CSC373 Worksheet 5 Solution

August 12, 2020

1. *Proof.* Assume that a flow network $G = (V, E)$ violates the assumption that the network contains a path $s \rightsquigarrow v \rightsquigarrow t$ for all vertices $v \in V$. Let u be a vertex for which there is no path $s \rightsquigarrow u \rightsquigarrow t$.

   I must show such that there is no flow at vertex u. That is, there exists a maximum flow f in G such that $f(u, v) = f(v, u) = 0$ for all vertices $v \in V$.

   Assume for the sake of contradiction that there is some vertex u with flow f. That is, there exists some vertices $v \in V$ such that $f(u, v) > 0$ or $f(v, u) > 0$.

   I see that three cases follows, and I will prove each separately.

   1. **Cases 1:** $f(u, v) = 0$ and $f(v, u) > 0$

      Here, assume that $f(u, v) = 0$ for all $v \in V$ and $f(v, u) > 0$ for some $v \in V$.

      Then, we can write $\sum_{v \in V} f(u, v) = 0$ and $\sum_{v \in V} f(v, u) > 0$

      But this violates the flow conservation property (i.e $\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u)$)

      Thus, by proof by contradiction, $f(u, v) = 0$ and $f(v, u) = 0$ for all $v \in V$ and all $u \in V$ with no path $s \rightsquigarrow u \rightsquigarrow t$.

   2. **Cases 2:** $f(u, v) > 0$ and $f(v, u) = 0$

      Here, assume that $f(u, v) > 0$ for some $v \in V$ and $f(v, u) = 0$ for all $v \in V$.

      Then, by similar work as case 1, the same result follows.

3. **Cases 3:** $f(u,v) > 0$ and $f(v,u) > 0$

   Here, assume that $f(u,v) > 0$ and $f(v,u) > 0$ for some $v \in V$.

   Since $s \rightsquigarrow v \rightsquigarrow t$ and u is connected by some vertices v, we can write $s \rightsquigarrow u \rightsquigarrow t$.

   Then, this violates the fact in header that the vertex u has no path $s \rightsquigarrow u \rightsquigarrow t$.

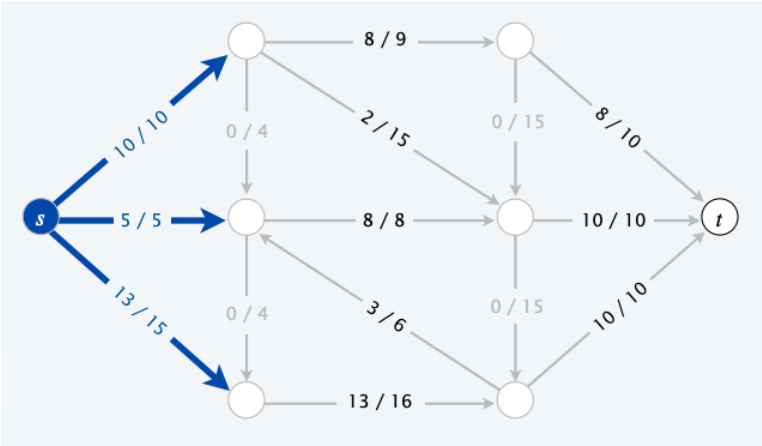
   Thus, by proof by contradiction, $f(u,v) = 0$ and $f(v,u) = 0$ for all $v \in V$ and all $u \in V$ with no path $s \rightsquigarrow u \rightsquigarrow t$.

□

Notes

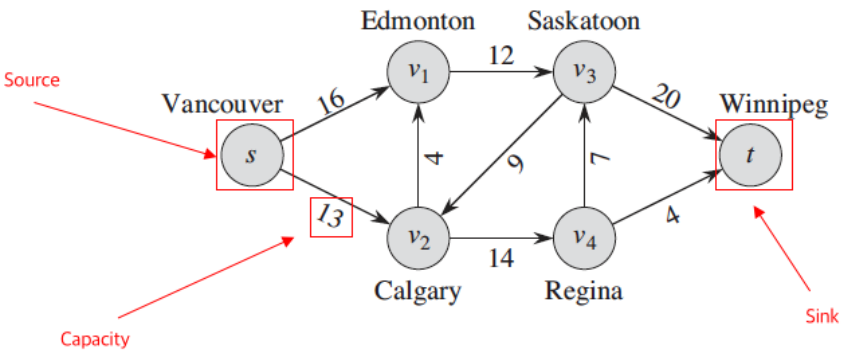
- **Maximum Flow:**
  - Finds a flow of maximum value [1]

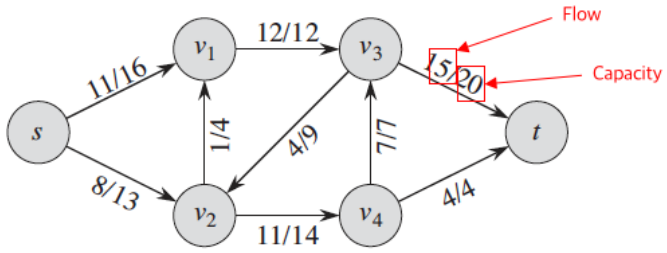
    Example



    Here, the maximum flow is $10 + 5 + 13 = 28$

- **Flow Network:**
  - $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative capacity $c(u, v) \geq 0$.
  - Two vertices must exist: **source** s and **sink** t
  - **path** from source s to vertax v to sink t is represented by $s \rightsquigarrow v \rightsquigarrow t$





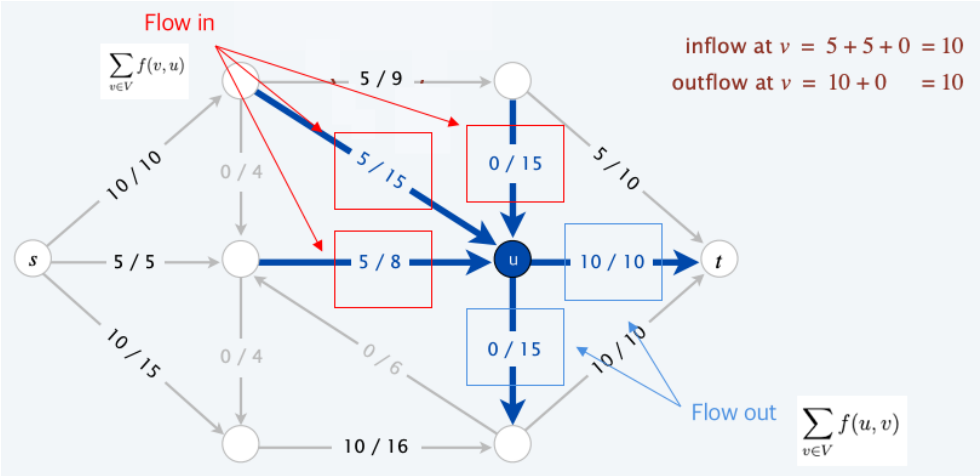
- **Capacity:**
  - Is a non-negative function $f : V \times V \to \mathbb{R}_{\geq 0}$
  - Has **capacity constraint** where for all $u, v \in V$ $0 \leq f(u, v) \leq c(u, v)$
    - Means flow cannot be above capacity constraint
- **Flow:**
  - Is a real valued function $f : V \times V \to \mathbb{R}$ in G
  - Satisfies **capacity constraint** (i.e for all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$)
  - Satisfies **flow conservation**

    For all $u \in V - \{s, t\}$, we require

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v) \tag{1}$$

    - Means flow into vertex u is the same as flow going out of vertex u. [1]
    - $\sum_{v \in V} f(u, v)$ means flow <u>out of</u> vertex u
    - $\sum_{v \in V} f(v, u)$ means flow <u>into</u> vertex u
    - $v \in V$ in $\sum_{v \in V} f(u, v)$ means all vertices that are an edge away from vertex u

    <u>Example:</u>



References

1) Princeton University, Network Flow 1, link

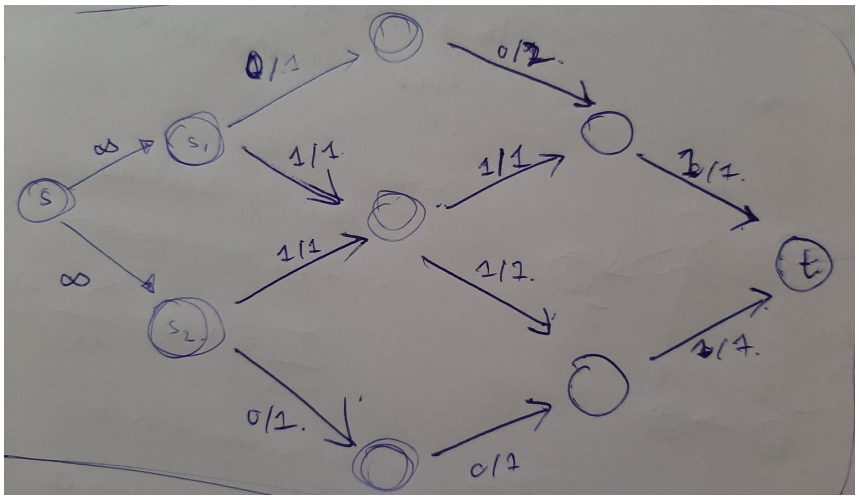
2. I need to formulate the problem of determining whether both of professor Adam's two children can go to the same school as maximum-flow problem.

   The problem statement tells us the following:

   1. There is 1 supersource (location of home)
   2. There is 1 sink (location of school)
   3. There are two sources (s_1 as child 1, s_2 as child 2)
   4. Edge (u, v) has capacity of 0 or more (0 representing unavailable sidewalk, 1 for sidewalk with capacity of 1, 2 for street with capacity of 2 and so on)
   5. Each vertex represents corner of intersection, and two children can have their paths crossing here.
   6. Has flow of 2, 1 or 0 (1 is where one of the two children walking on the road. 0 is none.)

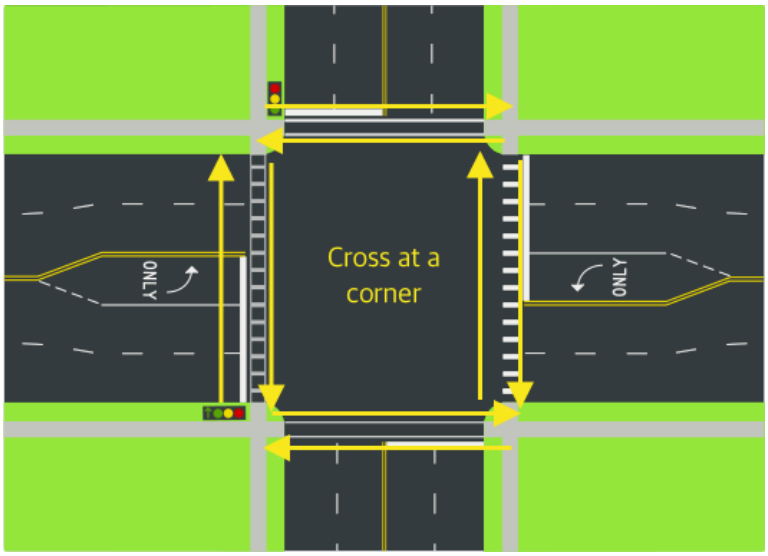
   Here we are to find whether children must go on to a vertex and out to the same edge with the flow of 2, or determine whether there is only edge to school with capacity of 1 or less.

   If none, then both children can safely go to school.



Notes:

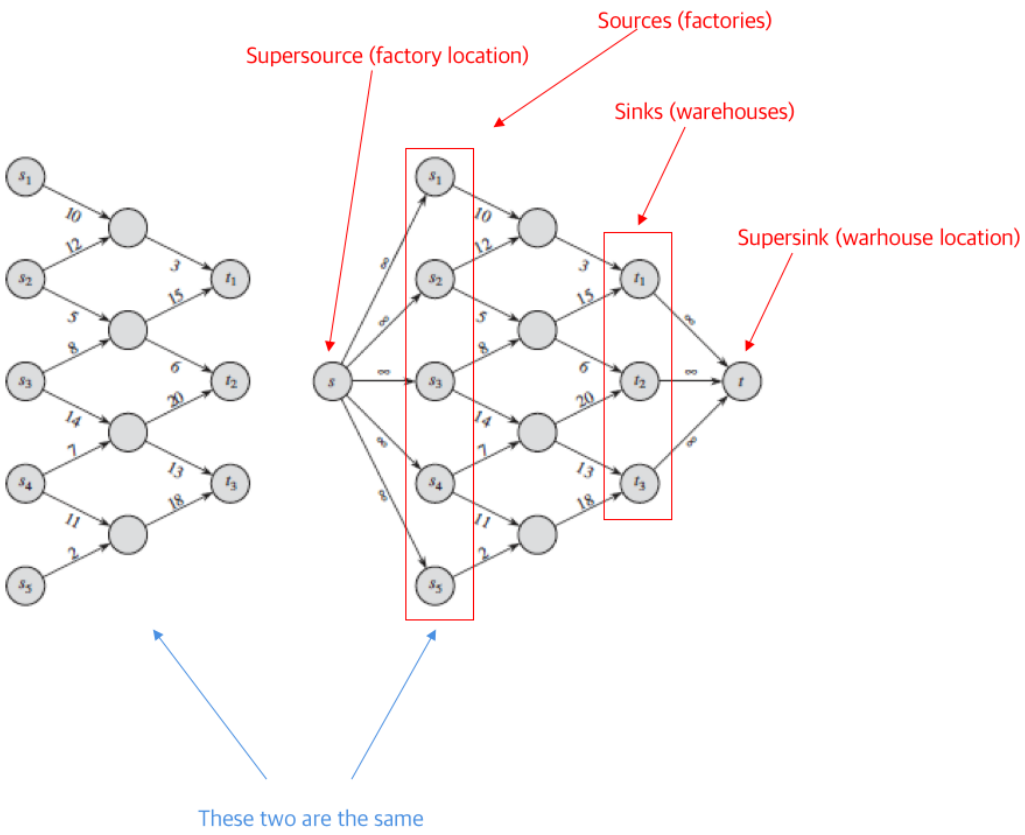
- **Cross at a Corner**
  - Means to walk across the street at a corner of the intersection.



- **Multiple Sources and Sinks**
  - Has edges (s, s_i) where $i = 1...n$ and (t_j, t) where $j = 1...n$ with capacity of ∞

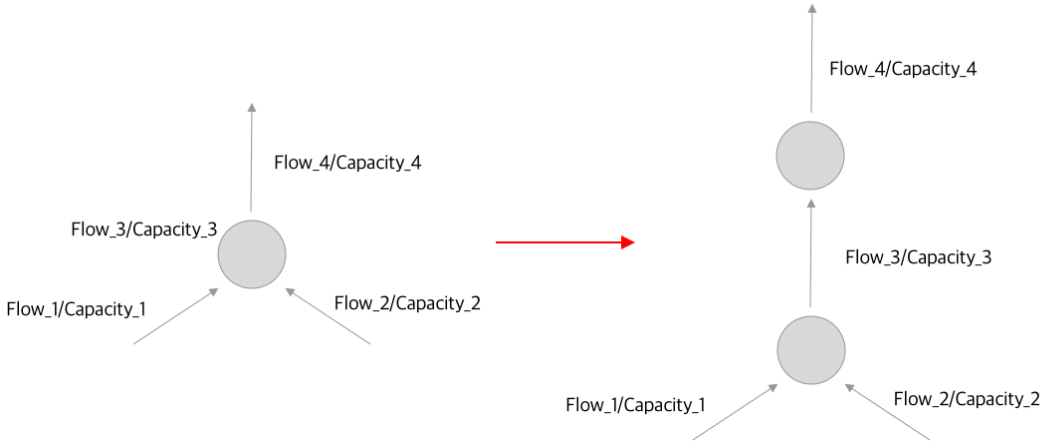
Example:

Lucky Puck Company having a set of m factories $\{s_1, s_2, ..., s_m\}$, and a set of n warehourses and n warehouses $\{t_1, t_2, ..., t_n\}$



3. I need to show how to transform a flow network $G = (V, E)$ with vertex capacities into an equivalent flow network $G' = (V', E')$ without vertex capacities.

   For each vertex capacities, change as follows.



   After transformation, there will be m more edges and verticies, where m represents the number of vertex capacities in G.

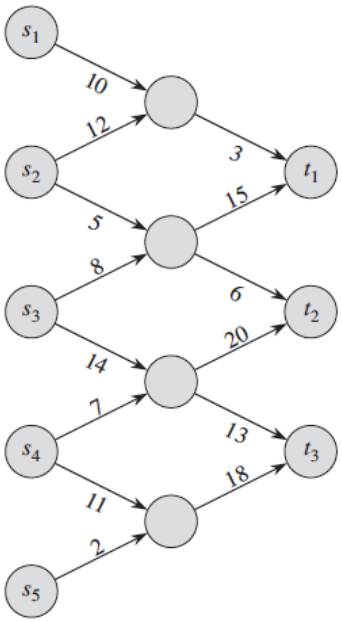
   Notes:

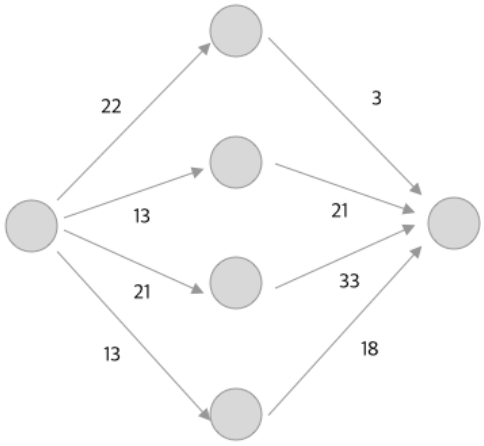
- **Vertex Capacities**
  - Each vertex v has limit $l(v)$ on how much flow can pass through v

4. I need to show how to convert the problem of finding a flow f that obeys the constraints into the problem of finding a maximum flow in a single source, single-sink flow network

   The steps are as follows:

   - Combine all sources s_i into a single source s
   - Combine all sinks t_j into a single sink t
   - Connect source s to each adjacent vertex v with edge weight $\sum_i f(s_i, v) = p_i$
     - The total edge weight from s should be $\sum_i p_i$
   - Connect each adjacent vertex v of t to t with edge weight $\sum_j f(v, t_j) = q_j$
     - The total edge weight to t should be $\sum_j q_j$
   - Find a simple path from s to t with the maximum amount of total flow





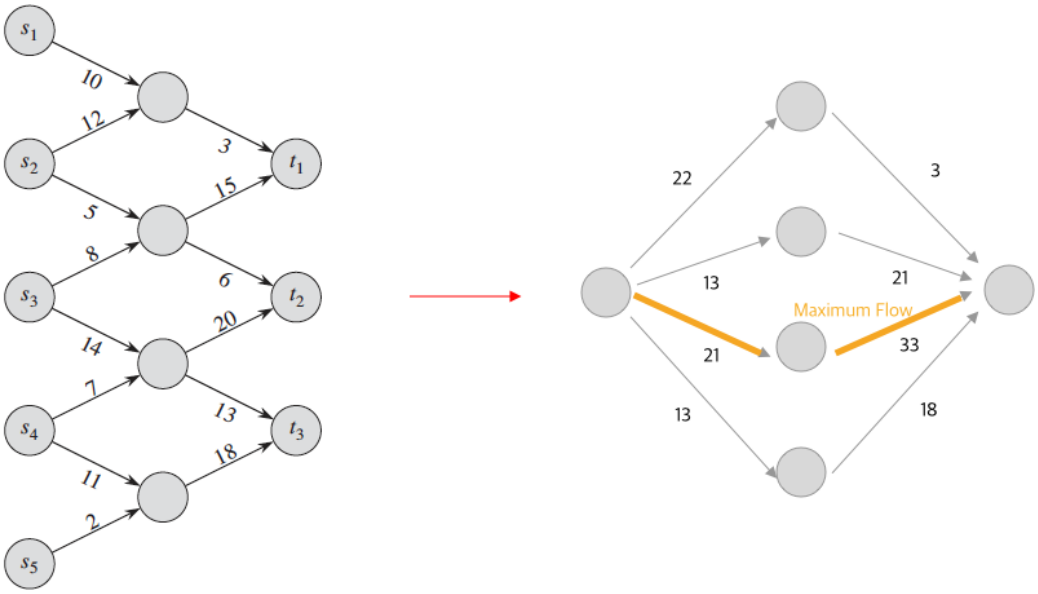
Correct Solution:

I need to show how to convert the problem of finding a flow f that obeys the constraints into the problem of finding a maximum flow in a single source, single-sink flow network

The steps are as follows:

- Combine all sources s_i into a single source s
- Combine all sinks t_j into a single sink t
- Connect source s to each adjacent vertex v with edge weight $\sum_i f(s_i, v) = p_i$
  - The total edge weight from s should be $\sum_i p_i$
- Connect each adjacent vertex v of t to t with edge weight $\sum_j f(v, t_j) = q_j$
  - The total edge weight to t should be $\sum_j q_j$
- Find a simple path from s to t with the maximum amount of total flow

Example



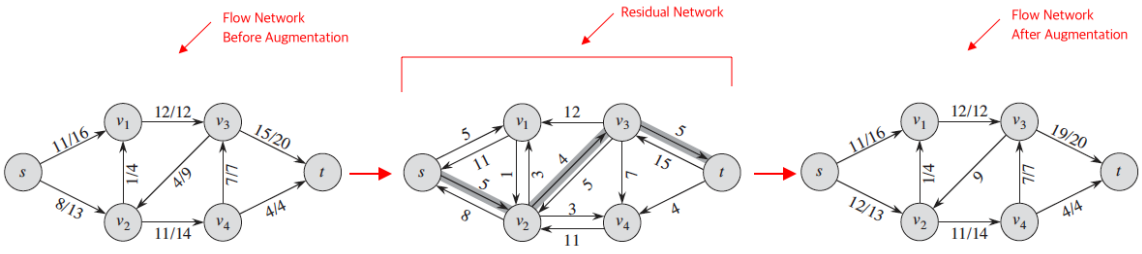
Notes:

- **Ford-Fulkerson Method**
  - Is a greedy algorithm that solves the maximum-flow problem
    - Determines maximum flow from start vertex to sink vertex in a graph
  - Called method (not algorithm) because several different implementations with different running time is used

FORD-FULKERSON-METHOD(G, s, t)

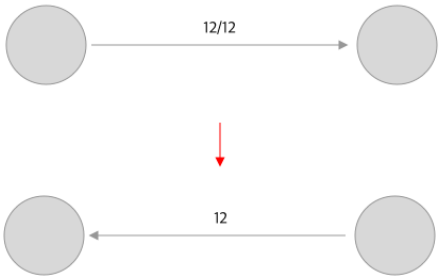
1 initialize flow f to 0
2 **while** there exists an augmenting path p in the residual network G_f
3 augment flow f along p
4 **return** f

- **Residual Network**
  - Indicates how muh more flow is allowed in each edge in the network graph [1]
  - Consists of edges with capacities that represents how we can change the flow on edges of G.
  - Provides roadmap for adding flow to the original flow network

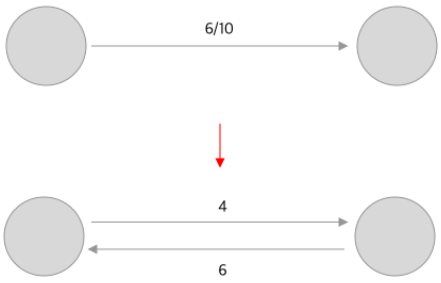


Steps

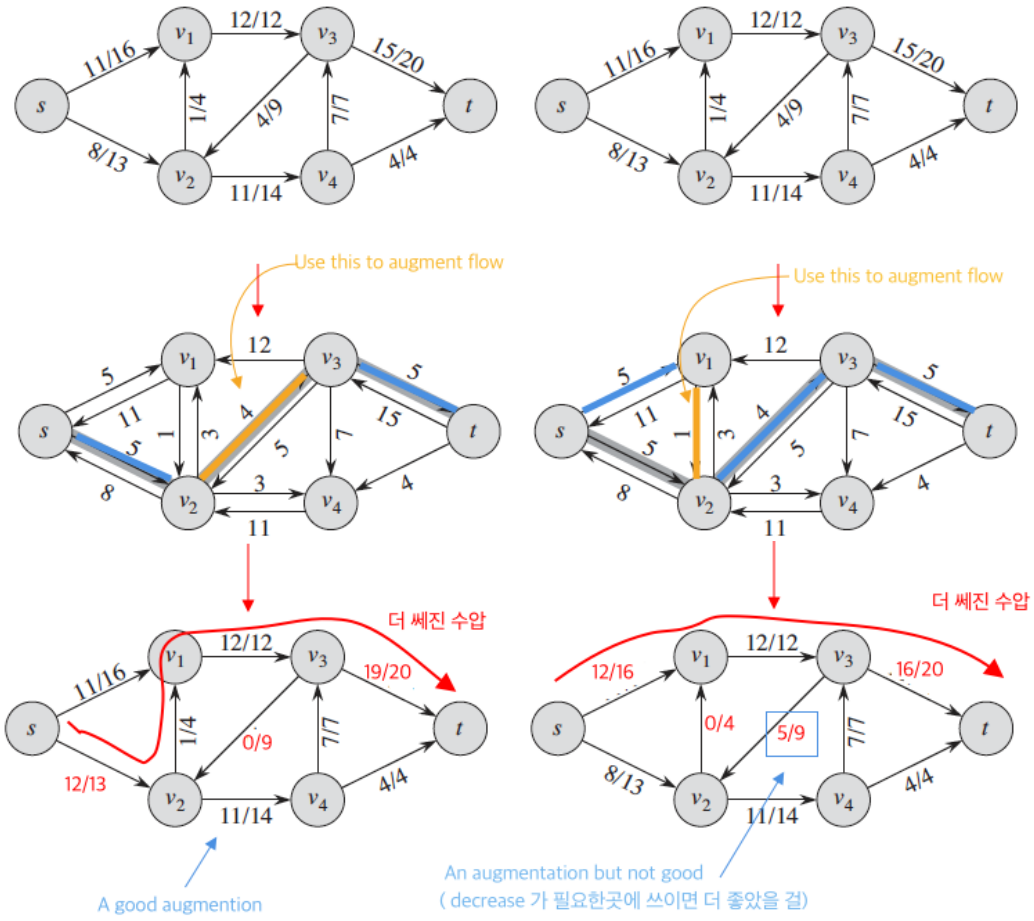
1) $Flow = Capacity$: Opposite arrow



2) $Flow < Capacity$:
   - $Flow$: Oppisite Arrow
   - $Capacity - Flow$: Current Arrow



- **Augmenting Path**
  - Is a path from source S to sink T where you can increase the amount of flow
  - Is a path that doesn't contain cycle (simple path) [2]



  - Edge (u, v) of an augmented path can be increased by upto $c_f(u, v)$ withhout violating the capacity constraint

- **Augmentation**
  - 한국어로 '불필요한 수압 decrease 해서 앞으로 가는 수압 더 쎄게 만들기'
  - Is symbolized by $f \uparrow f'$

* f is a flow in G
* f' is a flow in the residual network G_f

References

1) Hacker Earth, Maximum Flow, link
2) Stack Overflow, What Exactly Is Augmentation Path, link

5. The augmented flow satisfies flow conservation, but not capacity constraint.

Proof. Let $G = (V, E)$ be a flow network with sources s and sink t. Let f, f' be a flow in G. Let (u, v) be an edge in E where $u \in V - \{s, t\}$ and $v \in V$. We note that if $(u, v) \in E$, then $(v, u) \notin E$ and $f(v, u) = 0$. Thus, we can re-write the definition of flow augmentation (equation (26.4)) as

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) & [\text{If } (u, v) \in E] \\ 0 & [\text{Otherwise}] \end{cases} \tag{1}$$

which implies that the value of the augmentation of flow $f \uparrow f'$ on edge (u, v) is the sum of flow $f(u, v)$ and $f'(u, v)$ in G.

I need to show if the augmented flow of f and $f' \in G$ and satisfy the flow conservation property but not capacity constraint.

I will do so in parts

- **Part 1: Proving that $f \uparrow f'$ satisfies the flow conservation property**

  Here I prove that the augmented flow satisfies flow conservation. That is,

$$\sum_{v \in V} f \uparrow f'(u, v) = \sum_{v \in V} f \uparrow f'(v, u) \tag{2}$$

  .

  And indeed we have,

$$\sum_{v \in V} f \uparrow f'(u,v) = \sum_{v \in V} f(u,v) + f'(u,v) \quad \text{[By augmentation def.]} \tag{3}$$

$$= \sum_{v \in V} f(u,v) + \sum_{v \in V} f'(u,v) \tag{4}$$

$$= \sum_{v \in V} f(v,u) + \sum_{v \in V} f'(v,u) \quad \text{[By flow conserv. of } f \text{ and } f'\text{]} \tag{5}$$

$$= \sum_{v \in V} f(v,u) + f'(v,u) \tag{6}$$

$$= f \uparrow f'(v,u) \tag{7}$$

- **Part 2: Disproving that $f \uparrow f'$ satisfies the capacity constraint**

  Here, I need to disprove that the augmented flow satisfies capacity constraint. That is,

  $$(f \uparrow f')(u,v) > c(u,v) \tag{8}$$

  Let $f(u,v) = f'(u,v) = 10$ and $c(u,v) = 8$.

  Then, we can write $(f \uparrow f')(s,t) = 20$ and $c(u,v) = 8$.

  Thus, we can conclude the augmentation of flow doesn't satisfy capacity constraint.

□

Notes:

- I need clarification from professor about the meaning of $f' \in G$. Is f' a flow from flow network or residual network?
- I feel I am struggling because I am jumping to solution without understanding the problem
- I feel constructing a predicate logic would have helped to better understand this problem
- Noticed that a solution in University of Texas really elaborated on $f \uparrow f'(u,v)$ before moving onto strategizing and constructing a solution

capacity constraint property.

First, we prove that $f \uparrow f'$ satisfies the flow conservation property. We note that if edge $(u, v) \in E$, then $(v, u) \notin E$ and $f(v, u) = 0$. Thus, we can rewrite the definition of flow augmentation (equation (26.4)), when applied to two flows, as

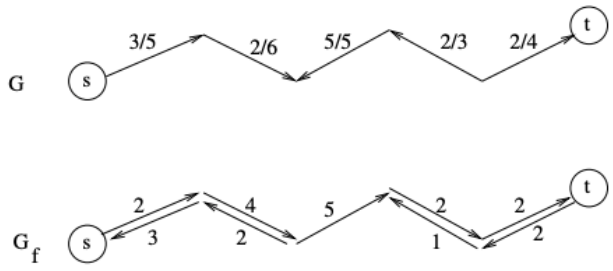
$$f \uparrow f'(u, v) = \begin{cases} f(u, v) + f'(u, v), & if (u, v) \in E \\ 0, & otherwise. \end{cases}$$

The definition implies that the new flow on each edge is simply the sum of the two flows on that edge. We now prove that in $f \uparrow f'$, the net incoming flow for each vertex equals the net outgoing flow. Let $u \notin \{s, t\}$ be any vertex of G. We have

- Noticed that a solution in University of Texas made quick sketches before laying the outline of proof
- **Flow Network (cont'd)** [Important!]
  - Flow network requires that
    1) $G = (V, E)$ is a directed graph
    2) each edge $(u, v) \in E$ has a non-negative capacity $c(u, v) \geq 0$
    3) If E contains an edge (u, v), then there is no edge (v, u) in the reverse direction (no anti-parallel edge)
- **Augmentation (cont'd)**
  - Flow value can be increased by

$$c_f(p) = \min_{(u,v) \in p} c_f(u, v) \tag{9}$$

Example:

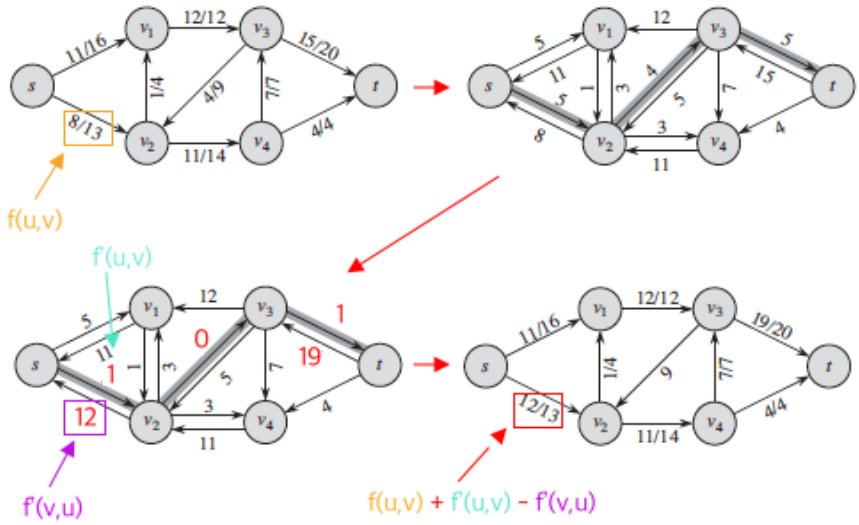


In this example, augmentation by 2.

  - Augmentation of flow f by f' or $f \uparrow f'$ is a function $V \times V \to \mathbb{R}$ is defined by

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u) & [\text{If } (u, v) \in E] \\ 0 & [\text{Otherwise}] \end{cases} \tag{10}$$

– Augmentation of flow $f \uparrow f'$ is the sum of flow on edge (u, v) in both flow network G and residual network G'



[NOTE!!] I really need to ask professor about this. I can't seem to understand using simple example why $f \uparrow f'(u, v) = f(u, v) + f'(u, v) - f'(v, u)$.

- **Proof of flow conservation for $f \uparrow f'$ when $f \in G$ and $f' \in G_f$**

Let $G = (V, E)$ be a flow network with sources s and sink t. Let f be a flow in G. Let G_f be a residual network of G induced by f and let f' be a flow in G_f. Let (u, v) be an edge in E where $u \in V - \{s, t\}$ and $v \in V$. We note that if $(u, v) \in E$, then $(v, u) \notin E$ and $f(v, u) = 0$. Thus, the definition

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u) & [\text{If } (u, v) \in E] \\ 0 & [\text{Otherwise}] \end{cases} \tag{1}$$

implies that the augmented flow $f \uparrow f'(u, v)$ on edge (u, v) is the sum of flow $f(u, v)$ in flow network G and flow $f'(u, v)$ minus its antiparallel flow $-f'(v, u)$ in residual flow network G'.

We now prove that the augmented flow satisfies flow conservation. That is,

$$\sum_{v \in V} f \uparrow f'(u, v) = \sum_{v \in V} f \uparrow f'(v, u) \tag{2}$$

.

And indeed we have

$$\sum_{v\in V} f \uparrow f'(u,v) = \sum_{v\in V} f(u,v) + f'(u,v) - f'(v,u) \quad \text{[By augmentation def.]} \tag{3}$$

$$= \sum_{v\in V} f(u,v) + \sum_{v\in V} f'(u,v) - \sum_{v\in V} f'(v,u) \tag{4}$$

$$= \sum_{v\in V} f(v,u) + \sum_{v\in V} f'(v,u) - \sum_{v\in V} f'(u,v) \quad \text{[By flow conserv. of } f \text{ and } f'\text{]} \tag{5}$$

$$= \sum_{v\in V} f(v,u) + f'(v,u) - f'(u,v) \tag{6}$$

$$= \sum_{v\in V} f \uparrow f'(v,u) \tag{7}$$

  - Flow in residual network also obey flow conservation

- **Proof of capacity constraint for $f \uparrow f'$ when $f \in G$ and $f' \in G_f$**

  Predicate Logic: $\forall f \in G,\ \forall f' \in G_f,\ \forall (u,v) \in E$ where $u, v \in V,\ 0 \leq (f \uparrow f')(u,v) \wedge (f \uparrow f')(u,v) \leq c(u,v)$

  Let $G = (V, E)$ be a flow network with sources s and sink t. Let f be a flow in G. Let G_f be a residual network of G induced by f and let f' be a flow in G_f. Let (u, v) be an edge in E where $u, v \in V$.

  I need to prove that $f \uparrow f'$ satisfies capacity constraint. That is, $0 \leq (f \uparrow f')(u,v) \wedge (f \uparrow f')(u,v) \leq c(u,v)$.

  I see there are two parts. I will prove each parts separately.

  1. **Part 1 ($0 \leq (f \uparrow f')(u,v)$)**

     Here, I need to show $0 \leq (f \uparrow f')(u,v)$. That is, $0 \leq f(u,v) + f'(u,v) - f'(v,u)$.

     And indeed we have,

$$(f \uparrow f')(u,v) = f(u,v) + f'(u,v) - f'(v,u) \tag{8}$$

$$\geq f(u,v) + f'(u,v) - c_f(v,u) \quad \text{[Since } f'(v,u) \leq c_f(v,u)\text{]} \tag{9}$$

$$= f(u,v) + f'(u,v) - f(u,v) \quad \text{[By def. of residual capacity]} \tag{10}$$

$$= f'(u,v) \tag{11}$$

$$\geq 0 \quad \text{[By cap. const. of } f' \text{ in } G_f \text{]} \tag{12}$$

$$\tag{13}$$

- $c_f(v,u) = f(u,v)$ is allowed

2. **Part 2 ($(f \uparrow f')(u,v) \leq c(u,v)$)**
Here, I need to show $(f \uparrow f')(u,v) \leq c(u,v)$. That is, $f(u,v)+f'(u,v)-f'(v,u) \leq c(u,v)$.

And indeed we have,

$$
\begin{aligned}
(f \uparrow f')(u,v) &= f(u,v) + f'(u,v) - f'(v,u) && (14)\\
&\leq f(u,v) + f'(u,v) && \text{[Since } f'(u,v) \geq 0 \text{ by cap. cons. of } f'] \quad (15)\\
&= f(u,v) + c_f(u,v) && \text{[Since } f'(u,v) \leq c_f(u,v)] \quad (16)\\
&= f(u,v) + (c(u,v) - f(u,v)) && \text{[By def of res. capacity]} \quad (17)\\
&= c(u,v) && (18)\\
& && (19)
\end{aligned}
$$

References

1) University of Teaxs, CSE 5311 Homework 5 Solution, link

6. **My Work:**
Let $G = (V, E)$ be an undirected graph.

I need to show how to determine the edge connectivity of G by running a maximum-flow algorithm.

First, I need to convert the undirected graph G to a directed graph.

I do so by assigning G' as a directed graph and transforming each edges in G to two directed edges (u,v) and (v,u).

Second, I need to setup graph G' as a flow network.

I do so by assigning each edge in G' with capacity of 1 and flow of 1, and assigning f' as max-flow in G' found using maximum-flow algorithm.

And together, I see that the directed graph G' has $\mathcal{O}(V)$ vertices and $\mathcal{O}(2E) = \mathcal{O}(E)$ edges.

Third, I need to find the edge connectivity of the directed graph G'. That is, the minimum number of edges that must be removed to disconnect G'. In other words, the minimum number of edges required to remove a vertex from graph G'.

And I do so by claiming that the edge connectivity of G' or G is $\min_{u \neq v} f'(u, v)$. That is, the minimum of max-flows over $u \in V$ in graph G'.

Since this algorithm runs over all $u \in V$, it runs at most $|V|$ times.

Suppose k is the edge connectivity of the graph and S is the set of k edges such that removal of S will disconnect the graph into 2 non-empty subgraphs G_1 and G_2. WLOG assume the node $u \in G$. Let w be a node in G_2. Since $u/neqw$, the value $f^*(u.w)$ will be computed by the algorithm. By the min-cut theorem $f^*(u, w)$ equals the min cut size between the pair (u, w), which is at most k since S duscibbects u and w. Therefore, we have

$$c* \leq f^*(u, w) \leq k \tag{1}$$

But c° cannot be smaller than k since that would imply a cut set of size smaller than k, contradicting the fact that k is the edge connectivity. Therefore $c^{\circ} = k$ and the algorithm returns the edge connectivity of the graph correctly.

Correct Solution:

Construct a directed graph G' from G by replacing each edge u, v in G by two directed edge (u, v) and (v, u) in G'. Let $f'(u, v)$ be the maximum flow value from u to v through G' with all edge capacities equal to 1. Pick an arbitrary node u and compute $f'(u, v)$ for all $v \neq u$.

We claim that edge connectivity equals $c^* = min_{v \neq u} f'(u, v)$. Therefore, the edge connectivity can be computed by running the maximum-flow algorithm $|V| - 1$ times on flow networks each having $|V|$ verticies and $2|E|$ edges.

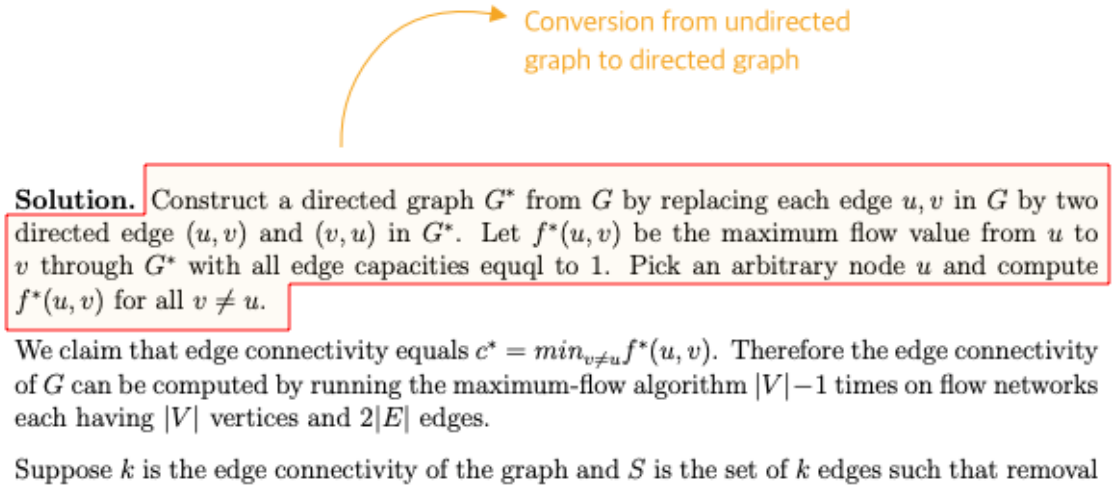
Suppose k is the edge connectivity of the graph and S is the set of k edges such that removal of S will disconnect the graph into 2 non-empty subgraphs G_1 and G_2. WLOG assume the node $u \in G$. Let w be a node in G_2. Since $u/neqw$, the value $f^*(u.w)$ will be computed by the algorithm. By the min-cut theorem $f^*(u, w)$ equals the min cut size between the pair (u, w), which is at most k since S duscibbects u and w. Therefore, we have

$$c* \leq f^*(u, w) \leq k \tag{2}$$

But c^* cannot be smaller than k since that would imply a cut set of size smaller than k, contradicting the fact that k is the edge connectivity. Therefore $c^* = k$ and the algorithm returns the edge connectivity of the graph correctly.

Notes

- I feel I should set myself a time, try, look at solution, and learn.
  - It feels like I am trying to ride a bicycle without parents' help when I can only ride a tricycle.
  - I need something that holds my bike and gradually let go as I become more comfortable.
  - It's taking too much time
- I feel that this maximum-flow algorithm maximizes flow value in a vertex with edges and antiparallel edges.
- I need clarification from professor about the role of maximum-flow algorithm.
- I noticed that the professor, instead of writing in parts like the first step, the second and the third step, he put everything together regarding conversion from undirected graph to flow network G' using let and construct, and pick.



Solution. Construct a directed graph G^* from G by replacing each edge u, v in G by two directed edge (u, v) and (v, u) in G^*. Let $f^*(u, v)$ be the maximum flow value from u to v through G^* with all edge capacities equql to 1. Pick an arbitrary node u and compute $f^*(u, v)$ for all $v \neq u$.

We claim that edge connectivity equals $c^* = min_{v \neq u} f^*(u, v)$. Therefore the edge connectivity of G can be computed by running the maximum-flow algorithm $|V|-1$ times on flow networks each having $|V|$ vertices and $2|E|$ edges.

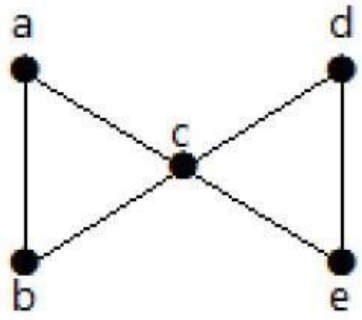
Suppose k is the edge connectivity of the graph and S is the set of k edges such that removal

And they all are relevant to the solution.

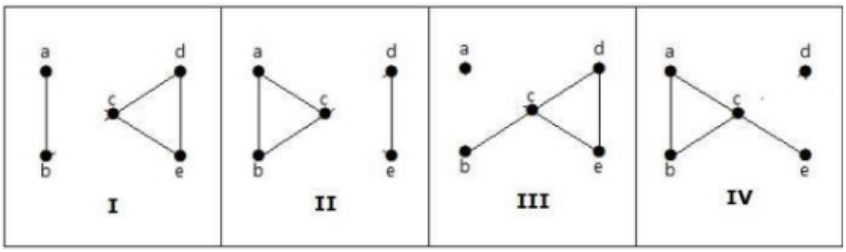
- I feel the solution was being super careful about not mentioning flow network
  - Flow network cannot have anti-parallel edges
- I noticed that the professor used the word 'claim' when he came up with his own algorithm that finds edge connectivity.

- **Maximum Flow:**
  - ~~Is a vertex in flow network with the maximum value of flow~~
  - Is equal to the capacity of minimum cut (Corollary 26.5)
- **Edge Connectivity:** Is the minimum number k of edges that must be removed to disconnect the graph. That is, the **number of edges in a smallest cut set of G** [2]

  Example:

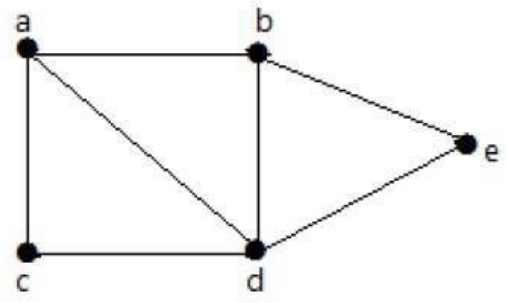


Here are the four ways to disconnect the graph by removing two edges −

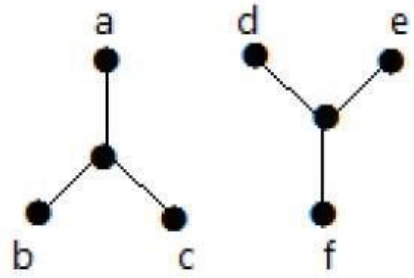


- **Connected:** A graph is said to be connected if there is a path between every pair of vertex

  Example:



Connected



Not Connected

References

1) National Taiwan University, Voluntary Exercise 3, link

2) TutorialsPoint, Graph Theory - Connectivity, link

7. Given the flow network G we will produce a flow network G' such that any minimum cut in G' is a minimum cut in G with the smallest number of edges (note that the minimum cut) with the smallest number of edges might not be unique). Then we execute any max-flow algorithm on G' determine any min-cut of G' and return the corresponding cut in G.

   Let $G = (V, E)$ be a flow network in integral capacities $c_e \geq 0$, $e \in E$. We define the flow network $G' = (V, E)$ with edge capacities $c'_e \geq 0$, $e \in E$ and we set $c'_e := |E| \cdot c_e + 1$.

   Claim: Any min-cut in G' is a min-cut of G with a smallest number of edges.

   Predicate Logic: MinCut$(G) \Rightarrow$ MinCut$(G') \wedge$ the number of edges crossing the cut is minimum

   Proof. Let $(S, V \setminus S)$ be a min cut in G' and let F be the set of edges crossing the cut. Then the capacity of the cut in G' is $\sum_{e \in F} c'_e = \sum_{c \in F} (|E| c_e + 1) = |F| + |E| \sum_{c \in F} c_e$ and the capacity of the corresponding cut in G is $C_G(S) := \sum_{e \in F} c_e$.

   We prove the claim by contradiction. At first assume that $(S, V \setminus S)$ is not a min-cut in G, i.e. there is a cut $(S', V \setminus S')$ in G with capacity smaller than $C_G(S)$. Let F' be the edges crossing this cut. Then the cut in G' has capacity

My Work

$$\begin{aligned}
\sum_{e\in F'} c'_e &= \sum_{e\in F'}(|E|\cdot c_e + 1) && (1)\\
&= |E|\sum_{e\in F'} c_e + \sum_{e\in F'} 1 && (2)\\
&= |E|\sum_{e\in F'} c_e + |F'| && (3)\\
&\le |E|(\sum_{e\in F} c_e - 1) + |F'| && [\text{Since } 0 \le \sum_{e\in F'} c_e < \sum_{e\in F} c_e] \quad (4)\\
&\le |E|(\sum_{e\in F} c_e - 1) + |E| && [\text{Since } |F| \le |E|] \quad (5)\\
&= |E|\sum_{e\in F} c_e - |E| + |E| && (6)\\
&= |E|\sum_{e\in F} c_e && (7)\\
&\le |E|\sum_{e\in F} c_e + |F| && (8)\\
&= \sum_{e\in F}(|E|\cdot c_e + 1) && (9)\\
&= \sum_{e\in F} c'_e && (10)
\end{aligned}$$

This is a contradction to $(S, V \setminus S)$ being a minimum cut of G'.

Now, we know that $(S, V\setminus S)$ is a minimum cut of G. Now, assume that there is a minimum cut $(S', V\setminus S')$ of G that uses fewer edges than the cut S. Again let F' be the edges crossing this cut. Then we can perform a similar calulation as above and we obtain that the cut S' in G' has capacity

My Work

$$\sum_{e \in F'} c'_e = \sum_{e \in F'} (|E| \cdot c_e + 1) \tag{11}$$
$$= \sum_{e \in F'} |E| \cdot c_e + |F'| \tag{12}$$
$$\leq \sum_{e \in F'} |E| \cdot c_e + |F| \qquad [\text{Since } |F'| < |F|] \tag{13}$$
$$= \sum_{e \in F} |E| \cdot c_e + |F| \qquad [\text{Since } \sum_{e \in F'} c_e = \sum_{e \in F} c_e] \tag{14}$$
$$= \sum_{e \in F} (|E| \cdot c_e + 1) \tag{15}$$
$$= \sum_{e \in F} c'_e \tag{16}$$
$$\tag{17}$$

This is a contradiction to $(S, V \setminus S)$ being a minimum cut of G', which proves the claim.

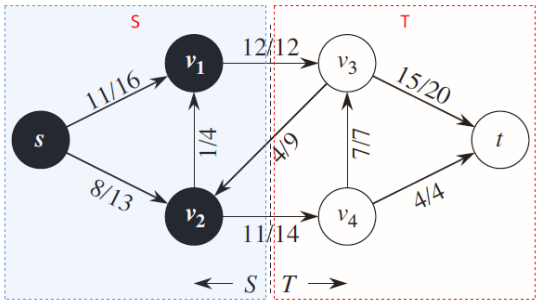
□

Notes

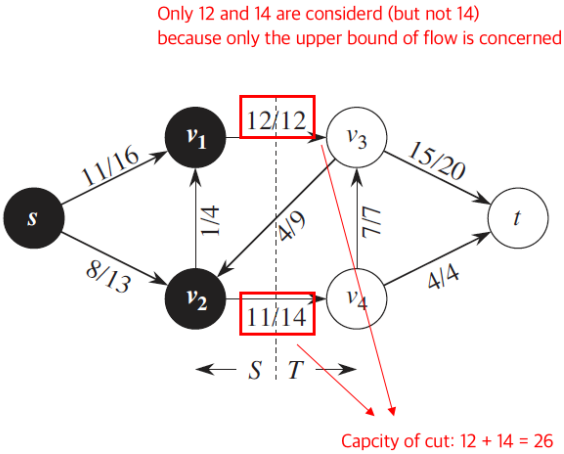
- I learned that $|E|$ in $c'_e = |E| \cdot c_e + 1$ is necessary.
- I learned that professor first proved the min-cut in G' is a min-cut in G, and then he used this fact to prove the number of edges crossing the cut in G must be minimum
- $A := B$
  - Means A and B are equal by definition (i.e. A is defined as B) [3]
  - Feels like the assignment operator = in programming languages

    Example

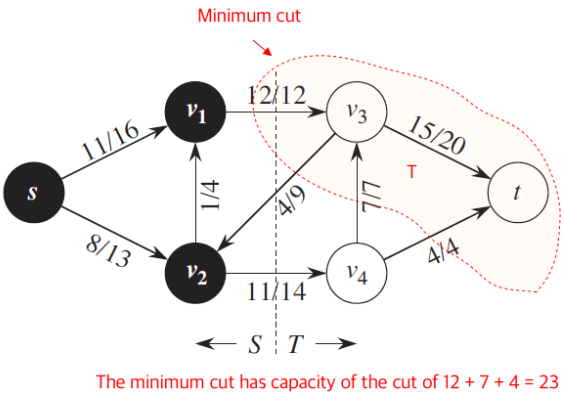
    $i := 2$ Means i is defined to have constant value 2.
- **Cut**
  - Is denoted using 'a cut (S, T)'.
  - Is a partition of V into S and $T = V - S$ such that $s \in S$ and $t \in T$
    - **Partition** means a grouping of its elements into non-empty subsets, in such a way that every element is included in exactly one subset [2]



- **Capacity of a Cut**
  - Is used as an **upper bound** on the flow from source to sink
  - Arrow in reverse direction is ignored
  - Is the maximum flow that can cross the cut from source to sink



- **Minimum Cut of a network**
  - Is a cut whose capacity is minimum over all cuts of the network

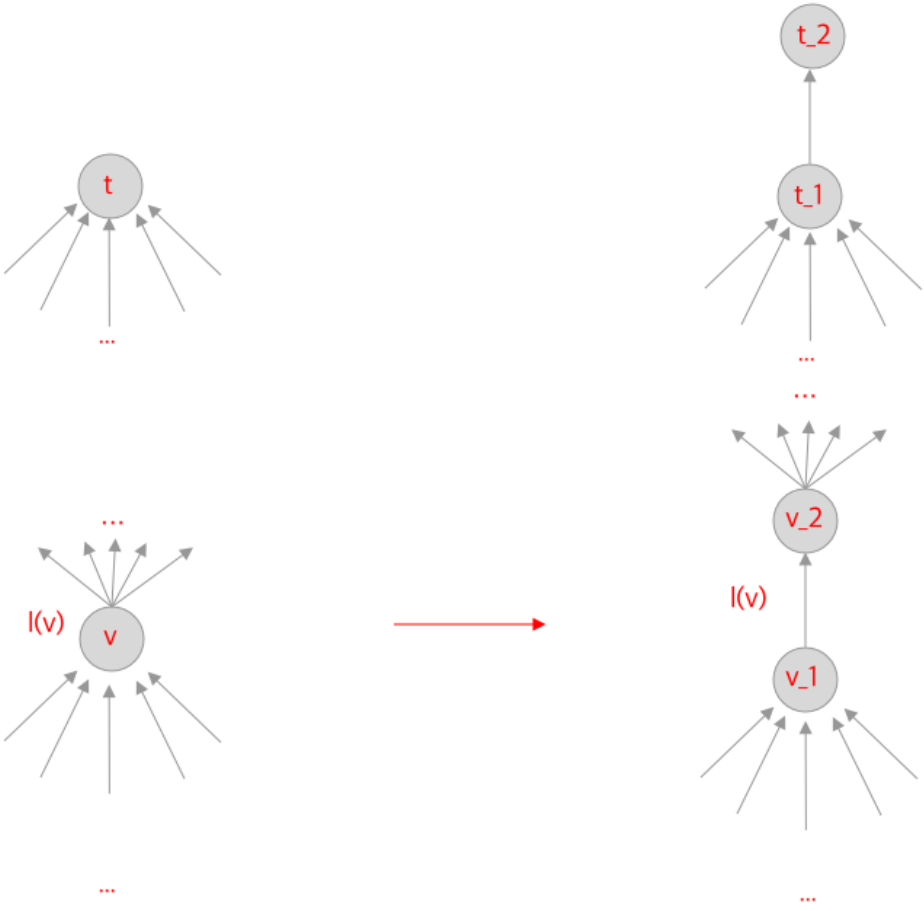


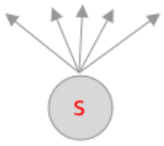
- **Integral Capacity**
  - Means capacity denoted by an integer value

References

1) University of Freiburg, Algorithm Theory, Winter Term 2016/17 Problem Set 5 Sample Solution, link
2) Wikipedia, Algorithm Theory, Partition of a set, link
3) Wolfram Math World, Defined, link

8. a) We will construct G' by splitting each vertex v of G into two verticies v_1, v_2, joined by an edge of capacity $l(v)$.All incoming edges of v are now incoming edges to v_1. All outgoing edges from v are now outgoing edges from v_2.





G

G'

More formally, construct $G' = (V', E')$ with capacity function c' as follows. For every $v \in V$, create two vertices $v_1, v_2 \in V'$. Add an edge $(v_1, v_2) \in E'$ with $c'(v_1, v_2) = l(v)$. For every edge $(u, v) \in E$, create an edge $(u_2, v_1) \in E'$ with capacity $c'(u_2, v_1) = c(u, v)$. Make s_1 and t_2 as the new source and target verticies in G'. Clearly,$|V'| = 2|V|$ and $|E'| = |E| + |V|$.

Let f be a flow in G that respects vertex capacities. Create a flow function f' in G' as follows. For each edge $(u, v) \in G$, let $f'(u_2, v_1) = f(u, v)$. For each vertex $u \in V - \{t\}$, let $f'(u_1, u_2) = \sum_{v \in V} f(u, v)$. Let $f'(t_1, t_2) = \sum_{v \in V} f(v, t)$.

We readily see that there is a one-to-one correspondence between flows that respect vertex capacities in G and flows in G'. For the capacity constraint, every edge in G' of the form (u_2, v_1) has a corresponding edge in G with a corresponding capacity and flow and thus satisfies the capacity constraint. For edges in E' of the form (u_1, u_2), the capacities reflect the vertex capacities in G. Therefore, for $u - V\{s, t\}$, we have $f'(u_1, u_2) = \sum_{v \in V} f(u, v) \leq l(u) = c'(u_1, u_2)$. We also have $f'(t_1, t_2) = \sum_{v \in V} f(v, t) \leq l(t) = c'(t_1, t_2)$. Note that this constraint also enforeces the vertex capacities in G.

Now, we prove flow conservation. By construction, every vertex of the form u_1 in G' has exactly one outgoing edge (u_1, u_2), and every incoming edge to u_1 corresponds to an incoming edge of $u \in G$. Thus, for all verticies $u \in V - \{s, t\}$, we have

My Work 1

$$\begin{aligned}
\text{incoming flow } u_1 &= \sum_{v_2 \in V'} f'(v_2, u_1) && && (1)\\
&= \sum_{v \in V} f(v, u) && \text{[By construction]} && (2)\\
&= \sum_{v \in V} f(u, v) && \text{[Since } f \text{ satisfies flow consv.]} && (3)\\
&= \sum_{v_1 \in V'} f'(u_2, v_1) && \text{[By construction]} && (4)\\
&= f'(u_1, u_2) && \text{[By construction]} && (5)\\
&= \text{outgoing flow } u_1 && && (6)
\end{aligned}$$

For t_1 we have

My Work 2

$$\begin{aligned}
\text{incoming flow } t_1 &= \sum_{v_2 \in V'} f'(v_2, t_1) && && (7)\\
&= \sum_{v \in V} f(v, t) && \text{[By construction]} && (8)\\
&= f'(t_1, t_2) && \text{[By construction]} && (9)\\
&= \text{outgoing flow } t_1 && && (10)
\end{aligned}$$

Verticies of the form u_2 has exactly one incoming edge (u_1, u_2), and every outgoing edge of u_2 corresponds to an outgoing edge of $u \in G$. Thus, for $u_2 \neq t_2$,

My Work 3

$$\begin{aligned}
\text{incoming flow } u_2 &= f'(u_1, u_2) && && (11)\\
&= \sum_{v \in V} f(v, u) && \text{[By construction]} && (12)\\
&= \sum_{v \in V} f(u, v) && \text{[Since } f \text{ satisfies flow consv.]} && (13)\\
&= \text{Outgoing flow } u_2 && && (14)
\end{aligned}$$

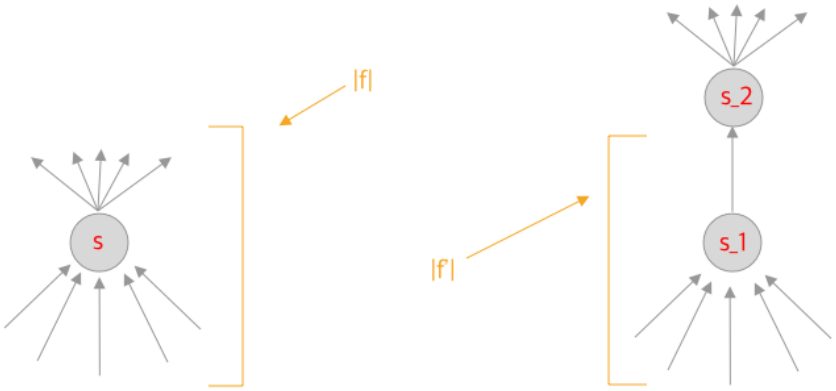
Finally, we prove that $|f'| = |f|$:

My Work 4

$$
\begin{aligned}
|f'| &= f'(s_1, s_2) && \text{[By construction]} && (15)\\
&= \sum_{v \in V'} f'(s_2, v) && \text{[By construction]} && (16)\\
&= \sum_{v \in V} f(s, v) && \text{[By construction]} && (17)\\
&= |f| && && (18)
\end{aligned}
$$

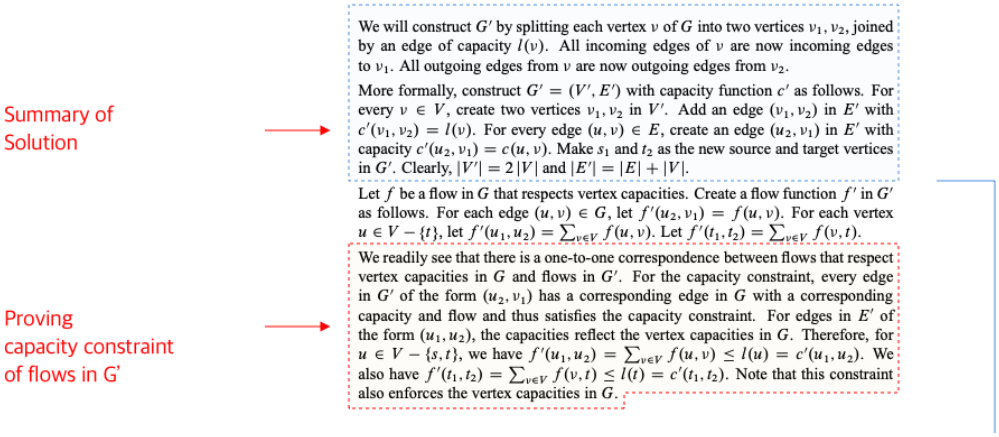
Notes

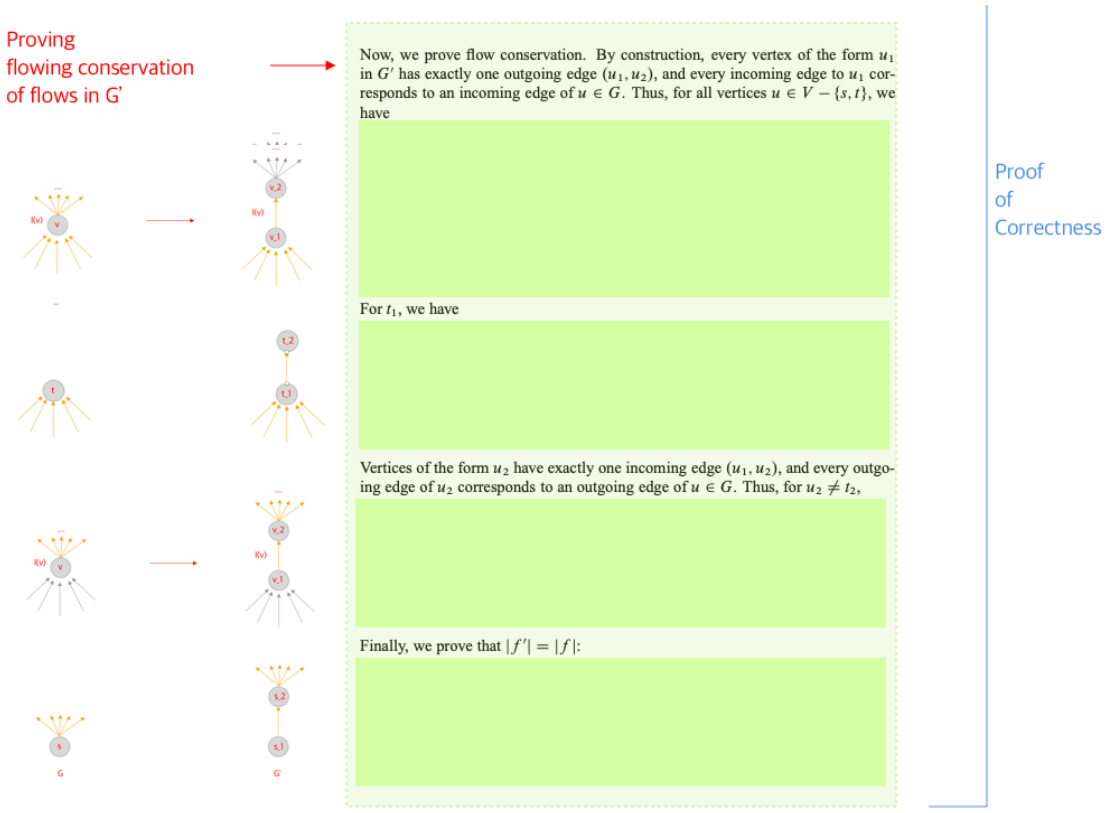
- **Flow (cont'd)**
  - Total flow out of the source and into the source is represented using $|\cdot|$



$$|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) \tag{19}$$

- Analysis of proof





References

1) CLRS Instructor's manual, Solution to Exercise 26.1-7 link

b) **Rough Work**

Goal: Find an efficient algorithm that solves the escape problem. That is, there are m vertex-disjoint paths from the starting points to any m different points on the boundary.

Inputs:

- Starting points $[(x_1, y_1), (x_2, y_2), (x_3, y_3)..., (x_m, y_m)]$ where $m \leq n^2$
- n (for $n \times n$ Grid)

Output:

- Boolean
  - True if there exists solution to grid escape problem
  - False otherwise

Brute Force Solution:

1) Let G be $n \times n$ grid

2) Mark starting points in grid as occupied
3) For each starting point
   - Find a shortest path from the starting point to unoccupied boundary
   - If there is a path, mark those paths occupied
   - Else, return False
4) Continue 3) until there are no starting points remaining

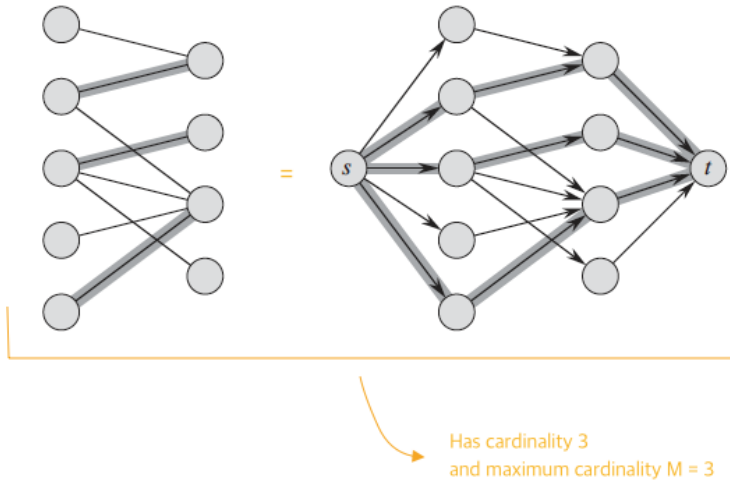
Improved Solution

Let S be the list of starting points $[(x_1, y_1), ..., (x_m, y_m)]$.

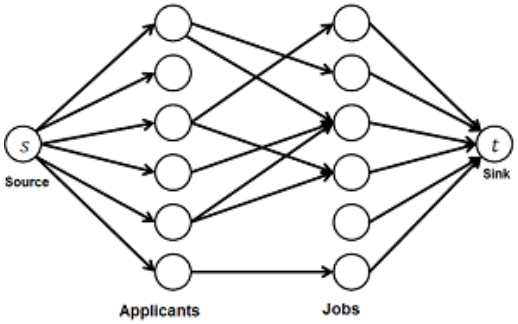
1) Construct $n \times n$ matrix
2) Store in M the value of a maximum flow using Ford-Fulkerson Method
3) If $M! = |S|$, return False
4) Else return True

Notes

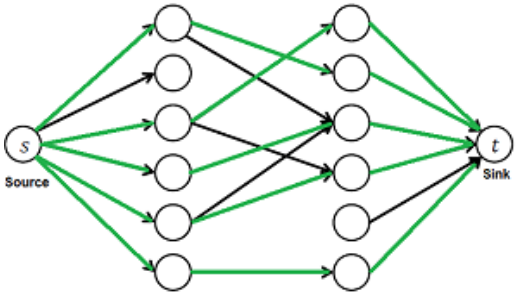
- **Maximum Bipartite Matching**
  - Casts some combinatorial problems as maximum-flow problems
  - Is a set of the edges chosen in such a way that no two edges share an endpoint



    * Cardinality means number of matching
  - Applications
    * Concurrent task management
  - Steps
    1. Create a flow network G' that correspond to matchings



2. Find the maximum flow using Ford-Fulkerson algorithm



   - Solves a graph in $\mathcal{O}(VE)$ time

- **Maximum Matching**
  - Is a matching of maximum cardinality
  - Is represented using M