i*,*j*) of a given network that satisfy the flow-conservation requirements and the capacity constraints.

Maximum Flow Problem

$$\text{maximize} \quad \sum_{j:(1,j)\in E} x_{1j}$$

subject to

$$\sum_{j:(j,i)\in E} x_{ji} - \sum_{j:(i,j)\in E} x_{ij} = 0 \quad \text{for } i = 2, \ldots, n-1$$

$$x_{ij} \leq u_{ij} \quad \text{for every edge } (i,j) \in E$$

$$x_{ij} \geq 0 \quad \text{for every edge } (i,j) \in E$$

Can be solved using simplex method or other linear programming solver.

However, the special structure allows a more efficient problem-specific algorithm.

Augmenting Path Method

Also called the Ford-Fulkerson method.

Start with the zero flow ($x_{ij} = 0$ for every edge)

On each iteration, try to find a *flow-augmenting path* from source to sink, i.e. path that can handle some additional flow

- If one is found, adjust the flow along the edges of this path to get a flow of increased value and try again
- If none found, the current flow is maximum. We'll prove correctness of this later…

Method, not algorithm, since how to find flow-augmenting path is not specified.

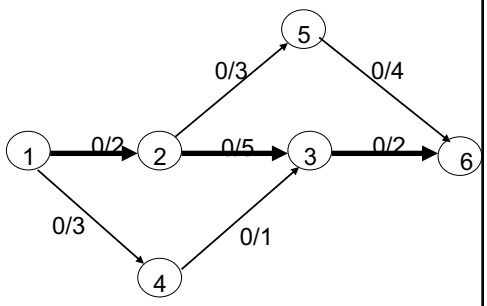
Ford-Fulkerson Example

Consider the earlier example.

The edges are annotated with assigned_flow/capacity.

At first, we have zero flow. Suppose we identify the flow augmenting path: 1→2 →3 →6

Since the smallest capacity along that path is 2, we update edges (1,2), (2,3) and (3, 6) to 2. (next slide)



Ford-Fulkerson Example

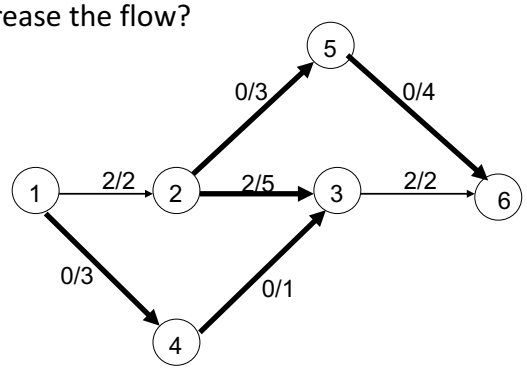
Since the smallest capacity along that path is 2, we update edges (1,2), (2,3) and (3, 6) to 2.

How can we further increase the flow?

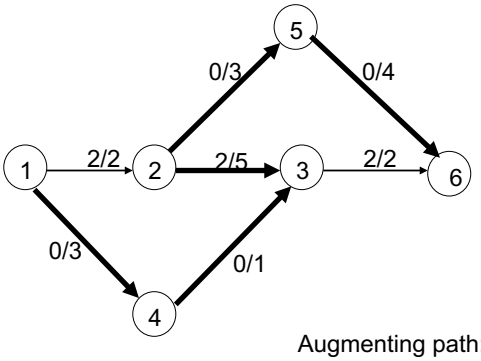
Consider adding 1 on path 1→4 →3.

Problem: Flow from 3→6 has capacity of 2

Solution: Add 1 unit of flow to 2→5 →6 syphoning it from 2→3



Ford-Fulkerson Example



Augmenting path:
1 →4 →3←2 →5 →6

Ford-Fulkerson Example

We now have the following with a max flow value of 3.
How could one show that this is indeed the maximum?