

SOICT Hanoi University of Science and Technology
 School of Information and Communications Technology

Discrete Mathematics

Nguyễn Khánh Phương
 Department of Computer Science
 School of Information and Communication Technology
 E-mail: phuongnk@soict.hust.edu.vn



PART 1

COMBINATORIAL THEORY

(Lý thuyết tổ hợp)


PART 2

GRAPH THEORY


(Lý thuyết đồ thị)

Content of Part 2

- Chapter 1. Fundamental concepts
- Chapter 2. Graph representation
- Chapter 3. Graph Traversal
- Chapter 4. Tree and Spanning tree
- Chapter 5. Shortest path problem
- Chapter 6. Maximum flow problem**



NGUYỄN KHÁNH PHƯƠNG
CS-SOICT-HUST


SOICT Hanoi University of Science and Technology
 School of Information and Communications Technology


Chapter 6

MAXIMUM FLOW PROBLEM

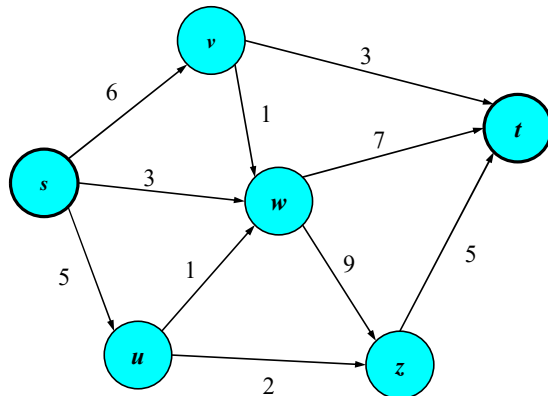
(Bài toán luồng cực đại trong mạng)

```

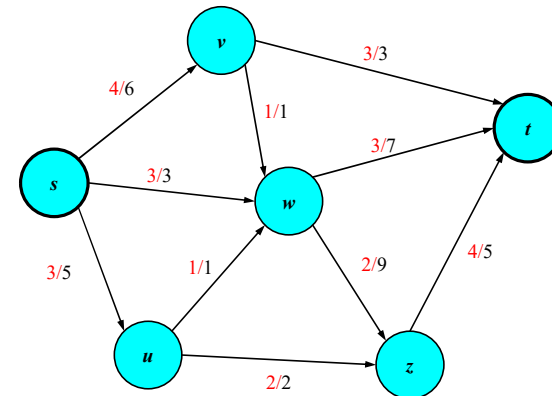
graph LR
    s((s)) -- "4/6" --> v((v))
    s -- "3/3" --> w((w))
    s -- "3/5" --> u((u))
    v -- "1/1" --> w
    u -- "1/1" --> w
    u -- "2/2" --> z((z))
    w -- "4/7" --> t((t))
    z -- "1/9" --> t
    z -- "3/5" --> t
  
```



Maximum Flow problem



Maximum Flow problem



Maximum flow with value $10 = 4 + 3 + 3 = 3 + 3 + 4$



Contents

1. Problem description and applications

2. Cut (Lát cắt)

3. Residual graph and Augmenting path

4. Ford-Fulkerson algorithm

5. Edmond-Karp algorithm

6. Some applications

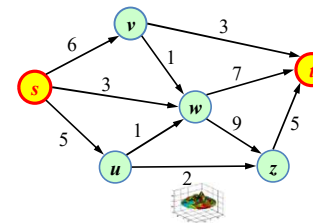


Network (Mạng)

Network is a directed graph $G = (V, E)$:

- There is only one vertex s without any incoming arcs, called **source vertex** (*đỉnh phát/nguồn*) and only vertex t without any outgoing arcs called **target vertex** (*đỉnh thu/đích*).
- Each edge e of G is assigned a nonnegative value $c(e)$ which is called capacity of e .

Example:



Flow in network (Luồng trong mạng)

Definition. Flow f in network $G=(V,E)$ is to assign value $f(e)$ on each edge e ($f(e)$ is flow on edge e) so that following conditions are satisfied:

1) Capacity rule:

For each edge e , $0 \leq f(e) \leq c(e)$

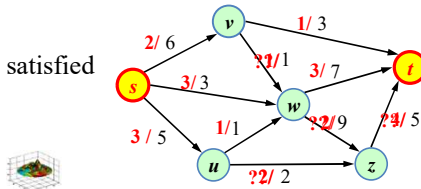
2) Conservation Rule (Điều kiện cân bằng luồng): Each $v \neq s, t$

$$\sum_{e \in E^-(v)} f(e) = \sum_{e \in E^+(v)} f(e)$$

$E^-(v)$ and $E^+(v)$ are sets of arcs entering and leaving vertex v , respectively.

Conditions 1) and 2) are satisfied
 $\Rightarrow f$ is flow in network

e.g. oil flowing through pipes,
 internet routing



Flow in network (Luồng trong mạng)

Definition. Flow f in network $G=(V,E)$ is to assign value $f(e)$ on each edge e ($f(e)$ is flow on edge e) so that following conditions are satisfied:

1) Capacity rule:

For each edge e , $0 \leq f(e) \leq c(e)$

2) Conservation Rule (Điều kiện cân bằng luồng): Each $v \neq s, t$

$$\sum_{e \in E^-(v)} f(e) = \sum_{e \in E^+(v)} f(e)$$

Definition. Value of flow f is

$$val(f) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e)$$

Edges going out of s

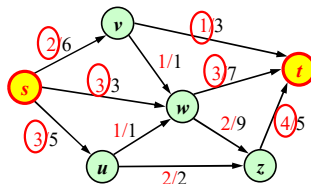
Edges going into t

(Equation (*) is obtained by summing up all the conservation rules.)



Flow in network: Example

Example:



- 2 values on each edge: value of flow on edge is in red, the other is capacity of edge.
- Conditions 1) and 2) are satisfied $\Rightarrow f$ is flow on the network.
- Value of flow:

$$8 = f(s,v) + f(s,u) + f(s,w) = f(v,t) + f(w,t) + f(z,t)$$



$$val(f) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e)$$

Flow in network (Luồng trong mạng)

Definition. Flow f in network $G=(V,E)$ is to assign value $f(e)$ on each edge e ($f(e)$ is flow on edge e) so that following conditions are satisfied:

1) Capacity rule:

For each edge e , $0 \leq f(e) \leq c(e)$

2) Conservation Rule (Điều kiện cân bằng luồng): Each $v \neq s, t$

$$\sum_{e \in E^-(v)} f(e) = \sum_{e \in E^+(v)} f(e)$$

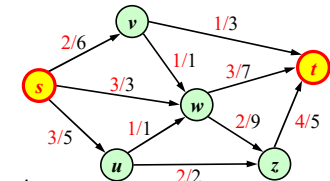
Definition. Value of flow f is

$$val(f) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e)$$

Edges going out of s

Edges going into t

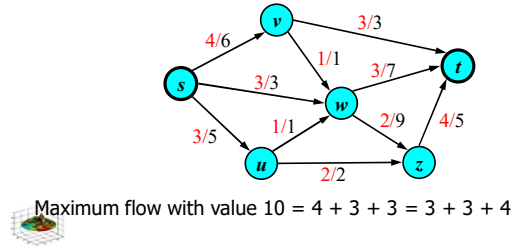
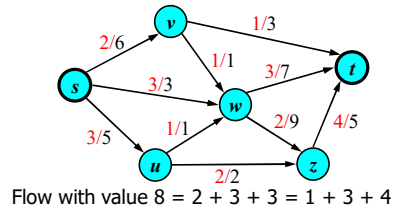
(Equation (*) is obtained by summing up all the conservation rules.)



Maximum Flow problem

A flow in network G is called maximum flow (luồng cực đại) if among all flows in G , it is the one having maximum value.

Maximum flow problem is the problem aiming to determine a feasible flow through the network with maximum value



Some applications

Network	Vertex	Arc	Flow
Network	transaction stations, computers, satellites	cables	voice, video, packets
Electrical network	gates, registers, processors	conductors	Power circuit
Mechanical	joints	rods, beams, springs	heat, energy
Irrigation	pumping stations, water source	pipelines	Liquid, water
Finance	bank	transitions	money
Traffic	airport, station	highways, flights	Goods, customers
Chemistry	sites	bonds	energy

Applications

- Some general flow problems:
 - The problem with multiple sources and sinks
 - The problem with limited capacities at nodes
- Airline scheduling
- Image segmentation
- Bipartite matching
- Data mining
- Project selection
- Network connectivity
- ...

Contents

- Problem description and applications
- Cut (Lát cắt)**
- Residual graph and Augmenting path
- Ford-Fulkerson algorithm
- Edmond-Karp algorithm
- Some applications

Cut

Cut is a partition vertex set of the network into 2 disjoint sets S and T with $s \in S, t \in T$.

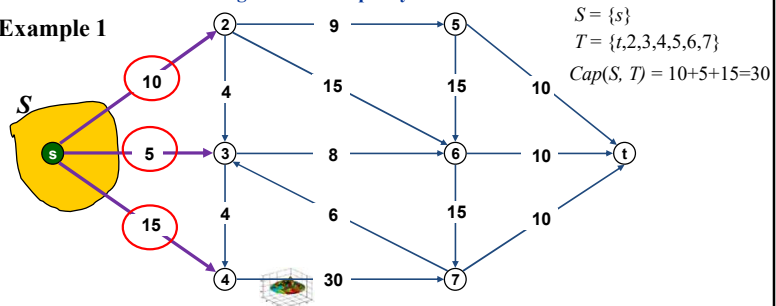
– Capacity (khả năng thông qua) $cap(S, T)$ of the cut (S, T) is:

$$cap(S, T) = \sum_{e \in S \rightarrow T} c(e),$$

where $S \rightarrow T := \{(v, w) \in E : v \in S, w \in T\}$

Min cut is the cut having minimum capacity.

Example 1



Cut

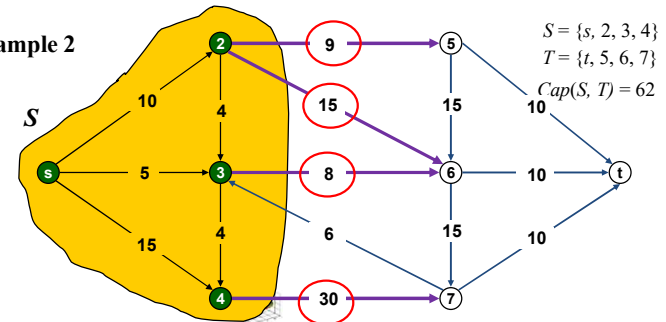
Cut is a partition vertex set of the network into 2 disjoint sets S and T with $s \in S, t \in T$.

– Capacity (khả năng thông qua) $cap(S, T)$ of the cut (S, T) is:

$$cap(S, T) = \sum_{e \in S \rightarrow T} c(e),$$

where $S \rightarrow T := \{(v, w) \in E : v \in S, w \in T\}$

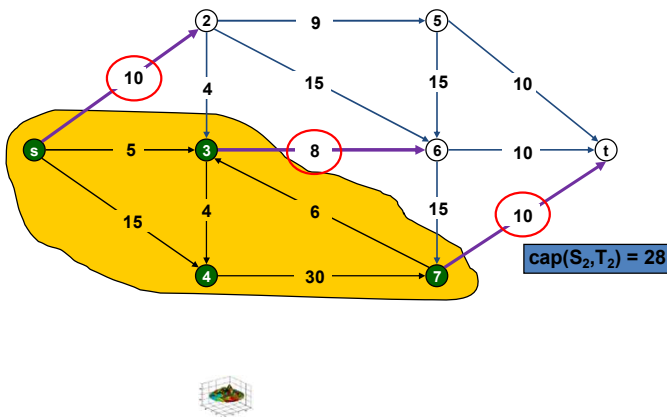
Example 2



Cut

Example 3:

Cut (S_2, T_2) , $S_2 = \{s, 3, 4, 7\}$, $T_2 = \{2, 5, 6, t\}$
 has the capacity = 28



Flow through the cut

Definition. Assume f is a flow in network and (S, T) is a cut. We call **value of flow through the cut (S, T)** is the value

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e)$$

where: $S \rightarrow T = \{(v, w) \in E : v \in S, w \in T\}$

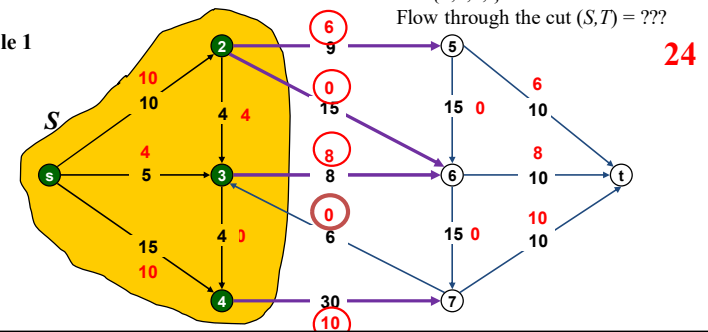
$T \rightarrow S = \{(v, w) \in E : v \in T, w \in S\}$

$S = \{s, 2, 3, 4\}$

$T = \{5, 6, 7, t\}$

Flow through the cut $(S, T) = ???$

Example 1



Flow through the cut (Luồng chảy qua lát cắt)

Definition. Assume f is a flow in network and (S, T) is a cut. We call **value of flow through the cut (S, T)** is the value

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) \quad 10+8+10 \quad 4$$

where: $S \rightarrow T = \{(v, w) \in E: v \in S, w \in T\}$

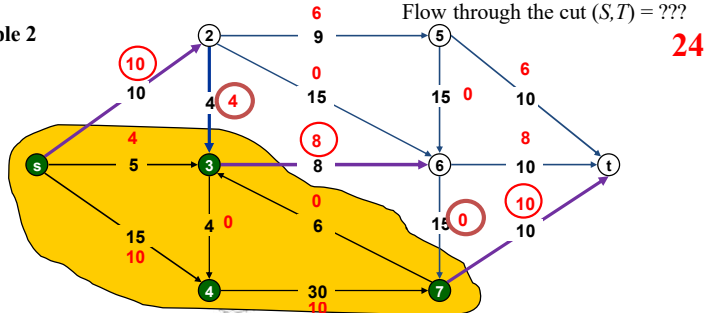
$T \rightarrow S = \{(v, w) \in E: v \in T, w \in S\}$

$S = \{s, 3, 4, 7\}$

$T = \{2, 5, 6, t\}$

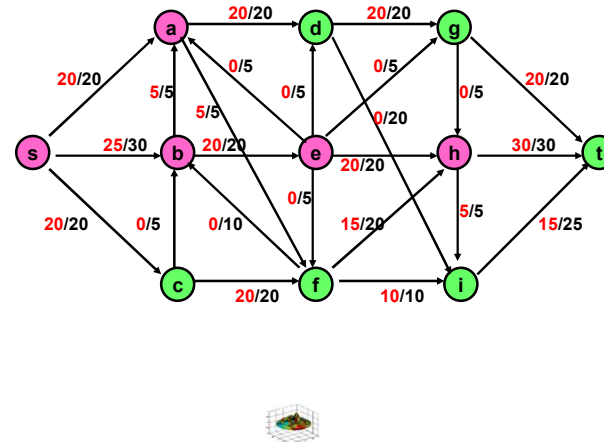
Flow through the cut $(S, T) = ???$

Example 2



What is $\text{Cap}(S, T)$ and $\text{Flow}(S, T)$

$S = \{s, a, b, e, h\}$, $T = \{c, f, i, d, g, t\}$



Flow and cut

Lemma 1. Assume f is flow, and (S, T) is a cut. Then

Flow through this cut is equal to the value of flow f .

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) = \sum_{e \in E^+(s)} f(e) - \sum_{e \in E^-(t)} f(e) = \text{val}(f)$$



Flow and cut

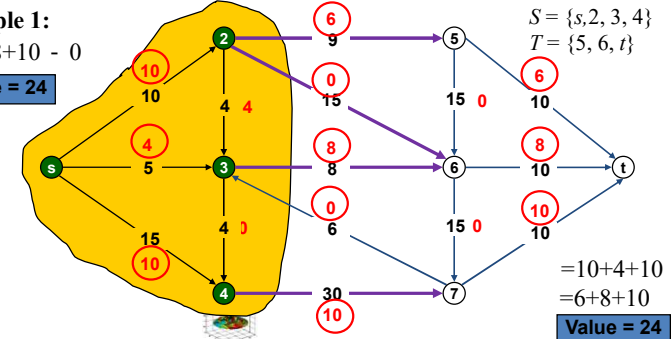
Lemma 1. Assume f is flow, and (S, T) is a cut. Then flow through this cut is equal to the value of flow f :

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) = \sum_{e \in E^+(s)} f(e) - \sum_{e \in E^-(t)} f(e) = \text{val}(f)$$

Example 1:

$6+0+8+10 - 0$

Value = 24



$= 10 + 4 + 10$

$= 6 + 8 + 10$

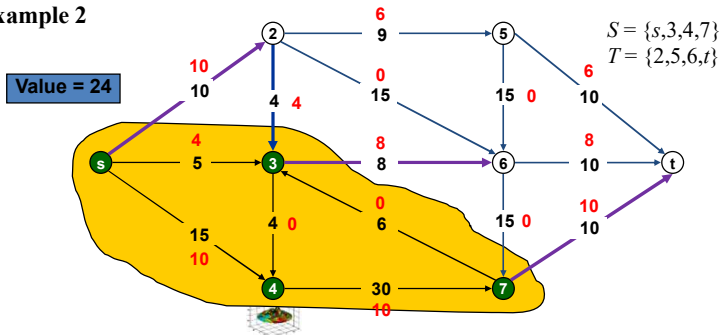
Value = 24

Flow and cut

Lemma 1. Assume f is flow, and (S, T) is a cut. Then flow through this cut is equal to the value of flow f :

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e) = \text{val}(f)$$

Example 2



Flow and cut

Lemma 1. Assume f is flow, and (S, T) is a cut. Then flow through this cut is equal to the value of flow f :

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) = \sum_{e \in E^+(s)} f(e) = \text{val}(f)$$

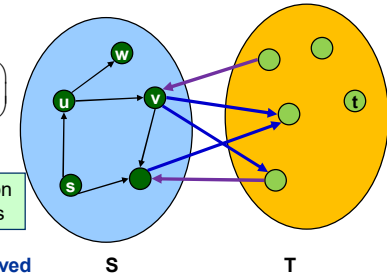
Proof: Sum of all flow conservation condition for all vertex $v \in S$:

$$0 = \sum_{v \in S} \left(\sum_{e \in E^+(v)} f(e) - \sum_{e \in E^-(v)} f(e) \right)$$

$$= \sum_{e \in E^+(s)} f(e) - \left(\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) \right)$$

Sum of flow on all blue arcs

Sum of flow on all purple arcs



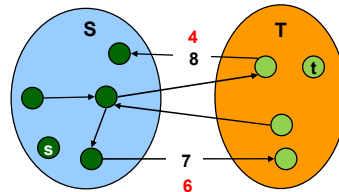
from that deduce the equation to be proved

Flow and cut

Lemma 2. Assume f is a flow, (S, T) is a cut. Then $\text{val}(f) \leq \text{cap}(S, T)$.

Proof

$$\begin{aligned} \text{val}(f) &= \sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) \\ &\leq \sum_{e \in S \rightarrow T} f(e) \\ &\leq \sum_{e \in S \rightarrow T} c(e) \\ &= \text{cap}(S, T) \end{aligned}$$



Max Flow and Min Cut (Luồng cực đại và lát cắt nhỏ nhất)

Corollary. Assume f is a flow, (S, T) is a cut. If $\text{val}(f) = \text{cap}(S, T)$ then f is maximum flow and (S, T) is minimum cut.

The flow in network G is called maximum flow if among all flows in G , its value is maximum

Minimum cut: the cut of minimum capacity

Proof: Consider f^* is arbitrary flow and (S', T') is arbitrary cut.

Using Lemma 2, we have:

$$\text{val}(f^*) \leq \text{cap}(S, T) = \text{val}(f) \leq \text{cap}(S', T')$$

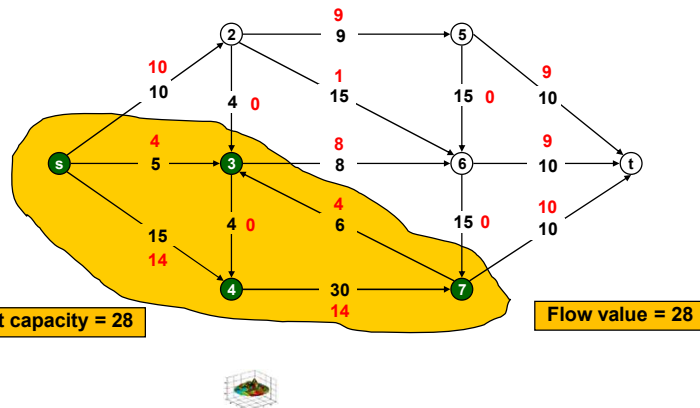
Lemma 2. Assume f^* is arbitrary flow and (S, T) is a cut. Then $\text{val}(f^*) \leq \text{val}(f)$ and $\text{val}(f) \leq \text{cap}(S, T)$

(S', T') is arbitrary cut $\Rightarrow (S, T)$ is min cut

$$\text{cap}(S, T) \leq \text{cap}(S', T')$$

Max-Flow Min-Cut Theorem Định lý về luồng cực đại và lát cắt nhỏ nhất

Theorem (Ford-Fulkerson, 1956): In a network, value of max flow is equal to capacity of min cut.



Value of flow f is $val(f) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e)$ $val(f) = 10+4+14 = 9+9+10=28$

Cut is a partition vertex set into 2 disjoint sets S and T with $s \in S, t \in T$.

– Capacity $cap(S, T)$ of cut (S, T) is value: **Min cut is the cut with min capacity.**

$$cap(S, T) = \sum_{e \in S \rightarrow T} c(e),$$

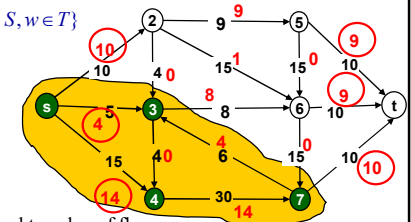
where $S \rightarrow T := \{(v, w) \in E : v \in S, w \in T\}$

– Flow through the cut (S, T) is the value

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e)$$

where: $S \rightarrow T = \{(v, w) \in E : v \in S, w \in T\}$

$T \rightarrow S = \{(v, w) \in E : v \in T, w \in S\}$



Lemma 1. Amount of flow through the cut is equal to value of flow:

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) = \sum_{e \in E^+(s)} f(e) = val(f)$$

Lemma 2. Value of flow is less than or equal to capacity of cut:

$$val(f) \leq cap(S, T).$$

If $val(f) = cap(S, T)$, then f is max flow and (S, T) is min cut.

Value of flow f is $val(f) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e)$ $val(f) = 10+4+14 = 9+9+10=28$
 $S = \{s, 3, 4, 7\}$
 $T = \{t, 2, 5, 6\}$

Cut is a partition vertex set into 2 disjoint sets S and T with $s \in S, t \in T$.

– Capacity $cap(S, T)$ of cut (S, T) is value: **Min cut is the cut with min capacity.**

$$cap(S, T) = 10+8+10=28 \quad cap(S, T) = \sum_{e \in S \rightarrow T} c(e),$$

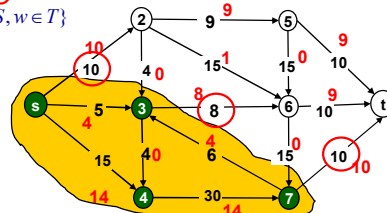
where $S \rightarrow T := \{(v, w) \in E : v \in S, w \in T\}$

– Flow through the cut (S, T) is the value

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e)$$

where: $S \rightarrow T = \{(v, w) \in E : v \in S, w \in T\}$

$T \rightarrow S = \{(v, w) \in E : v \in T, w \in S\}$



Lemma 1. Amount of flow through the cut is equal to value of flow:

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) = \sum_{e \in E^+(s)} f(e) = val(f)$$

Lemma 2. Value of flow is less than or equal to capacity of cut:

$$val(f) \leq cap(S, T).$$

If $val(f) = cap(S, T)$, then f is max flow and (S, T) is min cut.

Value of flow f is $val(f) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e)$ $val(f) = 10+4+14 = 9+9+10=28$
 $S = \{s, 3, 4, 7\}$
 $T = \{t, 2, 5, 6\}$

Cut is a partition vertex set into 2 disjoint sets S and T with $s \in S, t \in T$.

– Capacity $cap(S, T)$ of cut (S, T) is value: **Min cut is the cut with min capacity.**

$$cap(S, T) = 10+8+10=28 \quad cap(S, T) = \sum_{e \in S \rightarrow T} c(e),$$

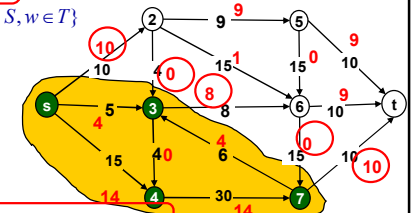
where $S \rightarrow T := \{(v, w) \in E : v \in S, w \in T\}$

– Flow through the cut (S, T) is the value

$$= 28 \quad \sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e)$$

where: $S \rightarrow T = \{(v, w) \in E : v \in S, w \in T\}$

$T \rightarrow S = \{(v, w) \in E : v \in T, w \in S\}$



Lemma 1. Amount of flow through the cut is equal to value of flow:

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) = \sum_{e \in E^+(s)} f(e) = val(f)$$

Lemma 2. Value of flow is less than or equal to capacity of cut:

$$val(f) \leq cap(S, T).$$

If $val(f) = cap(S, T)$, then f is max flow and (S, T) is min cut.

Value of flow f is $val(f) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e)$ $val(f) = 10+4+10 = 6+8+10=24$

Cut is a partition vertex set into 2 disjoint sets S and T with $s \in S, t \in T$.

- Capacity $cap(S, T)$ of cut (S, T) is value: **Min cut is the cut with min capacity.**
 $cap(S, T) = \sum_{e \in S \rightarrow T} c(e)$
 where $S \rightarrow T := \{(v, w) \in E : v \in S, w \in T\}$
- Flow through the cut (S, T) is the value
 $\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e)$
 where: $S \rightarrow T = \{(v, w) \in E : v \in S, w \in T\}$
 $T \rightarrow S = \{(v, w) \in E : v \in T, w \in S\}$

Lemma 1. Amount of flow through the cut is equal to value of flow:

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) = \sum_{e \in E^+(s)} f(e) = val(f)$$

Lemma 2. Value of flow is less than or equal to capacity of cut:
 $val(f) \leq cap(S, T)$.
 If $val(f) = cap(S, T)$, then f is max flow and (S, T) is min cut.

Value of flow f is $val(f) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e)$ $val(f) = 10+4+10 = 6+8+10=24$
 $S = \{s, 3, 4, 7\}$
 $T = \{t, 2, 5, 6\}$

Cut is a partition vertex set into 2 disjoint sets S and T with $s \in S, t \in T$.

- Capacity $cap(S, T)$ of cut (S, T) is value: **Min cut is the cut with min capacity.**
 $cap(S, T) = \sum_{e \in S \rightarrow T} c(e)$
 $cap(S, T) = 10+8+10=28$
 where $S \rightarrow T := \{(v, w) \in E : v \in S, w \in T\}$
- Flow through the cut (S, T) is the value
 $\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e)$
 $= 10+8+10 - 4 = 24$
 where: $S \rightarrow T = \{(v, w) \in E : v \in S, w \in T\}$
 $T \rightarrow S = \{(v, w) \in E : v \in T, w \in S\}$

Lemma 1. Amount of flow through the cut is equal to value of flow:

$$\sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) = \sum_{e \in E^+(s)} f(e) = val(f)$$

Lemma 2. Value of flow is less than or equal to capacity of cut:
 $val(f) \leq cap(S, T)$.
 If $val(f) = cap(S, T)$, then f is max flow and (S, T) is min cut.

Contents

1. Problem description and applications
2. Cut
- 3. Residual graph and Augmenting path**
4. Ford-Fulkerson algorithm
5. Edmond-Karp algorithm
6. Some applications

Greedy algorithm

Greedy algorithm:

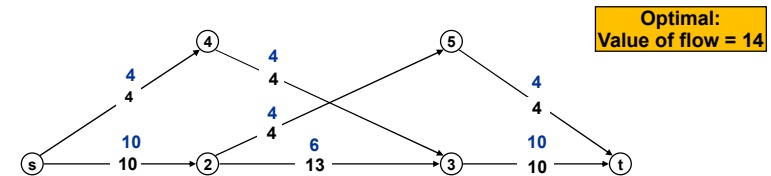
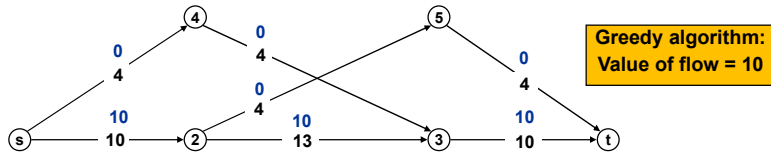
- Starting with flow 0 (value of flow = 0)
- Find path P from s to t such that each edge e satisfies $f(e) < c(e)$
- Augment flow along P .
- Repeat until can not find P

$s \rightarrow 4 \rightarrow 3 \rightarrow t$???
 $s \rightarrow 2 \rightarrow 5 \rightarrow t$???

Value of flow = 10

Greedy algorithm

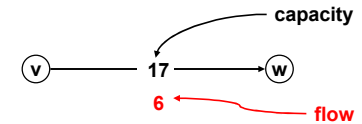
✚ Greedy algorithm does not provide optimal solution.



Residual graph (Đồ thị tăng luồng)

Network $G = (V, E)$.

- Arc $e = (v, w) \in E$
- Flow $f(e)$
- Capacity $c(e)$

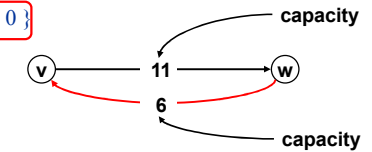


Residual graph: $G_f = (V, E_f)$.

- $E_f = \{e : f(e) < c(e)\} \cup \{e^R : f(e) > 0\}$
- Capacity of each arc e

$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$

$$e = (u, v) \Rightarrow e^R = (v, u)$$

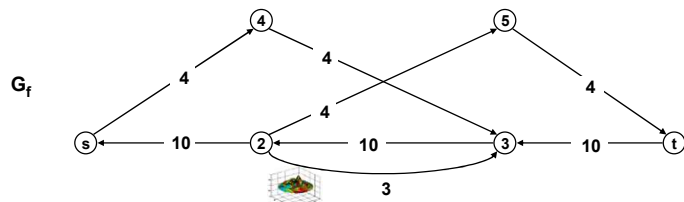
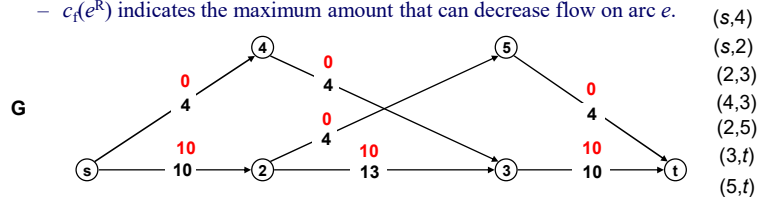


Residual graph - Example

Residual graph: $G_f = (V, E_f)$.

- $E_f = \{e : f(e) < c(e)\} \cup \{e^R : f(e) > 0\}$.
- $c_f(e)$ indicates the maximum amount that can increase flow on arc e .
- $c_f(e^R)$ indicates the maximum amount that can decrease flow on arc e .

$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$



Residual graph - Example

Residual graph: $G_f = (V, E_f)$.

- $E_f = \{e : f(e) < c(e)\} \cup \{e^R : f(e) > 0\}$.
- $c_f(e)$ indicates the maximum amount that can increase flow on arc e .
- $c_f(e^R)$ indicates the maximum amount that can decrease flow on arc e .

Augmenting path = path from s to t on the residual graph G_f .

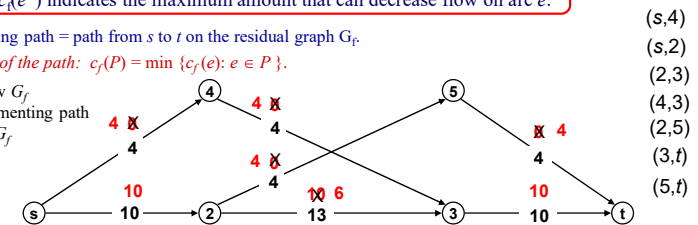
Capacity of the path: $c_f(P) = \min \{c_f(e) : e \in P\}$.

Build new G_f

Find augmenting path

On new G_f

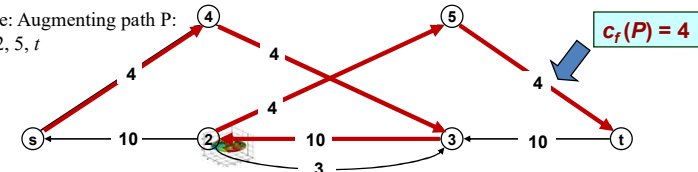
G_f



Example: Augmenting path P :

$s, 4, 3, 2, 5, t$

G_f



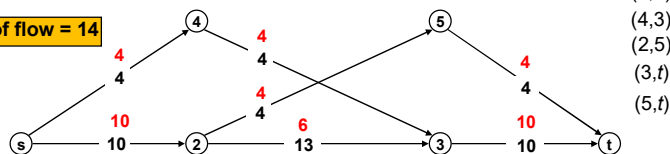
Augmenting path

Augmenting path = path from s to t on the residual graph.

- Max flow \Leftrightarrow could not find any augmenting path???

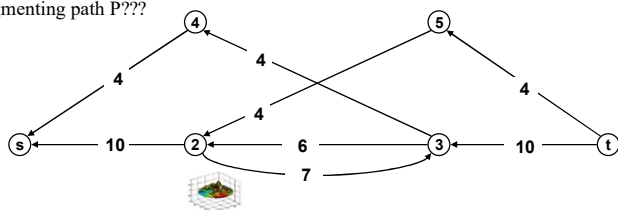
Value of flow = 14

G



Find augmenting path P???

G_f



(s,4)
(s,2)
(2,3)
(4,3)
(2,5)
(3,t)
(5,t)

Theorem about maximum flow and minimum cut

Augmenting path Theorem (Ford-Fulkerson, 1956): Flow is maximum if and only if there does not exist augmenting path on network.

Max flow and min cut Theorem (Ford-Fulkerson, 1956): the maximum possible flow in a network (from source to sink) is exactly equal to the minimum capacity of all possible cuts.

We will prove the following Theorem:

Theorem. Assume f is a flow in network. The following three statements are equivalent

- Can find the cut (S, T) such that $\text{val}(f) = \text{cap}(S, T)$.
- f is maximum flow.
- Could not find augmenting path to augment value flow f .



Prove theorem

- Prove.**
- Can find the cut (S, T) such that $\text{val}(f) = \text{cap}(S, T)$.
 - f is max flow.
- (i) \Rightarrow (ii)
- Proved by using Lemma 2.

Lemma 2. Assume f is a flow, and (S, T) is a cut. Then $\text{val}(f) \leq \text{cap}(S, T)$

- (ii) \Rightarrow (iii)
- Can not find any augmenting path to augment value of f .
- (ii) \Rightarrow (iii)
- Proof by Contrapositive: If can find an augmenting path, then f is not the maximum flow.
 - Indeed, if we could find a augmenting path P , then we augment value of flow f along P to obtain the flow f' with a greater value.



Prove theorem

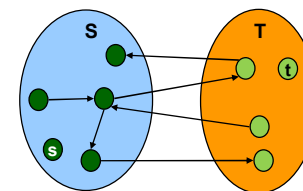
- (iii) \Rightarrow (i)
- Can not find any augmenting path to augment value of f .
 - Can find cut (S, T) such that $\text{val}(f) = \text{cap}(S, T)$.

- Assume (iii) $\Rightarrow f$ is flow and G_f does not contain any path from s to t .
- Let S the set of vertices reached from s in G_f
- From this definition about $s \in S$, and assumption (iii) $\Rightarrow t \notin S$
- Therefore, network flow f on each $e \in G$ must be

$$\begin{aligned} f(e) &= 0, e \in T \rightarrow S, \\ f(e) &= c(e), e \in S \rightarrow T \end{aligned}$$

- Then

$$\begin{aligned} \text{val}(f) &= \sum_{e \in S \rightarrow T} f(e) - \sum_{e \in T \rightarrow S} f(e) \\ &= \sum_{e \in S \rightarrow T} c(e) \\ &= \text{cap}(S, T) \end{aligned}$$



Given network G

Contents

1. Problem description and applications
2. Cut
3. Residual graph and Augmenting path
- 4. Ford-Fulkerson algorithm**
5. Edmond-Karp algorithm
6. Some applications



Ford – Fulkerson algorithm

Augmenting flow f along path P

```
float Augment(f, P)
{
    b ← cf(P)
    FOR e ∈ P
        IF (e ∈ E) // cạnh thuận
            f(e) ← f(e) + b
        ELSE // cạnh nghịch
            f(eR) ← f(e) - b
    RETURN f
}
```

Ford-Fulkerson algorithm

```
float Ford_Fulkerson(G, c, s, t)
{
    FOR e ∈ E // initialize flow f to 0
        f(e) ← 0
    Gf ← residual graph for f

    WHILE (there exists an augmenting path P)
    {
        f ← Augment(f, P)
        Update residual graph Gf
    }
    RETURN f
}
```

Computation time

Question: Is the Ford-Fulkerson algorithm a polynomial algorithm? (algorithm with computation time is bounded by a fixed degree polynomial of the input length)

- Answer: Not at all. If the maximum capacity is C then the algorithm may have to do C iterations.

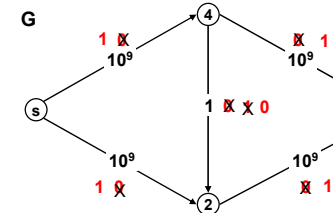
The following example shows how the algorithm can take a lot of iteration



Ford – Fulkerson algorithm is not polynomial algorithm

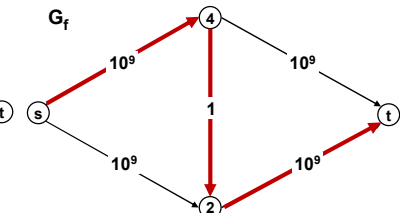
Augmenting path $P: s, 4, 2, t$

Augment flow along P



Augmenting path $P': s, 2, 4, t$

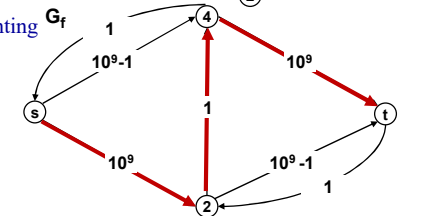
Augment flow along P'



Algorithm executes alternating augmenting flow along two paths P and P'

(s, 4)
(s, 2)
(4, 2)
(2, t)
(4, t)

2x10⁹ iterations



Computation time

Question: Is the Ford-Fulkerson algorithm a polynomial algorithm? (algorithm with computation time is bounded by a fixed degree polynomial of the input length)

- **Answer:** Not at all. If the maximum capacity is C then the algorithm may have to do C iterations.

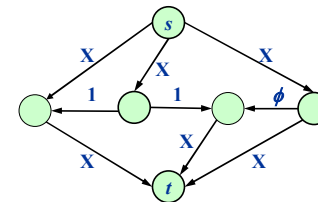
If the capacities of arcs on network is real, there is an example that the Ford-Fulkerson algorithm does not stop.

Zwick constructs examples showing that the algorithm may not stop, if the capacity of arcs is irrational



Example: Ford – Fulkerson algorithm does not terminate

Zwick constructs examples showing that the algorithm may not stop, if the capacity of arcs is irrational



There are 6 arcs with capacity = X , 2 arcs with capacity = 1 and one arc with capacity = $\phi = (\sqrt{5}-1)/2 \approx 0.618034...$



Assumption: All capacities on arcs are integers in the range $[0, C]$.

Immutability: Each value of flow value $f(e)$ and capacity $c_f(e)$ are always integer during algorithm execution.

Theorem: F-F algorithm stops after not more than $\text{val}(f^*) \leq nC$ iterations.

Proof.

After each augmenting flow, value of flow is augmented by at least 1

Corollary. Computation time of Ford-Fulkerson algorithm is $O(m.n.C)$

Corollary. If $C = 1$, then the computation time of algorithm is $O(mn)$.



Contents

1. Problem description and applications
2. Cut
3. Residual graph and Augmenting path
4. Ford-Fulkerson algorithm
5. **Edmond-Karp algorithm**
6. Some applications



How to select the augmenting path?

Be careful when select augmenting path, because

- Several options may lead to an exponential algorithm.
- Smart choice leads to polynomial algorithm.
- If capacity is irrational number, the algorithm may not stop

The goal: select augmenting path such that:

- Can find the augmenting path effectively.
- The algorithm requires as few iterations as possible..

Select augmenting path with

- Maximum capacity. (fat path - đường béo)
- Capacity is large enough. (capacity scaling)
- The number of edges along the way is the least. (shortest path)

Edmond-Karp algorithm

Applying BFS



5. Edmonds – Karp Algorithm

Edmonds and Karp, *JACM* 1972

- If the augmenting path is the shortest path from s to t , then the computation time of the algorithm is $O(|E|^2 |V|)$.



Ford-Fulkerson algorithm

```

float Ford_Fulkerson(G,c,s,t)
{
  FOR e ∈ E // initialize flow to 0
    f(e) ← 0
  Gf ← residual graph for flow f

  WHILE (there exists an augmenting path P)
  {
    f ← augment(f, P)
    update Gf
  }
  RETURN f
}
        
```

Edmonds – Karp algorithm

```

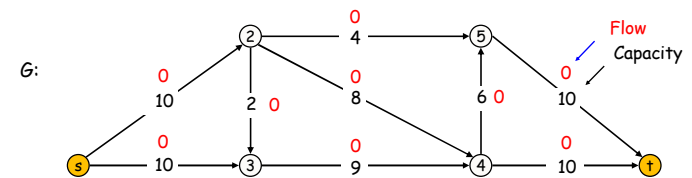
FOR e ∈ E
  f(e) ← 0
Gf ← residual graph for flow f
WHILE (there exists an augmenting path)
{
  find augmenting P by BFS
  f ← augment(f, P)
  update Gf
}
RETURN f
        
```

Find augmenting path by BFS:

- Easy to implement
- Augmenting path has the least edges

$O(|E|^2 |V|)$

Edmonds-Karp Algorithm



Value of flow = 0

Build the residual graph G_f

Edmonds – Karp algorithm

```

FOR e ∈ E
  f(e) ← 0
Gf ← residual graph for flow f
WHILE (there exists an augmenting path)
{
  find augmenting P by BFS
  f ← augment(f, P)
  update Gf
}
RETURN f
        
```



Edmonds-Karp Algorithm

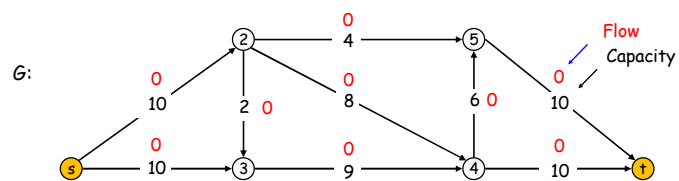
G :

Flow Capacity

Value of flow = 0

Build the residual graph G_f

57



Value of flow = 0

Build the residual graph G_f



57

Edmonds-Karp Algorithm

Diagram illustrating the Edmonds-Karp Algorithm on a flow network G .

The network G has nodes $s, 2, 3, 4, 5, t$ and edges with capacities and flows:

- $s \rightarrow 2$: Capacity 10, Flow 8 (marked with a red X).
- $s \rightarrow 3$: Capacity 10, Flow 0.
- $2 \rightarrow 3$: Capacity 2, Flow 0.
- $2 \rightarrow 4$: Capacity 8, Flow 8 (marked with a red X).
- $2 \rightarrow 5$: Capacity 4, Flow 0.
- $3 \rightarrow 4$: Capacity 9, Flow 0.
- $4 \rightarrow 5$: Capacity 6, Flow 0.
- $4 \rightarrow t$: Capacity 10, Flow 8 (marked with a red X).
- $5 \rightarrow t$: Capacity 10, Flow 0.

Red markings indicate flow adjustments:

- Red X on $s \rightarrow 2$ and $2 \rightarrow 4$ indicates flow reduction.
- Red X on $4 \rightarrow t$ indicates flow reduction.
- Red 0 on $s \rightarrow 3$ and $2 \rightarrow 5$ indicates flow increase.

Augmenting path $P: s, 2, 4, t$

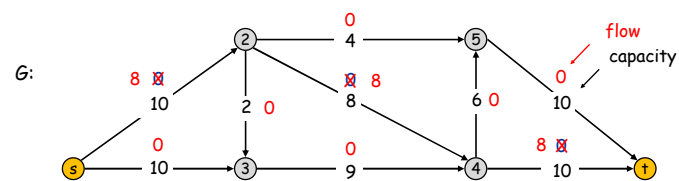
Value of flow = 8

Diagram illustrating the augmenting path $P: s, 2, 4, t$ and the flow augmentation:

The network G_f shows the flow after augmenting along path P by 8 units:

- $s \rightarrow 2$: Capacity 10, Flow 18 (red arrow).
- $s \rightarrow 3$: Capacity 10, Flow 0.
- $2 \rightarrow 3$: Capacity 2, Flow 0.
- $2 \rightarrow 4$: Capacity 8, Flow 0 (red arrow).
- $2 \rightarrow 5$: Capacity 4, Flow 0.
- $3 \rightarrow 4$: Capacity 9, Flow 0.
- $4 \rightarrow 5$: Capacity 6, Flow 0.
- $4 \rightarrow t$: Capacity 10, Flow 18 (red arrow).
- $5 \rightarrow t$: Capacity 10, Flow 0.

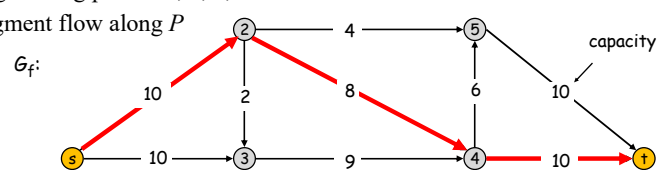
Red arrows indicate the augmenting path P and the flow augmentation.



Value of flow= 8

Augmenting path P : $s, 2, 4, t$

Augment flow along P



Edmonds-Karp Algorithm

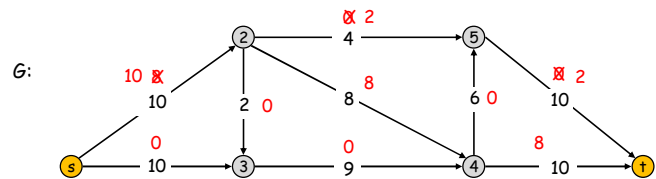
G :

Build residual graph G_f

Augmenting path P : $s, 2, 5, t$ Augment flow along P

Value of flow = 10

G_f :



Value of flow = 10

Build residual graph G_f

Augmenting path P : $s, 2, 5, t$ Augment flow along P

 $(s, 2)$

(S. 3)

 $(2, 3) G_f:$

(2, 5)

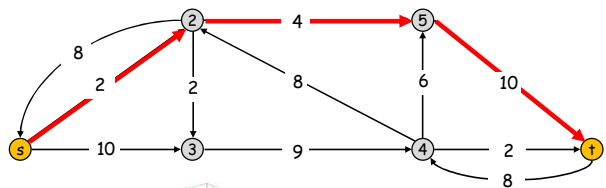
(2, 4)

(2, 4)

(3, 4)

 $(5, t)$

(4, 5)

 $(4, t)$ 

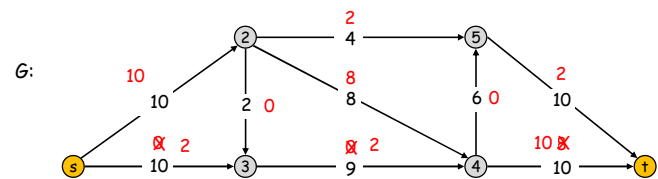
Edmonds-Karp Algorithm

Build residual graph G_f

Augmenting path $P: s, 3, 4, t$

Augment flow along P

Value of flow = 12

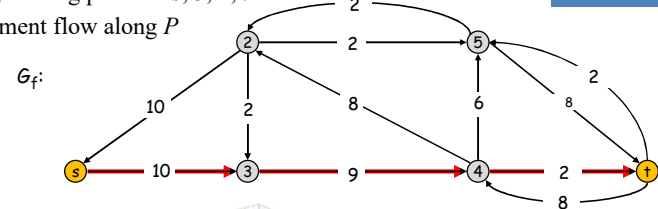


Value of flow = 12

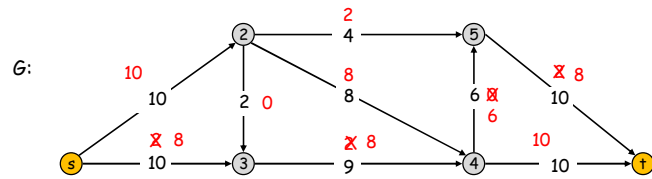
Build residual graph G_f

Augmenting path P : $s, 3, 4, t$

Augment flow along P



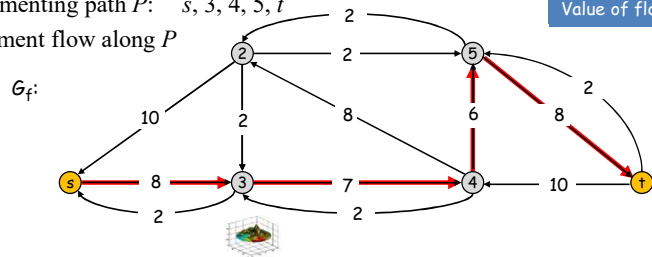
Edmonds-Karp Algorithm

Build residual graph G_f

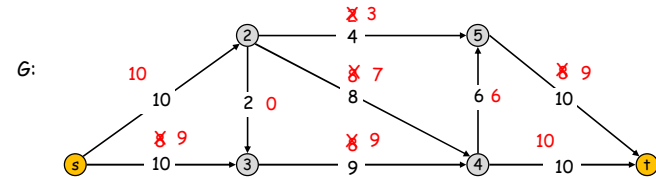
Augmenting path P : $s, 3, 4, 5, t$

Augment flow along P

Value of flow = 18



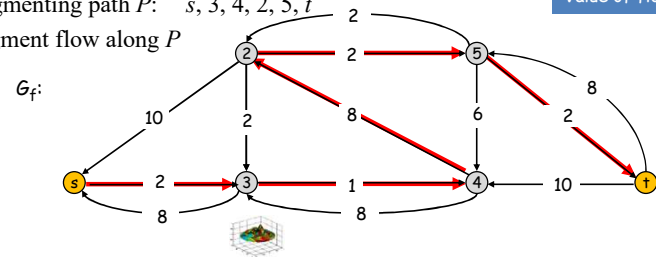
Edmonds-Karp Algorithm

| Build residual graph G_f

Augmenting path P : $s, 3, 4, 2, 5, t$

Augment flow along P

Value of flow = 19



Conclusion: - Min cut (S, T):	In graph G_f , S = set of vertices reached from s by $\text{BFS}(s)$ $S = \{s, 3\}$; $T = \{2, 4, 5, t\}$
---	---

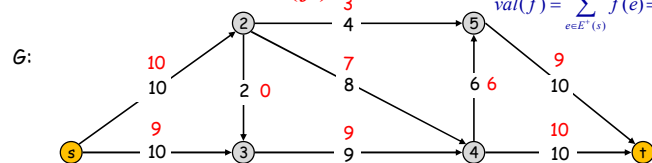
- Min cut (S, T):

S = {s, 3}; T = {2,4,5,t}

- Max flow:

$$\text{val}(f) = 10 + 9 = 19$$

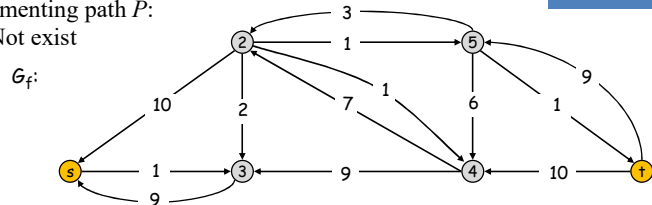
$$val(f) = \sum_{e \in E^+(s)} f(e) = \sum_{e \in E^-(t)} f(e)$$

Build residual graph G_f

Augmenting path P :

Not exist

Value of flow = 19



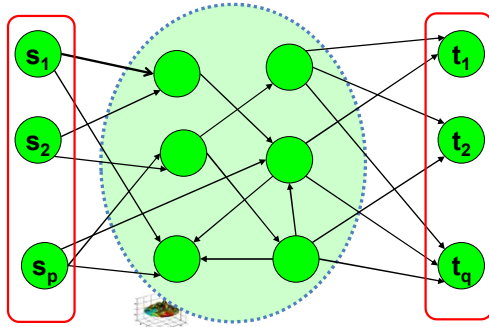
Applications

- Some general flow problems:
 - The problem with multiple sources and sinks
 - The problem with limited capacities at nodes
- Airline scheduling
- Image segmentation
- Bipartite matching
- Data mining
- Project selection
- Network connectivity
- ...



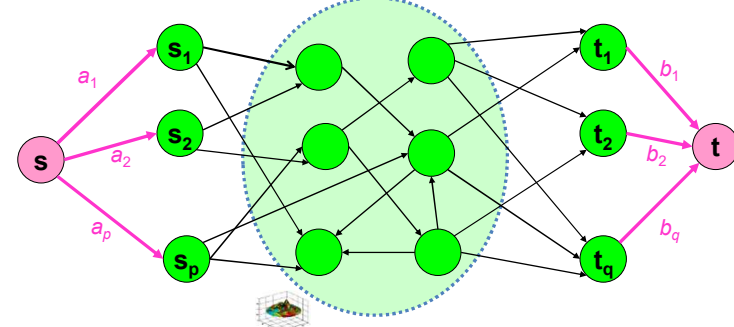
Network with multi-sources and multi-sinks

- Network G has p sources s_1, s_2, \dots, s_p with capacities of a_1, a_2, \dots, a_p and q sinks t_1, t_2, \dots, t_q with capacities of b_1, b_2, \dots, b_q
- Assume that flow can go from any source to all sinks
- Find the max flow from sources to sinks in this network



Network with multi-sources and multi-sinks

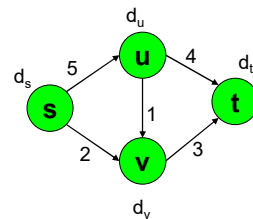
- Add a supersource s and a supersink t and edges connecting s to all sources, edges connecting all sinks to t
 - Capacity of arc (s, s_i) is equal to capacity of s_i
 - Capacity of arc (t_i, t) is equal to capacity of t_i
- The problem reduces to the single source single sink network



The problem with limited capacities at nodes

Network G , besides capacity on each arc $c(u, v)$, at each vertex $v \in V$: there is a capacity of vertex, denoted as $d(v)$, and require that sum of ingoing flow in each vertex v is not exceed the value $d(v)$:

$$\sum_{w \in V} f(w, v) \leq d(v)$$



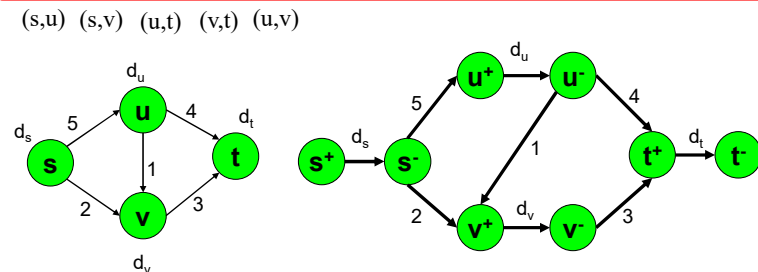
Find max flow from s to t in network G



The problem with limited capacities at nodes

Construct network G' such that:

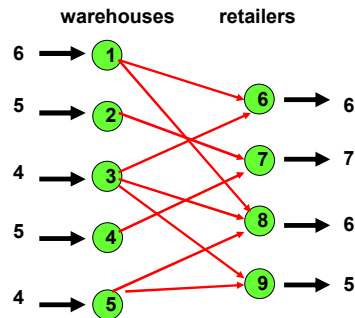
- each vertex v of G corresponds to 2 vertices v^+ and v^- of G' ,
- each arc (u, v) of G corresponds to arc (u^-, v^+) of G' ,
- each arc (v^+, v^-) of G' has capacity of $d(v)$, which is equal to capacity of vertex v in G .



The original problem reduces to the maximum flow problem in G'



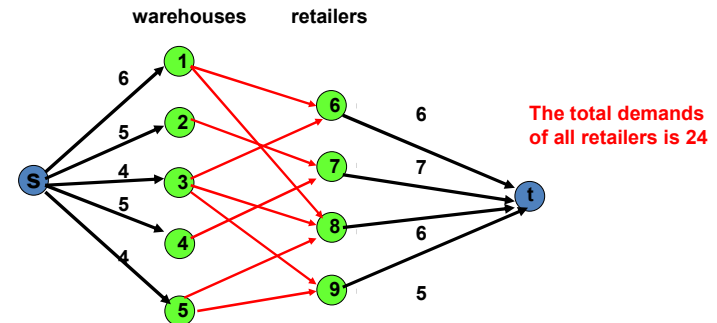
The delivery problem



Is there a way to delivery goods from the warehouses that meet the requirements of the retailers?



Reduces to maximum flow problem



There exists a 1-1 correspondence flow from s to t with a value of 24 with a delivery schedule that meets the requirements of the retailers.



Airline scheduling

- Desire to serve m specific flights:
 - Each flight i ($i=1, \dots, m$) specified by 4 parameters: origin airport O_i , destination airport D_i , departure time, arrival time.
 - V is the set of airport: $t(v, w)$ is time to fly from airport v to airport w
 - We can use a single plane for flight i and later for flight j if:
 - $D_i = O_j$ and there is enough time to perform maintenance on the plane between the two flights, or
 - We can add a flight that take plane from D_i to O_j if there is enough time for maintenance
- Flight j is called reachable from flight i if these two flights can use the same plane and one of two constraints above is satisfied.



Input: graph with directed arc (i, j) if flight j is reachable from flight i

→ Directed Acyclic graph

Goal: schedule all m flights using at most k planes



Image segmentation

- In the image segmentation problem, the input is an image, and we would like to partition it into background and foreground.

Example:



The (i) input image, and (ii) a possible segmentation of the image.



Image segmentation

- The input is a bitmap on a grid where every grid node represents a pixel. We convert this grid into a directed graph G , by interpreting every edge of the grid as two directed edges. See the figure below to see how the resulting graph looks like.

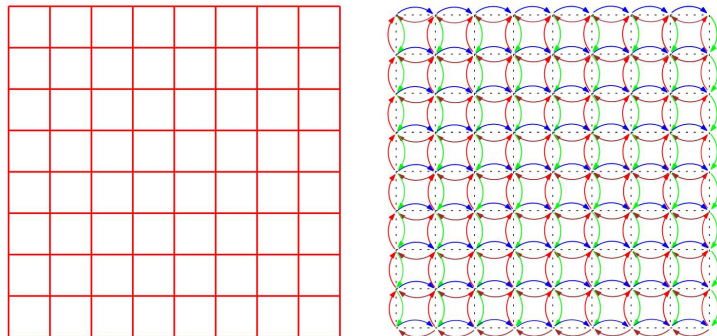


Image segmentation

The input for the problem is as follows:

- A bitmap of size $N \times N$, with an associated directed graph $G = (V, E)$.
- For every pixel i , we have a value $f_i \geq 0$, which is an estimate of the likelihood of this pixel to be in foreground (i.e., the larger f_i is the more probable that it is in the foreground)
- For every pixel i , we have (similarly) an estimate b_i of the likelihood of pixel i to be in background.
- We want the foreground/background boundary to be smooth:
 - For every two adjacent pixels i and j we have a separation penalty p_{ij} , which is the “price” of separating i from j (for placing one of them in the foreground and the other in the background). This quantity is defined only for adjacent pixels in the bitmap.



Image segmentation

Input: Pixel graphs $G(V, E)$, likelihood functions $f, b : V \rightarrow R^+$, penalty function $p : E \rightarrow R^+$

Output: Optimum labelling: partition V (the set of the pixels) into two sets F and B that maximize

$$q(F, B) = \sum_{i \in F} f_i + \sum_{i \in B} b_i - \sum_{(i,j) \in E, |F \cap \{i,j\}|=1} p_{ij}.$$



Image segmentation

$$\text{Let } Q = \sum_{i \in F \cup B} (f_i + b_i) \rightarrow \sum_{i \in F} f_i - \sum_{i \in B} b_i = Q - \sum_{i \in F} b_i - \sum_{i \in B} f_i$$

Therefore, maximizing

$$\begin{aligned} q(F, B) &= \sum_{i \in F} f_i + \sum_{i \in B} b_i - \sum_{(i,j) \in E, |F \cap \{i,j\}|=1} p_{ij} \\ &= Q - \sum_{i \in F} b_i - \sum_{i \in B} f_i - \sum_{(i,j) \in E, |F \cap \{i,j\}|=1} p_{ij} \end{aligned}$$

is identical to minimizing

$$q'(F, B) = \sum_{i \in F} b_i + \sum_{i \in B} f_i + \sum_{(i,j) \in E, |F \cap \{i,j\}|=1} p_{ij}$$

