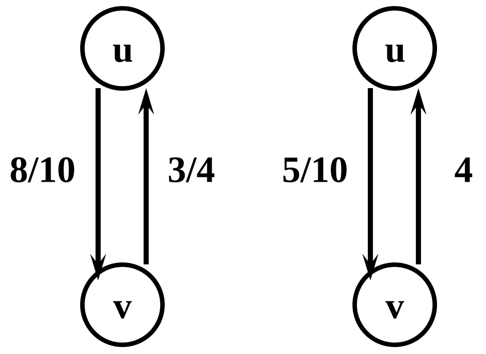
Maximum Flow (Part 2)

Net flow and value of a flow

Net Flow:
positive or negative value of f(u,v)



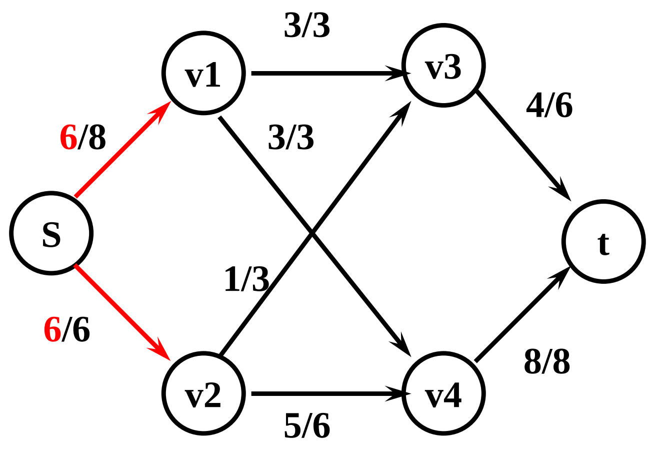
f(u,v) = 5

f(v,u) = -5

Value of a Flow f:

Def:

$$|f| = \sum_{v \in V} f(s,v)$$

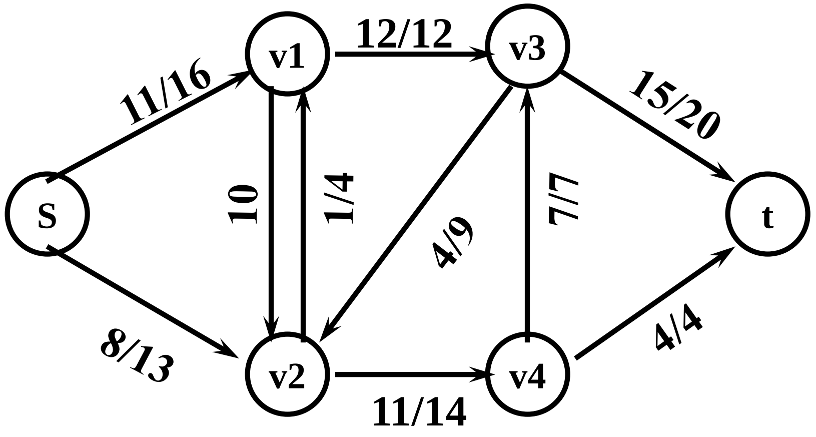


Ford Fulkerson – residual networks

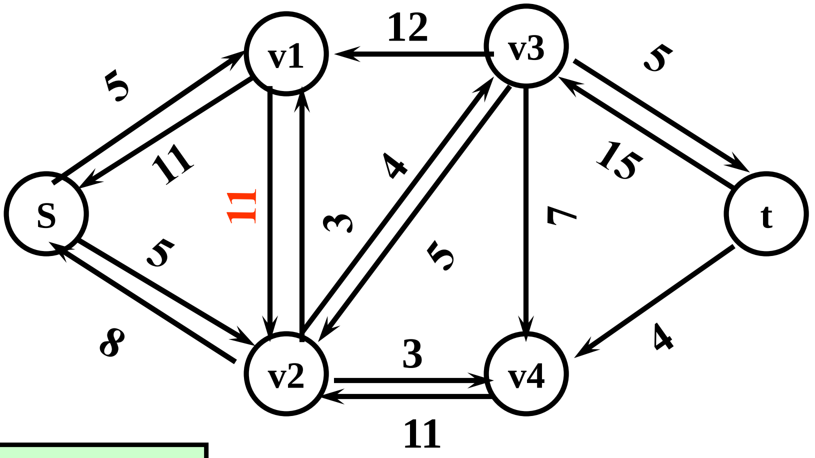
The residual network G_f of a given flow network G with a valid flow f consists of the same vertices $v \in V$ as in G which are linked with residual edges $(u,v) \in E_f$ that can admit more strictly positive net flow.

The residual capacity c_f represents the weight of each edge E_f and is the amount of additional net flow f(u,v) before exceeding the capacity c(u,v)

Flow network $G = (V,E)$



residual network $G_f = (V,E_f)$



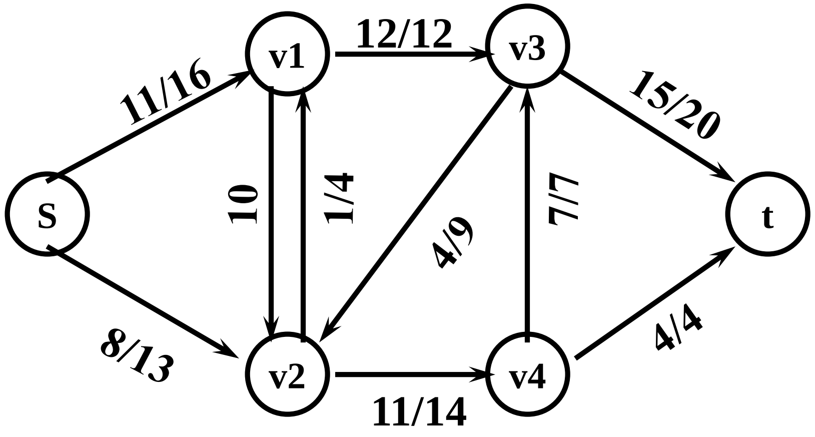
$$c_f(u,v) = c(u,v) - f(u,v)$$

Ford Fulkerson – augmenting paths

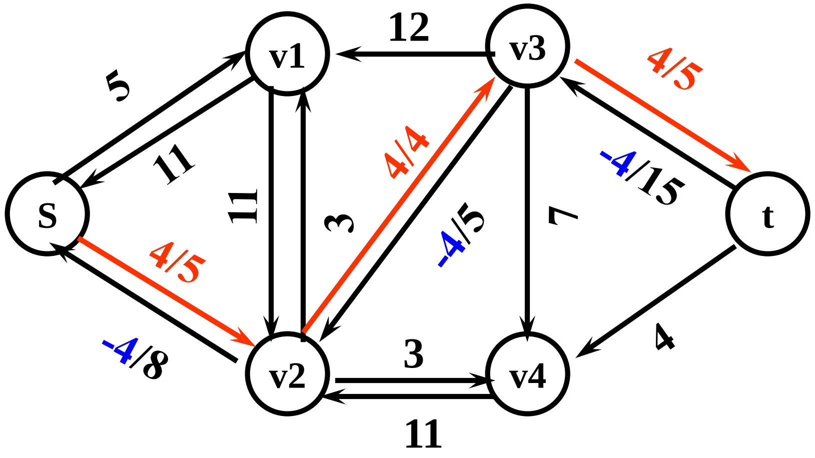
We define a flow: f_p: V x V $\rightarrow$ R such as:

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u,v) \text{ is on } p \\ -\,c_f(p) & \text{if } (v,u) \text{ is on } p \\ 0 & \text{otherwise} \end{cases}$$

Flow network G = (V,E)



residual network $G_f = (V,E_f)$



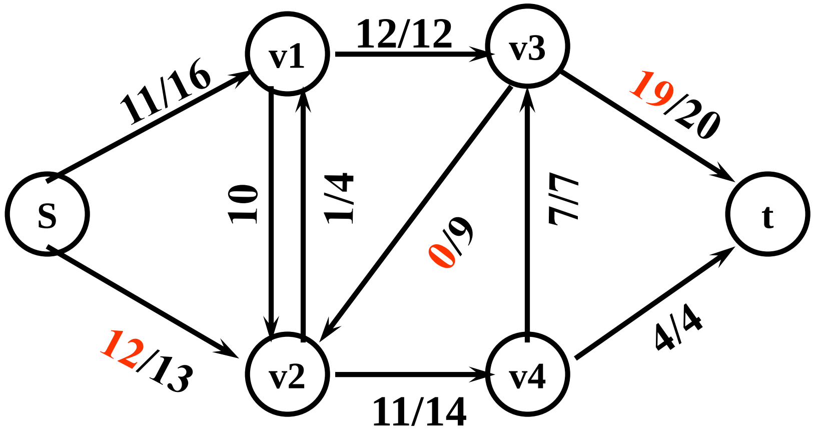
Our virtual flow f_p along the augmenting path p in G_f

Ford Fulkerson – augmenting the flow

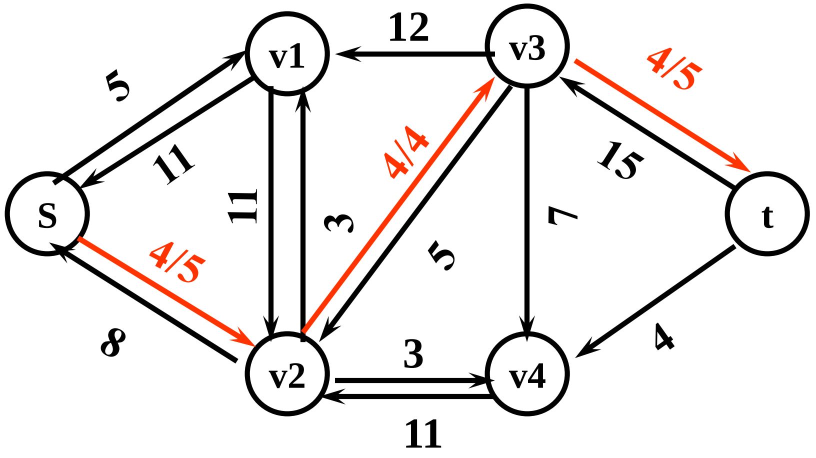
We define a flow: f_p: V x V $\rightarrow$ R such as:

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u,v) \text{ is on } p \\ -c_f(p) & \text{if } (v,u) \text{ is on } p \\ 0 & \text{otherwise} \end{cases}$$

Flow network G = (V,E)



residual network $G_f = (V,E_f)$



New flow: $\boxed{f' : V \times V \rightarrow R :\ f' = f + f_p}$

Our virtual flow f_p along the augmenting path p in G_f

Ford Fulkerson – new valid flow

proof of capacity constraint

Lemma:

$$\boxed{f' : V \times V \rightarrow R : \quad f' = f + f_p \qquad \text{in } G}$$

$$f_p(u,v) = \begin{cases} c_f(p) & \text{if } (u,v) \text{ is on } p \\ -\,c_f(p) & \text{if } (v,u) \text{ is on } p \\ 0 & \text{otherwise} \end{cases}$$

$$c_f(p) = \min\{c_f\,(u,v): (u,v) \text{ is on } p\}$$

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

Capacity constraint:

For all $u,v \in V$, we require $f(u,v) \leq c(u,v)$

Proof:

$$f_p\,(u\,,v) \leq c_f\,(u\,,v) = c\,(u\,,v) - f\,(u\,,v)$$

$$\Rightarrow (f + f_p)\,(u\,,v) = f\,(u\,,v) + f_p\,(u\,,v) \leq c\,(u\,,v)$$

Ford Fulkerson – new valid flow

proof of Skew symmetry

Lemma:

$f' : V \times V \rightarrow R : \quad f' = f + f_p \qquad$ in G

Skew symmetry:

For all $u,v \in V$, we require $f(u,v) = - f(v,u)$

Proof:

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

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

Ford Fulkerson – new valid flow

proof of flow conservation

Lemma:

$$\boxed{f' : V \times V \to R : \quad f' = f + f_p \qquad \text{in } G}$$

Flow conservation:

For all $u \in V - \{s,t\}$: $\sum_{v \in V} f(u,v) = 0$

Proof:

$$u \in V - \{s,t\} \Rightarrow \sum_{v \in V} (f + f_p)(u,v) = \sum_{v \in V} (f(u,v) + f_p(u,v))$$

$$= \sum_{v \in V} f(u,v) + \sum_{v \in V} f_p(u,v) = 0 + 0 = 0$$

Ford Fulkerson – new valid flow

Lemma:

$$\boxed{|(f + f_p)| = |f| + |f_p|}$$

Value of a Flow f:

Def:

$$\boxed{|f| = \sum_{v \in V} f(s,v)}$$

Proof:

$$|(f + f_p)| = \sum_{v \in V} (f + f_p)(s,v) = \sum_{v \in V} (f(s,v) + f_p(s,v))$$

$$= \sum_{v \in V} f(s,v) + \sum_{v \in V} f_p(s,v) = |f| + |f_p|$$

Ford Fulkerson – new valid flow

Lemma:

$\boxed{f' : V \times V \rightarrow R : \quad f' = f + f_p \qquad \text{in } G}$

$\boxed{|(f + f_p)| = |f| + |f_p| > |f|}$

Lemma shows:

if an augmenting path can be found then the above flow augmentation will result in a flow improvement.

Question: If we cannot find any more an augmenting path is our flow then maximum?

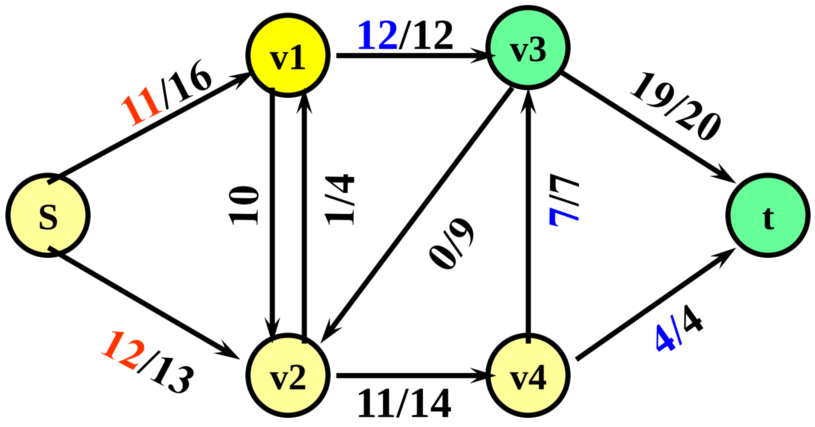
Idea: The flow in G is maximum $\Leftrightarrow$ the residual G_f contains no augmenting path.

Ford Fulkerson – cuts of flow networks

Lemma:

the value of a flow in a network is the net flow across any cut of the network

$$\boxed{f(S,T) = |f|}$$



Ford Fulkerson – cuts of flow networks

Assumption:

the value of a flow in a network is the net flow across any cut of the network

Lemma: $\boxed{f(S,T) = |f|}$

Proof:

$f(S, T) = f(S, V-S)$

$= f(S, V) - f(S, S)$

$= f(S, V) = f(s \cup [S-s], V)$

$= f(s, V) + f(S-s, V)$

$= f(s, V) = |f|$

Working with flows:

$X,Y,Z \subseteq V$ and $X \cap Y = \emptyset$

$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y)$

$f(X, X) = 0$

$f(X, Y) = - f(Y, X)$

$f(u, V) = 0$ for all $u \in V \setminus \{s, t\}$

$f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$

$f(Z, X \cup Y) = f(Z, X) + f(Z, Y)$

Ford Fulkerson – cuts of flow networks

Assumption:

The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G

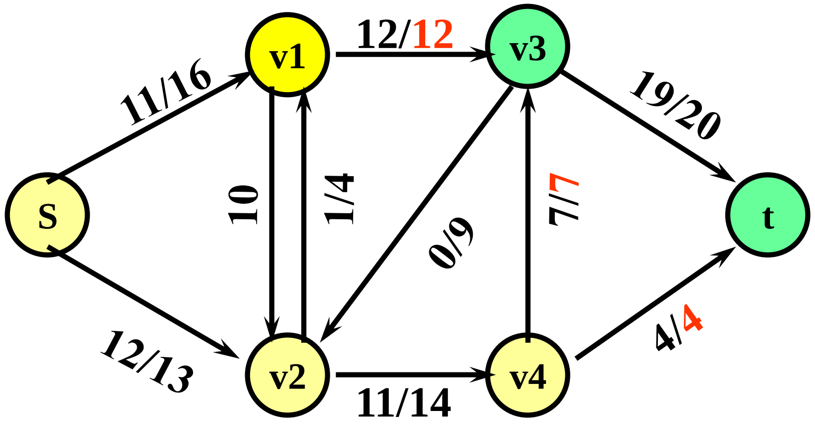
Lemma: $\boxed{|f| \leq c(S, T)}$

$$|f| = f(S,T)$$

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

$$\leq \sum_{u \in S} \sum_{v \in T} c(u,v)$$

$$= c(S,T)$$



F. Fulkerson: Max-flow min-cut theorem

If f is a flow in a flow network G = (V,E) with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. | f | = c (S, T) for some cut (S, T) of G.

proof:

(1) $\Rightarrow$ (2):

We assume for the sake of contradiction that f is a maximum flow in G but that there still exists an augmenting path p in G_f.

Then as we know from above, we can augment the flow in G according to the formula: $f' = f + f_p$. That would create a flow f´that is strictly greater than the former flow f which is in contradiction to our assumption that f is a maximum flow.

F. Fulkerson: Max-flow min-cut theorem

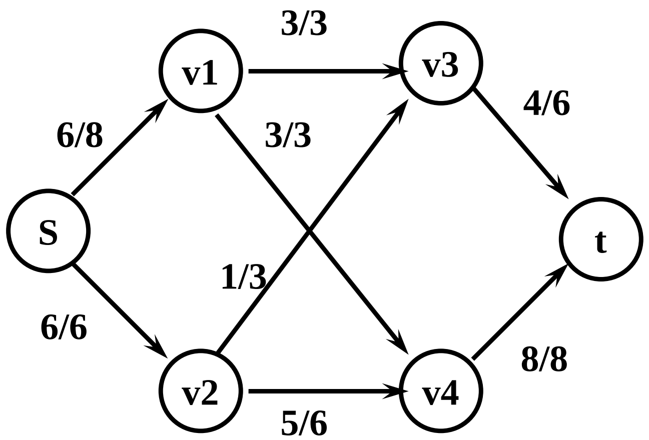
If f is a flow in a flow network G = (V,E) with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. | f | = c (S, T) for some cut (S, T) of G.

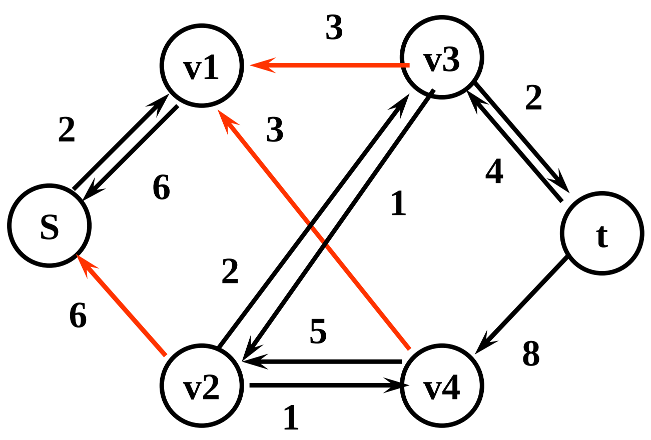
proof:

(2) $\Rightarrow$ (3):

original flow network G



residual network G_f



F. Fulkerson: Max-flow min-cut theorem

If f is a flow in a flow network G = (V,E) with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. | f | = c (S, T) for some cut (S, T) of G.

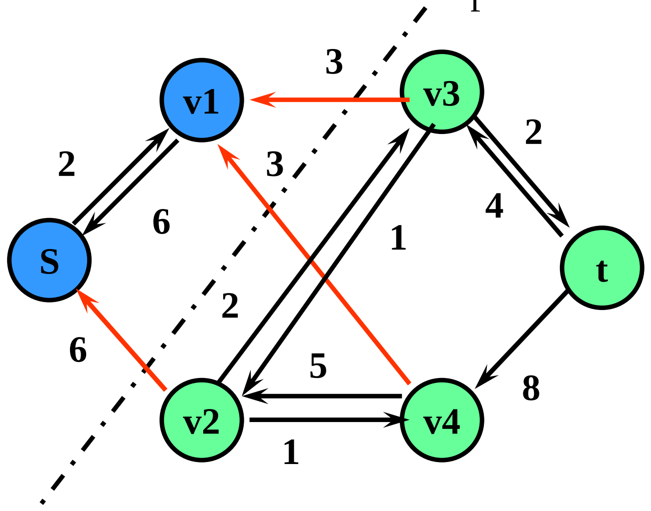
proof:

(2) $\Rightarrow$ (3): Define

$S = \{v \in V \mid \exists \text{ path } p \text{ from } s \text{ to } v \text{ in } G_f\}$

$T = V \setminus S$ (note $t \notin S$ according to (2))

residual network G_f



F. Fulkerson: Max-flow min-cut theorem

If f is a flow in a flow network G = (V,E) with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. | f | = c (S, T) for some cut (S, T) of G.

proof:

(2) $\Rightarrow$ (3): Define

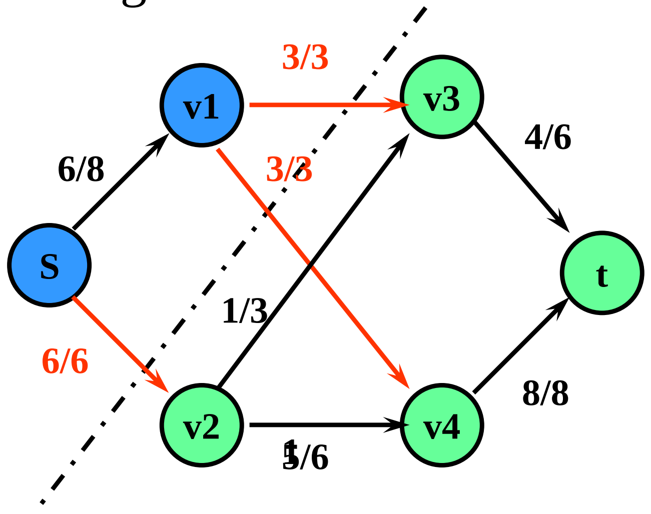
$S = \{v \in V \mid \exists \text{ path } p \text{ from } s \text{ to } v \text{ in } G_f\}$

$T = V \setminus S$ (note $t \notin S$ according to (2))

$\Rightarrow$ for $\forall\, u \in S, v \in T$: f (u, v) = c (u, v) (otherwise $(u, v) \in E_f$ and $v \in S$)

$\Rightarrow$ | f | = f (S, T) = c (S, T)

original network G



F. Fulkerson: Max-flow min-cut theorem

If f is a fow in a flow network G = (V,E) with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. | f | = c (S, T) for some cut (S, T) of G.

proof:

(3) $\Rightarrow$ (1): as proofed before | f | = f (S, T) $\leq$ c (S, T)

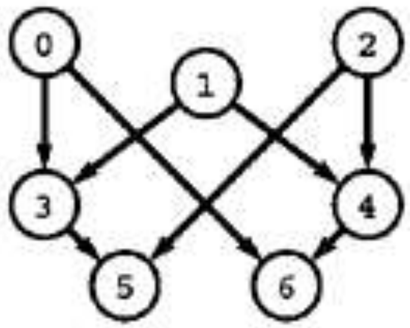
the statement of (3) : | f | = c (S, T) implies that f is a maximum flow

Reduction to Maximum Flow

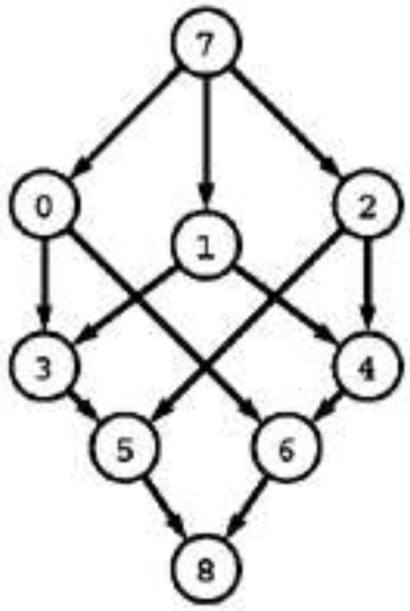
R1: Multiple sources and sinks

- Problem:
  - What if you have a problem with more than one source and more than one sink?
  - The network at the top has three sources (0, 1, and 2) and two sinks (5 and 6).
- Reduction: Create a network which
  - is a copy of the original network
  - with the addition of a new source 7 and
  - a new sink 8.
  - There is an edge from 7 to each original-network source with capacity equal to the sum of the capacities of that source's outgoing edges,
  - an edge from each original-network sink to 8 with capacity equal to the sum of the capacities of that sink's incoming edges.

	cap
0-3	2
0-6	3
1-3	3
1-4	1
2-4	1
2-5	1
3-5	2
4-6	3



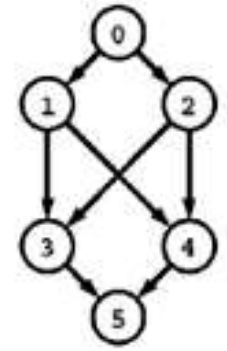
	cap
0-3	2
0-6	3
1-3	3
1-4	1
2-4	1
2-5	1
3-5	2
4-6	3
7-0	5
7-1	4
7-2	2
5-8	3
6-8	6



R2: Removing vertex capacities

- Problem:
  - Given a flow network, find a maxflow satisfying additional constraints specifying that the flow through each vertex must not exceed some fixed capacity.

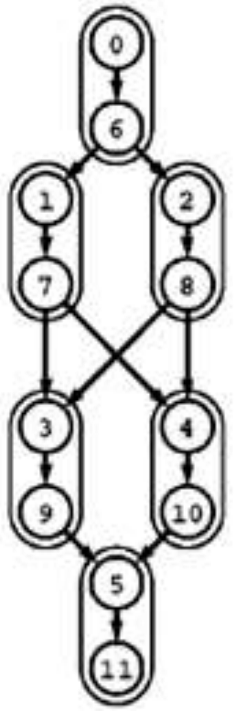
	cap
0-1	2
0-2	3
1-3	3
1-4	1
2-3	1
2-4	1
3-5	2
4-5	3



	0	1	2	3	4	5
capV	4	4	3	1	3	4

- Reduction: Create a new network
  - Associate a new vertex u* (where u* denotes u+V) with each vertex u,
  - add an edge u-u* whose capacity is the capacity of u,
  - include an edge u*-v for each edge u-v.

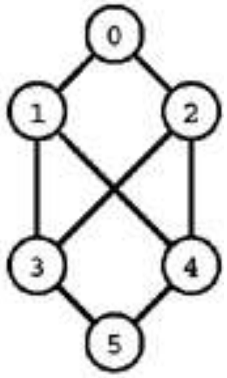
	cap
6-1	2
6-2	3
7-3	3
7-4	1
8-3	1
8-4	1
9-5	2
10-5	3
0-6	4
1-7	4
2-8	3
3-9	1
4-10	3
5-11	4



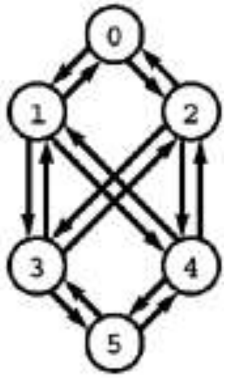
R4: Reduction from undirected networks

- Problem:
  - Given an undirected weighted graph find the maximum flow
- Reduction: Create a network s.t.
  - we can consider it to be a directed network with edges in each direction.

	cap
0-1	2
0-2	3
1-3	3
1-4	1
2-3	1
2-4	1
3-5	2
4-5	3



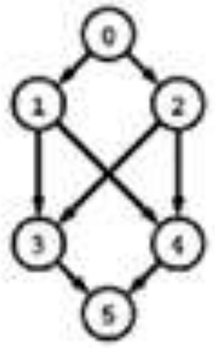
	cap
0-1	2
1-0	2
0-2	3
2-0	3
1-3	3
3-1	3
1-4	1
4-1	1
2-3	1
3-2	1
2-4	1
4-2	1
3-5	2
5-3	2
4-5	3
5-4	3

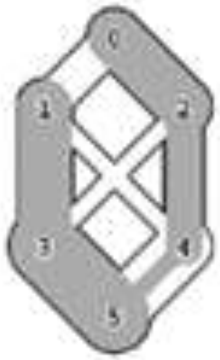


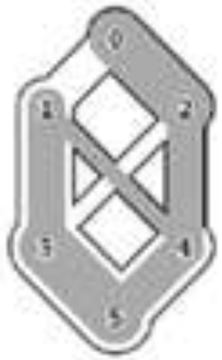
R5: Feasible flow

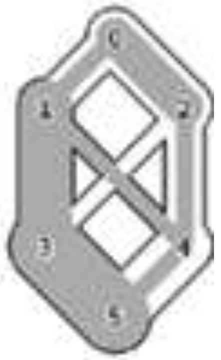
- Problem:
  - Suppose that a weight is assigned to each vertex in a flow network and is to be interpreted as supply (if positive) or demand (if negative), with the sum of the vertex weights equal to zero. Define a flow to be feasible if the difference between each vertex's outflow and inflow is equal to that vertex's weight (supply if positive and demand if negative). Given such a network, determine whether or not a feasible flow exists.

	cap	flow	flow	flow
0-1	2	0	0	1
0-2	3	3	3	2
1-3	3	3	2	3
1-4	1	0	1	1
2-3	1	0	0	0
2-4	2	2	2	1
3-5	4	4	3	4
4-5	3	1	2	1





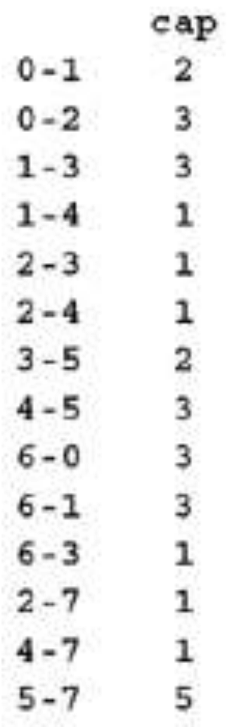




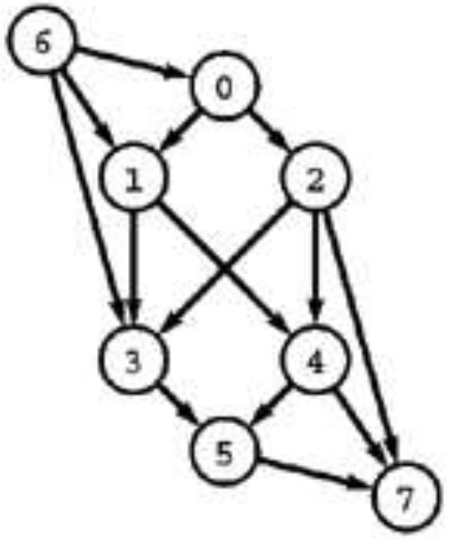
	0	1	2	3	4	5
supply	3	3		1		
demand			1		1	5

R5: Reduction from feasible solution

- Reduction: Create a network
  - by adding edges from a new source vertex to the supply vertices (each with capacity equal to the amount of the supply) and
  - edges to a new sink vertex from the demand vertices (each with capacity equal to the amount of the demand).
  - The network has a feasible flow if and only if this network has a flow (a maxflow) that fills all the edges from the sink and all the edges to the source.

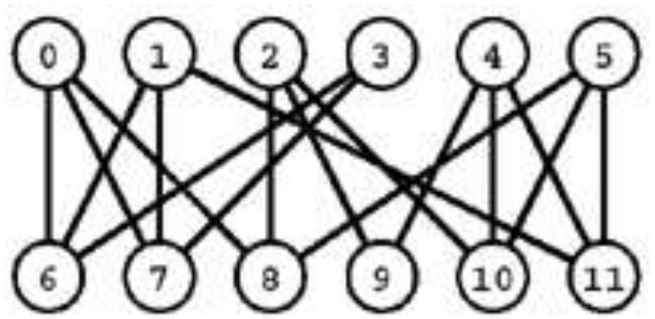


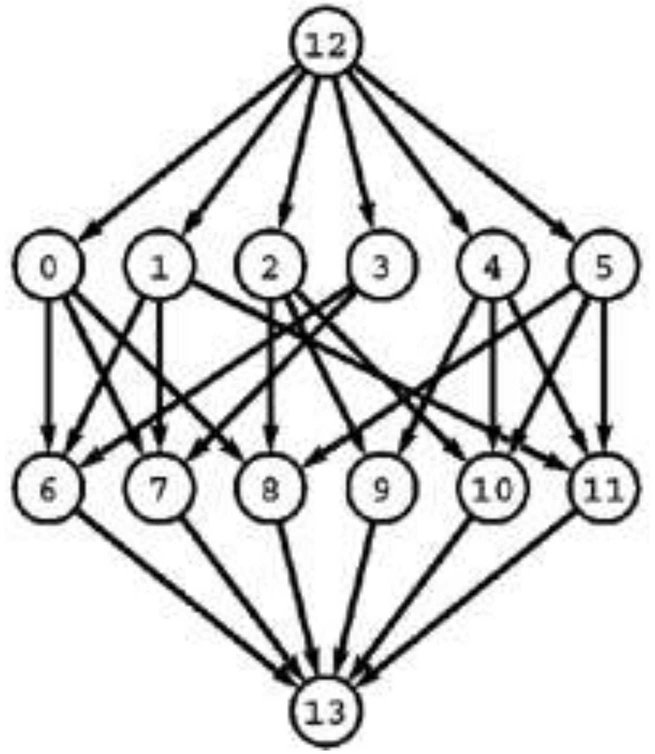
	cap
0-1	2
0-2	3
1-3	3
1-4	1
2-3	1
2-4	1
3-5	2
4-5	3
6-0	3
6-1	3
6-3	1
2-7	1
4-7	1
5-7	5



R6: Bipartite matching

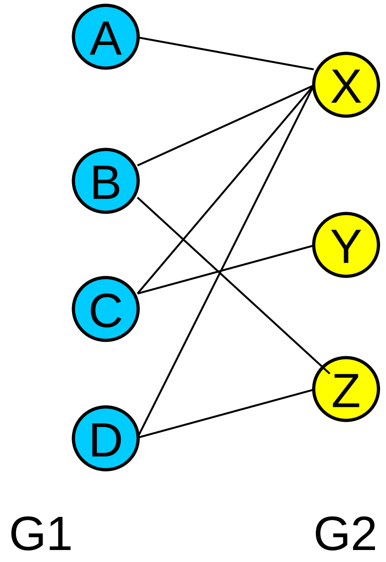
- Problem:
  - Given a bipartite graph, find a set of edges of maximum cardinality such that each vertex is connected to at most one other vertex.
- Reduction: Construct an st-network
  - by directing all the edges from the top row to the bottom row,
  - adding a new source with edges to each vertex on the top row,
  - adding a new sink with edges to each vertex on the bottom row, and
  - assigning capacity 1 to all edges.
  - In any flow, at most one outgoing edge from each vertex on the top row can be filled and at most one incoming edge to each vertex on the bottom row can be filled, so a solution to the maxflow problem on this network gives a maximum matching for the bipartite graph.

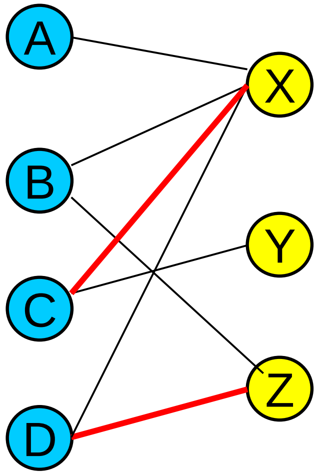




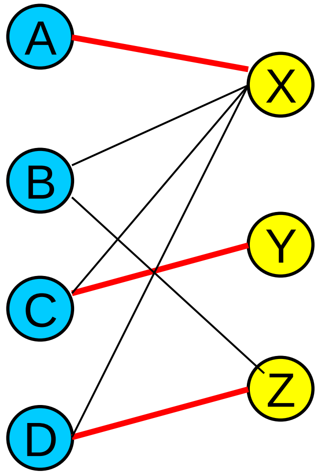
Model for Matching Problem

- Group1 on leftmost set, Group2 on rightmost set, edges if they are compatible





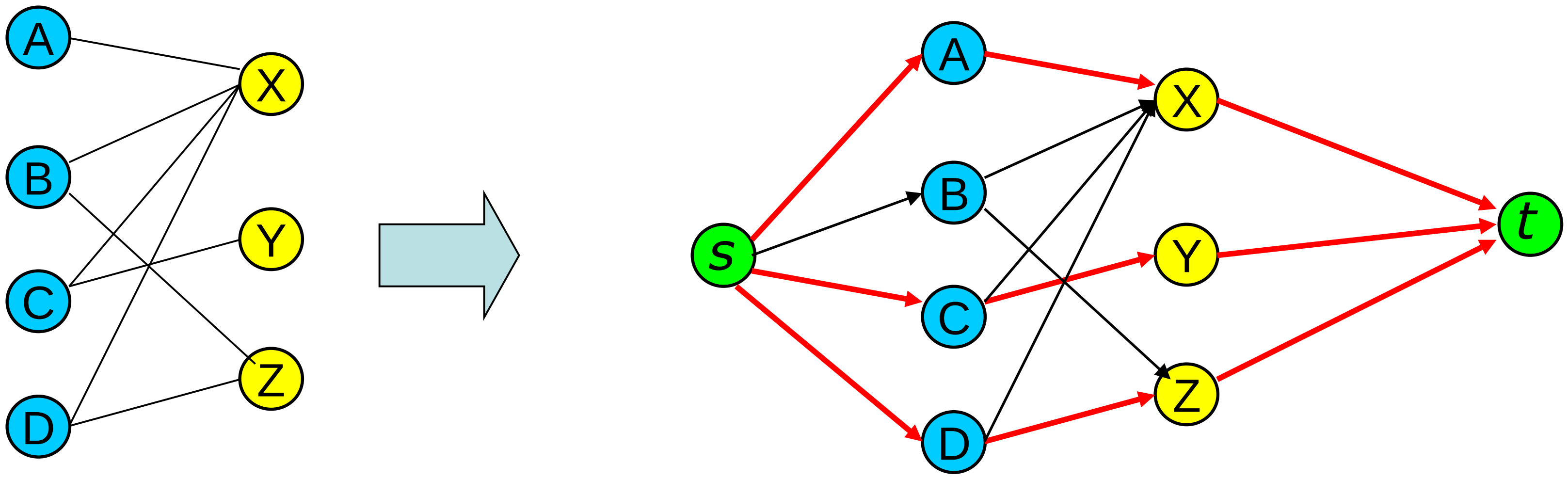
A matching



Optimal matching

Solution Using Max Flow

- Add a supersouce, supersink, make each undirected edge directed with a flow of 1



Since the input is 1, flow conservation prevents multiple matchings

Edge Connectivity

- What is the minimum number of edges that need to be removed to separate a given graph into two pieces?
- Find a set of edges of minimal cardinality that does this separation.

Edge Connectivity for undirected graph

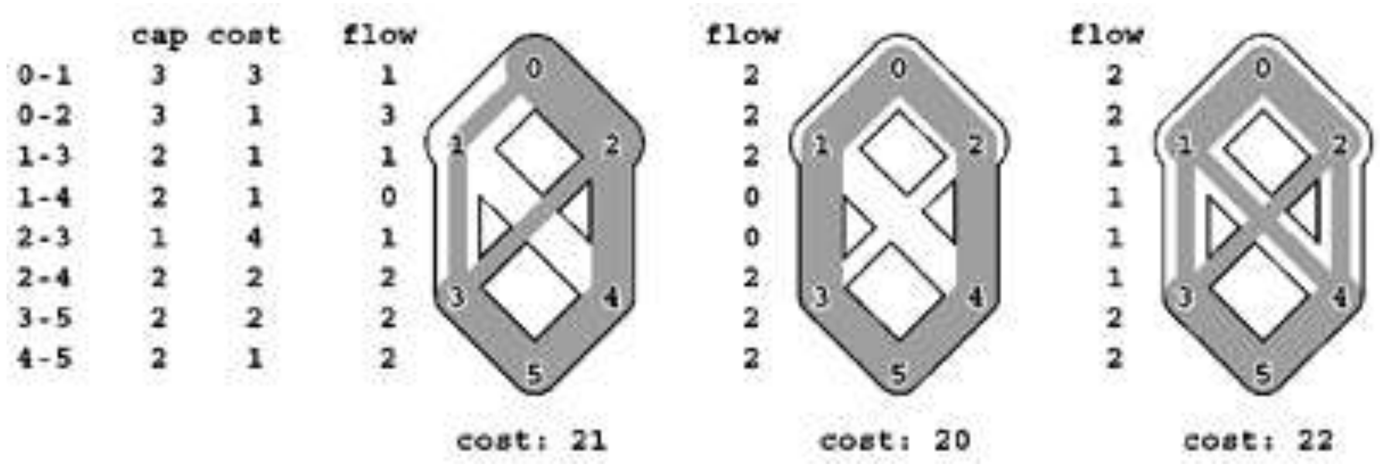
- Theorem:
  - The time required to determine the edge connectivity of an undirected graph is $O(E^2)$.
- Proof:
  - We can compute the minimum size of any cut that separates two given vertices by computing the maxflow in the st-network formed from the graph by assigning unit capacity to each edge.
  - The edge connectivity is equal to the minimum of these values over all pairs of vertices.
  - We do not need to do the computation for all pairs of vertices,
  - Let s* be a vertex of minimal degree in the graph.
  - Note that the degree of s* can be no greater than 2E/V.
  - Consider any minimum cut of the graph.
    - By definition, the number of edges in the cut set is equal to the graph's edge connectivity.
    - The vertex s* appears in one of the cut's vertex sets, and the other set must have some vertex t,
    - so the size of any minimal cut separating s* and t must be equal to the graph's edge connectivity. Therefore, if we solve V-1 maxflow problems (using s* as the source and each other vertex as the sink), the minimum flow value found is the edge connectivity of the network.
  - Now, any augmenting-path maxflow algorithm with s* as the source uses at most 2E/V paths;
  - so, if we use any method that takes at most E steps to find an augmenting path, we have a total of at most (V – 1)(2 E/V)E steps to find the edge connectivity and that implies the stated result.

Vertex Connectivity

- What is the minimum number of vertices that need to be removed to separate a given graph into two pieces?
- Assignment

Min Cost Maximum Flow

- Flow cost:
  - The flow cost of an edge in a flow network with edge costs is the product of that edge's flow and cost. The cost of a flow is the sum of the flow costs of that flow's edges.
- Mincost maxflow:
  - Given a flow network with edge costs, find a maxflow such that no other maxflow has lower cost.



	cap	cost	flow	flow	flow
0-1	3	3	1	2	2
0-2	3	1	3	2	2
1-3	2	1	1	2	1
1-4	2	1	0	0	1
2-3	1	4	1	0	1
2-4	2	2	2	2	1
3-5	2	2	2	2	2
4-5	2	1	2	2	2

- These flows all have the same (maximal) value, but their costs (the sum of the products of edge flows and edge costs) differ. The maxflow in the center has minimal cost (no maxflow has lower cost).

Extended def: “Residual Network”

- Residual network
  - Given a flow in a flow network with edge costs, the residual network for the flow has the
    - same vertices as the original and
    - one or two edges in the residual network for each edge in the original,
    - For each edge u-v in the original, let f be the flow, c the capacity, and x the cost.
      - If f is positive, include an edge v-u in the residual with capacity f and cost -x;
      - if f is less than c, include an edge u-v in the residual with capacity c-f and cost x.

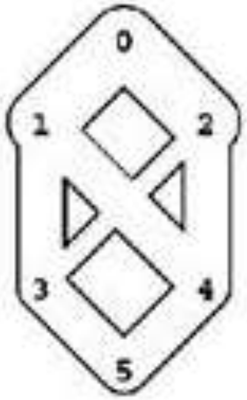
Mincost Maxflo Theorem

- Theorem:
  - A maxflow is a mincost maxflow if and only if its residual network contains no negative-cost (directed) cycle.

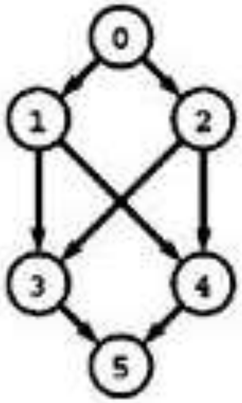
How to solve Mincost Maxflow Problem

- Algorithm:
  - Find a maxflow.
  - Augment the flow along any negative-cost cycle in the residual network, continuing until none remain.
- Implementation:
  - We use any maxflow implementation to find the initial maxflow and the Bellman–Ford algorithm to find negative cycles

	cap	cost	flow
0-1	3	3	0
0-2	3	1	0
1-3	2	1	0
1-4	2	1	0
2-3	1	4	0
2-4	2	2	0
3-5	2	2	0
4-5	2	1	0



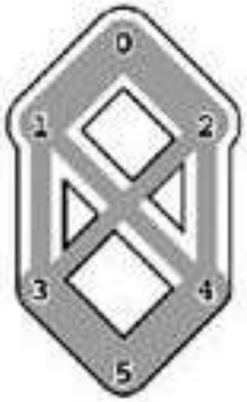
0-1	3
0-2	3
1-3	2
1-4	2
2-3	1
2-4	2
3-5	2
4-5	2



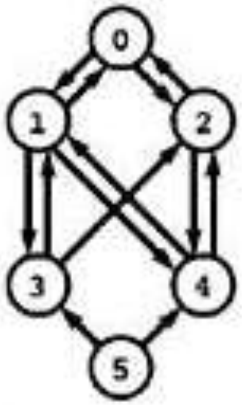
initial maxflow

	cap	cost	flow
0-1	3	3	2
0-2	3	1	2
1-3	2	1	1
1-4	2	1	1
2-3	1	4	1
2-4	2	2	1
3-5	2	2	2
4-5	2	1	2

total cost: 22



0-1	1	1-0	2
0-2	1	2-0	2
1-3	1	3-1	1
1-4	1	4-1	1
		3-2	1
2-4	1	4-2	1
		5-3	2
		5-4	2

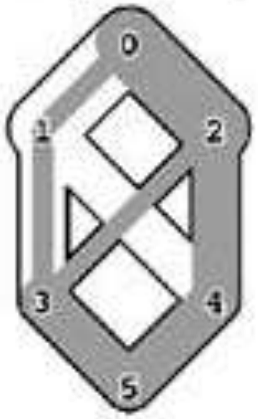


negative cycles: 4-1-0-2-4
3-2-0-1-3
3-2-4-1-3

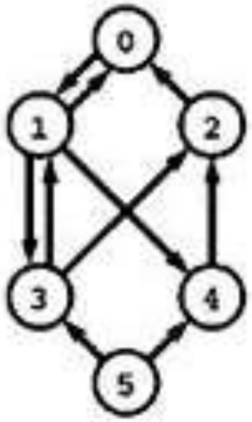
augment +1 on 4-1-0-2-4 (cost -1)

	cap	cost	flow
0-1	3	3	1
0-2	3	1	3
1-3	2	1	1
1-4	2	1	0
2-3	1	4	1
2-4	2	2	2
3-5	2	2	2
4-5	2	1	2

total cost: 21



0-1	2	1-0	1
		2-0	3
1-3	1	3-1	1
1-4	2		
		3-2	1
		4-2	2
		5-3	2
		5-4	2

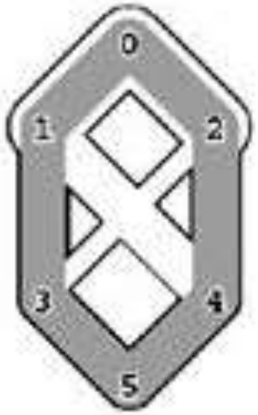


negative cycle: 3-2-0-1-3

augment +1 on 3-2-0-1-3 (cost -1)

	cap	cost	flow
0-1	3	3	2
0-2	3	1	2
1-3	2	1	2
1-4	2	1	0
2-3	1	4	0
2-4	2	2	2
3-5	2	2	2
4-5	2	1	2

total cost: 20



0-1	1	1-0	2
0-2	1	2-0	2
		3-1	2
1-4	2		
2-3	1		
		4-2	2
		5-3	2
		5-4	2