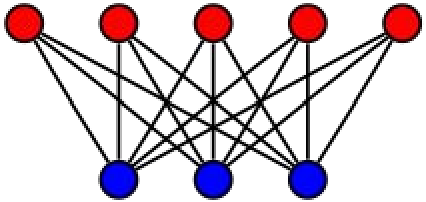
Maximum Bipartite Matching And Connected Problems

Mike Mirzayanov
Codeforces, Saratov State U

Definitions

Graph (undirected simple finite graph) is $G=(V,E)$, where $0<|V|<\infty$ and E is a set of unordered pairs from V.

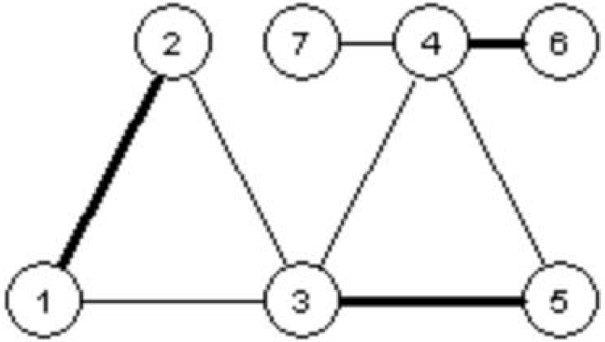
Bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets A and B (which means A and B are each independent sets) such that every edge connects a vertex in A to one in B. Vertex sets A and B are usually called the parts of the graph.



König's criterion. Graph is bipartite if and only if it does not contain an odd cycle.

Maximum Matching

Matching or **independent edge set** in a graph is a set of edges without common vertices.

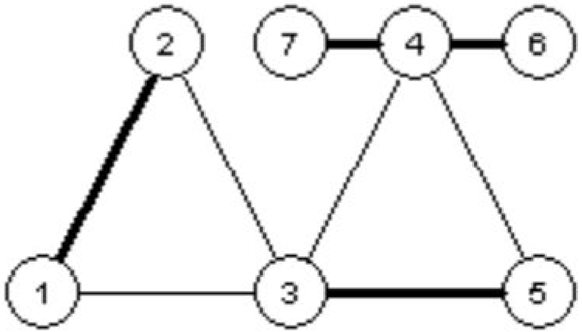


Maximum matching size is 3

The vertices 1, 2, 3, 4, 5 and 6 are **matched**.

Minimum Edge Cover

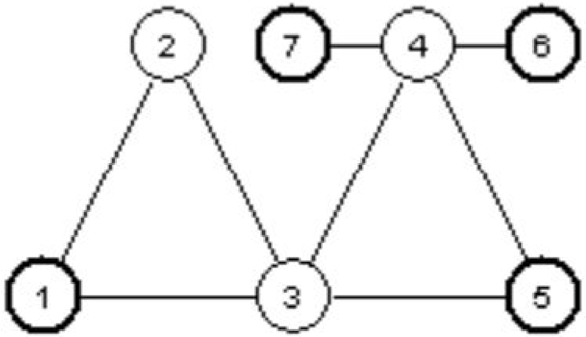
Edge cover of a graph is a set of edges such that every vertex of the graph is incident to at least one edge of the set.



Minimum edge cover size is 4.

Maximum Independent Vertex Set

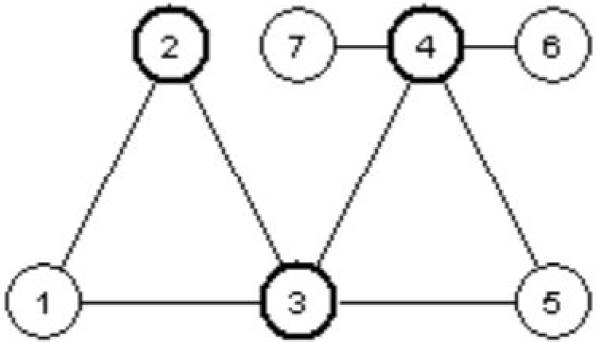
Independent set or **stable set** is a set of vertices in a graph, no pair of which is connected.



Maximum independent vertex set size is 4.

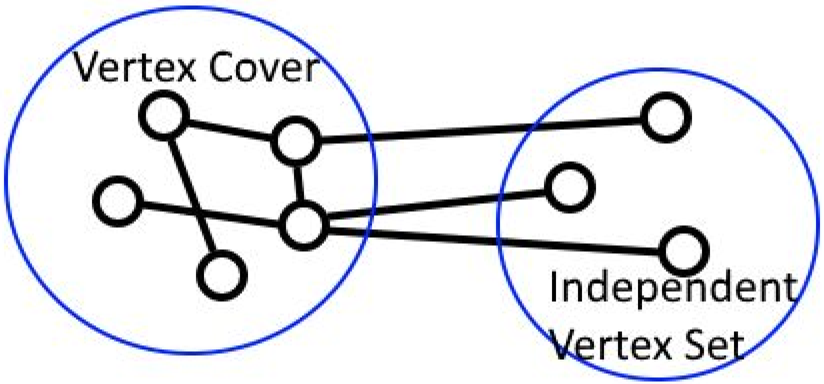
Minimum Vertex Cover

Vertex cover (sometimes node cover) of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set.



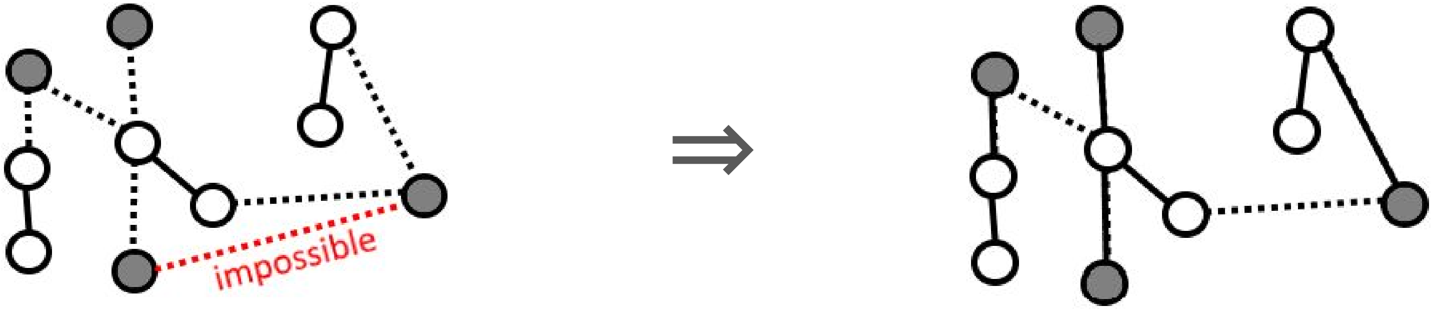
Minimum vertex cover size is 3.

MVC is complement of MIVS and vice versa



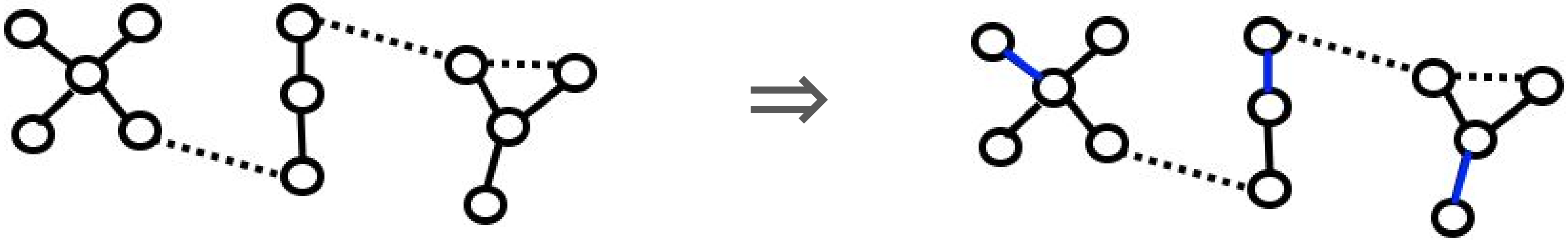
$$|MVC|+|MIVS|=n$$

MEC from MM



$$n\text{-}2|MM|+|MM|=n\text{-}|MM| \Rightarrow |MEC| \leq n\text{-}|MM|$$

MM from MEC



$$\textit{components}=n-|MEC| \Rightarrow |MM| \geq n-|MEC|$$

MM and MEC connection

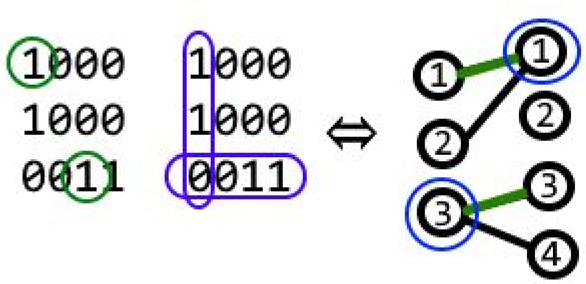
$$|MEC| \leq n-|MM| \text{ and } |MM| \geq n-|MEC|$$

$$\Leftrightarrow$$

$$|MEC|+|MM|=n$$

MM and MIVS connection (bipartite only!)

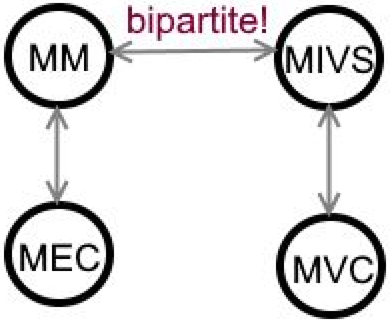
Kőnig's theorem. In a matrix of zeros and ones, the maximum number of ones, no two in a line (row or column), equals the minimum number of lines needed to cover all the ones.



In bipartite graphs

$$|MM| = |MIVS|$$

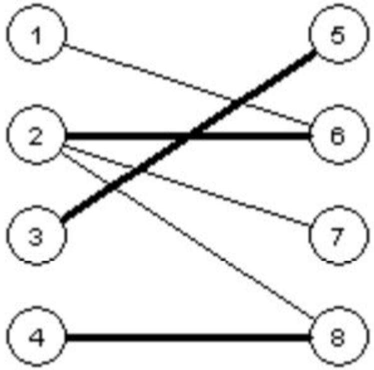
Connection of 4 problems



In practice, solve MM at first, find MEC, MIVS, MVC using MM.

Augmenting Paths

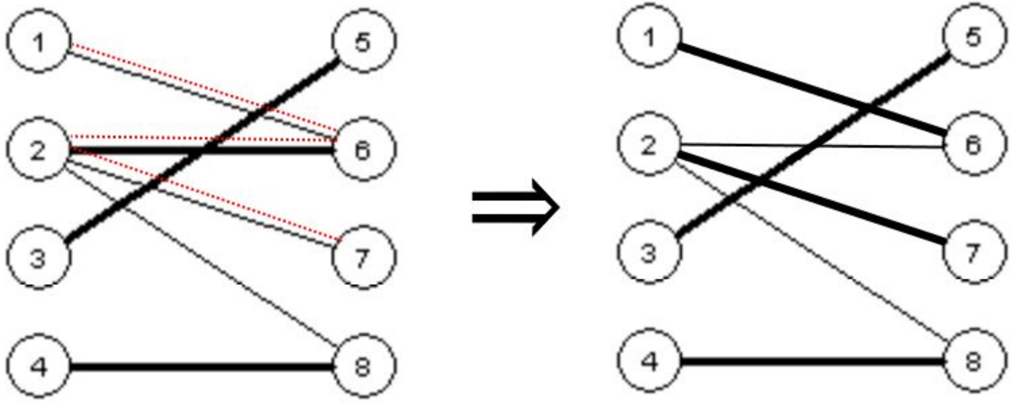
An **augmenting path** starts with unmatched vertex, ends with unmatched vertex and alternates relatively to matching *M*.



Example: P=(1,6,2,7)

Augment Matching

$M := M \oplus P$, where $\oplus$ is a symmetric difference



Berge's lemma

A matching M in a graph G is maximum if and only if there is no augmenting path with M.

Generic algorithm

$M := \varnothing$

while augmenting path P exists

$\quad M := M \oplus P$

MM Algorithm

Naive Approach

Use DFS/BFS on each iteration to find *P*, increment *M* using *P*.

Kuhn Algorithm

Find and increment in a single DFS. Kuhn algorithm has interesting properties.

Kuhn Algorithm

<table>
<tr>
<td>

```
function maxMatching(G)
  M := ∅
  for a ∈ A do
    fill(used, FALSE)
    tryKhun(a)
  return M
end
```

M:B⟶A, $M[y]$=-1 if y is unmatched,
$M[y]=x$ if $(x,y)\in M$

</td>
<td>

```
function tryKuhn(a)
  if used[a] then
    return FALSE
  used[a] := TRUE
  for b is neighbour(a)
    if (M[b] == -1) or
tryKhun(M[b]) then
      M[b]:=a
      return TRUE
  return FALSE
end
```

</td>
</tr>
</table>

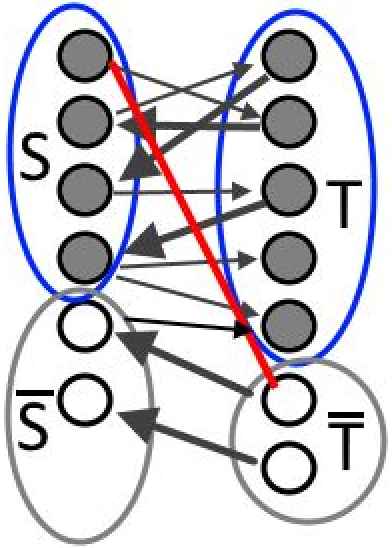
Kuhn Algorithm Properties

- Works in O(|*A*|^3+|B|), where *A* is the smallest part.
- After the *i*-th iteration of **for** in maxMatching *M* is *MM* for graph containing the first *i* vertices from *A* and all the vertices from *B.*
- A vertex matched once becomes matched forever.
- A vertex *a* from *A* unmatched after root call tryKuhn(*a*) stays unmatched forever.

Consequence

- If vertices in *A* have weights and the goal is to minimize the total weight of matched with MM vertices in *A*: sort vertices in *A* by weight and use Kuhn Algorithm.

Minimal Vertex Cover by Maximum Matching

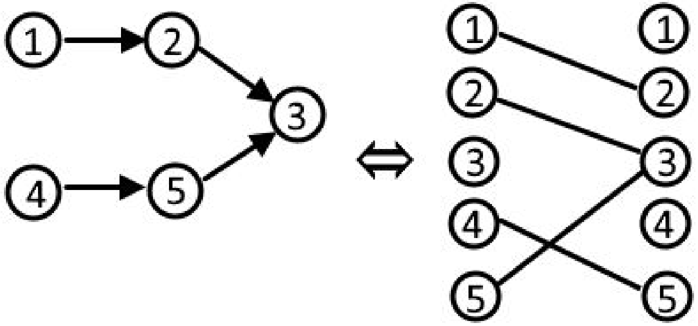


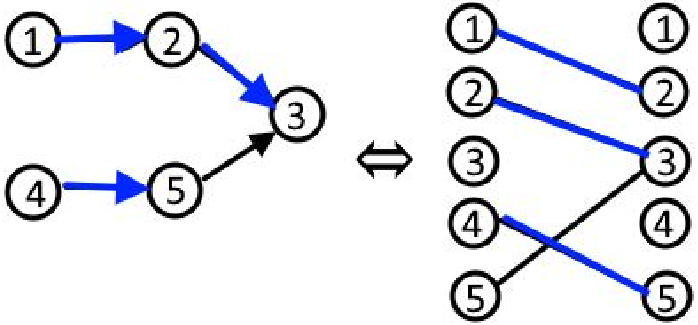
- *S* is vertices of *A* marked by DFS from unmatched vertices of *A.*
- *T* is vertices of *B* marked by DFS from unmatched vertices of *A.*

$$MVC = \overline{S} \cup T$$

Cover DAG by Minimal Number of Paths

Bijection:





A matching ⇒ each vertex in directed graph
has indegree at most 1,
outdegree at most 1.

So matching ⇒ cycles and paths, but it was a DAG ⇒ paths.

MM Algorithm Speedup Heuristics (1)

1. Greedy Initialization

```
M := maximumMatchingGreedy(G)
for a ∈ A do
    fill(used, FALSE)
    if M[a]==-1 then
        tryKhun(a)
return M
----
```

Algorithm is not the Kuhn Algorithm

MM Algorithm Speedup Heuristics (2)

2. Loops Split

```
function tryKuhn(a)
  if used[a] then return FALSE
  used[a] := TRUE
  for b is neighbour(a)
    if M[b] == -1 then { M[b]:=a; return TRUE }
  for b is neighbour(a)
    if tryKhun(M[b]) then { M[b]:=a; return TRUE }
  return FALSE
end
```

Questions

Any questions?