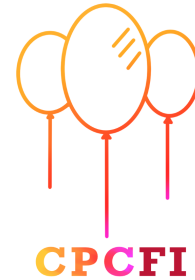




Lecture: Flow Networks
Unit: 7.4
Instructor: Carlos C.L



Flow network

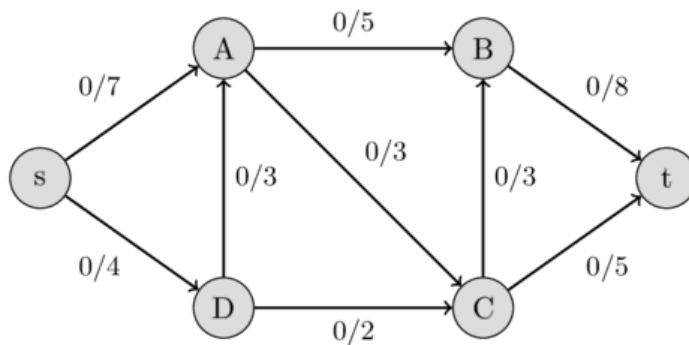
A directed graph $G(V, E)$ combined with a capacity function $c: E \rightarrow \{R > 0\}$ and two vertices, a source(s) and a sink(t).

Flow

A function $f: E \rightarrow \{R > 0\}$. It must satisfy the following conditions:

$$f(e) \leq c(e)$$

$$\sum f((u, v)) = \sum f((v, u)) \text{ for any node } v$$



We can prove the following holds given this conditions:

$$\sum f((s, u)) = \sum f((u, t)) = \text{flow of the network}$$

Ford-Fulkerson (published 1956)





Our problem is to find the flow function f , that maximizes the flow of the network.

Residual Graph

The residual graph is the same graph but with **reverse edges** and a different capacity function, the **residual capacity**.

Residual capacity

How much we can increase the flow for a given edge:

$$rc(e) = c(e) - f(e)$$

Reverse edges

For any edge (u, v) in the original graph, we define a reverse edge (v, u) in the residual graph

$$f((v, u)) = -f((u, v))$$

$$c((v, u)) = 0$$

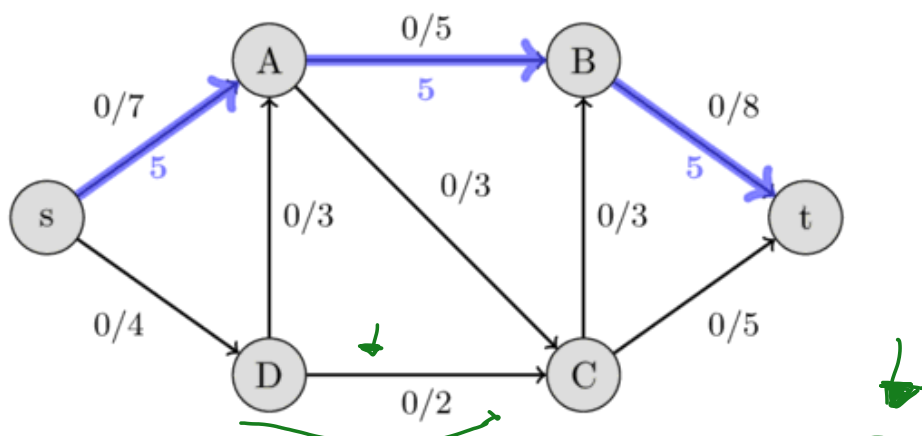
$$rc((v, u)) = f((u, v))$$

Augmented path

It's a path on the residual graph with all edges having positive residual capacity.

Bottleneck capacity

The lowest residual capacity on an augmented path.



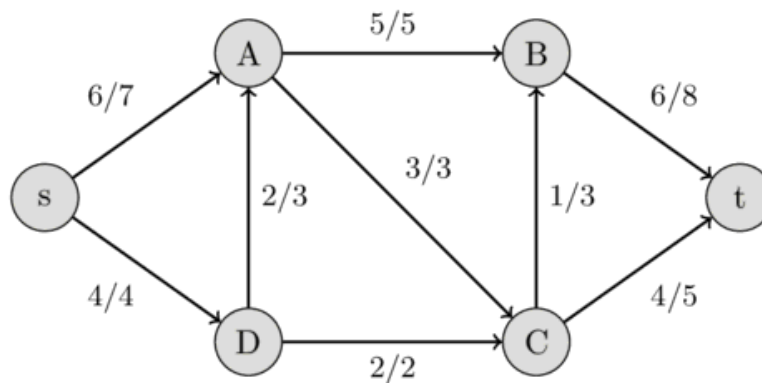
SPFA
 $O(M)$

~~dfs~~ ~~dfs~~ ~~dfs~~
 ~~$O(MN)$~~

Algorithm

1. Find augmented path from s to t (use ~~BFS~~ or ~~DFS~~)
2. Find bottleneck capacity = bc
3. Augment each edge on the path
$$f((u, v)) += bc$$
$$f((v, u)) -= bc$$
$$rc((u, v)) -= bc$$
$$rc((v, u)) += bc$$
4. Repeat until no augmented paths can be achieved

Solution



Edmonds Karp

If we use BFS for step one then we are solving with Edmonds Karp.

Complexity:

Time: $O(EF)$, E = #Aristas, F = Max flow

Time Edmonds Karp: $O(VE^2)$

Space Edmonds Karp: $O(V+E)$

Problems:

[CSES Download Speed](#)

Dinic's Algorithm

Solves max flow problem.

Dinic's Algorithm | Network Flow | Graph Theory

Layered Network

A layered network is a new graph network constructed in the following way:

1. Build residual network
2. Build level array
 $\text{level}[v] = \text{min distance (unweighted) from } s \text{ to } v \text{ using only edges with } rc(e) > 0$
3. Only keep edges where (u, v)
 $\text{level}[v] = \text{level}[u] + 1$

Augmenting path on a layered graph

Notice an augmenting path on a layered graph will satisfy:

1. A path e_1, \dots, e_k on a residual graph where:
 - $rc(e_i) > 0$ for all i
 - $\text{level}[e_{i+1}] = \text{level}[e_i] + 1$ for all $i < k$

Blocking flow

A flow function, where any path from s to t contains and edge where $rc(e) = 0$

Algorithm

1. Build level array with BFS

$\text{level}[v] = \text{min distance (unweighted) from } s \text{ to } v \text{ using only edges with } rc(e) > 0$

2. If sink not reached while doing level array return max flow
3. Find disjoint (do not share edges) augmenting paths from s to t until a blocking flow is reached, for each augmenting path update flow with bottleneck capacity (bc):

$$\begin{aligned} f((u, v)) &+= bc \\ f((v, u)) &- = bc \\ rc((u, v)) &- = bc \\ rc((v, u)) &+= bc \end{aligned}$$

4. Repeat step 1

Complexity

Time complexity: $O(V^2E)$

Proof

We first need to prove the following:

$$\text{level}_{i+1}[t] > \text{level}_i[t]$$

Since $\text{level}_i[t]$ cannot be greater than V , the algorithm must terminate in less than V iterations.

In each iteration we will have to check at most all edges E to find the

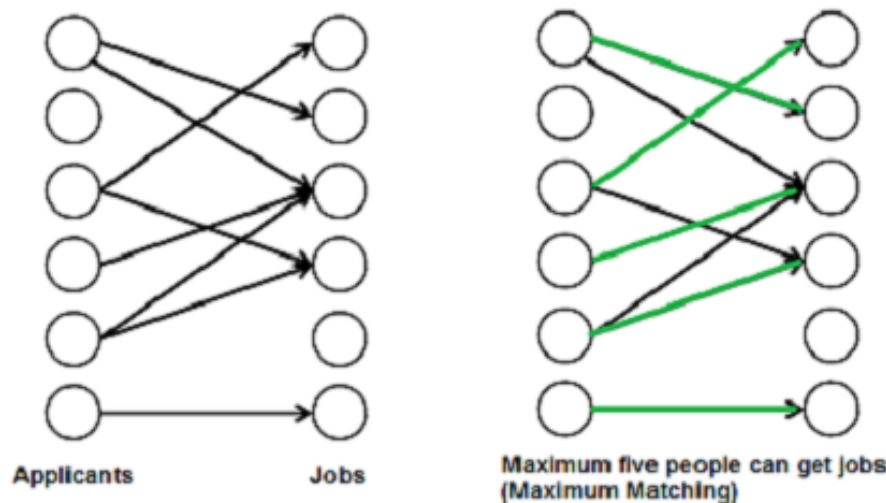
For unit networks

A unit network is a network where all edges have unit capacity.

Time complexity: $O(V^{1/2}E)$

Unweighted Bipartite Matching

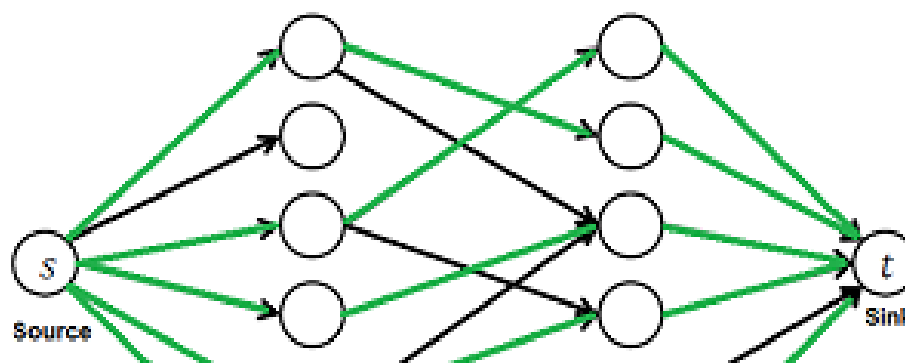
We have a bipartite graph representing applicants and jobs. There is an edge (applicant, job) if the applicant has the requirements to do the job.



We want to **maximize the number of applicants that get a job.**

Idea

1. Transform problem to max flow problem
2. Make an imaginary source and target, connect source to applicant and jobs to applicants
3. Find max flow in this graph



The maximum flow from source to sink is five units. Therefore, maximum five people can get jobs.

Problems:

CSES School Dance

Matching Problems

Common matching variations		
	Easier	Harder
Easier	Bipartite	Non-Bipartite
Unweighted Edges	<ul style="list-style-type: none"> • Max flow algorithms • Repeated augmenting paths with dfs • Hopcroft-Karp 	<ul style="list-style-type: none"> • Edmond's blossom algorithm
Weighted Edges	<ul style="list-style-type: none"> • Min cost max flow algorithms • Hungarian algorithm (perfect matching) • LP network simplex 	<ul style="list-style-type: none"> • DP solution for small graphs
Harder		

Resources:

[Unweighted Bipartite Matching](#) | [Network Flow](#) | [Graph Theory](#)

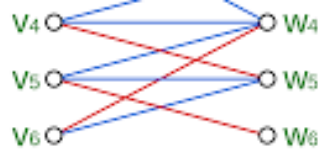
Assignment problem

First Example:

There are N jobs and N workers. Some jobs can only be done by some workers, and we know the cost it takes for the worker to do the job. Make a 1:1 assignment (each worker gets assigned only one job and each job gets assigned only one worker) so that the total cost is minimized.

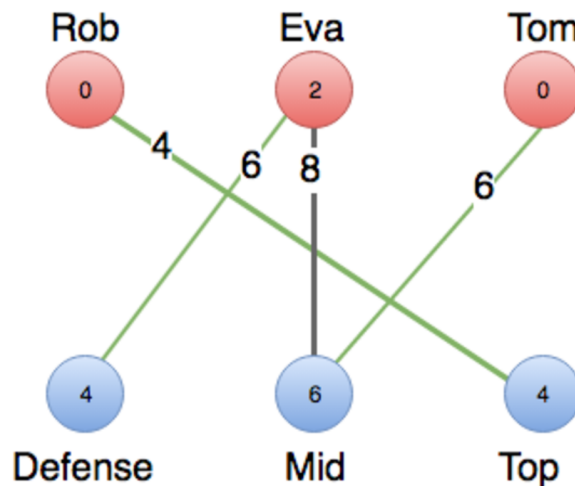
	W1	W2	W3	W4	W5	W6	
V1	1	1	0	0	0	0	V1
V2	1	1	1	1	0	0	V2
V3	0	1	1	0	0	0	V3

V4	0	0	1	1	1	0
V5	0	0	0	1	1	1
V6	0	0	0	1	1	0



Second Example:

A football coach wants to assign a position to each of its players, he knows the skill of each player for the positions. **Make a 1:1 assign (each player has only one position and each position has only one player) so that the total skill is maximized.**



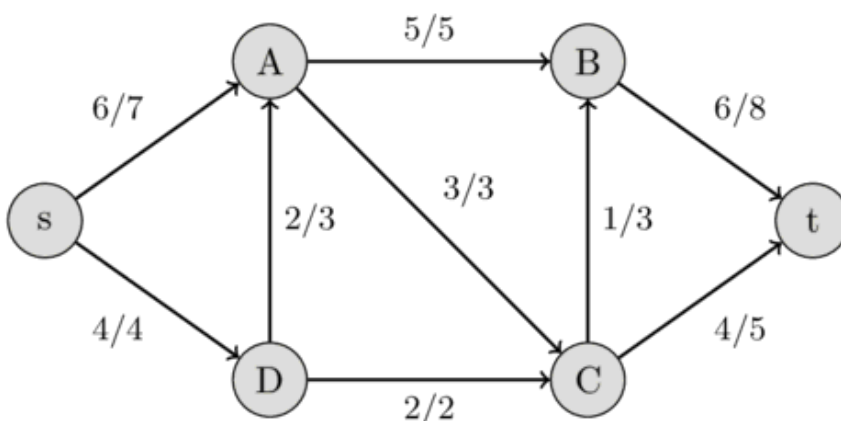
Min Cost Max Flow Problem

Suppose we have a network flow, $G(V, E, \text{capacity})$ and a **cost function**, $\text{cost}: E \rightarrow \{R > 0\}$.

$\text{cost}(e)$ = cost per unit flow for edge e

$\text{Total cost}(G) = \sum \text{cost}(e) * \text{flow}(e)$

We want to find the max flow in this network **and the lowest possible total cost to achieve this flow.**



Algorithm:

1. Find augmented path from s to t **with the lowest possible cost** (use Bellman Ford or SPFA)
2. Find bottleneck capacity = bc
3. Augment each edge on the path
$$f((u, v)) += bc$$
$$f((v, u)) -= bc$$
$$rc((u, v)) -= bc$$
$$rc((v, u)) += bc$$
4. Repeat until no augmented paths can be achieved

Solve Initial problem using Min cost Max Flow:

- What nodes should we create and how should we connect them to the graph?

Complexity:

Time: $O(V^5)$

Hungarian Algorithm

It can solve the assignment problem on $O(V^3)$

Problems:

[CSES Police Chase](#)

Resources:

[Brilliant – Matching Problem](#)

Topological Sort

Motivation

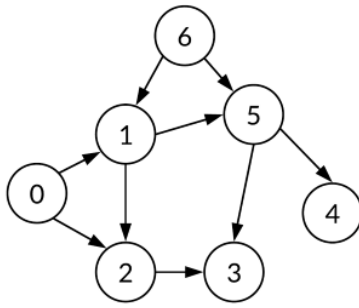
Suppose we are given a set of classes and an ordering in which the classes must be taken, this is, a number of pairs (x, y) where:

(x, y) implies x must be taken before y

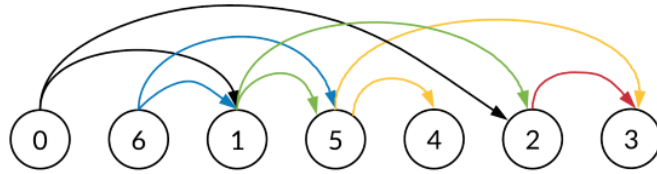
Return any valid ordering (permutation of courses) that is valid, this is, we can take the courses following that

ordering.

Unsorted graph



Topologically sorted graph



Algorithm

1. Build degree array:

$\text{degree}[i] = \text{\#Nodes pointing to } i$

2. Create queue, which will contains nodes with $\text{degree}[v] = 0$

3. Pop node u from queue and update degree for its child's:

$\text{degree}[v] -= 1$ for all v child's of u

4. Add all nodes with degree 0 to the queue.

5. Repeat from step 3 until there are no nodes left

Problems

[Leetcode Course Schedule](#)

[CSES Game Routes](#)