Network flow

Input: $\begin{cases} \text{directed graph } G(V, E) \\ c - \text{int} (\geq 0) \\ s, t \end{cases}$ $\rightarrow$ max flow $s \to t$

flow: function $f(e)$ — amount on edge e

Constraint $\begin{cases} \text{respecting capacities} & 0 \leq f(e) \leq c(e) \\ \text{flow conservation} & \sum_{\text{entering}} f(e) = \sum_{\text{leaving}} f(e) \end{cases}$

Residual graph $\begin{cases} \text{forward edge: } c(e) - f(e) \\ \text{reverse edge: } f(e) \end{cases}$

Augmenting paths

P — s-t path in residual graph G_f

bottleneck (P, f) — smallest capacity across all edges in P

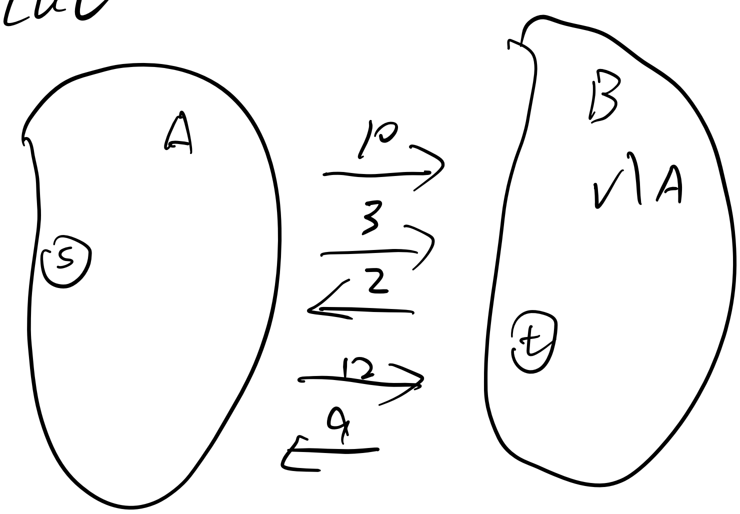
Ford-Fulkerson Algorithm

```
flow = 0
for each edge (u, v) in G:
    flow(u, v) = 0
while there is a path, p, from s -> t in residual network G_f:
    residual_capacity(p) = min(residual_capacity(u, v) : for (u, v) in p)
    flow = flow + residual_capacity(p)
    for each edge (u, v) in p:
        if (u, v) is a forward edge:
            flow(u, v) = flow(u, v) + residual_capacity(p)
        else:
            flow(u, v) = flow(u, v) - residual_capacity(p)
return flow
```

n vertices
$2m$ edges $\} \Rightarrow O((m+n) \cdot C)$

C ↑ max flow

cut



sum of capacity leaves A

$cap(A,B) = 25$

$$v(f) = f^{out}(A) - f^{in}(A)$$
$$\leq cap(A,B)$$
$$\Rightarrow \max v(f) \leq \min cap(A,B)$$

Max flow = min cut

Edmonds-Karp Algorithm $O(|V| \cdot |E|^2)$

L use BFS to find shortest s-t path in G_f

critical edge - saturate $\Rightarrow$ bottleneck(P, f) = capacity(e)

Bipartite Matching

perfect : flow $= n = |U| = |V|$

Hall's Marriage Theorem

perfect matching $\Rightarrow$ $|N(S)| \geq |S| \quad \forall S \subseteq U$

Set of nodes in V adjacent to node in S

Maximum # of edge disjoint $s \to t$ paths
= max flow = min cut

Merger's Theorem

edge-disjoint s-t path = minimum number of edges (removal disconnects s and t)

demand $\leftrightarrow$ loss

$d > 0 \rightarrow$ demand node

$d < 0 \rightarrow$ supply node

$d = 0 \rightarrow$ Transshipment node

G': supply node $\rightarrow$ add edge with $c = -d$

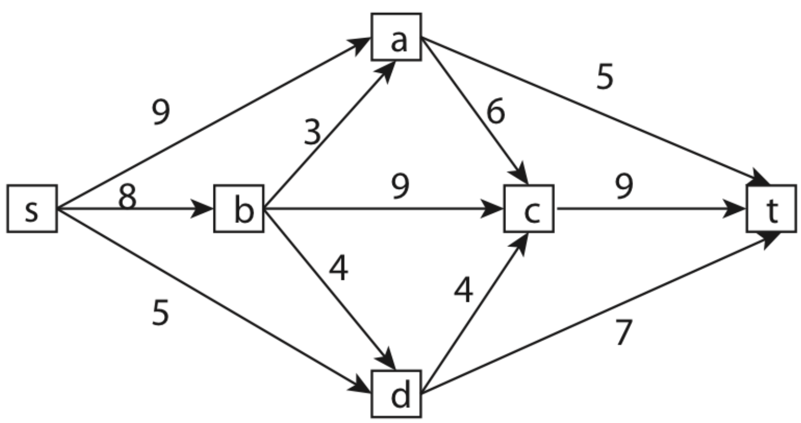
demand node $\rightarrow$ add edge with $c = d$

G has circulation iff G' has max flow

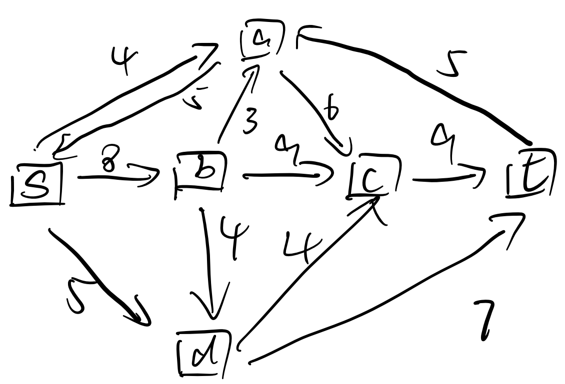
$$\sum_{\substack{v \\ d(v)>0}} d(v) = \sum_{\substack{v \\ d(v)<0}} -d(v)$$

Tutorial 4

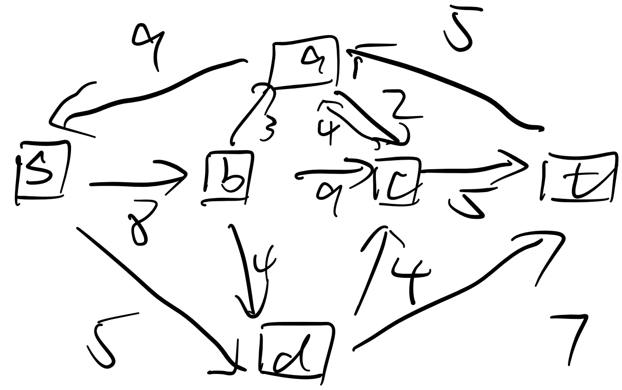
1)



① (s, a, t)   r.c. = 5 |f| = 5



1



2

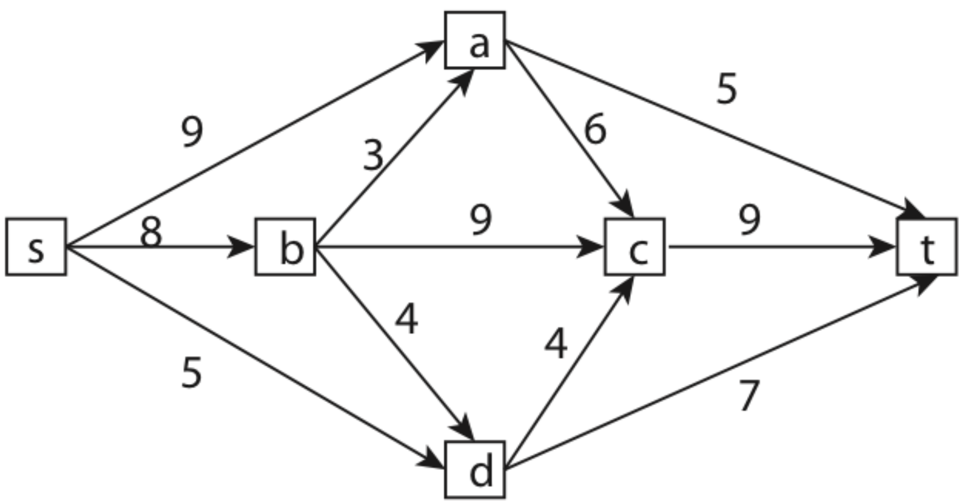
② (s, a, c, t) r.c. = 4 |f| = 9

③ (s, b, c, t) r.c. = 5 |f| = 14

④ (s, d, t) r.c = 5 |f| = 19

⑤ (s, b, d, t) r.c = 2 |f| = 21

(b) Consider the cut $X_0 = (S = \{s, b, c, d\}, T = \{a, t\})$. Identify all forward and all backward edges across X_0. Compute the capacity of X_0.



(s, a) = 9

(b, a) = 3

9+3+9+7 = 28

(c, t) = 9

(d, t) = 7

(a, c) = 6

(c) Find a cut in the network whose capacity is equal to the value of the flow you computed in part (a). (This provides a guarantee that your flow is indeed maximum.) Use the idea outlined in the proof of correctness of the Ford-Fulkerson algorithm.

Looking at final residual graph

$\{s, a, b, c, d\}$ $\{t\}$

Q2 Graph Modifications

In this problem, we will consider what happens to the maximum flow when the flow network G is modified slightly.

(a) TRUE/FALSE: In any network G with integer edge capacities, there always exists an edge e such that increasing the capacity of e increases the maximum flow value in G.

(b) Suppose we are given a network G with n nodes, m edges, and integer edge capacities, and we are also given a flow f in G of maximum value. We now increase the capacity of a specific edge e by one. Give an $O(m + n)$ time algorithm to find a maximum flow in the updated network.

a) False

$s \xrightarrow{1} e \xrightarrow{1} t$

b) consider G_f — residual graph

if already contains forward edge for e → f not change

if does not contain → add edge with c=1
then check max flow

Q3 Teaching Assignment

Suppose there are m courses: $c_1, \ldots, c_m$. For each $j \in \{1, \ldots, m\}$, course c_j has s_j sections. There are n professors: $p_1, \ldots, p_n$. For each $i \in \{1, \ldots, n\}$, professor p_i has a teaching load of ℓ_i and likes to teach the subset of courses $A_i \subseteq \{c_1, \ldots, c_m\}$.

Your goal is to use the network flow paradigm to design an algorithm, which either finds an assignment of professors to courses satisfying the following constraints or reports that no such assignment exists.

- Each professor p_i must be assigned exactly ℓ_i courses.
- Each course c_j must be assigned to exactly s_j professors.
- No professor should be be assigned a course that they do not like to teach.
- No professor can teach multiple sections of the same course.

(a) Describe your full algorithm. That is, describe the network flow instance created (nodes, edges, and edge capacities) and how your algorithm uses a maximum flow in this instance to determine if a valid assignment of professors to courses exists, and output one if it does.

(b) Prove that your reduction is correct. That is, prove that there exists a valid assignment of professors to courses if and only if your algorithm finds one.

(c) What is the worst-case running time of your full algorithm if you use the naïve Ford-Fulkerson algorithm to solve the network designed in part (a)?

a) $s \xrightarrow{l_i} P_i \xrightarrow{1/0} C_j \xrightarrow{S_j} t$

Ford- Fulkerson → find max flow

requirements:

$(s, p_i) = l_i$ $\quad f(c_j, t) = l_j$

$\sum_{-m \to j : m} e_{mj} = s_j$

b) 1-1 correspondence between f and assignments

$\Downarrow$

$f(p_i, c_j) = 1 \iff$

c) $m+n+2$ - vertices

at most $m+n+mn$ edges

at most nm capacity

$O(m^2n^2) \in (m+n+mn)mn = m^2n + m^2n^2 + mn^2$

Tutorial 5

Q1 Standard Form

Consider the following linear program.

$$\begin{aligned} \min\ & 4x + 3y - 6z \\ \text{s.t.}\ & y - 3z \geq 2x + 2 \\ & 3x + 2y + 5z = 10 \\ & x, z \geq 0 \end{aligned}$$

(a) Convert this LP into the standard form.

(b) Write the dual of the LP from Part (a).

a) max $-4x-3y+6z$

s.t. $2x - y + 3z \leq -2$

$3x+2y+5z \leq 10$

$-3x-2y-5z \leq -10$

$x, z \geq 0$

replace $y = y' - y''$

$\Downarrow$

max $-4x-3(y'-y'')+6z$

$\vdots$

$x, y', y'', z \geq 0$

b) max:- $z = -4x - 3y' + 3y'' + 6z$

$2xy_1 - y'y_1 + y''y_1 + 3zy_1 \leq -2y_1$

$3xy_2 + 2y'y_2 - 2y''y_2 + 5zy_2 \leq 10y_2$

$-3xy_3 - 2y'y_3 + 2y''y_3 - 5zy_3 \leq -10y_3$

$(\underbrace{2y_1 + 3y_2 - 3y_3}_{\geq -4})x + y'(\underbrace{-y_1 + 2y_2 - 2y_3}_{\geq -3})$

$$+ y''(y_1 - 2y_2 \overset{\geq 3}{+} 2y_3) + z(3y_1 + 5y_2 - 5y_3)^{\geq 6}$$

$$\leq -2y_1 + 10y_2 - 10y_3$$

$$\Rightarrow \min \ -2y_1 + 10y_2 - 10y_3$$

Q2 Simple Scheduling with Prerequisites (SSP)

You are given n jobs with a list of durations $d_1, d_2, \ldots, d_n$. For every pair of jobs (i, j), you are also given a boolean $p_{i,j}$: if this is true, then job i must finish before job j can begin (i.e. job i is a prerequisite for job j).

Your goal is to find start times $s_1, s_2, \ldots, s_n$ for the jobs (no job can start earlier than time 0) such that the total time to complete all jobs is minimized while ensuring that the prerequisite constraints are met. Write a linear program to solve this problem.

Minimize T

subject to $T \geq s_i + d_i$

$s_j \geq s_i + d_i \quad \forall\, i \neq j$ and $P_{i,j}$ is true

$s_i \geq 0$

Q3 Integer Linear Programming

Suppose you are writing down a binary integer linear program (i.e., an optimization problem with a linear objective, linear constraints, and each variable taking a value in $\{0, 1\}$). Three of the binary variables in your program are x, y, and z; you have already placed the constraint: $x, y, z \in \{0, 1\}$.

Now, you want to encode the following relationships between x, y, and z. Show how to do so using linear constraints. Briefly justify your answers.

(a) Logical AND, $z = x \wedge y$: You want z to be 1 whenever both x and y are 1, and 0 otherwise.

(b) Logical OR, $z = x \vee y$: You want z to be 1 whenever at least one of x and y is 1, and 0 otherwise.

(b) Logical NOT, $z = \neg x$: You want z to be 1 whenever x is 0, and 0 otherwise.

a) $z \le x$

$z \le y$

$z \ge x+y-1$

b) $z \ge x$

$z \ge y$

$z \le x+y$

c) $z = 1-x$