

CSE 3318 Notes 16: Network Flows and Bipartite Matching

(Last updated 7/19/24 1:41 PM)

CLRS 26.1-26.3

<http://dl.acm.org.ezproxy.uta.edu/citation.cfm?doid=2632661.2628036>

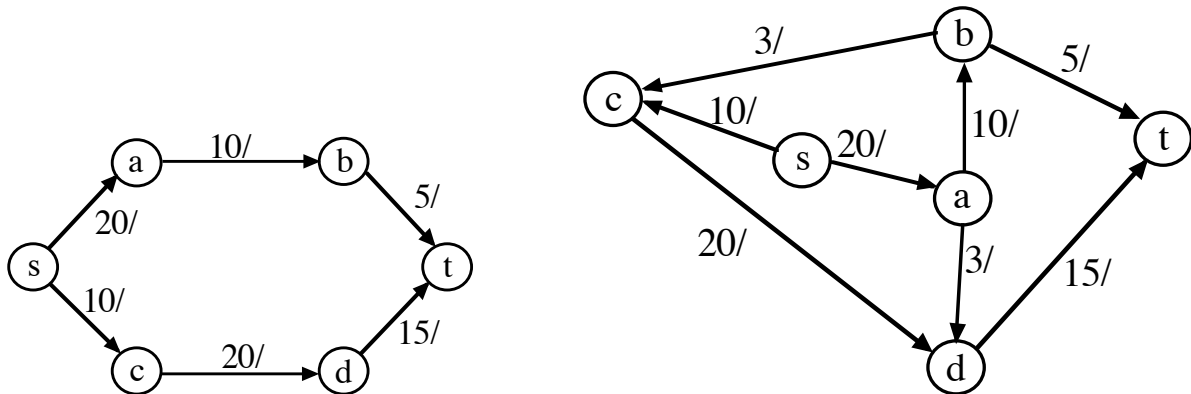
16.A. MAXIMUM FLOW CONCEPTS

Given a positive weighted (capacitated) directed graph with a designated *source* (s) and *sink* (t), determine the maximum possible flow through the “network”, i.e.

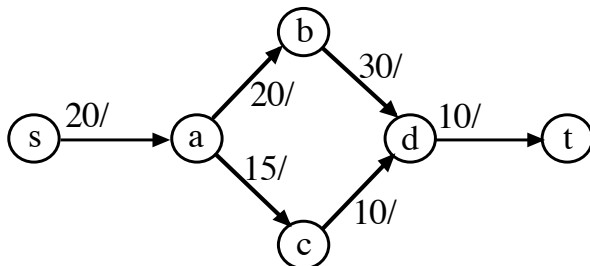
Flow Conservation: inflow = outflow, except at source (inflow = 0) and sink (outflow = 0)

Goal: maximize outflow at source (equivalently, maximize inflow at sink)

Examples:



Note: Solution is not necessarily unique

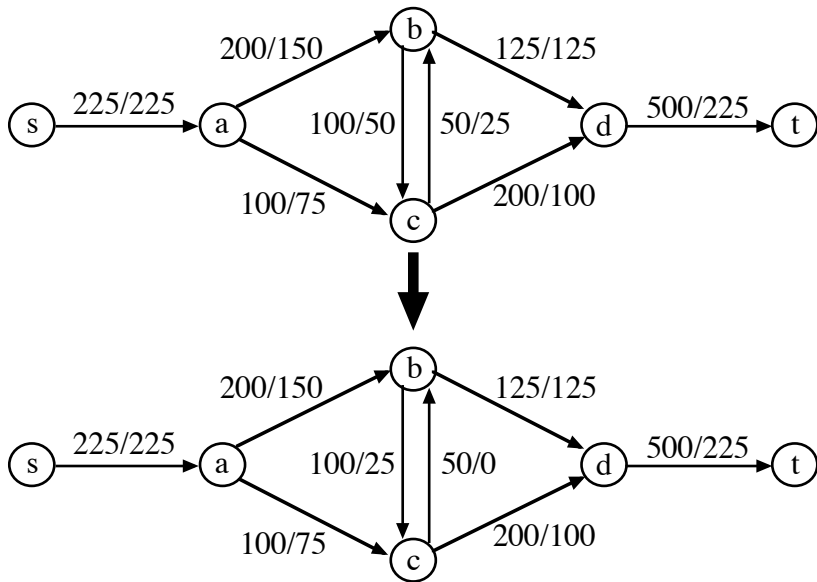


Applications:

- Communication system modeling
- Pipelines
- Mechanical systems
- Highways
- Matching (see section 16.H)

Observations:

A solution can be found that does not have flow on both edges (x, y) and (y, x) .
(Observation does not say that the smaller capacity edge should be removed.)

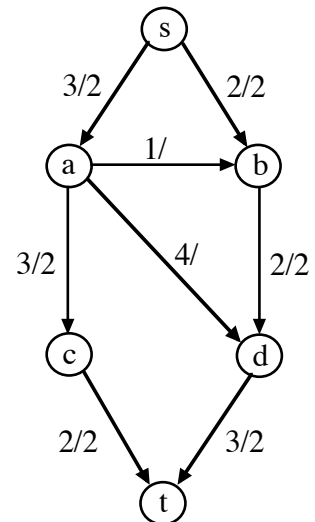
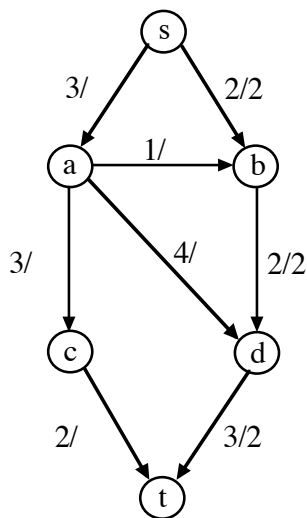
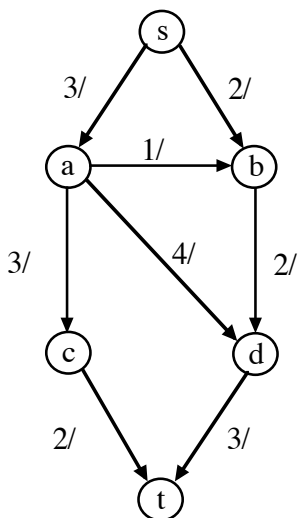


Solution is *not optimal* if an “augmenting path” (A.P.) from the source to sink can be found.

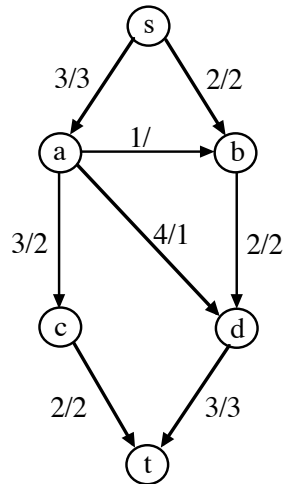
Example:

A.P. s, b, d, t / 2 units

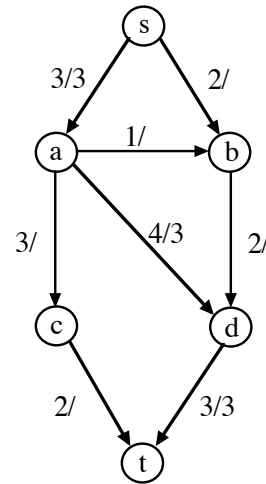
A.P. s, a, c, t / 2 units



A.P. $s, a, d, t / 1$ unit



Suppose that first A.P. was $s, a, d, t / 3$ units



To fix this problem, an A.P. may “reverse” and “redirect” a flow:

A.P. $s, b, d, a, c, t / 2$ units

Remaining Question: If there are no A.P.s (as generalized for previous example), is the flow maximized?

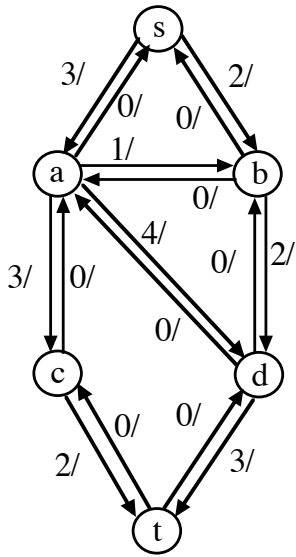
YES \Rightarrow Leads to Ford-Fulkerson Method and Max-flow, Min-cut Theorem.

16.B. FORD-FULKERSON METHOD

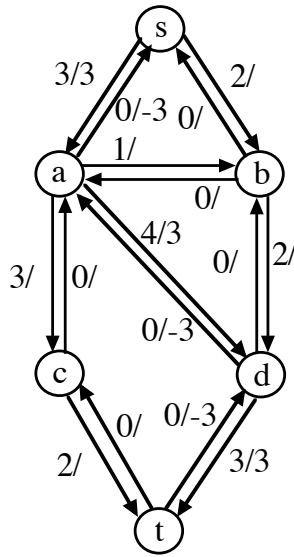
Search for A.P.s is facilitated by maintaining a *residual network*:

1. Initially has the same edges/capacities as input network.
2. The inverse of every edge is included (often with zero capacity).
3. If the flow from vertex u to vertex v is positive, $f(u,v) > 0$, then the flow from v to u will be negative, i.e. $f(u,v) = -f(v,u)$.
4. An A.P. is found in the residual network by finding a path of edges where every edge has remaining capacity (“unsaturated”), $f(u,v) < c(u,v)$. The *incremental flow* is the minimum difference, $c(u,v) - f(u,v)$, along the path.
5. An A.P. is recorded *twice* in the residual network by:
 - a. Adding the incremental flow to $f(u,v)$ for every edge in the A.P.
 - b. Subtracting the incremental flow from $f(v,u)$ for every edge in the reverse path.

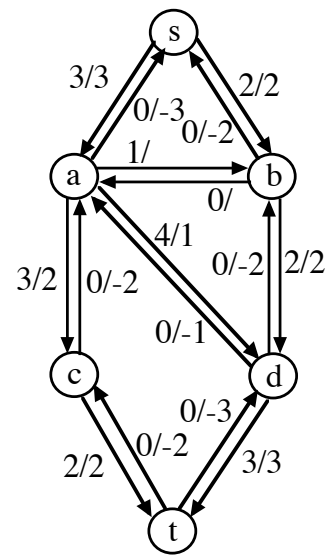
The initial residual network for the last example is:



A.P. s, a, d, t / 3 units



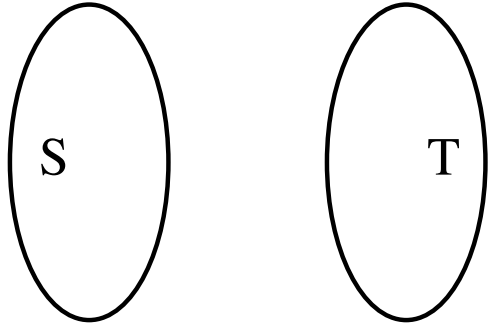
A.P. s, b, d, a, c, t / 2 units



16.C. MAX-FLOW MIN-CUT THEOREM

Cut: A partition of V into two sets S (“source side”) and T (“sink side”) such that

1. $S \cap T = \emptyset$
2. $S \cup T = V$
3. $s \in S$
4. $t \in T$



Capacity of a cut:

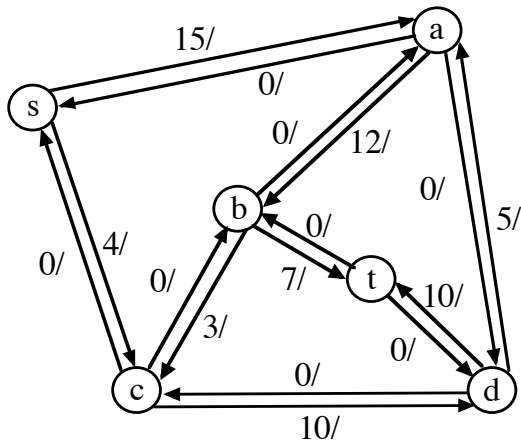
$$\sum_{x \in S} \sum_{y \in T} \text{capacity}(x,y)$$

Lemma: The capacity of any cut is an upper bound on the maximum flow. (will prove later)

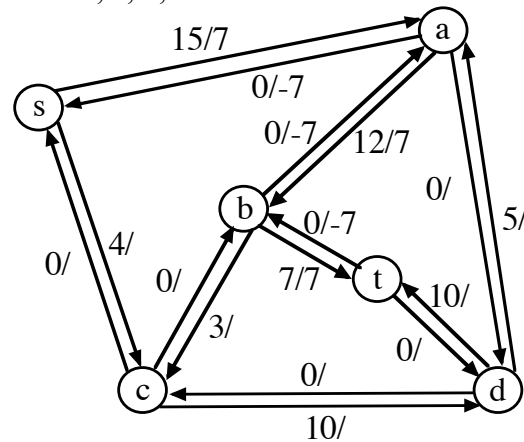
In general, a min-cut is determined from the residual network at termination of the max-flow algorithm:

1. S includes all vertices reachable from the source using unsaturated edges.
2. T is everything else.

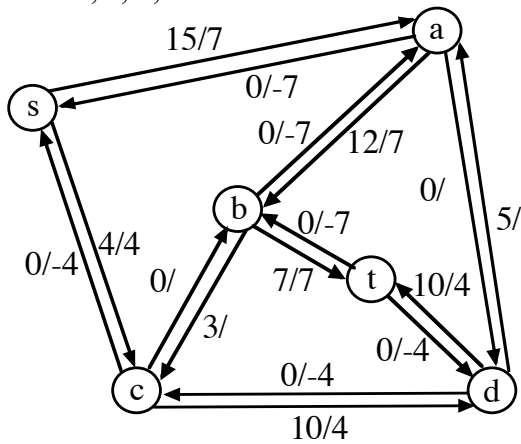
Initial residual network:



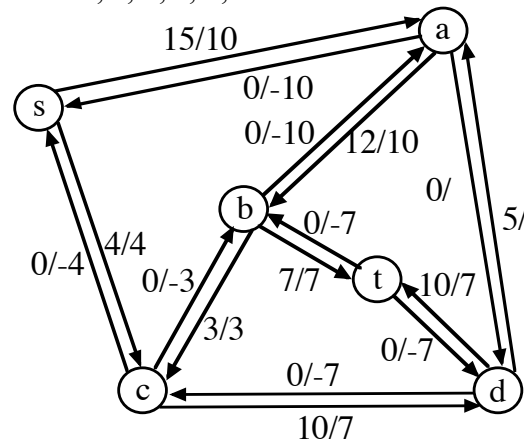
A.P. s, a, b, t / 7 units



A.P. s, c, d, t / 4 units



A.P. s, a, b, c, d, t / 3 units



The following sequence of A.P.s also leads to the same final residual network:

s, a, b, c, d, t / 3 units

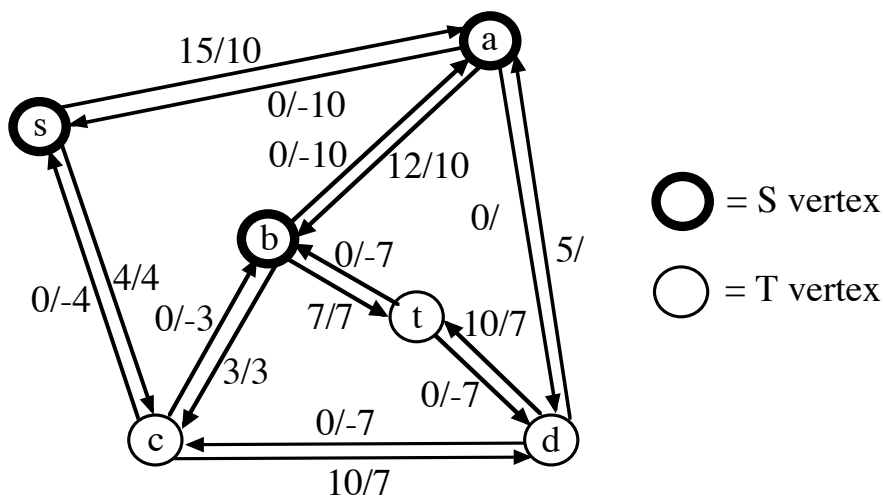
s, c, d, a, b, t / 4 units

s, a, d, c, b, t / 3 units

s, a, b, c, d, t / 3 units

s, a, d, t / 1 unit

Searching the unsaturated edges of the residual network yields the cut $S = \{s, a, b\}$ $T = \{t, c, d\}$ with capacity 14.



16.D. IMPLEMENTING FORD-FULKERSON

Using Edmonds-Karp BFS

4: 5 \leftarrow 4 \leftarrow 3 \leftarrow 0

7: 5 \leftarrow 2 \leftarrow 1 \leftarrow 0

3: 5 \leftarrow 4 \leftarrow 3 \leftarrow 2 \leftarrow 1 \leftarrow 0

Edge capacities/flows

0 \rightarrow 3 4/4

0 \rightarrow 1 15/10

1 \rightarrow 2 12/10

2 \rightarrow 5 7/7

2 \rightarrow 3 3/3

3 \rightarrow 4 10/7

4 \rightarrow 5 10/7

4 \rightarrow 1 5/0

Max flow is: 14

S side: 0 1 2

T side: 3 4 5

Using maximum capacity paths

7: 5 \leftarrow 2 \leftarrow 1 \leftarrow 0

4: 5 \leftarrow 4 \leftarrow 3 \leftarrow 0

3: 5 \leftarrow 4 \leftarrow 3 \leftarrow 2 \leftarrow 1 \leftarrow 0

Edge capacities/flows

0 \rightarrow 3 4/4

0 \rightarrow 1 15/10

1 \rightarrow 2 12/10

2 \rightarrow 5 7/7

2 \rightarrow 3 3/3

3 \rightarrow 4 10/7

4 \rightarrow 5 10/7

4 \rightarrow 1 5/0

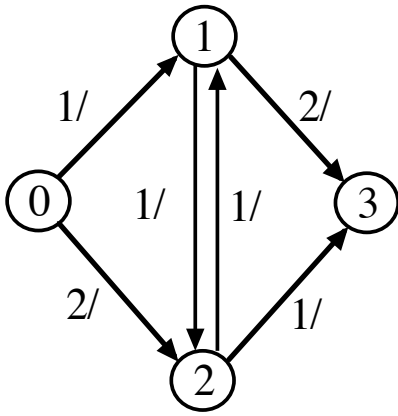
Max flow is: 14

S side: 0 1 2

T side: 3 4 5

An inefficient implementation (<http://ranger.uta.edu/~weems/NOTES3318/ff.c>) is easily implemented using adjacency matrices for the residual network, but wastes time scanning over nonexistent edges during breadth-first search.

An efficient implementation (<http://ranger.uta.edu/~weems/NOTES3318/ffLab.c>) involves additional details for building compressed adjacency lists for the residual network. The following example illustrates the phases:



```

% a.out
4 6
0 1 1
0 2 2
1 2 1
2 1 1
1 3 2
2 3 1
Input & inverses:
  i tail head  cap
  0  0   1   1
  1  1   0   0
  2  0   2   2
  3  2   0   0
  4  1   2   1
  5  2   1   0
  6  2   1   1
  7  1   2   0
  8  1   3   2
  9  3   1   0
 10  2   3   1
 11  3   2   0
qsort CPU 0.000027
Sorted edges:
  i tail head  cap
  0  0   1   1
  1  0   2   2
  2  1   0   0
  3  1   2   0
  4  1   2   1
  5  1   3   2
  6  2   0   0
  7  2   1   1
  8  2   1   0
  9  2   3   1
 10  3   1   0
 11  3   2   0

```

Coalesced edges:

i	tail	head	cap
0	0	1	1
1	0	2	2
2	1	0	0
3	1	2	1
4	1	3	2
5	2	0	0
6	2	1	1
7	2	3	1
8	3	1	0
9	3	2	0

set inverses CPU 0.000035

Initialized residual network:

Vertex firstEdge

0	0
1	2
2	5
3	8

=====

4	10
---	----

i	tail	head	cap	inv
0	0	1	1	2
1	0	2	2	5
2	1	0	0	0
3	1	2	1	6
4	1	3	2	8
5	2	0	0	1
6	2	1	1	3
7	2	3	1	9
8	3	1	0	4
9	3	2	0	7

3<-1<-0 adds 1 incremental flow

3<-2<-0 adds 1 incremental flow

3<-1<-2<-0 adds 1 incremental flow

3 augmenting paths

S side of min-cut:

0

T side of min-cut:

1

2

3

total flow is 3

Ford-Fulkerson time 0.000293

flows along edges:

0->1 has 1

0->2 has 2

1->3 has 2

2->1 has 1

2->3 has 1

16.E. FORD-FULKERSON CORRECTNESS

Definitions: Given a cut (S, T):

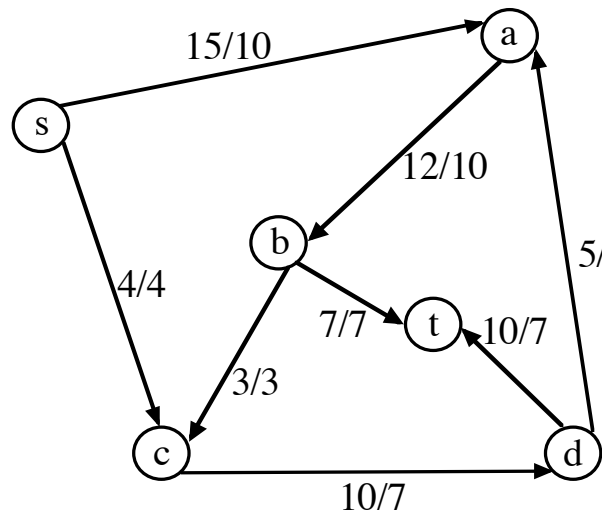
1. *Capacity* = sum of capacities over the edges going from the S-side to the T-side:

$$\sum_{x \in S} \sum_{y \in T} \text{capacity}(x, y)$$

2. Given a flow assignment (obeys flow conservation) for a network, the *net flow* across a cut is the sum:

$$\sum_{x \in S} \sum_{y \in T} f(x, y) - \sum_{x \in T} \sum_{y \in S} f(x, y)$$

(Consider only the positive flow values, not the negative flow values.)



Cut 1:

$$S = \{s\}$$

$$T = \{t, a, b, c, d\}$$

Capacity = 19

Flow Across Cut =

Cut 2:

$$S = \{s, a, c\}$$

$$T = \{t, b, d\}$$

Capacity = 22

Flow Across Cut =

Cut 3:

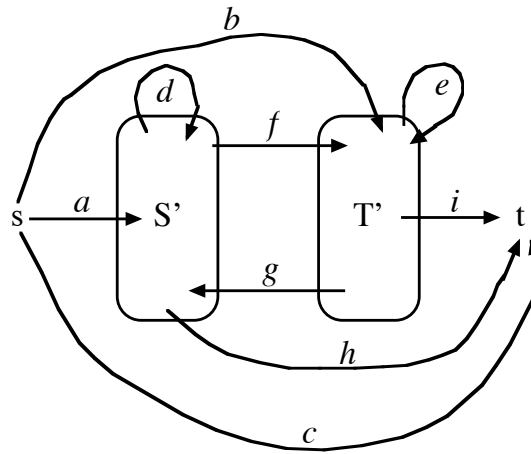
$$S = \{s, a, b\}$$

$$T = \{t, c, d\}$$

Capacity = 14

Flow Across Cut =

Claim 1: Net flow across any cut = Amount of flow (from source to sink)



Amount of flow = $f(a) + f(b) + f(c) = f(h) + f(i) + f(c)$

Observe S' : $f(a) + f(g) = f(f) + f(h)$, so $f(a) = f(h) + f(f) - f(g)$

Net flow across cut = $f(b) + f(c) + f(h) + f(f) - f(g)$, but previous step allows substitution of $f(a)$
 $= f(b) + f(c) + f(a) = \text{Amount of flow}$

Claim 2: Flow from s to t is bounded by any cut's capacity

Capacity of cut $\geq f(b) + f(c) + f(f) + f(h)$ and $f(f) + f(h) \geq f(a)$ (since $f(f) + f(h) = f(a) + f(g)$)

Thus, capacity of cut $\geq f(a) + f(b) + f(c) = \text{Amount of flow}$

(Note: In a min-cut $f(g) = 0$.)

Max-flow Min-cut Theorem: If f is a flow assignment, then the following are equivalent:

1. f is a max-flow.
2. No A.P.s in residual network.
3. The amount of flow for f is the same as the capacity of some cut

Proof: $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 1$

$1 \Rightarrow 2$: By contradiction. If there is an A.P., then the flow may be increased.

$2 \Rightarrow 3$: If there are no A.P.s, then search on unsaturated edges of residual network gives a min-cut.

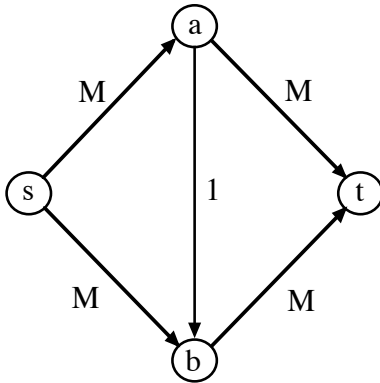
$3 \Rightarrow 1$: By contradiction. If f is not a max-flow, then f cannot be as large as the capacity of any cut.

16.F. FORD-FULKERSON COMPLEXITY ISSUES

Will assume that capacities are integers. Otherwise, termination (due to arithmetic accuracy or irrational numbers) becomes an issue.

The original Ford-Fulkerson technique makes no assumption about choosing an A.P.

Classic bad case (unlikely in practice).



Can lead to $2M$ A.P.s that each contribute one unit of flow.

Since the number of bits (in the input file) to represent a number of magnitude M is $\Theta(\log M)$, Ford-Fulkerson has the theoretical potential to take exponential time.

16.G. EDMONDS-KARP VARIANT OF FORD-FULKERSON

Concept: Choose A.P. using BFS on residual network to obtain a path with smallest number of edges.

Guarantees $O(VE)$ A.P.s and $O(VE^2)$ time.

Critical edge on an A.P.

Based on $\min \{ \text{capacity} - \text{flow} \}$ for all edges in an A.P.

No capacity will remain on edge after A.P. is recorded (saturated).

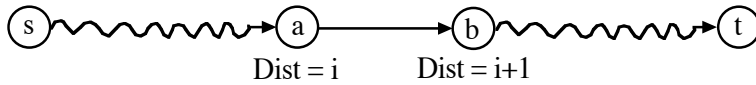
Observations

Edge may be the critical edge for several A.P.s.

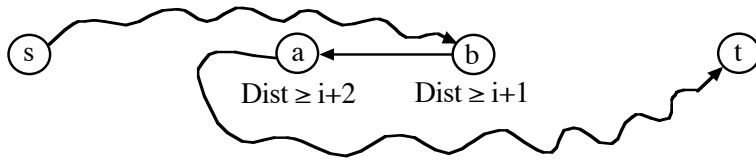
A vertex cannot get closer to source in later rounds of BFS. (See Lemma 26.7 in CLRS)

Flow must be sent in opposite direction by another A.P. before an edge can become critical again.

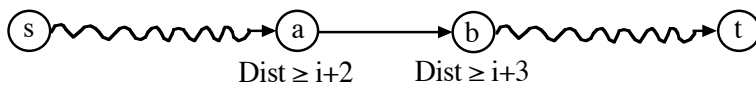
First time critical



Later



Second time critical



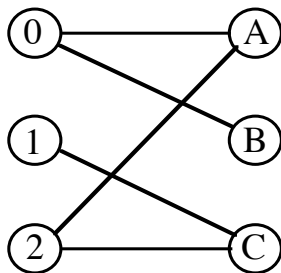
The number of distinct distances available for the tail of a repeated critical edge is bounded above by $(V-2)/2$. The number of edges that become critical is bounded above by E . Thus, $O(VE)$ A.P.s overall.

16.H. BIPARTITE MATCHING - Classic application of network flows.

Consider a bipartite graph with two disjoint sets of vertices, U (employees) and V (jobs), along with a set of edges showing which employees can handle which jobs (no teamwork and no multi-tasking superstars).

You would like to choose the maximum size subset of edges to match employees to jobs.

Example:



The following construction gives a corresponding instance of network flow.

Include source (s) and sink (t).

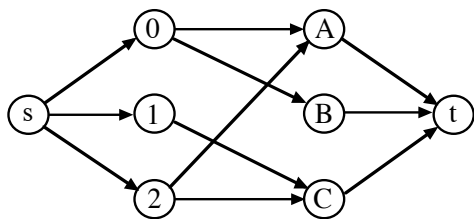
Direct all edges from U to V.

Include an edge from the source to each U vertex.

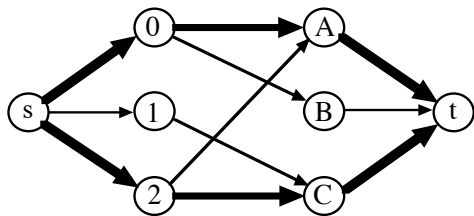
Include an edge from each V vertex to the sink.

Set each capacity to one.

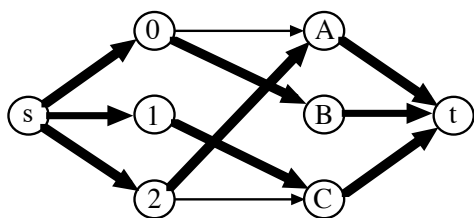
For the example:



Suppose the first two A.P.s are $s, 0, A, t$ and $s, 2, C, t$. (This may be done during a preprocessing phase that scans the adjacency list structure for unpaired vertices.)



But recall that in the residual network there will be a unit of residual capacity on $C \rightarrow 2$ and $A \rightarrow 0$, so the A.P. $s, 1, C, 2, A, 0, B, t$ is available.



This gives a (perfect) matching of $\{(0, B), (1, C), (2, A)\}$. Note that each A.P. causes a net increase of only one pair.