5800
Max Flows 2

mar 27/28 2022

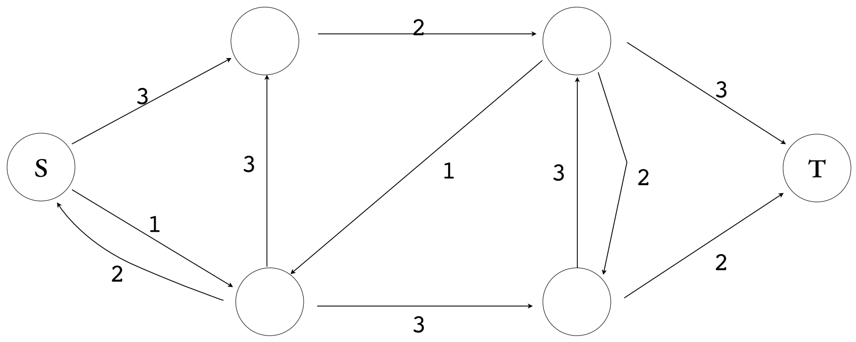
shelat

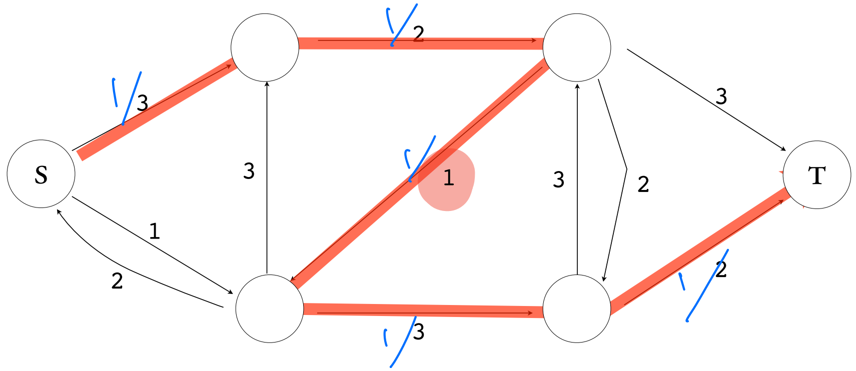
Ford-Fulkerson

INITIALIZE $f(u,v) \leftarrow 0 \; \forall u,v$

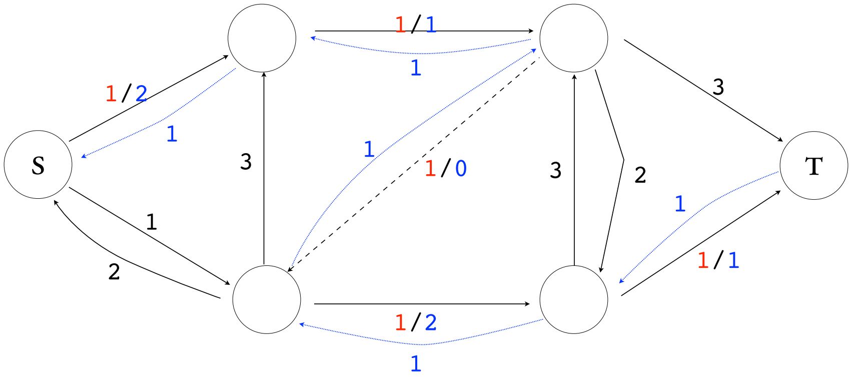
WHILE EXISTS AN AUGMENTING PATH p IN G_f

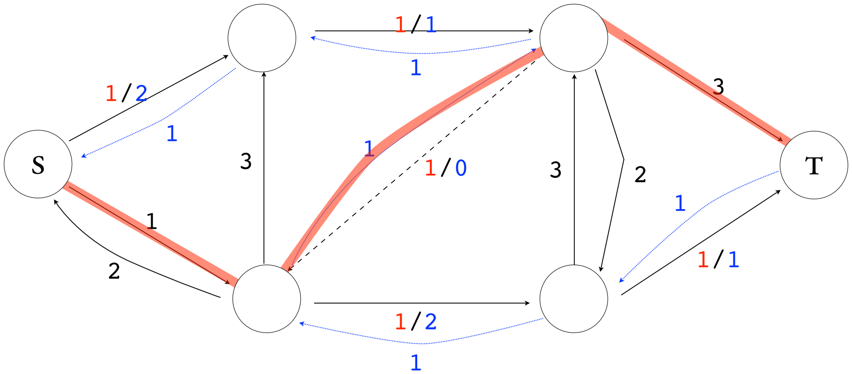
AUGMENT f WITH $c_f(p) = \min_{(u,v)\in p} c_f(u,v)$


S
T
3
1
2
3
2
3
1
3
2
3
3
2

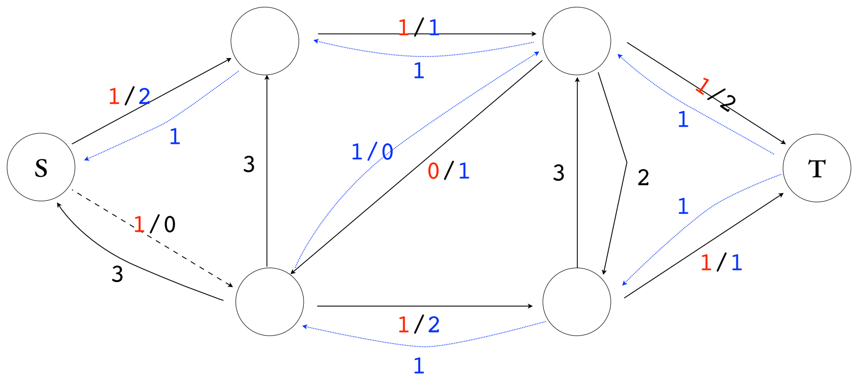


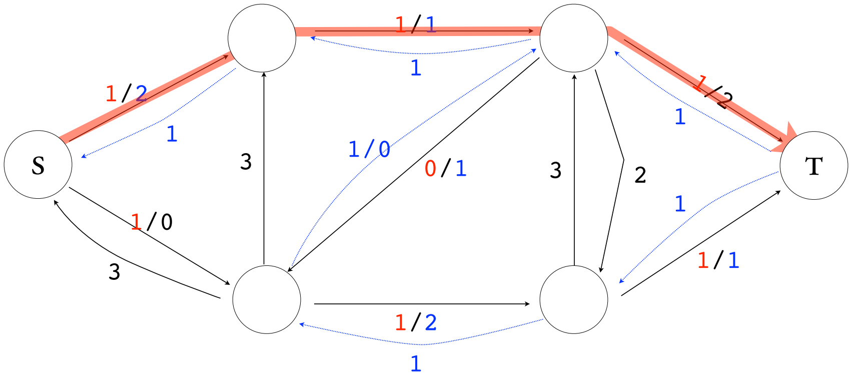
S
T
1/3
1/2
1/1
1/3
1/2
3
3
3
2
1
2


S
T
1 / 2
1
1
1 / 1
1
3
3
1
1 / 0
2
3
2
3
1
1 / 1
1 / 2
1

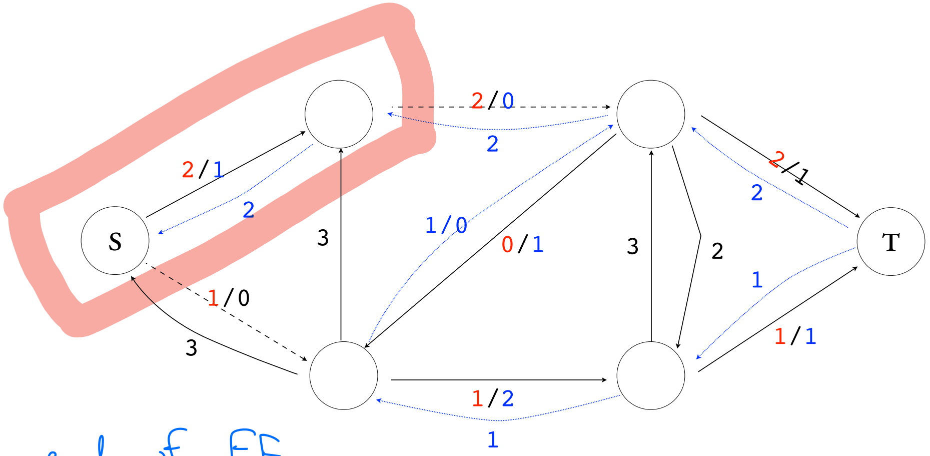


S
T
1/1
1
1/2
1
3
1
1/0
3
2
3
1
2
1
1/1
1/2
1


S
T
1/2
1
1/1
1
3
1/0
0/1
1/0
3
3
2
1/2
1
1
1/2
1
1/1



S
T
1/2
1
1/1
1
1/2
1
3
1/0
0/1
3
2
1/0
3
1
1/1
1/2
1



At the end of FF,
we can identify a cut whose value is equal to
the value of the flow.

FORD-FULKERSON

INITIALIZE $f(u,v) \leftarrow 0 \; \forall u,v$

WHILE EXISTS AN AUGMENTING PATH p IN G_f

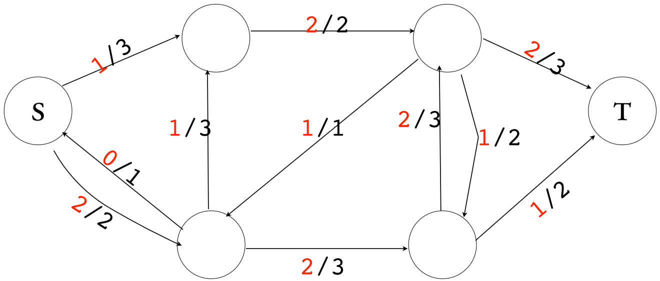
AUGMENT f WITH $c_f(p) = \min_{(u,v) \in p} c_f(u,v)$

TIME TO FIND AN AUGMENTING PATH: $\Theta(E+V)$ DFS, BFS.

NUMBER OF ITERATIONS OF WHILE LOOP: $|f|$

potentially a problem.

FOR ANY $f, (S, T)$ IT HOLDS THAT $|f| \leq ||S, T||$



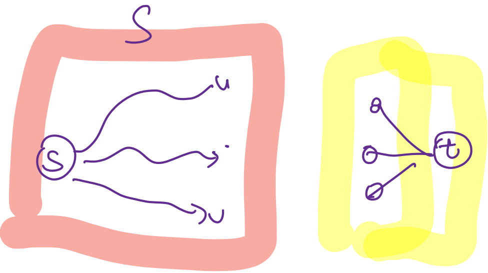
Thm: max flow = min cut

$$\max_f |f| = \min_{S,T} ||S,T||$$

IF F IS A MAX FLOW, THEN GF HAS NO AUGMENTING PATHS.

Define the set $S = \{ v \mid \exists$ a path from s to v in G_f and $c_f(p) > 0 \}$

these are the nodes one can still "reach" from s.



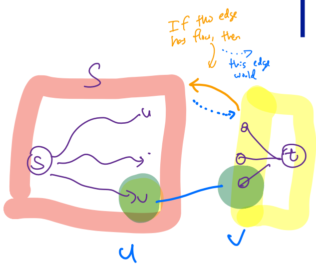
Define $T = V - S$.

Note that $s \in S$. Note $t \in T$. (why?)

$\Rightarrow (S, T)$ is a cut.

Because if $t \in S$, then there is still one more augmenting path !!

Thm: max flow = min cut (cont)



① Consider some $u \in S$ and $v \in T$.

$f(u,v) = c(u,v)$. why? If there was residual c_f capacity left, then v would have been in the set S.

$c_f(u,v) = 0 \Rightarrow f(u,v) = c(u,v)$

② $f(v,u) = 0$ for any (orange) $v \in T$, $u \in S$. If $f(v,u) > 0$, then there exists a residual edge u,v w/ $f(u,v) > 0$. and so v would be in S.

③

$$|f| = \sum_{u \in S} \left[\sum_{v \in T} f(u,v) - \sum_{v \in T} f(v,u) \right]$$

$$= \sum_{u \in S} \sum_{v \in T} c(u,v) - \cancel{\sum_{u \in S} \sum_{v \in T} f(v,u)}$$

this term is 0 by ②

$$= ||S,T||$$

$\Rightarrow$ that f has to be the MAX FLOW.

FORD-FULKERSON

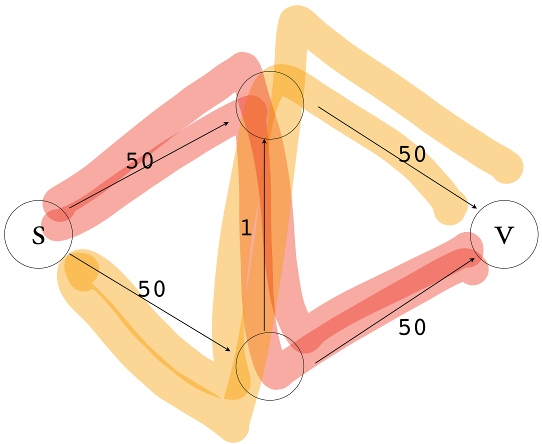
INITIALIZE $f(u,v) \leftarrow 0 \; \forall u,v$

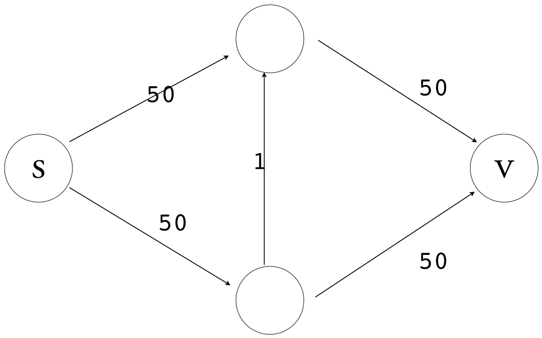
WHILE EXISTS AN AUGMENTING PATH p IN G_f

AUGMENT f WITH $c_f(p) = \min_{(u,v)\in p} c_f(u,v)$

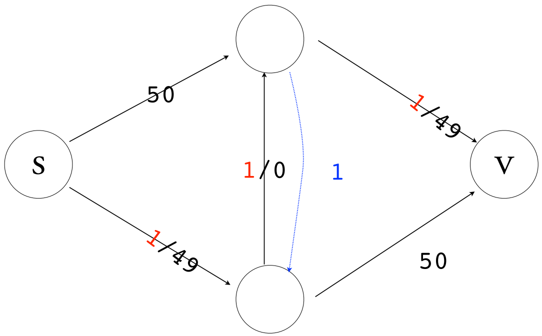
TIME TO FIND AN AUGMENTING PATH:

NUMBER OF ITERATIONS OF WHILE LOOP:

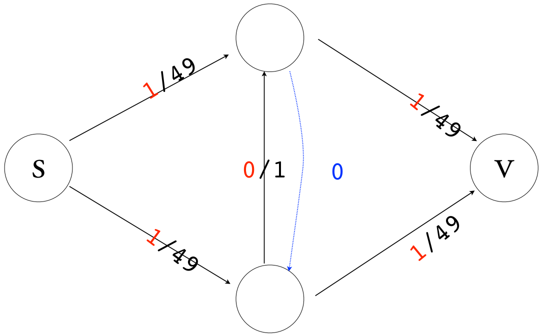

S
V
50
50
1
50
50



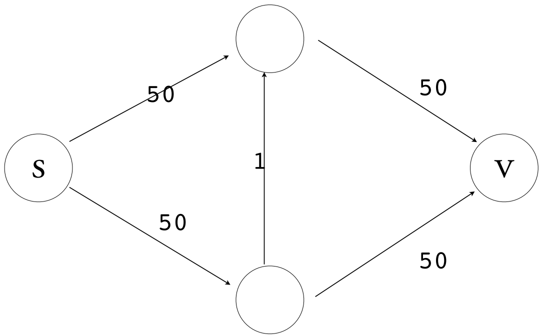
50
50
S
1
V
50
50



S
V
50
1/49
1/0
1
1/49
50


1/49
1/49
S
0/1
0
V
1/49
1/49

root of the problem



Edmonds-Karp 2

for a graph G,
and flow f,

choose path with fewest edges first. (BFS)

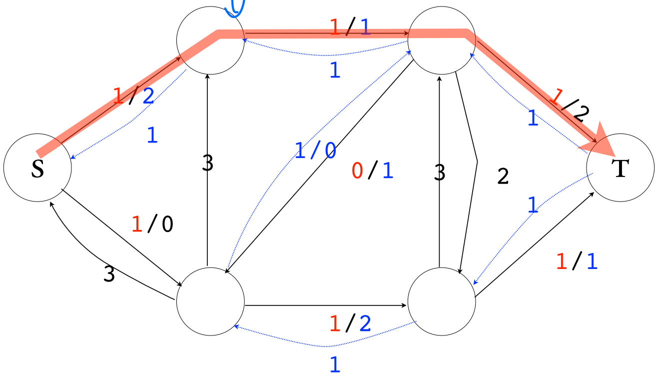
$\delta_f(s, v)$: fewest # of edges on a path from s to v
in the residual graph G_f

$\delta_f(s, v)$ increases monotonically thru exec

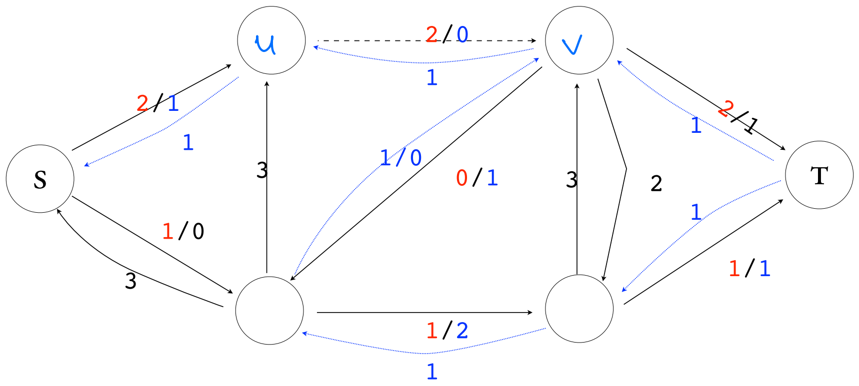
$$\delta_{i+1}(v) \geq \delta_i(v)$$

index i corresponds to the number of augmenting paths that have been found so far.

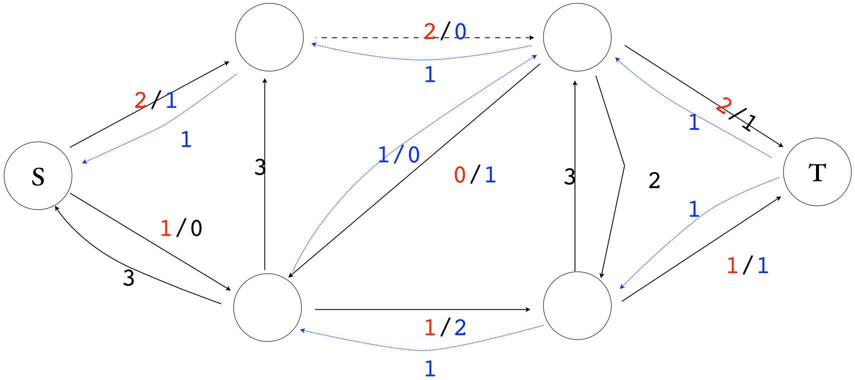
the critical bottleneck on the path from S to v.



for every augmenting path, some edge is critical.



critical edges are removed in next residual graph.

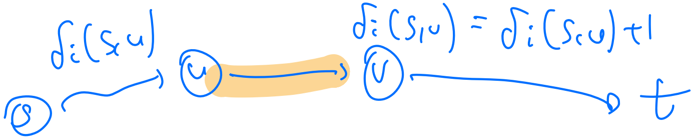


key idea: how many times can an edge be critical?

[when we are using the EK2 idea of always pushing along the shortest path from s to t] timeline of augmenting paths using EK2.

→ the ith augmenting flow that EK found.

Lets say that edge (u,v) is the critical edge at time i for the 1st time



Therefore, at time $i+1$, the edge (u,v) has 0 capacity and there is some edge from (v,u).

$$\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$$

first time (u,v) is critical:

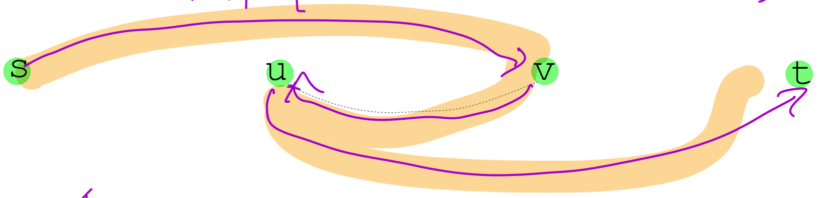
the iteration at which (u,v) is an edge in G_f.

time i+1: (u,v) is critical: $$\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$$

time j: Edge (u,v) STRIKES BACK

what must occur for the edge (u,v) to be added back??

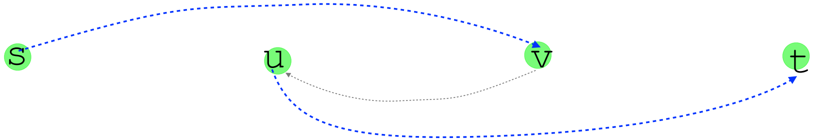
We must have found a shortest path $s \rightsquigarrow v \to u \rightsquigarrow t$



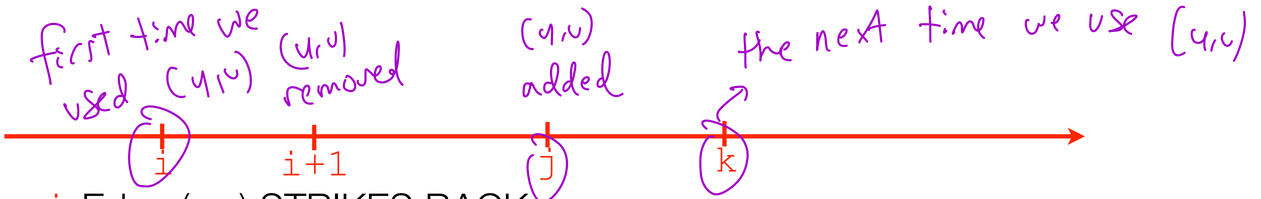
$$\delta_j(s,u) = \delta_j(s,v) + 1$$

time i+1: (u,v) is critical: $\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$

time j: Edge (u,v) STRIKES BACK



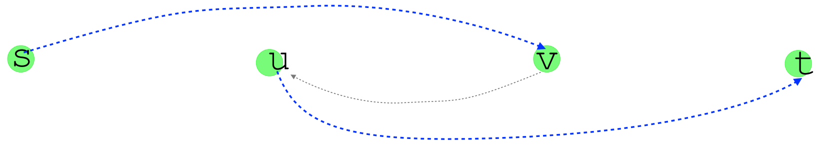
$$\delta_j(s,u) = \delta_j(s,v) + 1$$



time j: Edge (u,v) STRIKES BACK

$$\delta_{i+1}(s,v) \geq \delta_i(s,u) + 1$$

$$\delta_j(s,u) = \delta_j(s,v) + 1$$

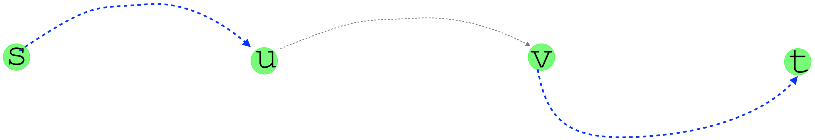


$$\delta_j(s,u) \geq \left(\delta_{i+1}(s,u) + 1\right) + 1 = \delta_{i+1}(s,u) + 2$$

2nd time (u,v) is used

time k: RETURN OF THE (u,v) critical

$$\delta_k(s,u) \geq \delta_i(s,u) + 2$$



QUESTION: How many times can (u,v) be critical?

$$\frac{|V|}{2}$$

each edge critical only $\frac{|V|}{2}$ times.

there are only E edges.

Every augmenting path has at least 1 critical edge

ergo, total # of augmenting paths: $\Theta(E \cdot V)$

time to find an augmenting path: $\Theta(E)$

BFS

total running time of E-K algorithm:

worst case: $\Theta(E^2 V)$

FF	$O(E\|f^*\|)$	Any augmenting path.
EK2	$\Theta(E^2V)$	BFS.
PUSH-RELABEL		
(Tarjan) et al FASTER PUSH-RELABEL	$\Theta(V^3)$	

GoldBerg-Rao $O\left(\min\{E^{2/3}, V^{1/2}\} \cdot E \cdot \log\left(\frac{V^2}{E}\right) \cdot \log(U)\right)$

max value of a capacity

$V^{1.2/3} \cdot \log(\ldots)$ $V^{2.5} \cdot \log(\ldots)$

Bipartite Matchings

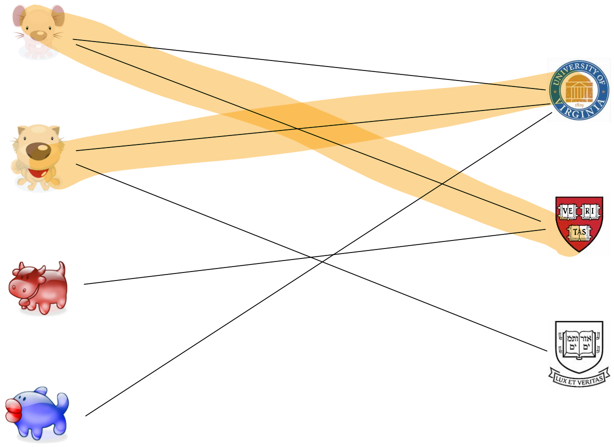
maximum bipartite matching

$V = (L, R)$

All edges in the graph go from L to R.

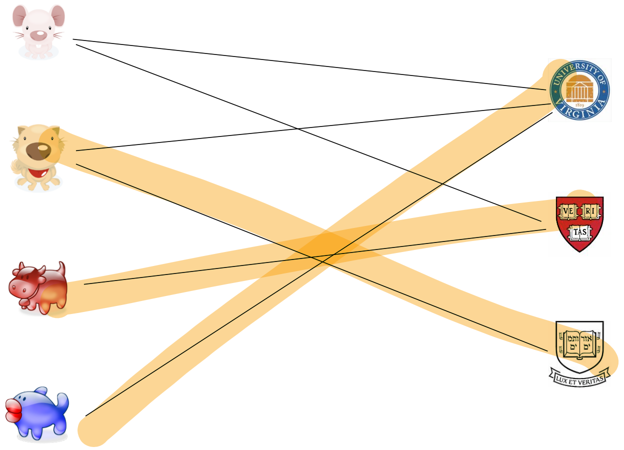
Left nodes

Right nodes



only 2 students were matched.

maximum bipartite matching



3 students can be matched.

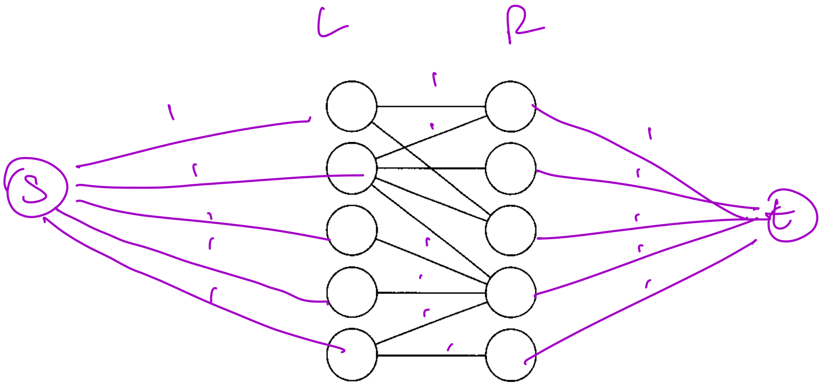
bipartite matching

PROBLEM: Given as input a bipartite graph $G = ((L, R), E)$,

Find the largest subset of edges $M \subseteq E$ such that each node occurs at most once in the set M.

algorithm

G'



Add a s, t.

Add edges from s to {L} and {R} to t.

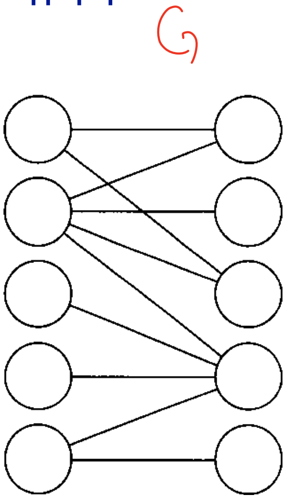
Set all capacities to 1.

algorithm

1. MAKE NEW G'
FROM INPUT G.

2. RUN ~~FF~~ EK ON G'

3. OUTPUT ALL MIDDLE EDGES
WITH FLOW F(E)=1.



FF always produces integral flows
if capacities are integral.

correctness

input

IF G HAS A MATCHING OF SIZE K, THEN G' has a MAXFLOW of K

Proof: Given a matching $M \subseteq E$ of size $|M| = K$, such that each node occurs at most once in M, construct a flow f as follows: for all $e \in M$, $f(e) = 1$

for all $e = (u,v) \in M$, $f(s,u) = 1$

$f(v,t) = 1$

0 otherwise.

① f is a valid flow.

capacity constraint: $f(u,v) \leq c(u,v)$

flow constraint: ✓

② $|f| = K$. because $\sum_{u \in L} f(s,u) = K$.

correctness

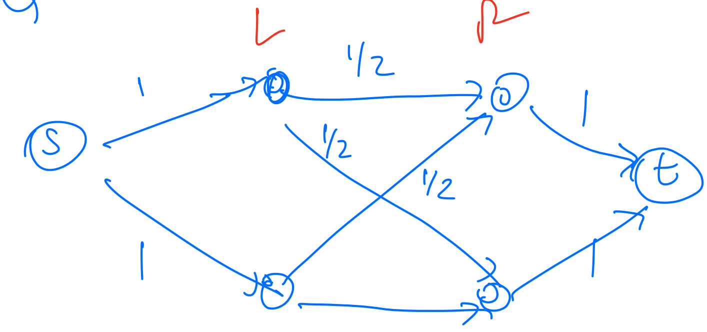
IF G' HAS A FLOW OF K, THEN G HAS A BIPARTITE MATCHING of K.

Proof: Consider all edges in G' between L and R with $f(e) = 1$.

Add e to M. $\Rightarrow |M| = K$. each $e = (u, v)$ is s.t. u can only have one unit of inflow from s, so it can occur at most once in M.

Same for each v.

G':



This is a valid MAX flow!!

But our reduction procedure produces $M = \{\}$. (empty set)

integrality theorem

IF CAPACITIES ARE ALL INTEGRAL, THEN FF returns an integral flow.

Proof: By induction. In FF, f begins as 0. Thus integral.

Spse f is integral after i steps of FF.

Consider the $(i+1)^{st}$ step. Since f is integral, all residual capacities on G_f are integral. FF finds an augmenting path. The min capacity will be integral, and thus the update to f will remain integral on step $i+1$.

correctness

IF G' HAS A FLOW OF K, THEN G HAS K-MATCHING.

G' has integral flows. Therefore, each flow value can be only 0 or 1 because capacities are 1.

[Rest of the argument from before]

running time

$O(E \cdot |f|) < O(E \cdot V)$

edge-disjoint paths

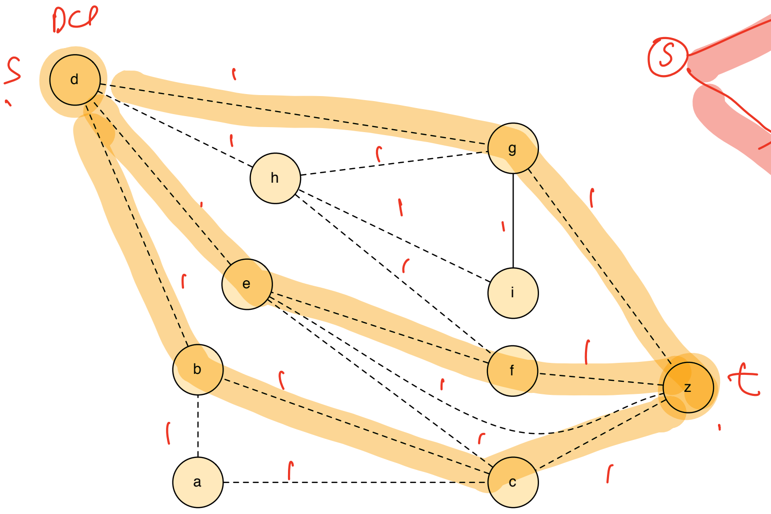
INPUT: graph G

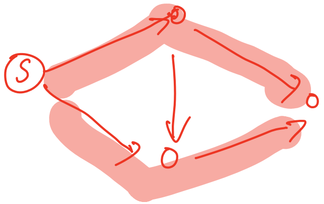
source and destination

s, t.

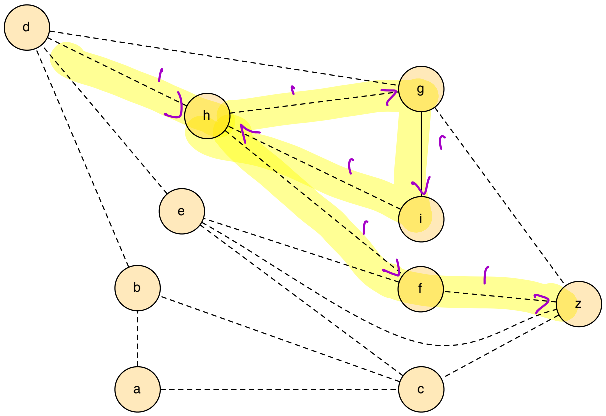
output: # of edge disjoint paths.

(the actual paths)





algorithm



add capacity 1 to each edge

1. Compute max flow
2. Remove all edges with $f(e) = 0$.
3. Walk from s.
    1. If you reach a node you have visited before, erase flow along path
    2. If you reach t, add this path to your set, erase flow along path.

analysis

capacity for each edge

IF G HAS K DISJOINT PATHS, THEN there (G, c) has a flow $|f| = K$.

why is this true??

→ simply assign a flow of 1 on each edge on the K disjoint paths.

(a) capacity constraint will be satisfied because $f(e) \leq c(e)$ for all $e \in E$.

(b) flow constraint: (for you to ponder)

(I will post on the 447 website)

analysis

(G, c)

IF G' HAS A FLOW OF K, THEN there exist k edge disjoint paths.

To show: $\exists$ k edge disjoint paths among the edges in G' for which $f(e) = 1$.

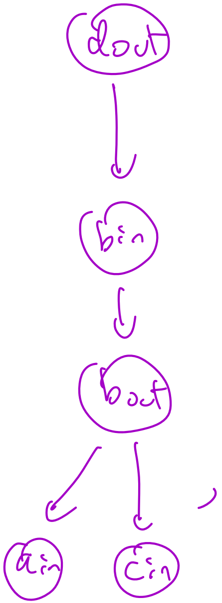
This argument is by induction: Start w/ i paths and a G with flow k.

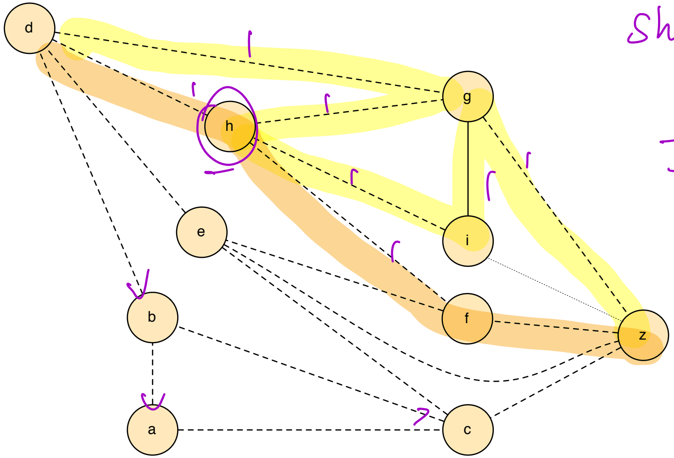
→ After running the procedure in step 3, the resulting graph has (i+1) paths and G with flow k-1.

(proof as an exercise, posted on website)

vertex-disjoint paths

example:



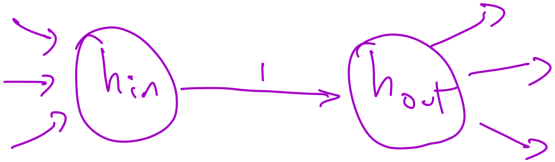


⟹ edge disjoint paths may still share a common node.

Input: graph G, and s, t ∈ V.

Output: # of vertex disjoint paths.

for each node, replace:



BASEBALL ELIMINATION

	W	L	Left	A	P	N	M
				Against			
ATL	83	71	8	-	1	6	1
PHL	80	79	3	1	-	0	2
NY	78	78	6	6	0	-	0
MONT	77	82	3	1	2	0	-

MONT cannot win the NLE.

BASEBALL ELIMINATION

Some team must have 77 wins.

	W	L	Left	N	B	Bo	T	D
NY	75	59	28		3	8	7	3
BAL	71	63	28	3		2	7	4
BOS	69	66	27	8	2			
TOR	63	72	27	7	7			
DET	49	86	27	3	4			

(Against: N, B, Bo, T, D)

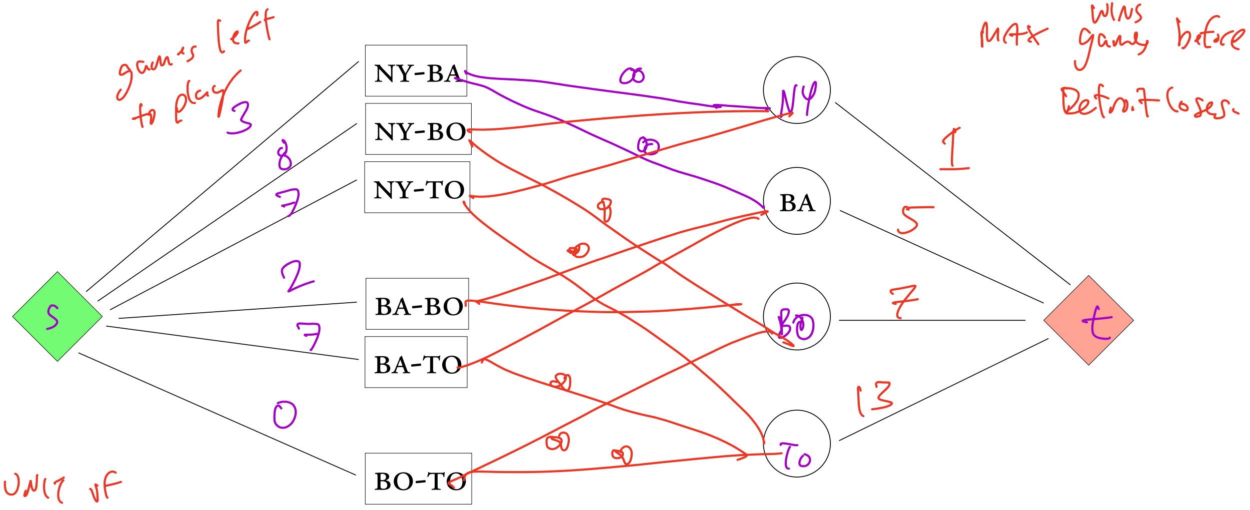
75
71
69
63
278
+ 27
305

76.25
4) 305
28
25
24
1

CAN Detroit win??

Run Flow:

Better non-trivial

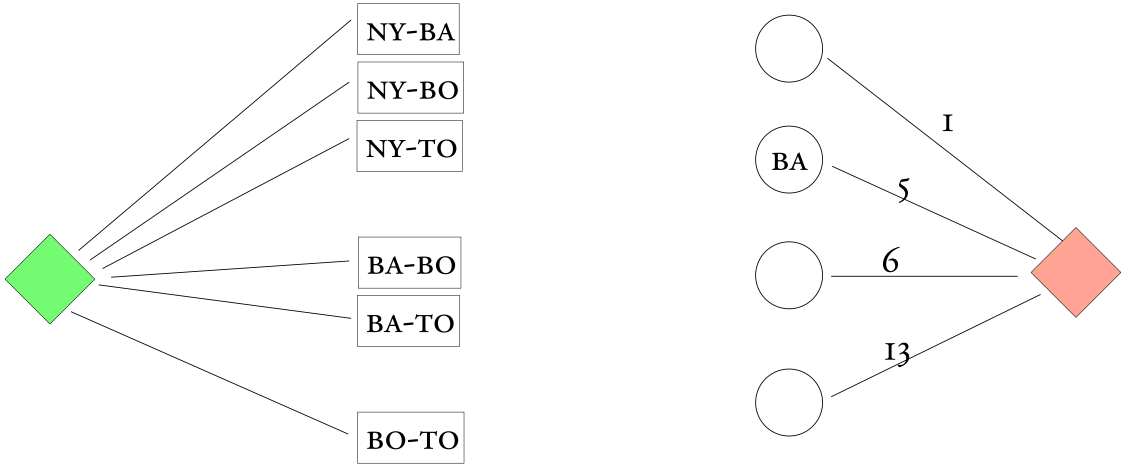


unit of flow = 1 win

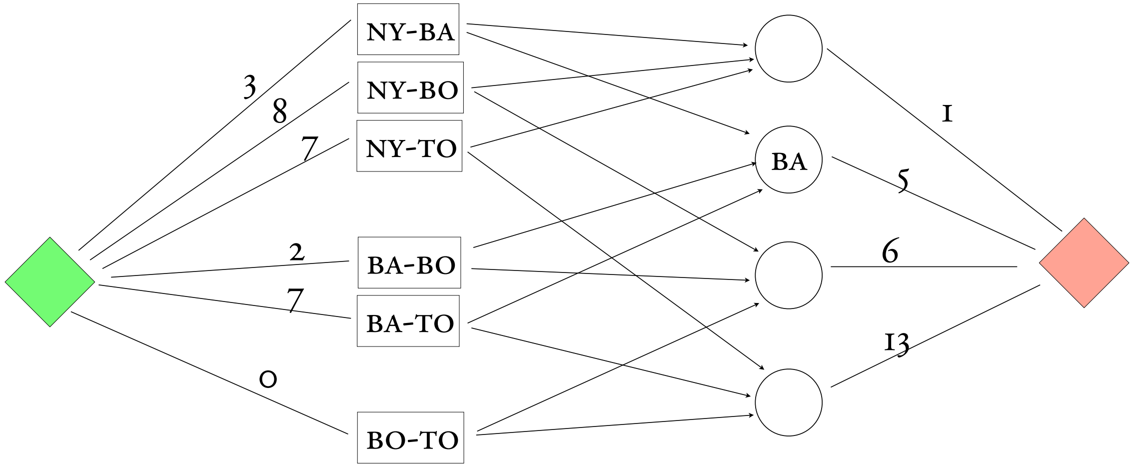
NY-SFO

	W	L	Left	N	B	Bo	T	D
NY	75	59	28		3	8	7	3
BAL	71	63	28	3		2	7	4
BOS	69	66	27	8	2			
TOR	63	72	27	7	7			
DET	49	86	27	3	4			

76



	W	L	Left	N	B	Bo	T	D
NY	75	59	28		3	8	7	3
BAL	71	63	28	3		2	7	4
BOS	69	66	27	8	2			
TOR	63	72	27	7	7			
DET	49	86	27	3	4			



	W	L	Left	N	B	Bo	T	D
NY	75	59	28		3	8	7	3
BAL	71	63	28	3		2	7	4
BOS	69	66	27	8	2			
TOR	63	72	27	7	7			
DET	49	86	27	3	4			

BASEBALL ELIMINATION

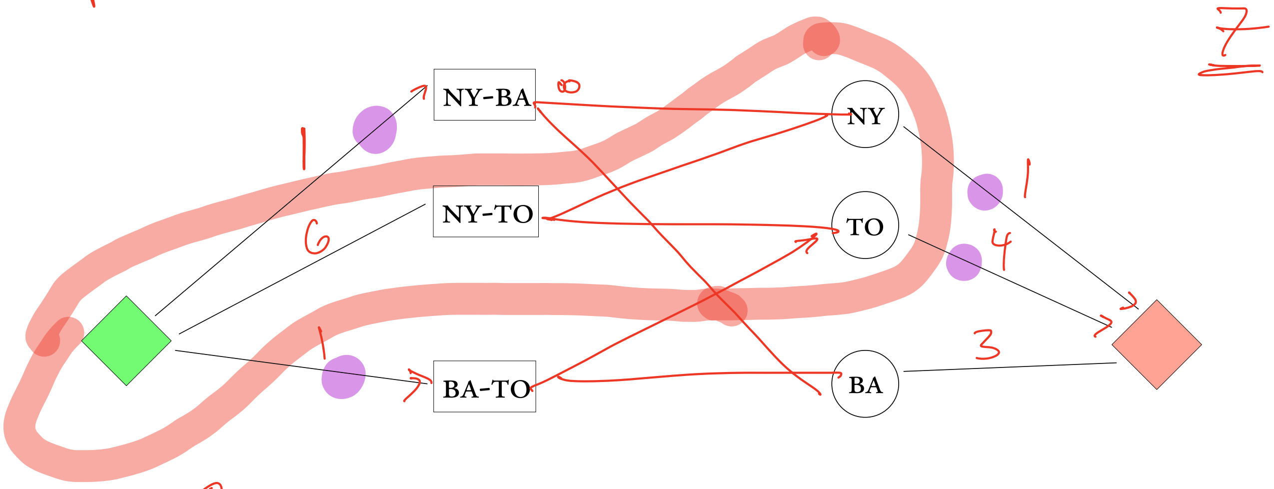
	W	Against N	B	Bo	T
NY	90		1	4	6
BAL	88	1		4	1
BOS	79	4	4		4
TOR	87	6	1	4	

$79 + 12 = \underline{91}$

$$\begin{array}{r} 90 \\ 88 \\ 87 \\ \hline 265 \\ 8 \\ \hline \end{array}$$

$$3 \overline{)273}^{91}$$

FLOW ≤ ANY CUT



this cut has value 7 which is less than the 8 games left.

	W	L	Left	N	B	Bo	T
NY	90				1	4	6
BAL	88			1		4	1
BOS	79			4	4		4
TOR	87			6	1	4	

Why it works

Thm: A team T has been eliminated if the maxflow of graph G is less than the total number of games left between the other teams in the league.