Competitive Algorithm Design and Practice
Bipartite Matching
2014/04/30

Guan Yu, Chen (kevinx6000)
kevinx6000@gmail.com

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

Outline

- Pre-concept
  - Matching
  - Cardinality vs Weighted
  - Bipartite Graph
- Maximum Cardinality Bipartite Matching
  - Flow Modeling
  - Alternating Path
  - Augmenting Path Algorithm

Matching

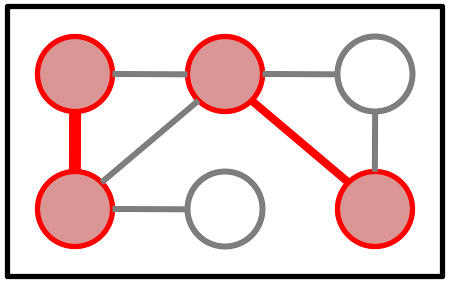
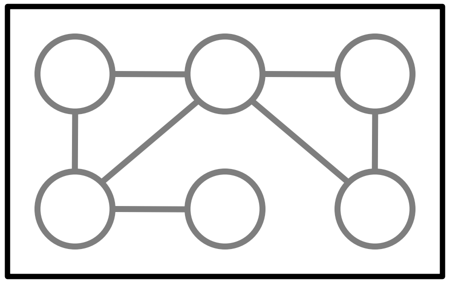
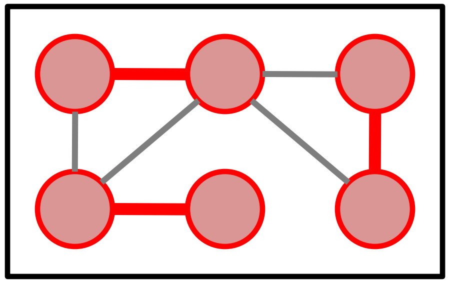
Matching

- 匹配
- A set of edges in a graph without common vertices.

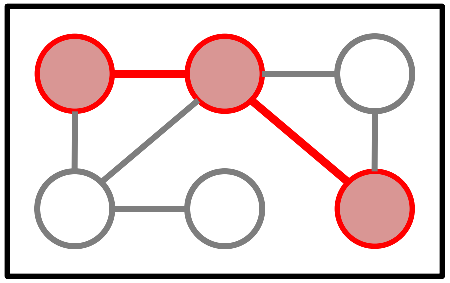
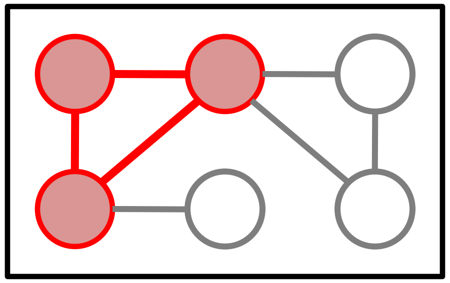
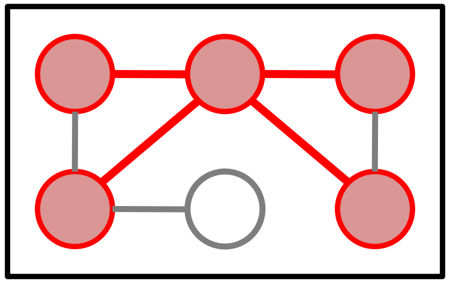
Matching

- 匹配
- A set of edges in a graph without common vertices.

Matching

Not Matching

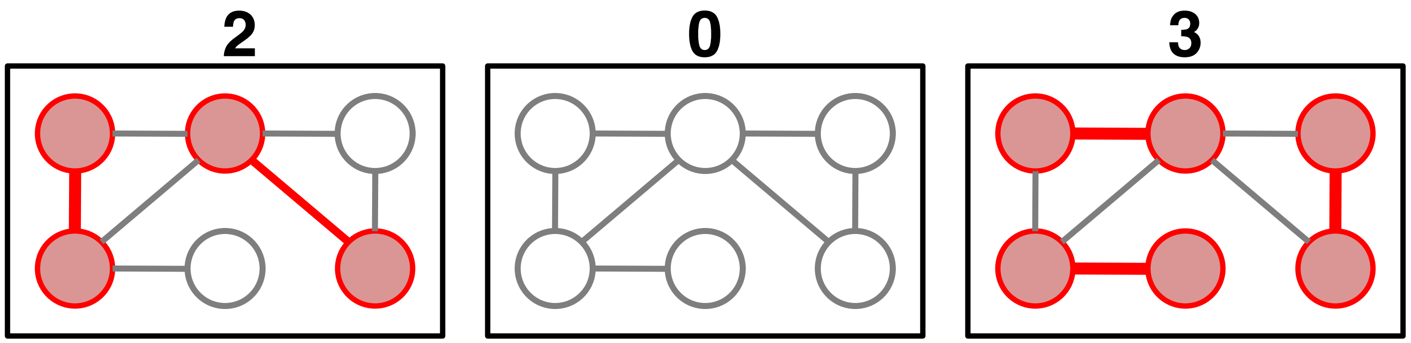
Cardinality vs Weighted

Cardinality vs Weighted

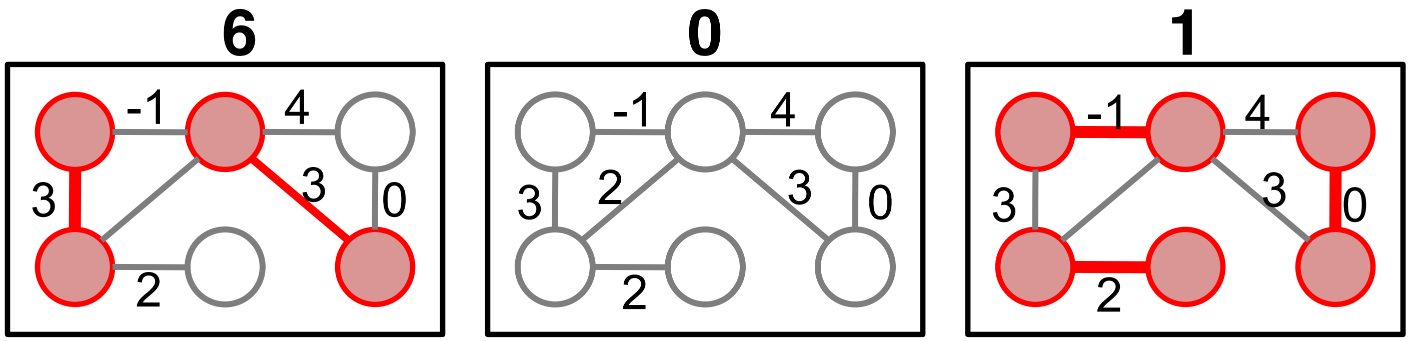
- Cardinality: 個數
- Weighted:　權重

Cardinality vs Weighted

- Cardinality: 個數



- Weighted:　權重



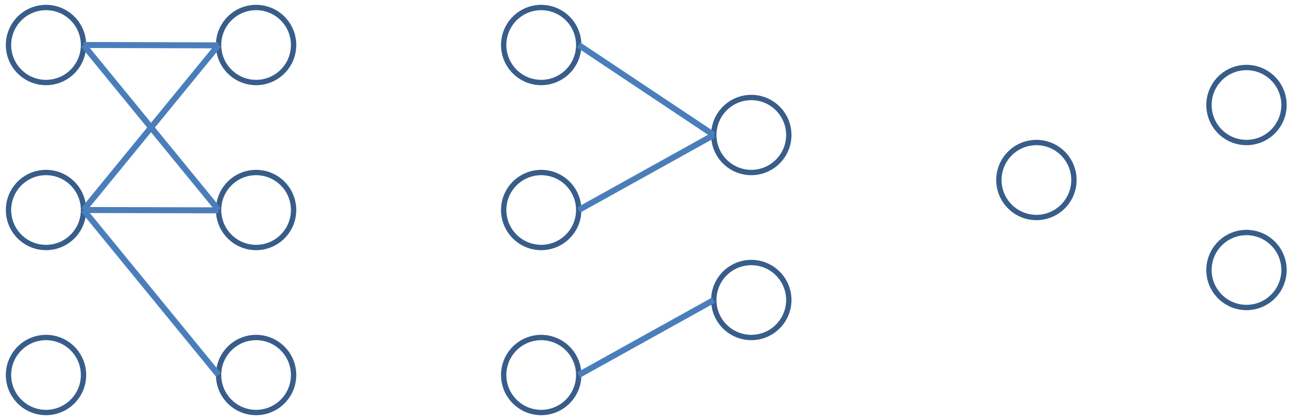
Matching

- Maximum (Cardinality) Matching
  - 最大匹配數
- Maximum Weighted Matching
  - 最大總權重
- Maximum Weighted Maximum Cardinality Matching
  - 最大匹配數的前提下，最大總權重
- And so on…

Bipartite Graph

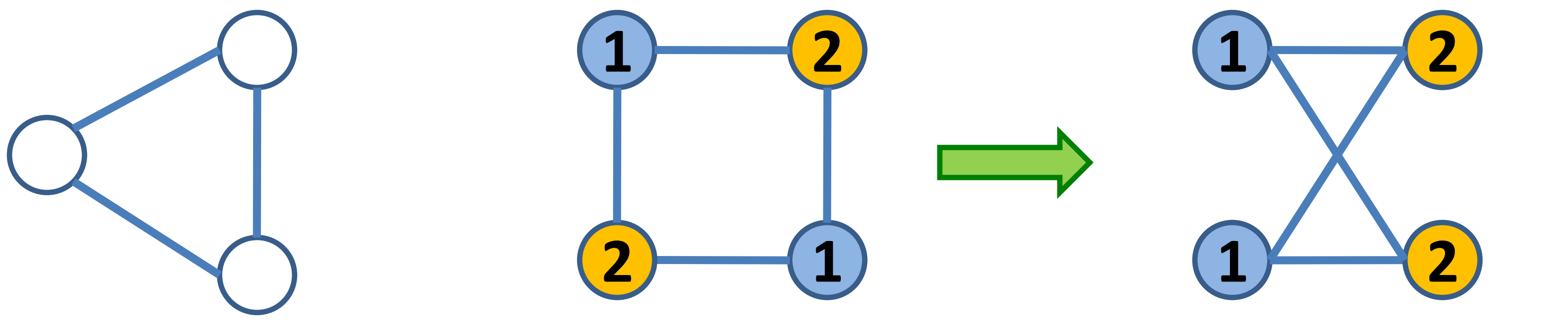
Bipartite Graph

- 二分圖
- 可以分成兩群，每群內彼此間沒有edge



Bipartite Graph

- 或者，一張圖上不存在odd (length) cycle
- 若給一張不存在odd cycle的圖，
  可用DFS/BFS標號將圖分成兩群

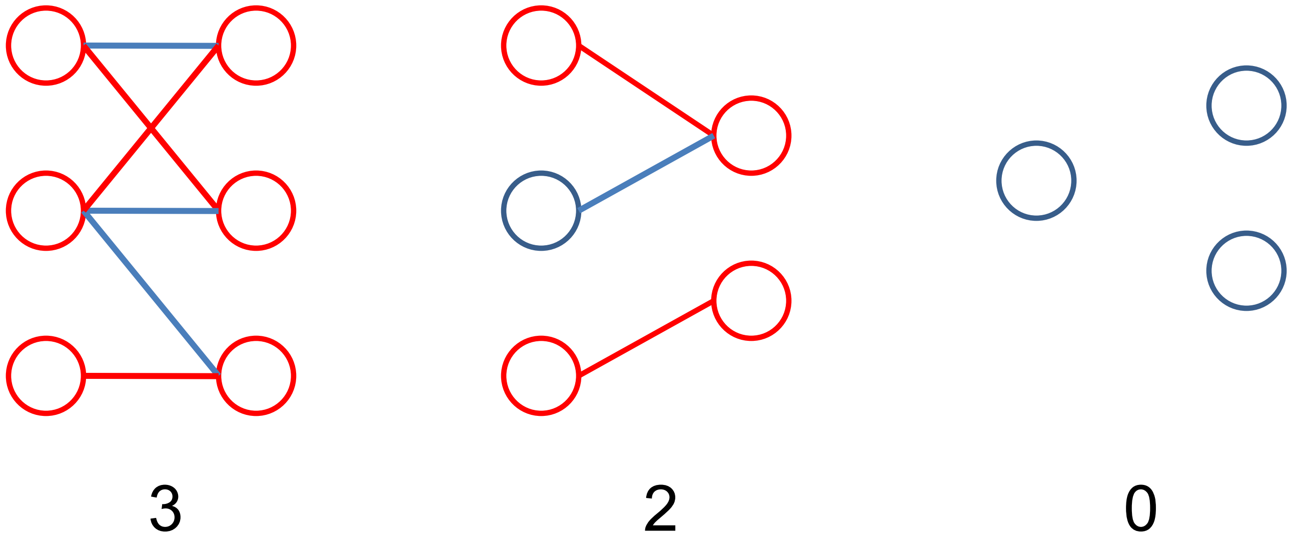


odd cycle　　even cycle　　Bipartite Graph

Maximum Cardinality Bipartite Matching

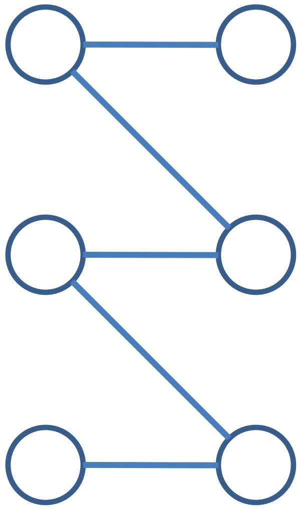
Maximum Bipartite Matching

- 二分圖最大匹配(數)



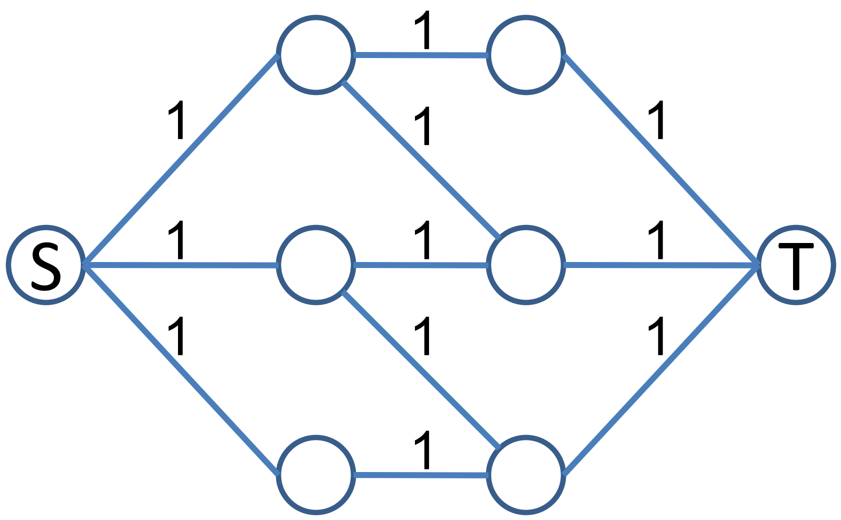
Maximum Bipartite Matching

- Flow modeling



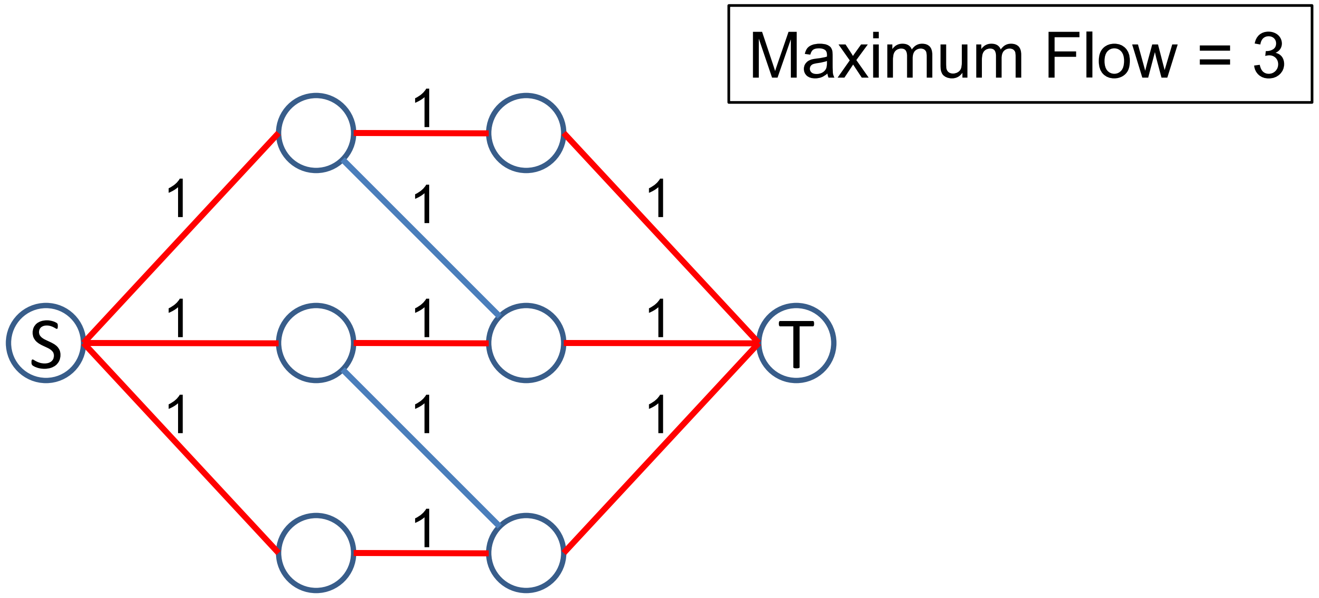
Maximum Bipartite Matching

- Flow modeling



Maximum Bipartite Matching

- Flow modeling



Maximum Bipartite Matching

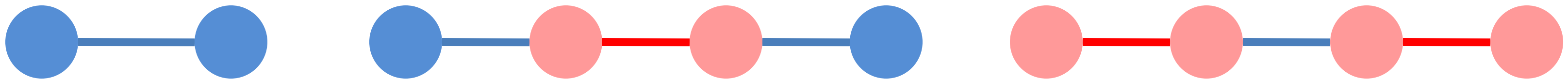
- Flow modeling
  - Edmonds-Karp: $O(VE^2)$
  - Dinic: $O(V^2E)$

- Faster algorithm…?

Augmenting Path Algorithm

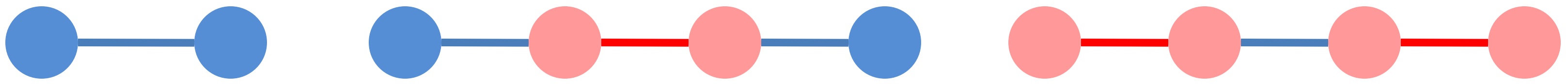
Augmenting Path Algorithm

- Alternating Path (交錯路徑)
  - 匹配邊與未匹配邊交替出現的路徑



Augmenting Path Algorithm

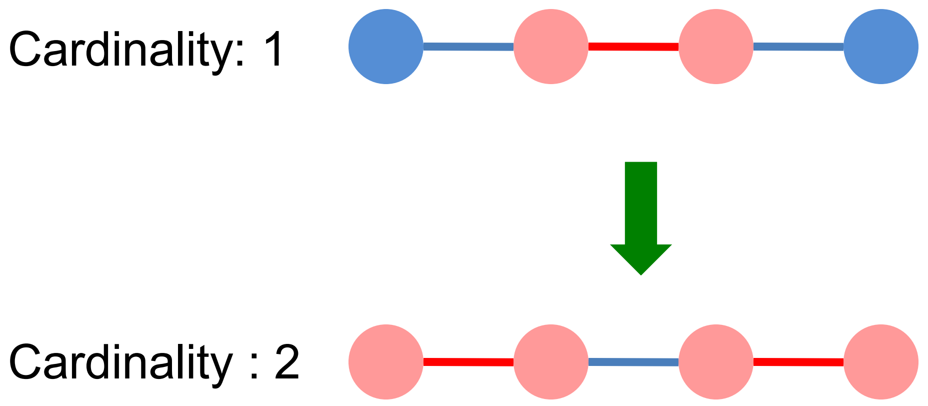
- Augmenting Path(增廣路徑)
  - 起點與終點都是未匹配點的alternating path



Yes　　Yes　　No

Augmenting Path Algorithm

- 觀察:
  - 將augmenting path的匹配邊與未匹配邊對調，匹配數量**加1**，且不影響匹配正確性

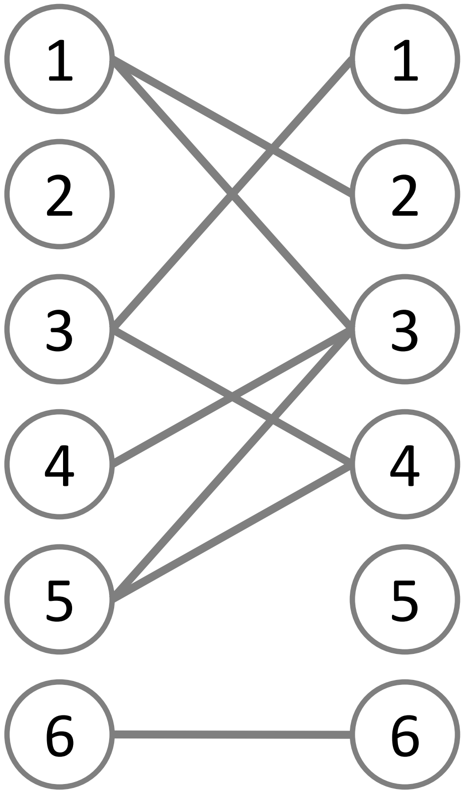


Augmenting Path Algorithm

- Algorithm:
  1. 枚舉左邊這群的每個點，嘗試找尋augmenting path
  2. 每次找到augmenting path，對調匹配與未匹配邊

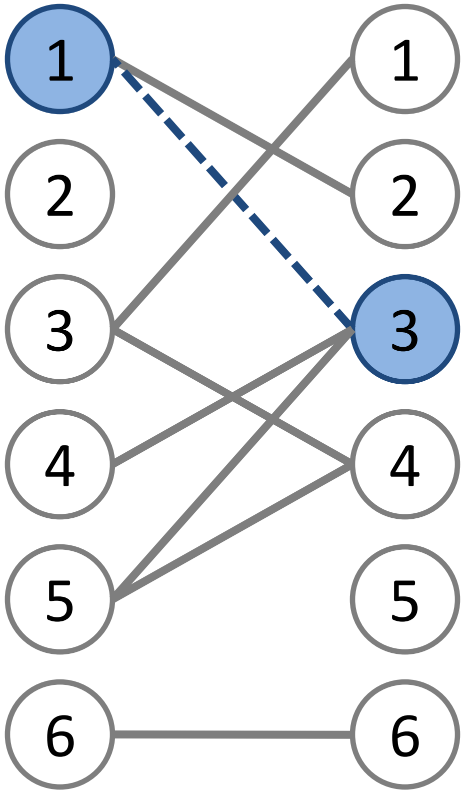
Augmenting Path Algorithm

- Example



Augmenting Path Algorithm

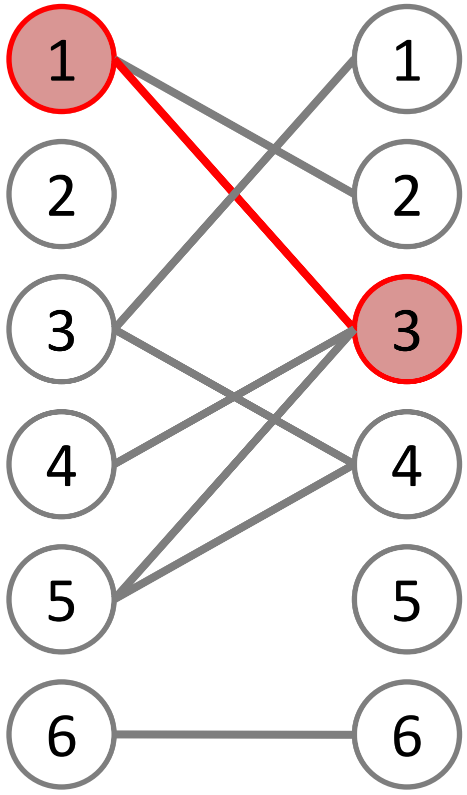
- Example



找到augmenting path:
1-3

Augmenting Path Algorithm

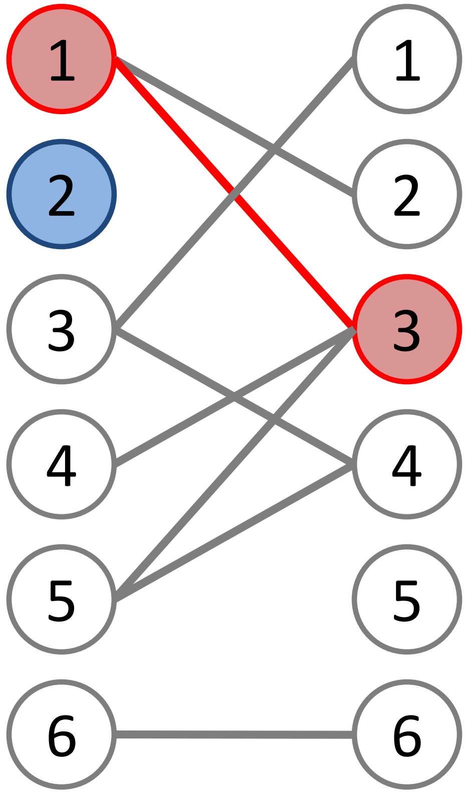
- Example



對調，匹配數+1

Augmenting Path Algorithm

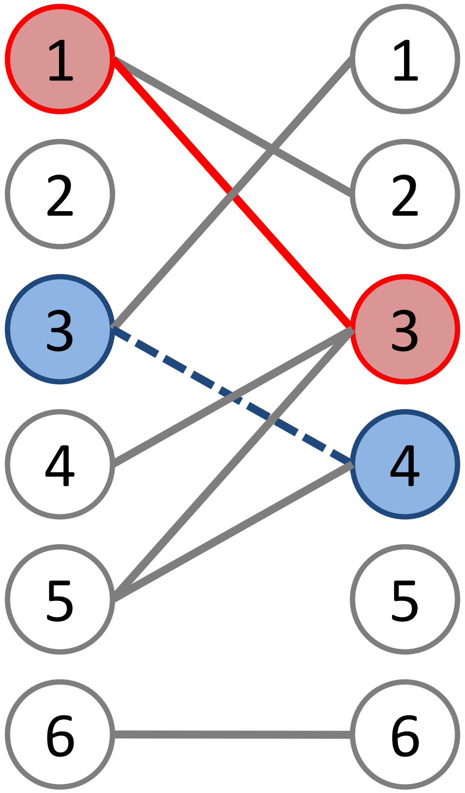
- Example



沒有augmenting path

Augmenting Path Algorithm

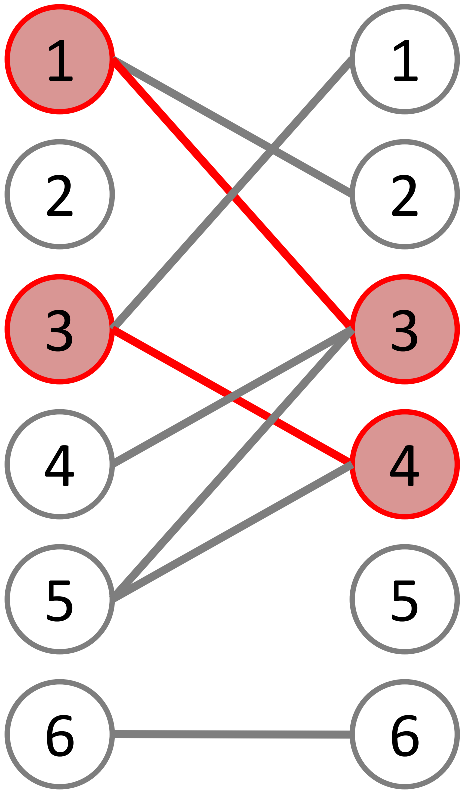
- Example



找到augmenting path:
3-4

Augmenting Path Algorithm

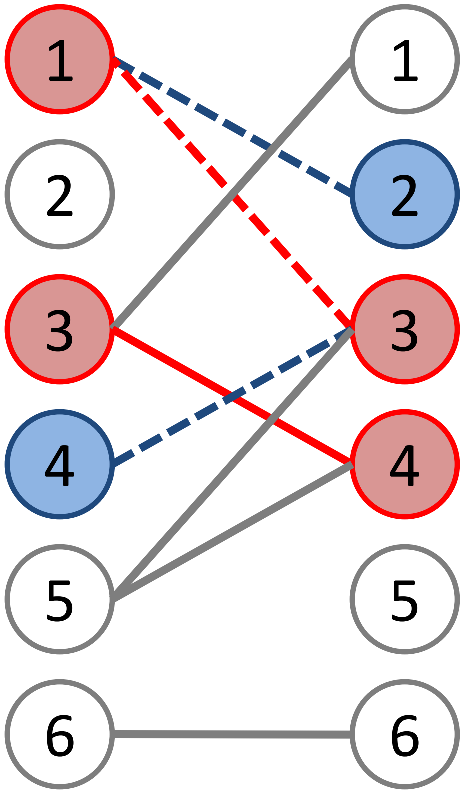
- Example



對調，匹配數+1

Augmenting Path Algorithm

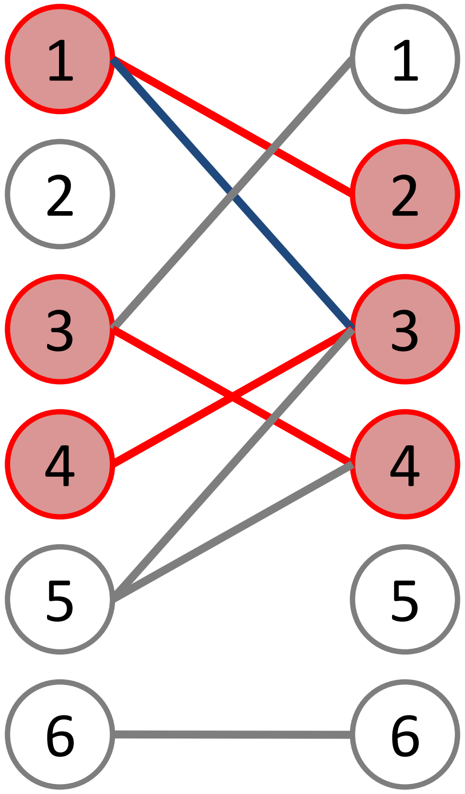
- Example



找到augmenting path:
4-3-1-2

Augmenting Path Algorithm

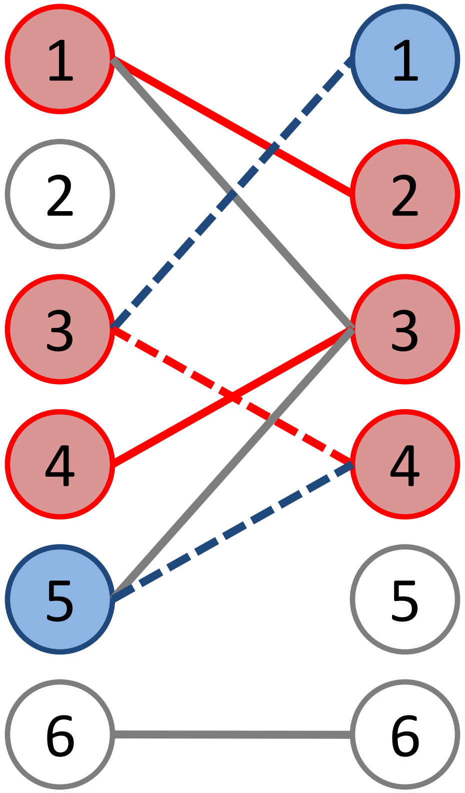
- Example



對調，匹配數+1

Augmenting Path Algorithm

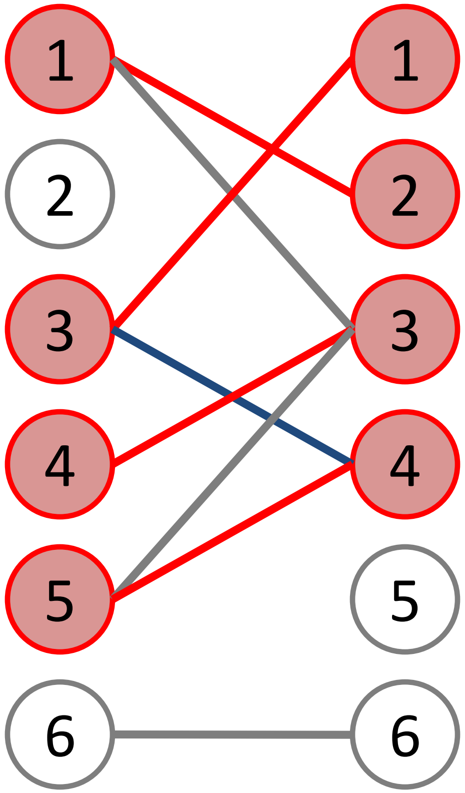
- Example



找到augmenting path:
5-4-3-1

Augmenting Path Algorithm

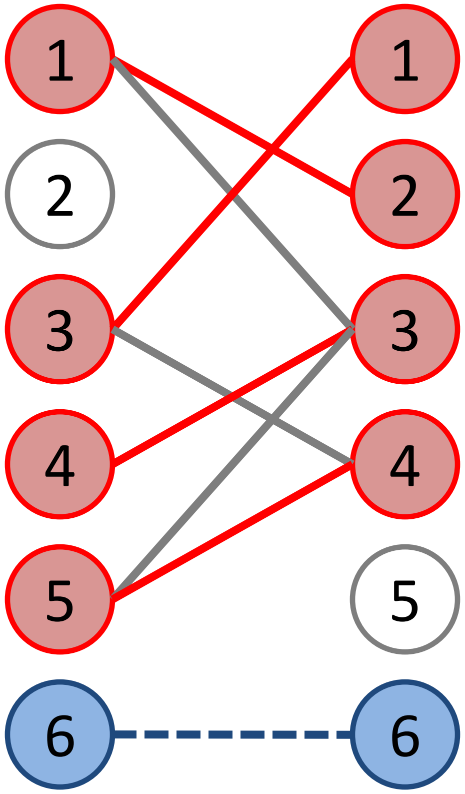
- Example



對調，匹配數+1

Augmenting Path Algorithm

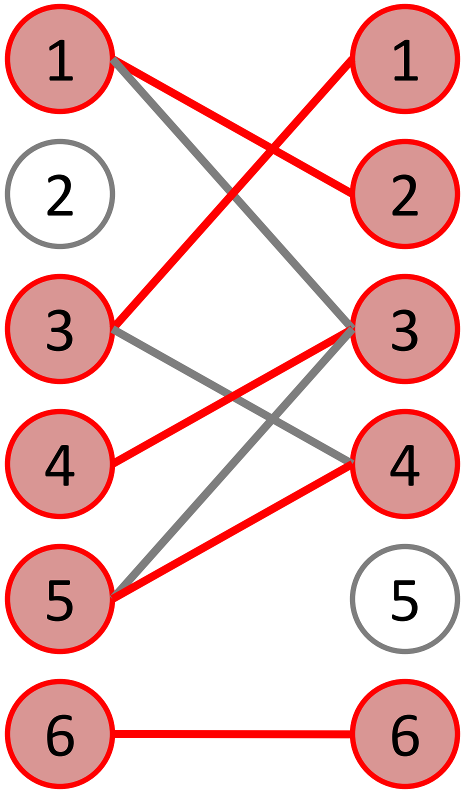
- Example



找到augmenting path:
6-6

Augmenting Path Algorithm

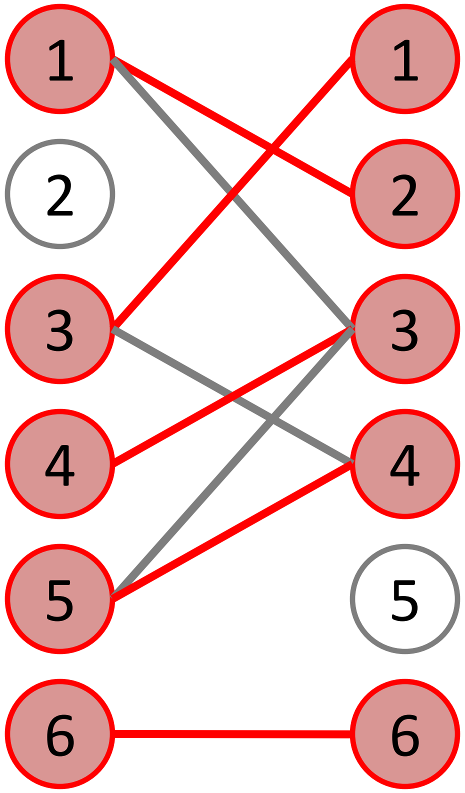
- Example



對調，匹配數+1

Augmenting Path Algorithm

- Example



最大匹配數: 5
1-2
3-1
4-3
5-4
6-6

Augmenting Path Algorithm

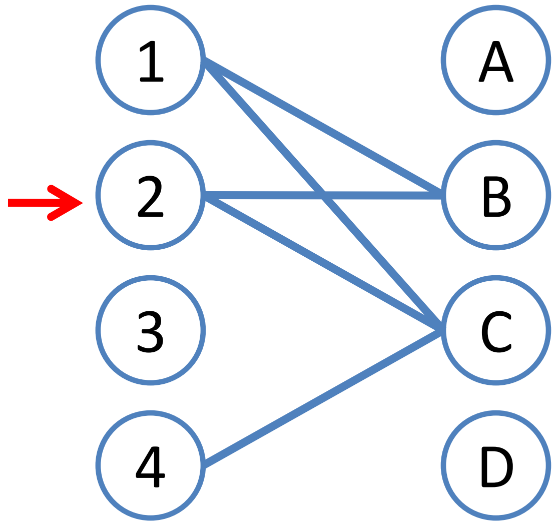
- 找augmenting path:
  - Backtracking?

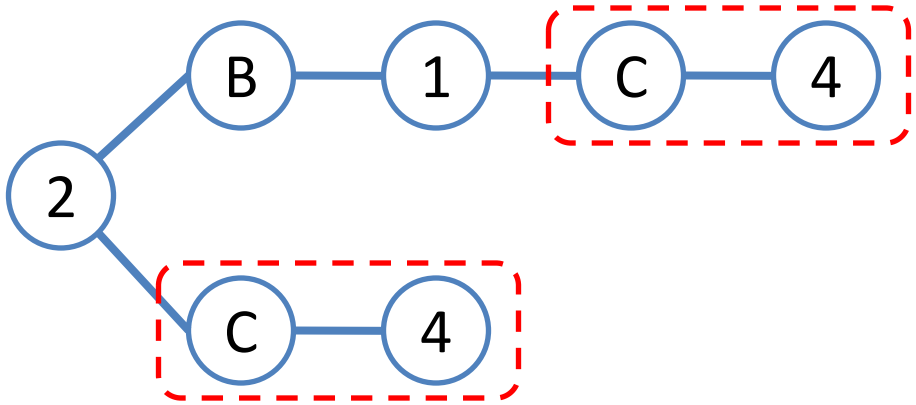
Augmenting Path Algorithm

- 找augmenting path:
  - Backtracking? **NO!**
  - **DFS/BFS**! Why?

Augmenting Path Algorithm

- 找augmenting path:
  - Backtracking? **NO!**
  - **DFS/BFS**! Why?

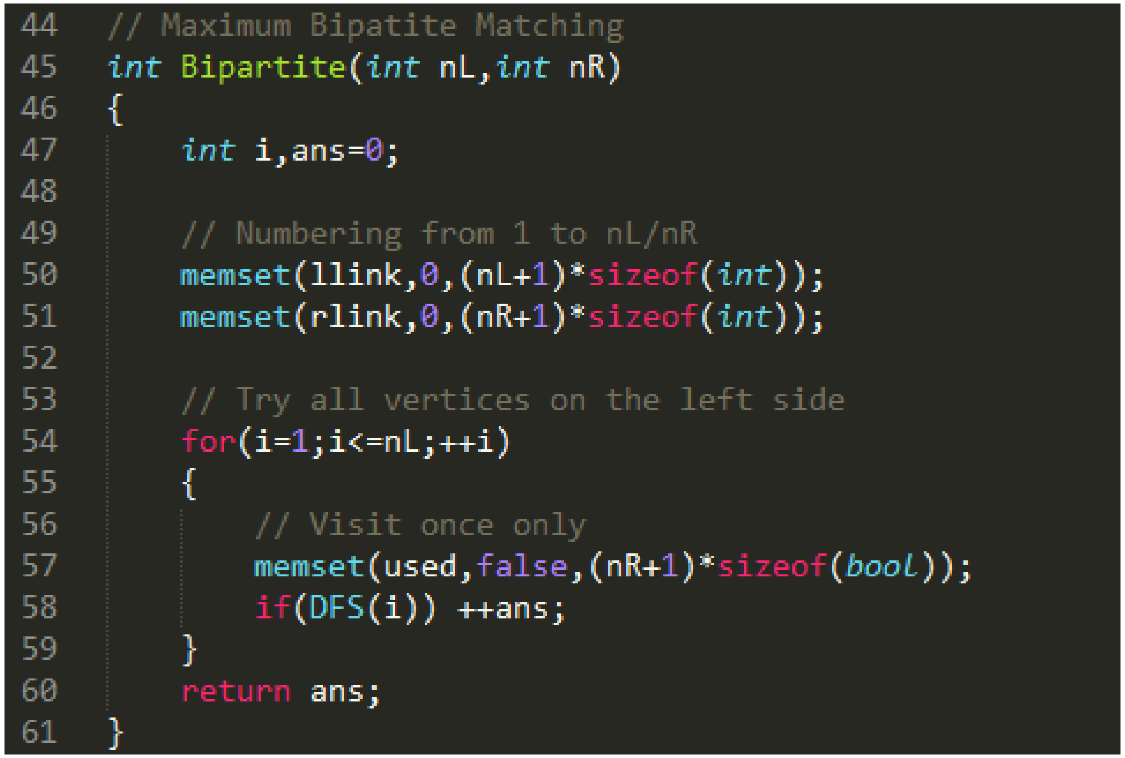




對於右邊visit過的點，不需要重新visit

Augmenting Path Algorithm

- Code



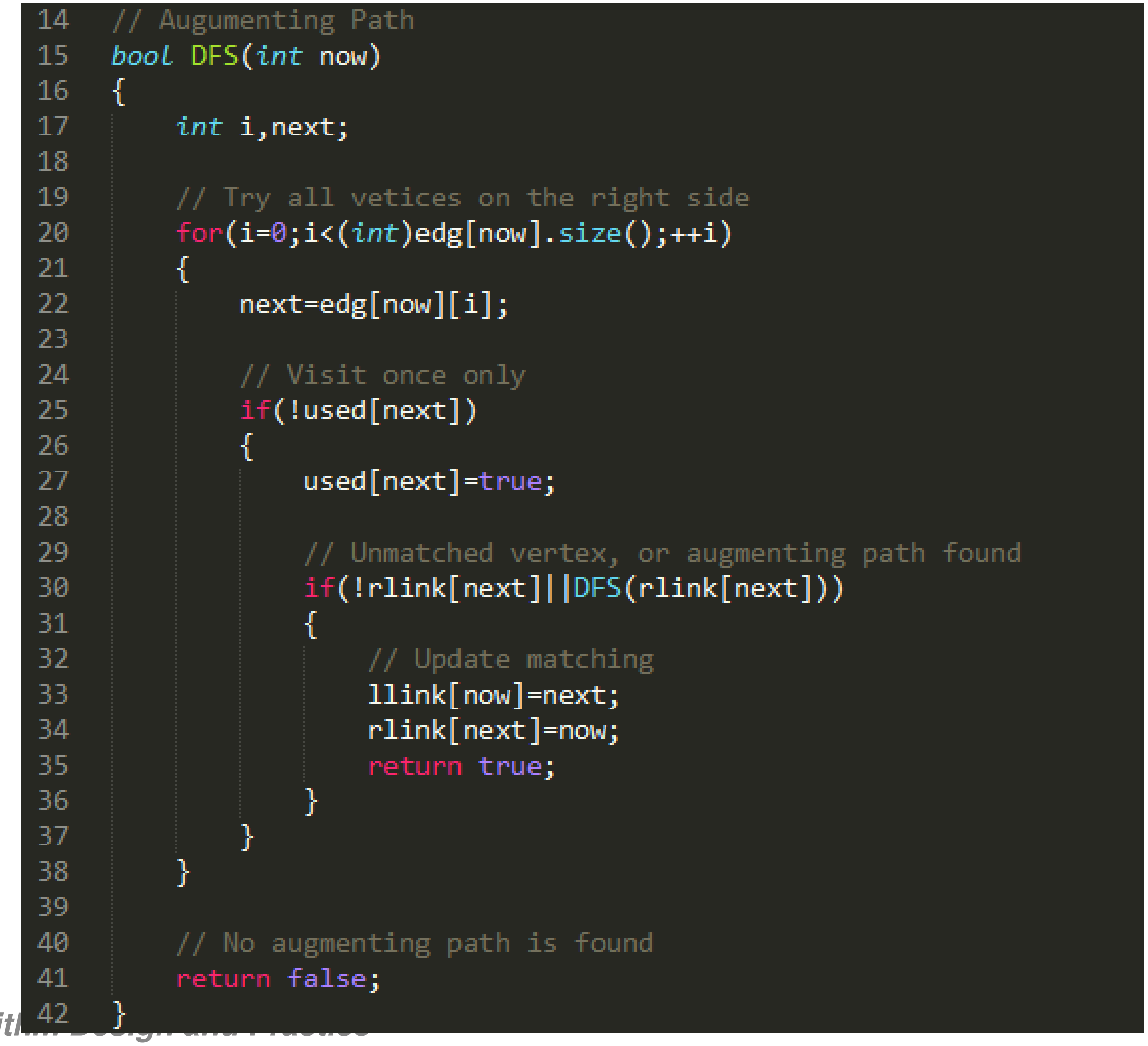
```
// Maximum Bipatite Matching
int Bipartite(int nL,int nR)
{
    int i,ans=0;

    // Numbering from 1 to nL/nR
    memset(llink,0,(nL+1)*sizeof(int));
    memset(rlink,0,(nR+1)*sizeof(int));

    // Try all vertices on the left side
    for(i=1;i<=nL;++i)
    {
        // Visit once only
        memset(used,false,(nR+1)*sizeof(bool));
        if(DFS(i)) ++ans;
    }
    return ans;
}
```

Augmenting Path Algorithm

- Code



```cpp
// Augumenting Path
bool DFS(int now)
{
    int i,next;

    // Try all vetices on the right side
    for(i=0;i<(int)edg[now].size();++i)
    {
        next=edg[now][i];

        // Visit once only
        if(!used[next])
        {
            used[next]=true;

            // Unmatched vertex, or augmenting path found
            if(!rlink[next]||DFS(rlink[next]))
            {
                // Update matching
                llink[now]=next;
                rlink[next]=now;
                return true;
            }
        }
    }

    // No augmenting path is found
    return false;
}
```

Practice

- UVa 10080
  - 或POJ 2536

Thank you for your attention!