NCKU Programming Contest Training Course
2013/05/22

Pin-Chieh Huang (free999)

Pinchieh.huang@gmail.com

http://myweb.ncku.edu.tw/~P76014143/20130522_Flow.rar

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

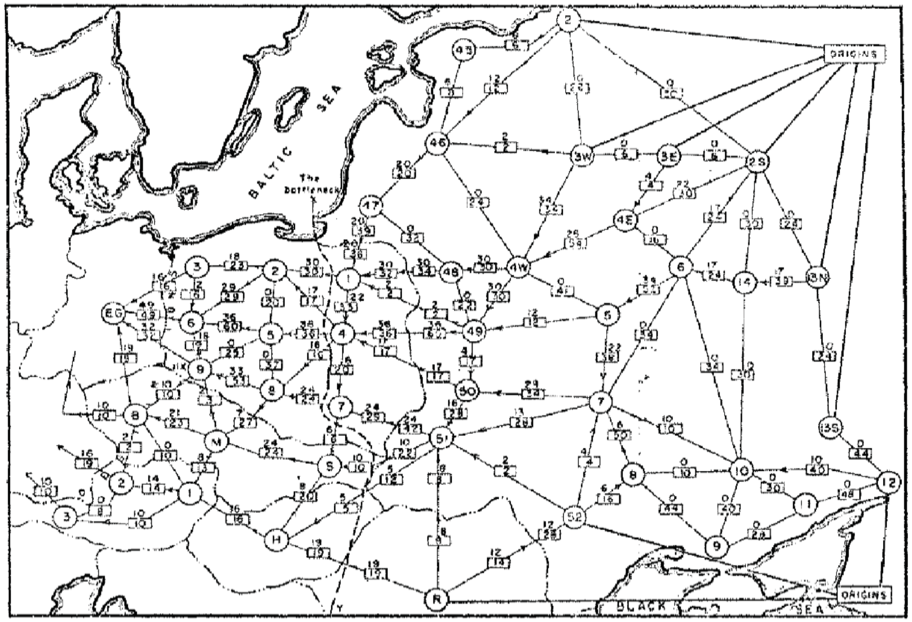
Outline

Maximum Network Flow

Maximum Flow

- Network flow problem

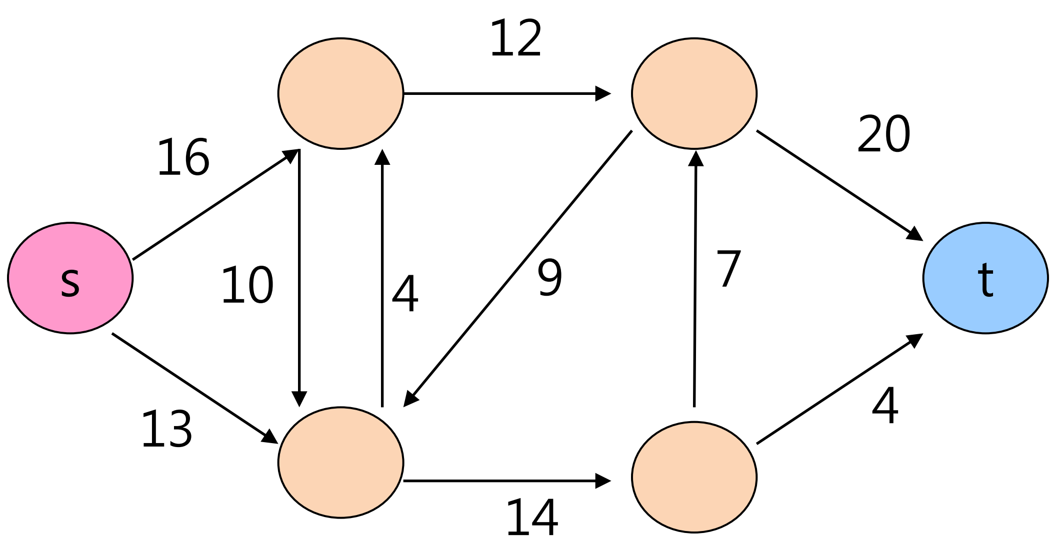
Soviet Rail Network, 1955



Reference: *On the history of the transportation and maximum flow problems.*
Alexander Schrijver in Math Programming, 91: 3, 2002.

Maximum Flow

- Network flow problem
  - A flow network G=(V,E): a directed graph, where each edge (u,v)∈E has a nonnegative capacity c(u,v)>=0.
  - If (u,v)∉E, we assume that c(u,v)=0.
  - two distinct vertices :a **source s** and a **sink t.**

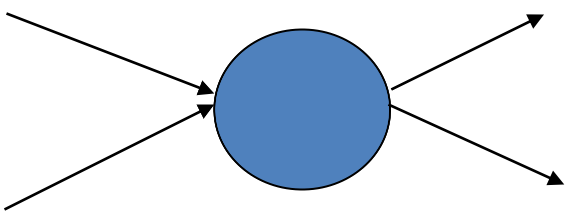


Maximum Flow

- G=(V,E): a flow network with capacity function c.
- *s* -- the source and *t* -- the sink.
- A flow in G: a real-valued function f:V*V → R satisfying the following two properties:
- Capacity constraint: For all u,v ∈V,

  we require f(u,v) ≤ c(u,v).
- Flow conservation: For all u ∈V-{s,t}, we require

$$\sum_{e.in.v} f(e) = \sum_{e.out.v} f(e)$$



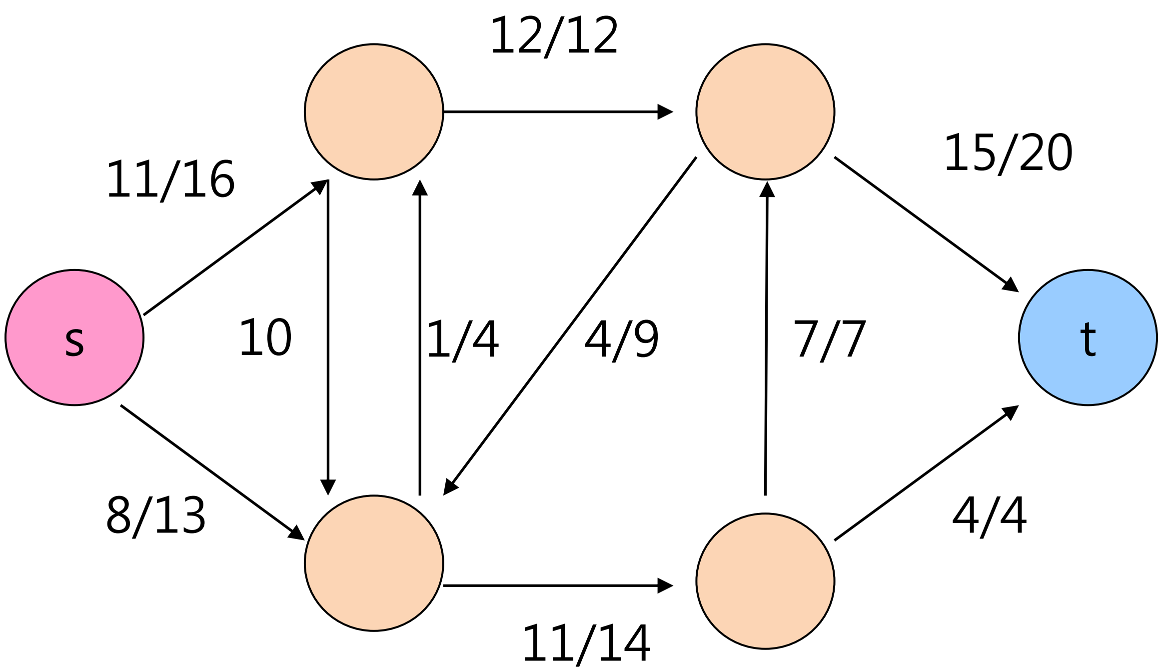
Maximum Flow

- The quantity f (u,v) is called the net flow from vertex u to vertex v.
- The value of a flow is defined as

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

  - The total flow from source to any other vertices.
  - The same as the total flow from any vertices to the sink.

Maximum Flow



A flow f in G with value $|f| = 19$

Maximum Flow

- Given a flow network G with source s and sink t
- Find a flow of maximum value from s to t.
- How to solve it efficiently ?

made by electron & free999

Maximum Flow

- This section presents the Ford-Fulkerson method for solving the maximum-flow problem. We call it a "method" rather than an "algorithm" because it encompasses several implementations with different running time. The Ford-Fulkerson method depends on three important ideas that transcend the method and are relevant to many flow algorithms and problems: residual networks, augmenting paths, and cuts. These ideas are essential to the important max-flow min-cut theorem, which characterizes the value of maximum flow in terms of cuts of the flow network.

Maximum Flow

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

  initialize flow *f* to *0*

  while there exists an *augmenting* path *p*

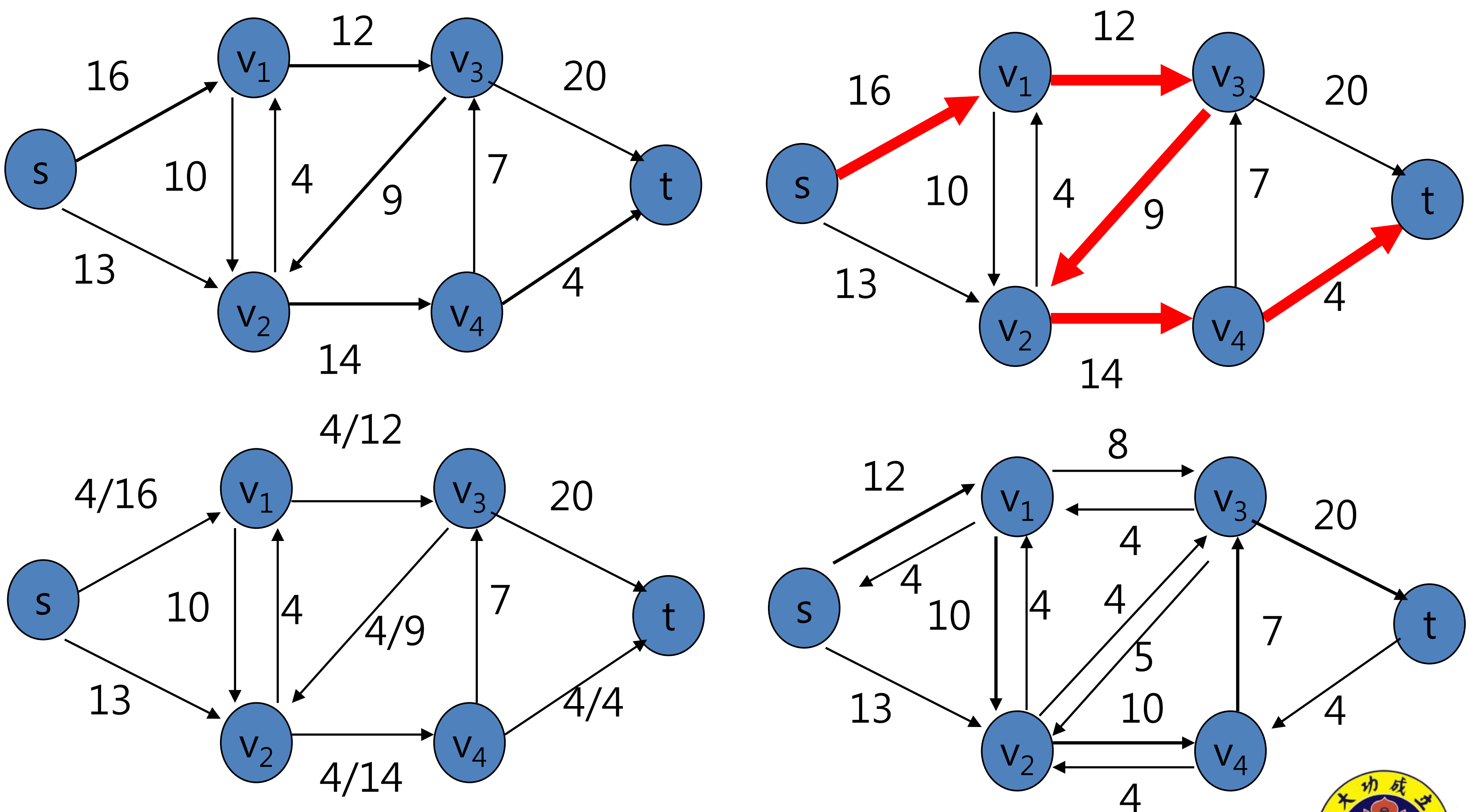
  do *augment* flow *f* along *p*

  return *f*

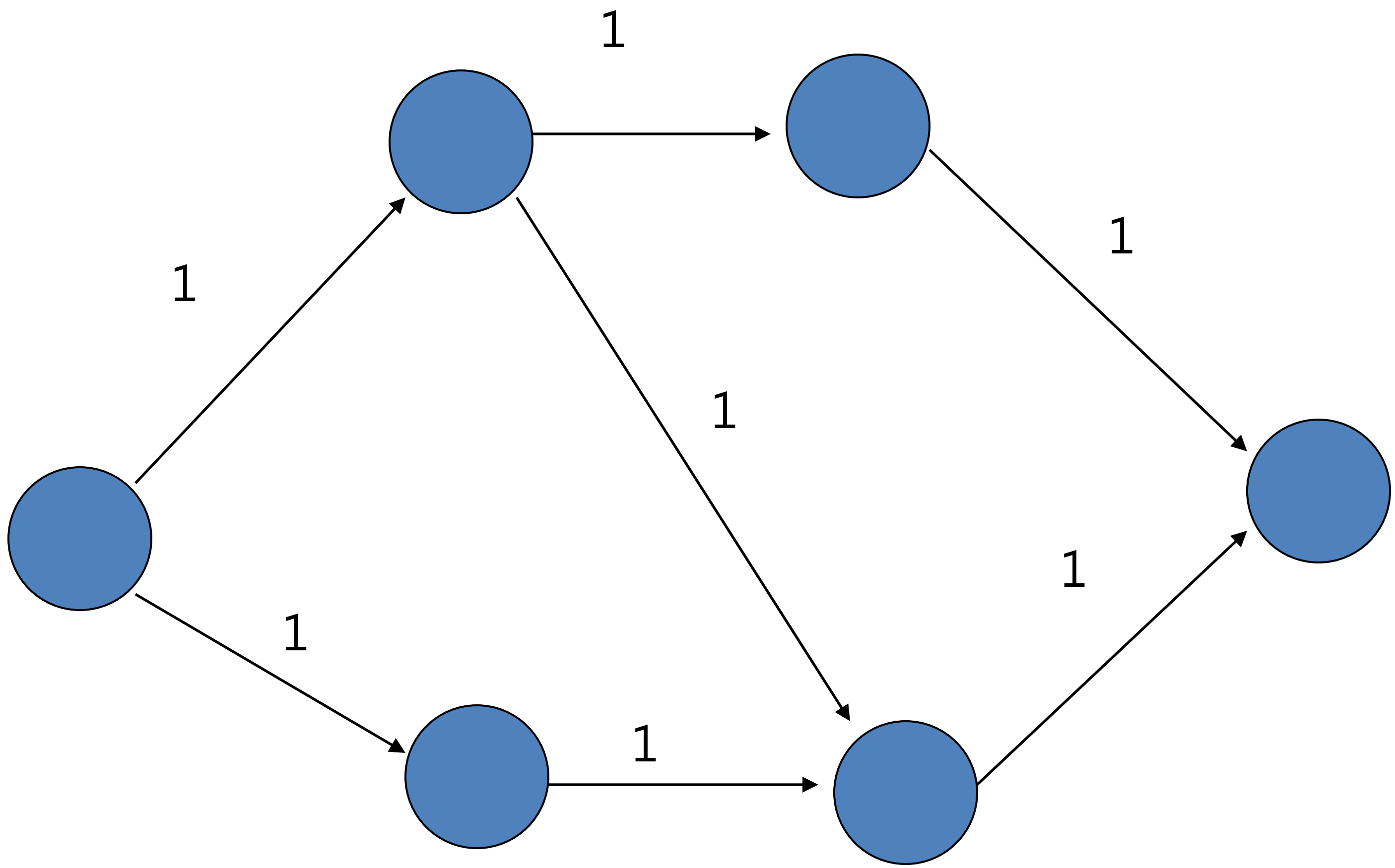
Maximum Flow

- Given a flow network and a flow, the **residual network** consists of edges that can admit more net flow.
- $G=(V,E)$ --a flow network with source s and sink t
- f: a flow in G.
- The amount of additional net flow from u to v before exceeding the capacity $c(u,v)$ is the residual capacity of (u,v), given by:
  - In the regular direction: $c_f(u,v)=c(u,v)-f(u,v)$
  - in the other direction: $c_f(v, u)=c(v, u)+f(u, v)$.

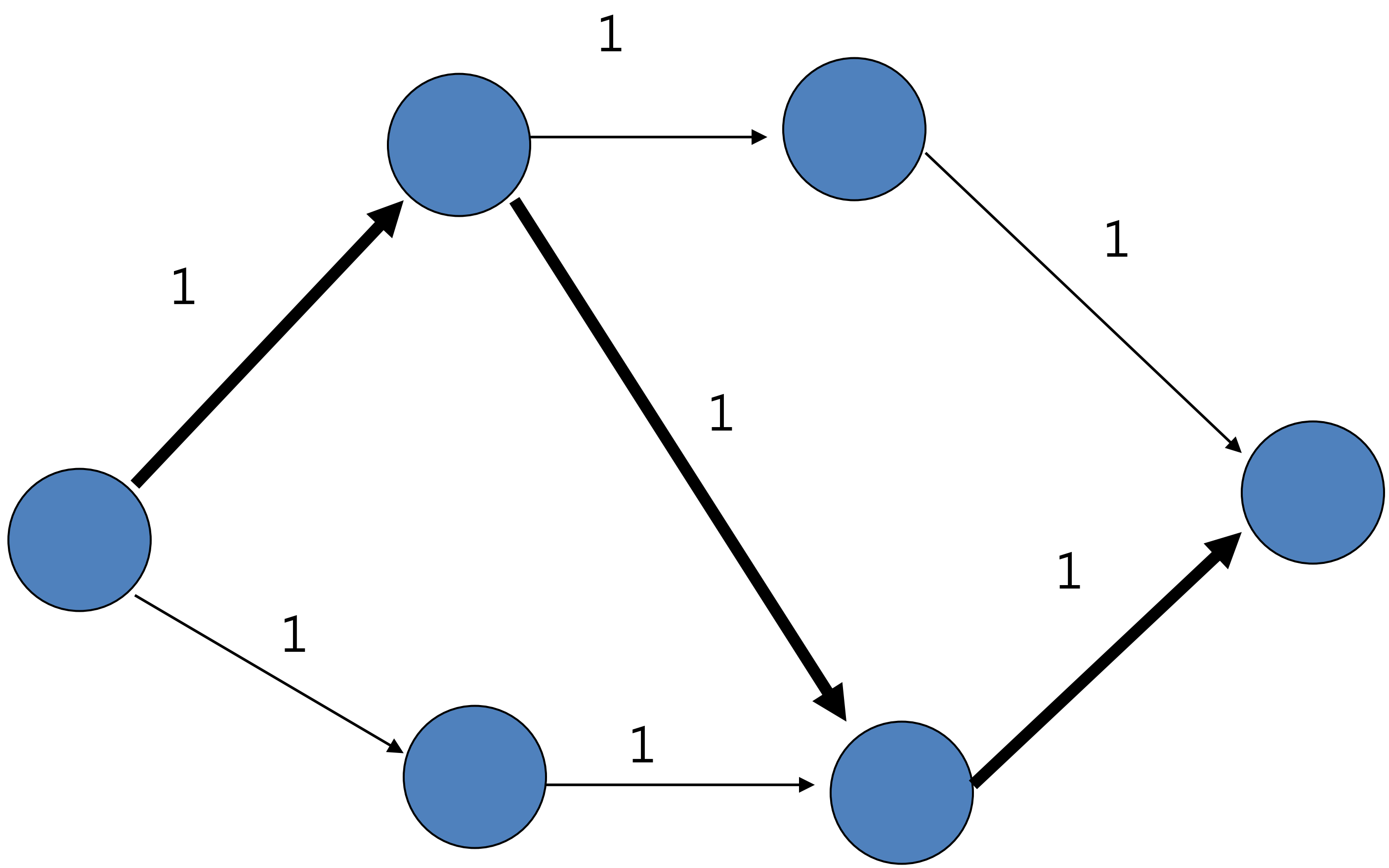
Maximum Flow



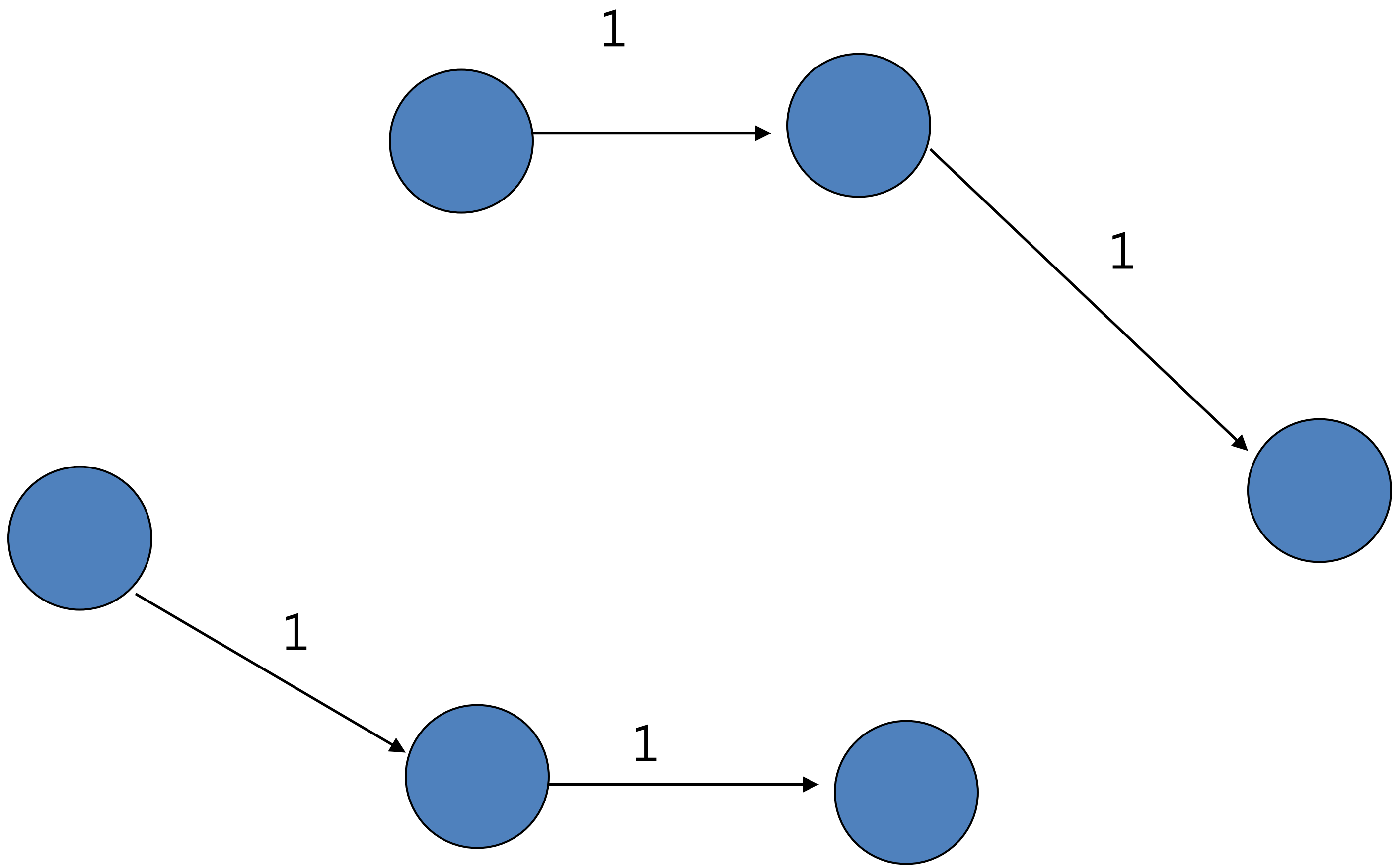
Maximum Flow



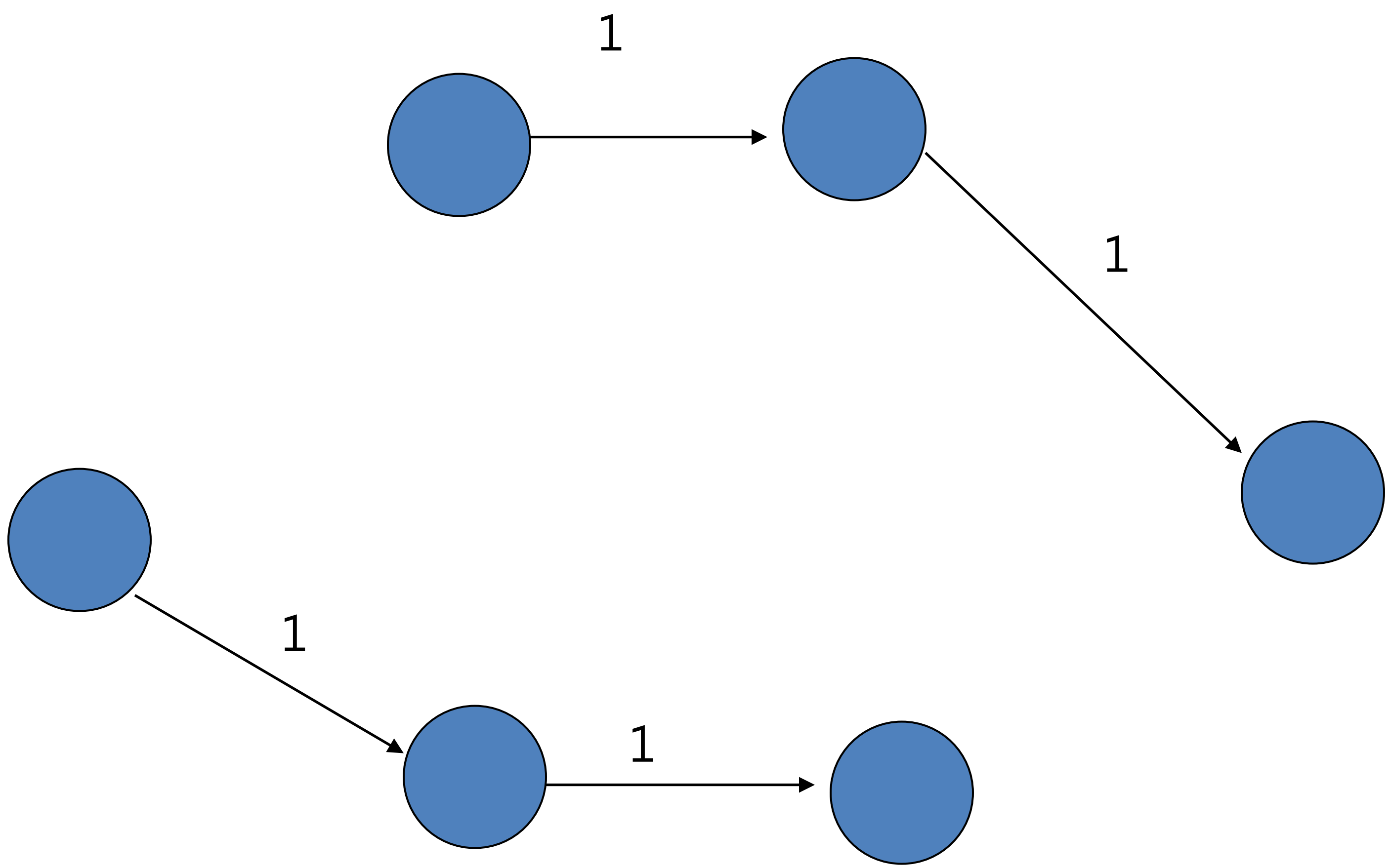
Maximum Flow



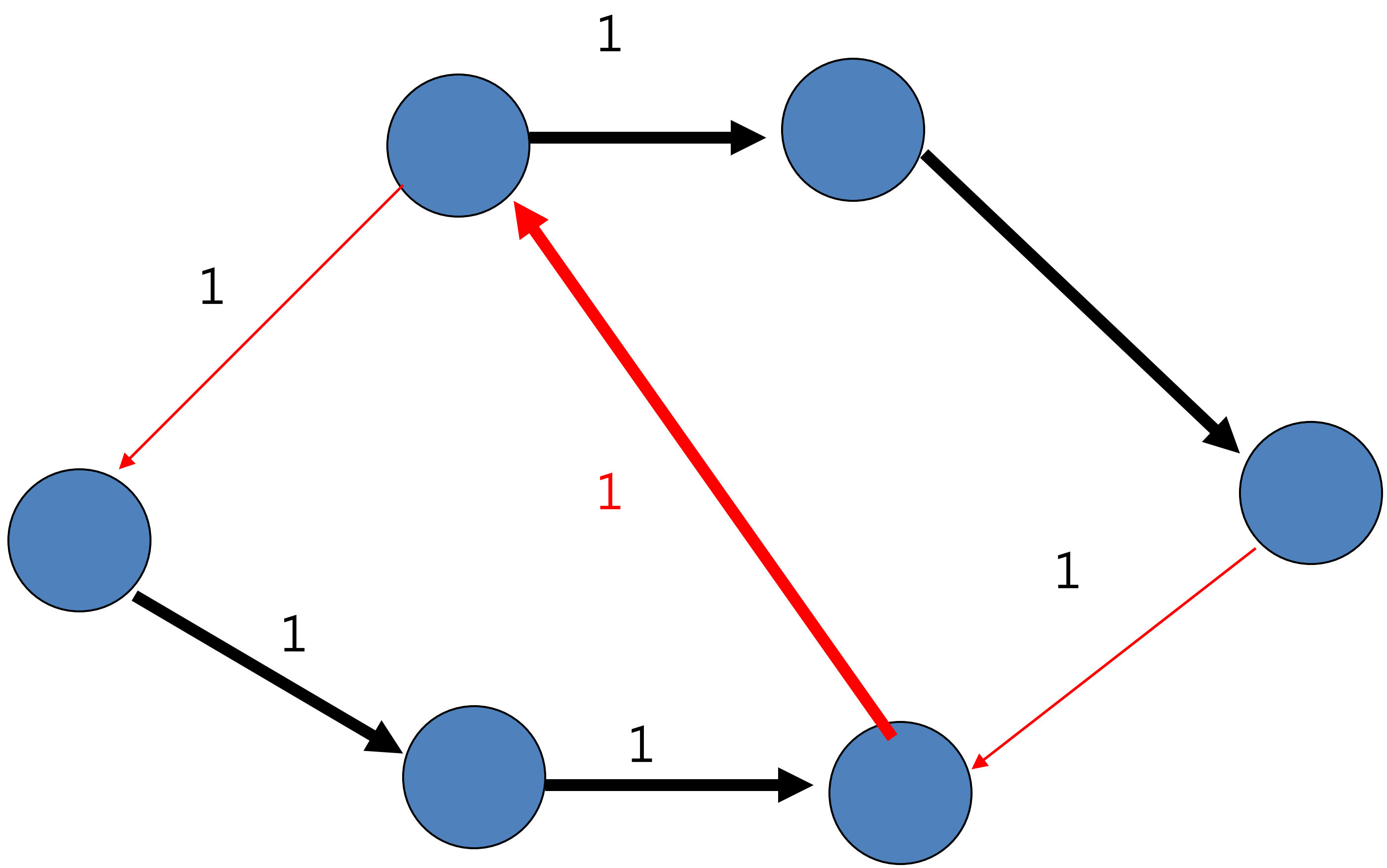
Maximum Flow

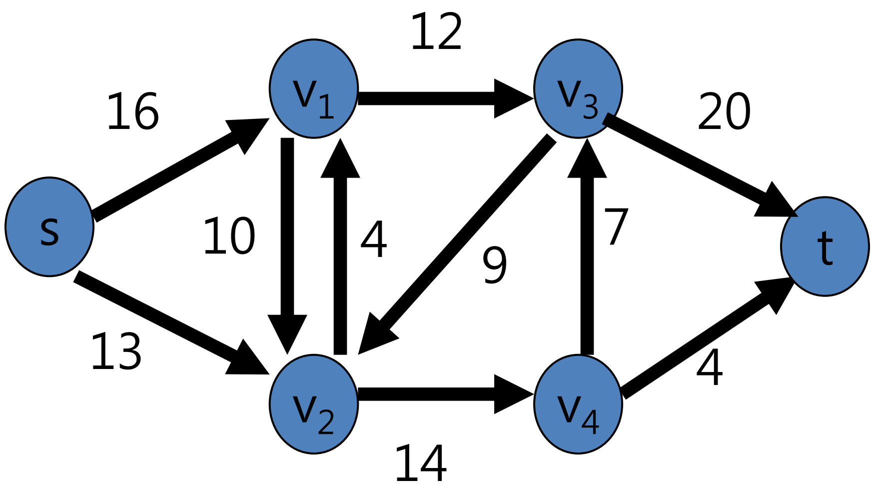


Maximum Flow

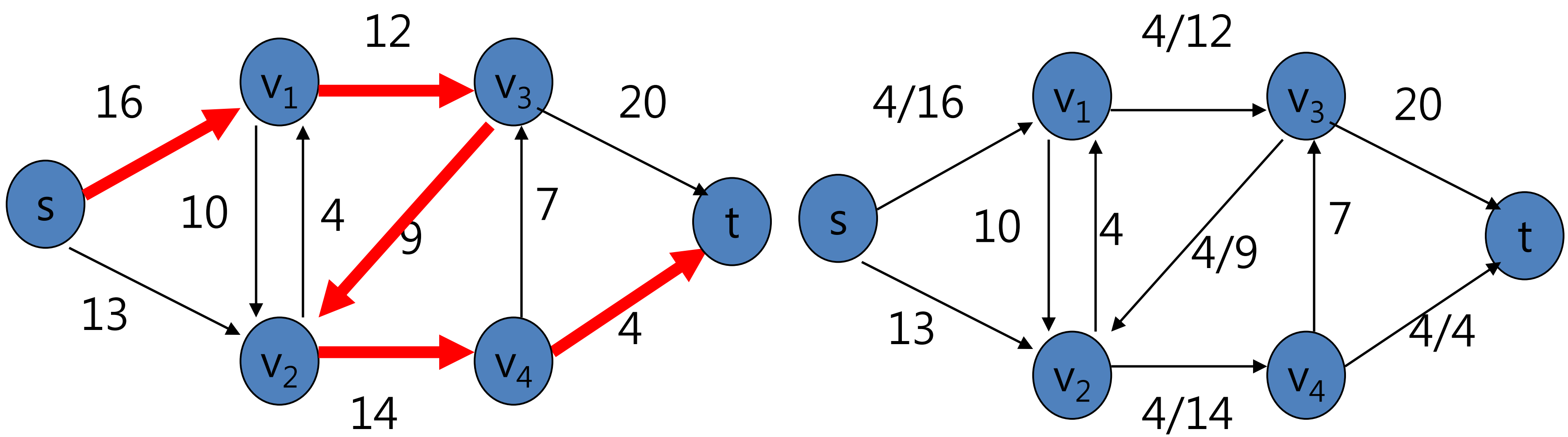


Maximum Flow

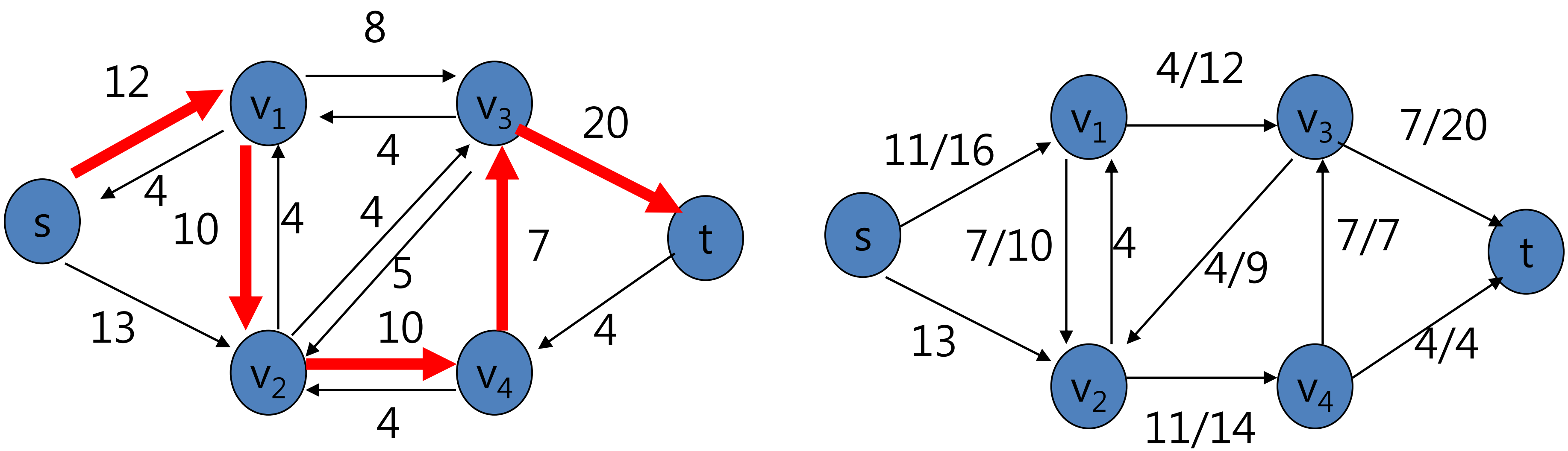




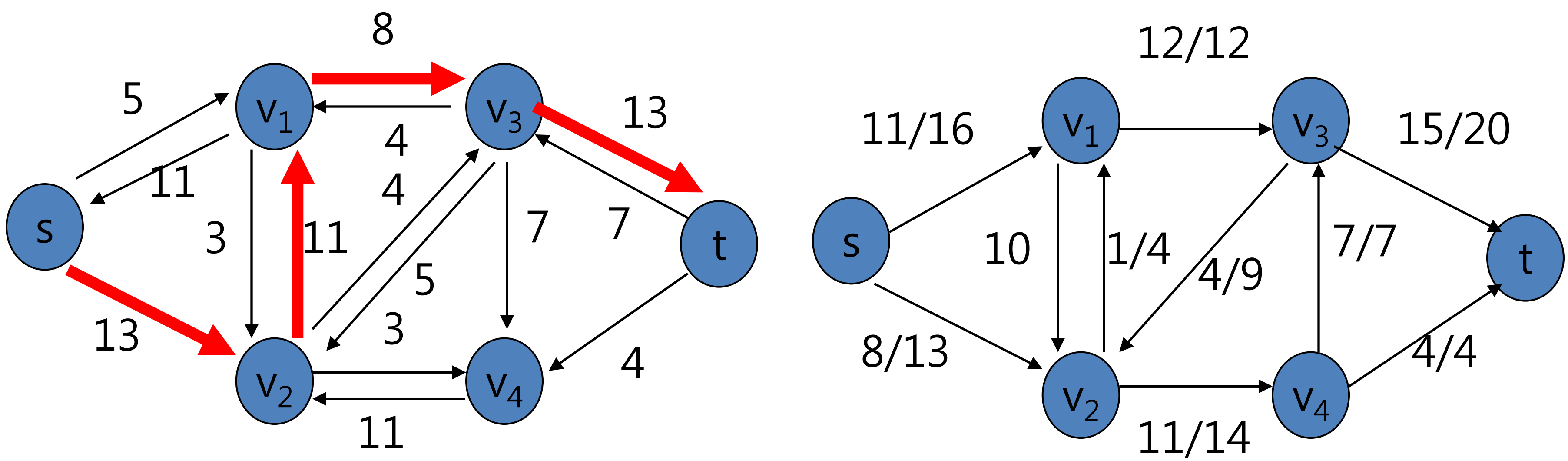
Initial



(a)


s
v_1
v_2
v_3
v_4
t
12
8
4
20
4
10
4
4
5
7
13
10
4
4
s
v_1
v_2
v_3
v_4
t
11/16
4/12
7/20
7/10
4
4/9
7/7
13
11/14
4/4

(b)



s
v_1
v_2
v_3
v_4
t
5
11
13
3
11
8
4
4
5
3
7
7
13
4
11
s
v_1
v_2
v_3
v_4
t
11/16
8/13
10
1/4
12/12
4/9
11/14
7/7
15/20
4/4

(c)

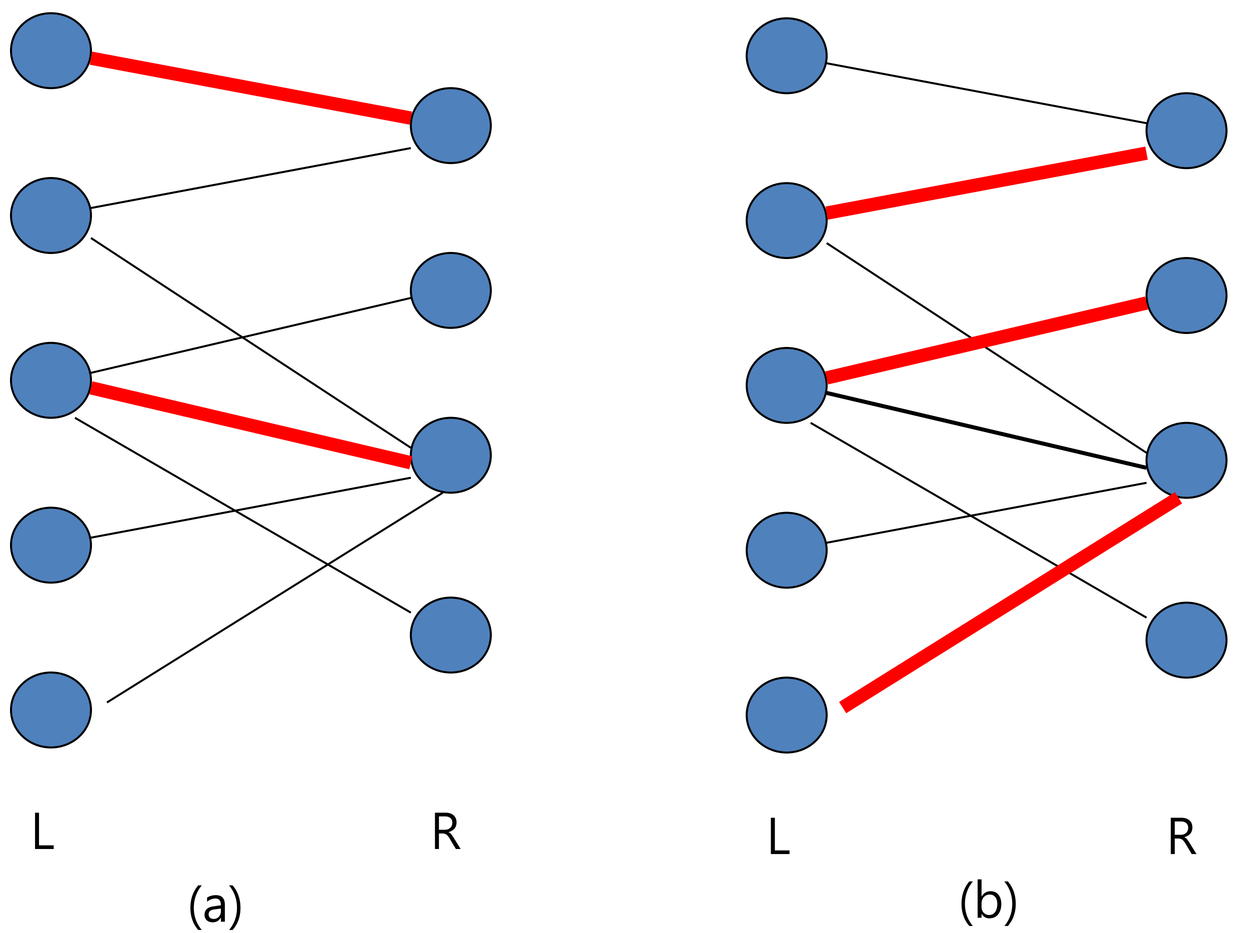
Example

- Uva 820

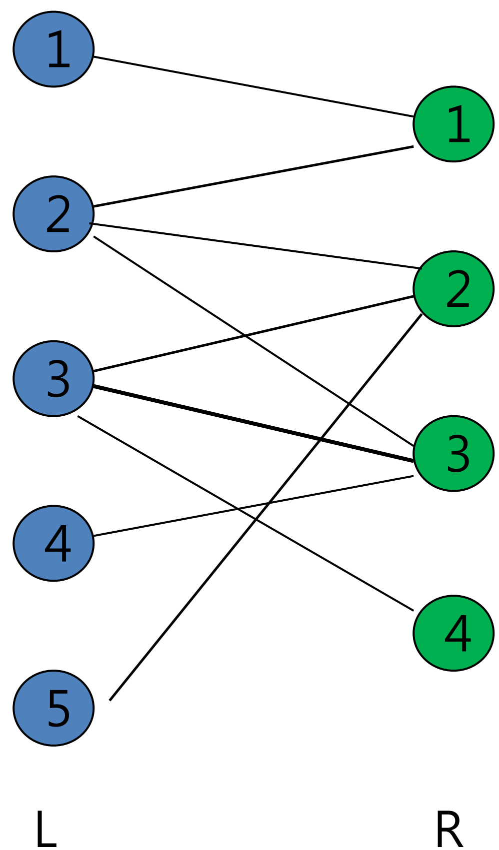
Outline

Bipartite Matching

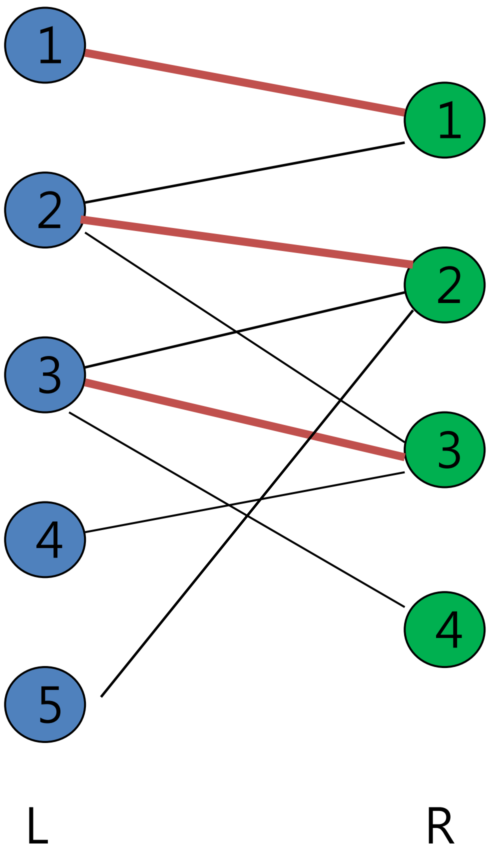
Bipartite Matching



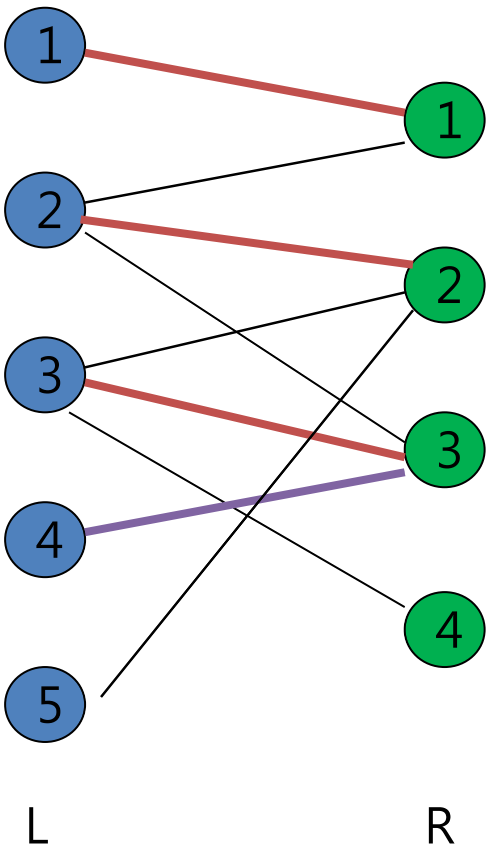
Bipartite Matching



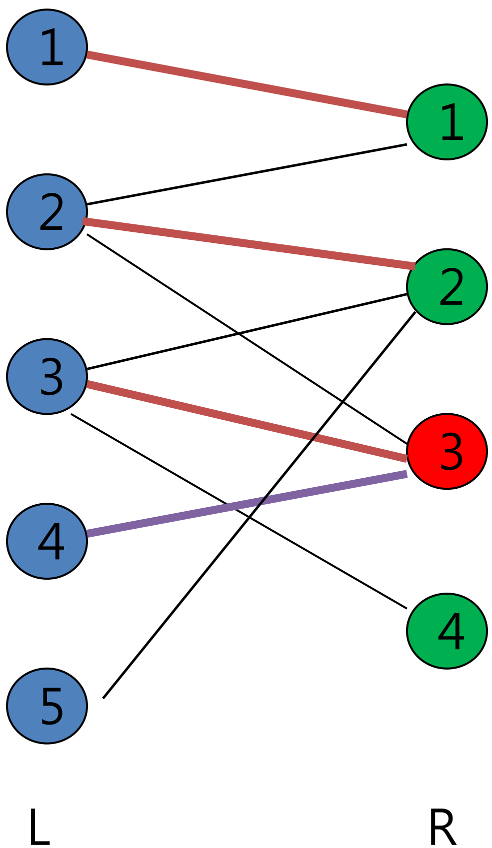
Bipartite Matching

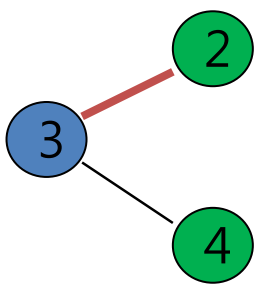


Bipartite Matching

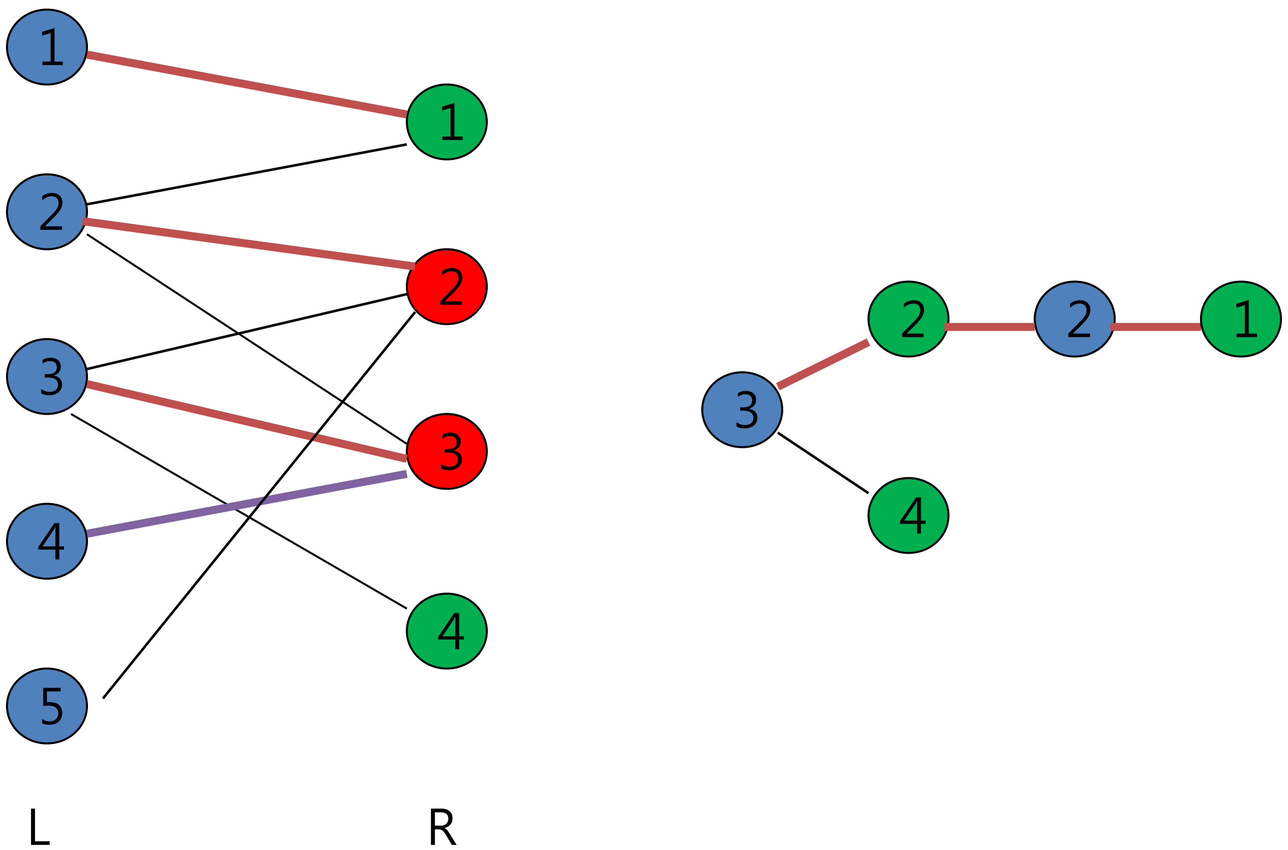


Bipartite Matching

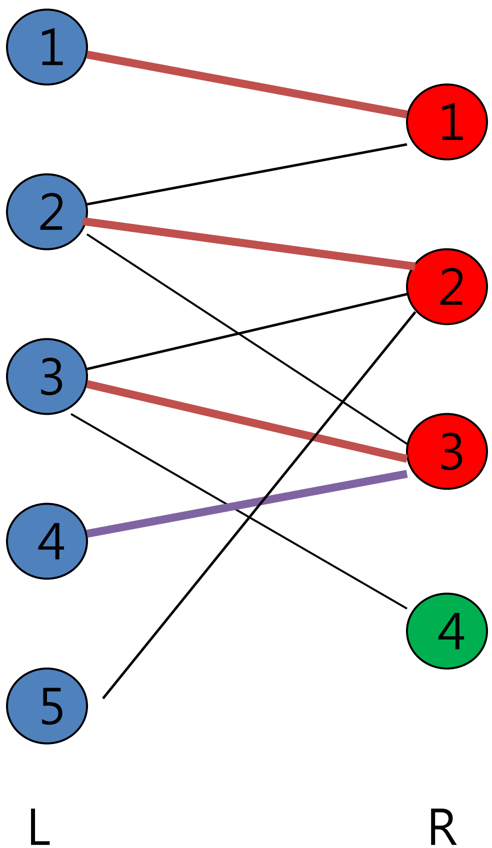


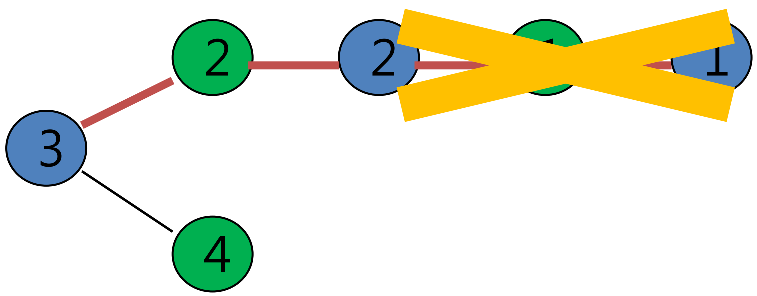


Bipartite Matching

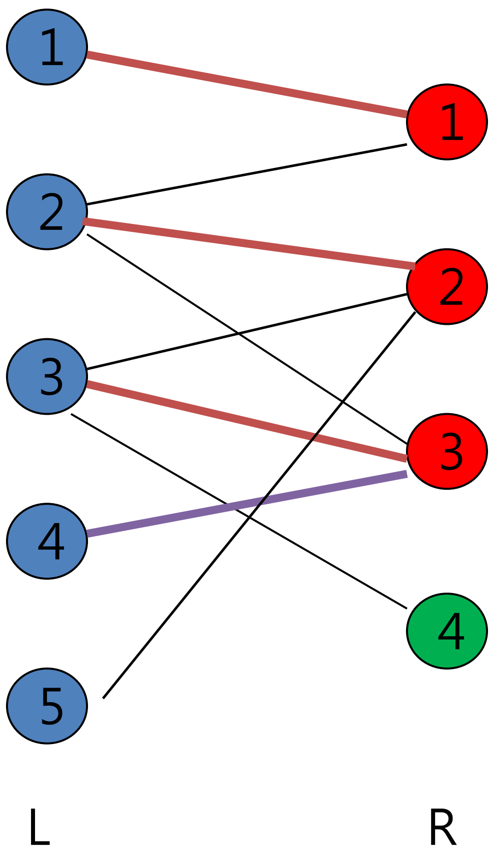


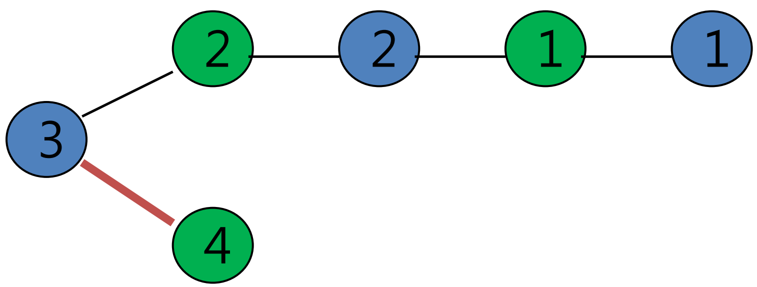
Bipartite Matching



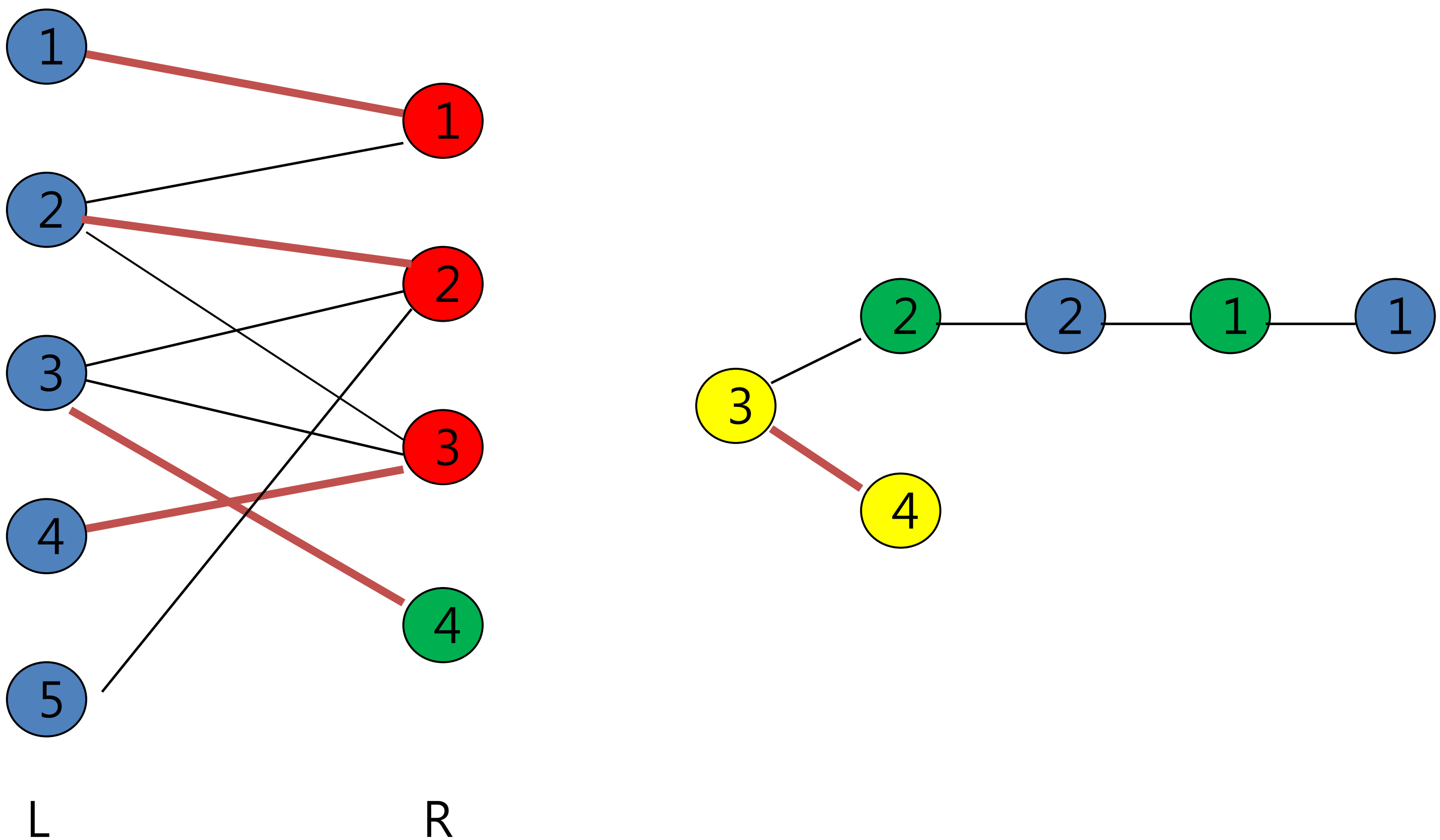


Bipartite Matching





Bipartite Matching



Bipartite Matching

```
int bipartite_matching()
{
    // 全部的點初始化為未匹配點。
    memset(mx, -1, sizeof(mx));
    memset(my, -1, sizeof(my));
    int c=ini_matching();  // 能連的先連一連
    // 依序把x中的每一個點作為擴充路徑的端點，並嘗試尋找擴充路徑
    for (int x=1; x<=nx; ++x)
        if (mx[x] == -1)    // x為未匹配點
        {
            // 開始Graph Traversal
            memset(vy, false, sizeof(vy));
            if (DFS(x)) c++;
        }
    return c;
}
```

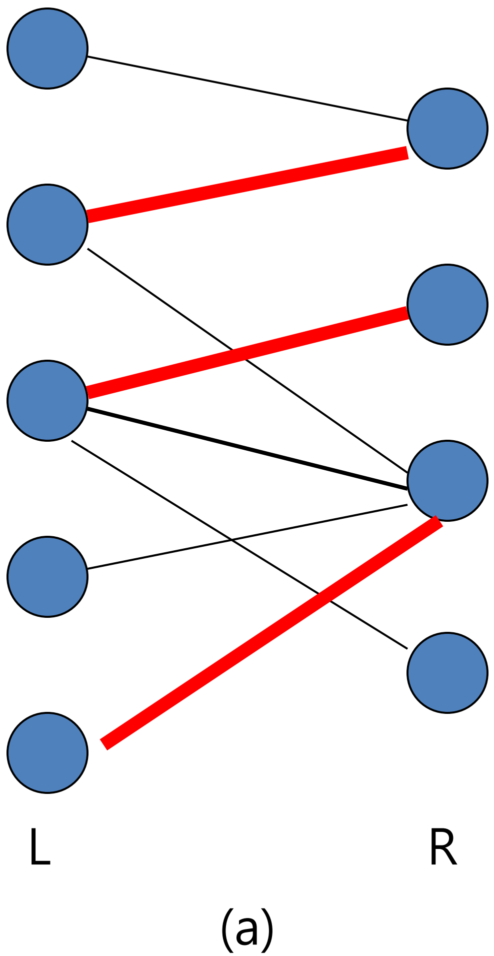
Bipartite Matching

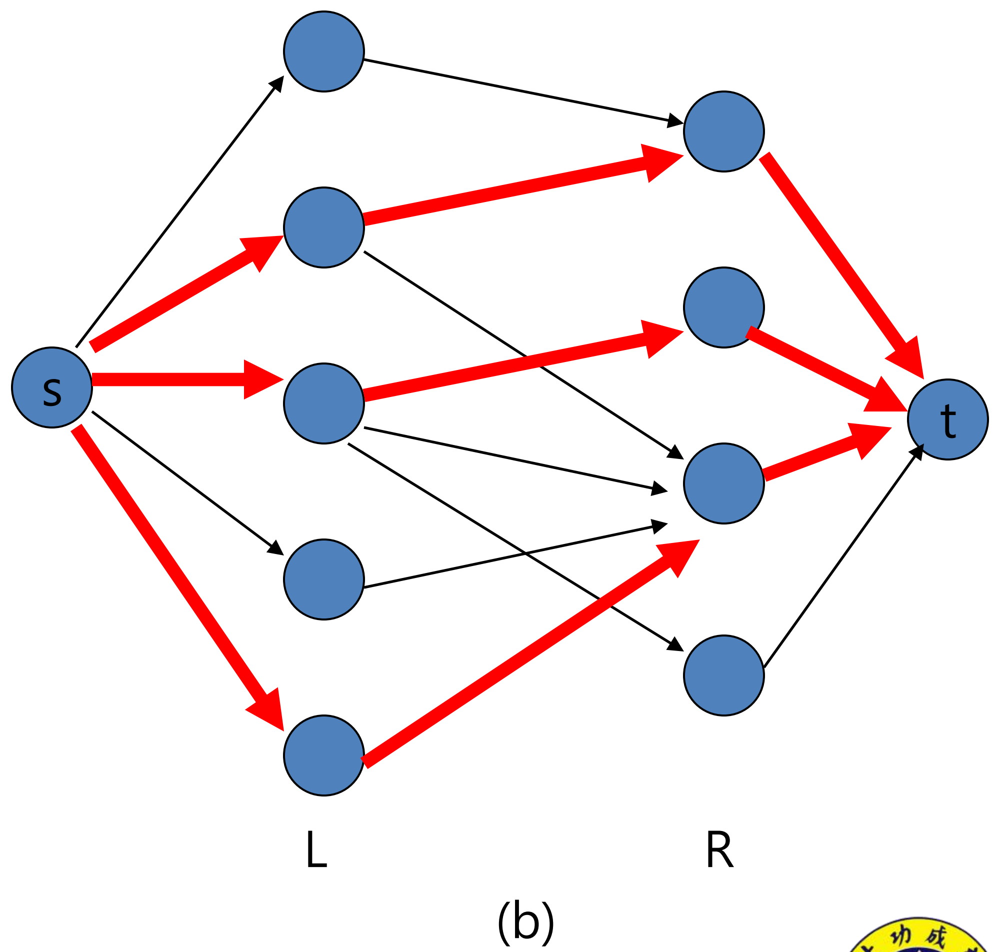
```
bool DFS(int x)
{
    for (int y=1; y<=ny; ++y)
        if (adj[x][y] && !vy[y])
        {
            vy[y] = true;
            // 找到擴充路徑
            if (my[y] == -1 || DFS(my[y]))
            {
                mx[x] = y; my[y] = x;
                return true;
            }
        }
    return false;
}
```

Example

- POJ 1274

Bipartite Matching





Homework

- POJ:
  - **Flow**: 1149, 1459, 2112, 2289, 2396, 2455, 2584, 3189, 3228
  - **Matching**: 1274, 1325, 1422, 1466, 1469, 2060, 2226, 2239, 2446, 2536, 2594, 2724, 3020, 3041, 3207, 3216, 3343, 3692
- UVa:
  - **Flow**: 820, 10330, 10779, 563, 10511, 10983, 10806, 10380
  - **Matching**: 259, 670, 753, 10080, 10092, 10243, 10418, 10984, 663