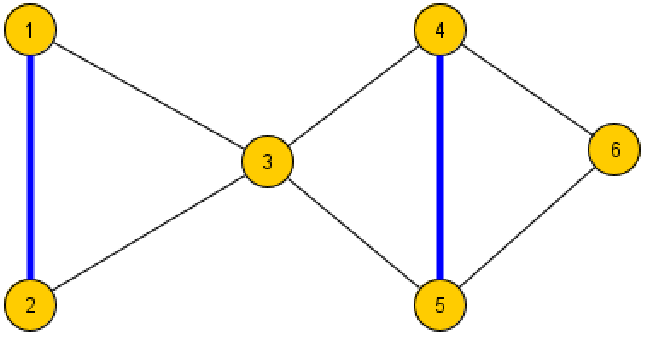
Matchings

Definitions

A matching (паросочетание) M in G is a subset of the edges $M \subseteq E$ such that each node appears in at most one edge in M.

If M is a matching, the two ends of each edge of M are said to be matched under M, and each vertex incident with an edge of M is said to be covered by M.

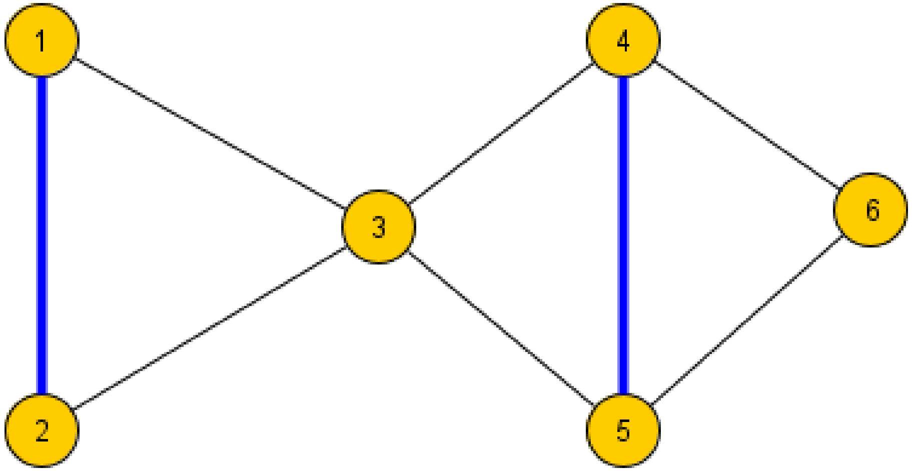


$M = \{\{1, 2\}, \{4, 5\}\}$.

Vertices 1, 2, 4, 5 are covered by M.

Maximum Matchings

A *perfect matching* *(совершенное паросочетание)* is one which covers every vertex of the graph, a *maximum matching* *(наибольшее паросочетание)* one which covers as many vertices as possible.

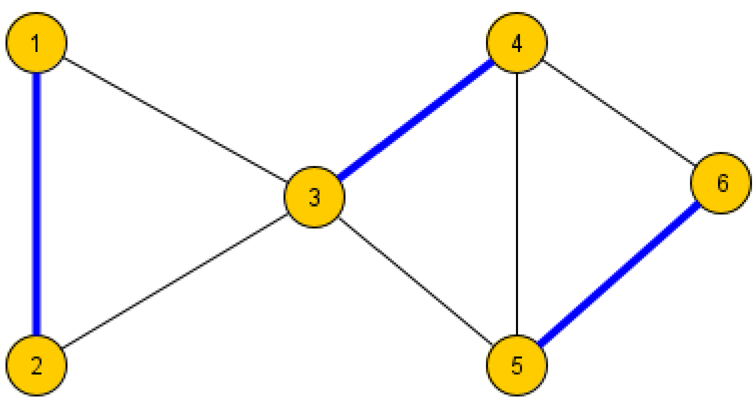


Vertices 3 and 6 are not covered by M_1.

So, M_1 is not a perfect matching.

However, M_2 = {{1, 2}, {3, 4}, {5, 6}} is a perfect matching.

Remarks.
Every perfect matching is a maximum matching.
However, the converse is false.
If G has perfect matching, then $|V|$ is even.
However, the converse is false;
for example, $K_{1,3}$ and $K_{2,4}$ do not have perfect matchings.



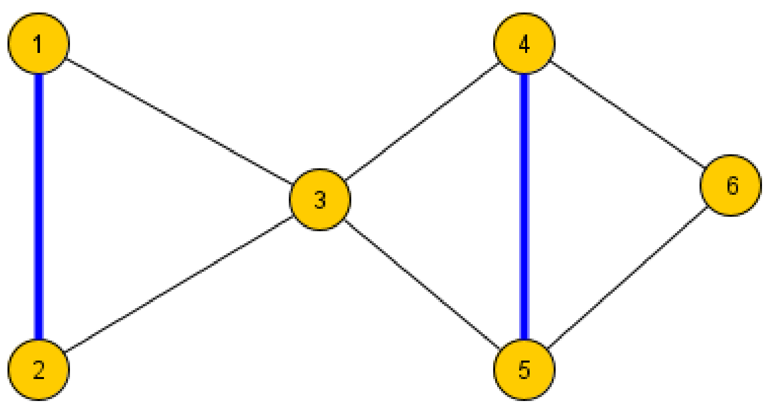
A graph is *matchable* if it has a perfect matching.

Not all graphs are matchable.

Indeed, no graph of odd order can have a perfect matching, because every matching clearly covers an even number of vertices.

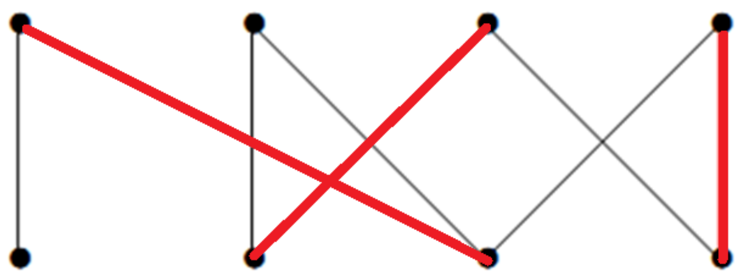
A *maximal matching* *(максимальное паросочетание)* is one which cannot be extended to a larger matching.

Equivalently, it is one which may be obtained by choosing edges in a greedy fashion until no further edge can be incorporated.

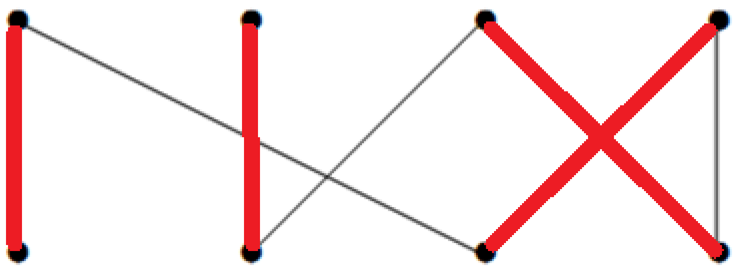


Example 2. Maximal and maximum matching in a bipartite graph.

Given a bipartite graph, it is easy to find a maximal matching, that is, one that cannot be made larger simply by adding an edge: just choose edges that do not share endpoints until this is no longer possible



(a) maximal matching



(b) perfect matching

Problem 1 The Maximum Matching Problem

Given*: a graph G,*

Find*: a maximum matching M* in G.*

There are many questions of practical interest which, when translated into the language of graph theory, amount to finding a maximum matching in a graph.

Problem 2 The Assignment Problem

A certain number of jobs are available to be filled. Given a group of applicants for these jobs, fill as many of them as possible, assigning applicants only to jobs for which they are qualified.

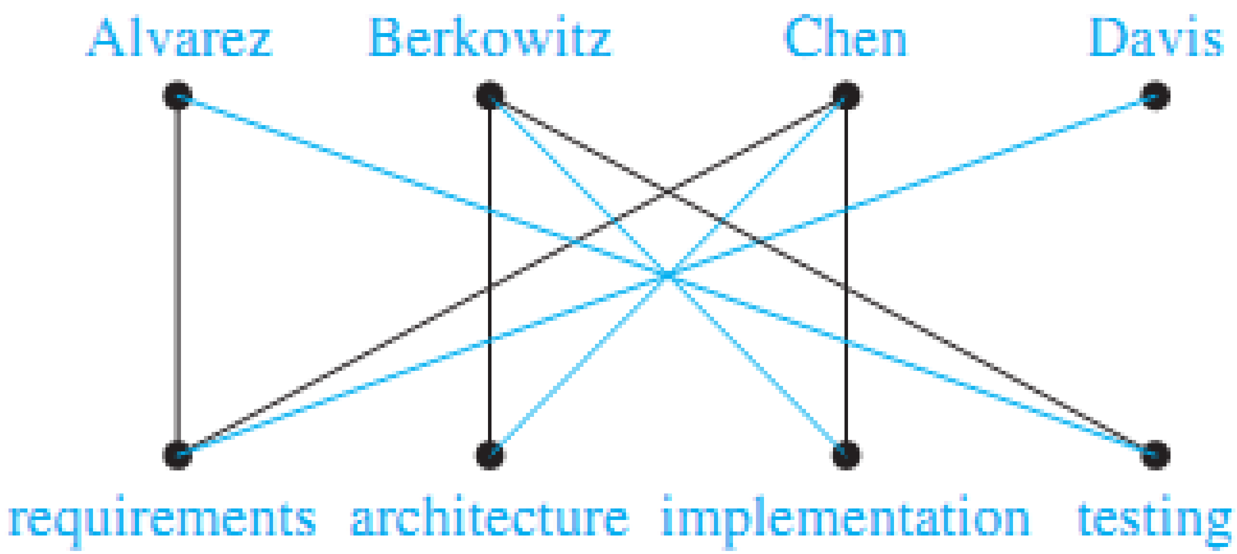
This situation can be represented by means of a bipartite graph $G[X, Y]$ in which X represents the set of applicants, Y the set of jobs, and an edge xy with $x \in X$ and $y \in Y$ signifies that applicant x is qualified for job y.

An assignment of applicants to jobs, one person per job, corresponds to a matching in G, and the problem of filling as many vacancies as possible amounts to finding a maximum matching in G.

EXAMPLE 3 A group has 4 employees: Alvarez, Berkowitz, Chen, and Davis; and suppose that 4 jobs need to be done to complete Project 1: requirements, architecture, implementation, and testing.

Suppose that Alvarez has been trained to do requirements and testing; Berkowitz has been trained to do architecture, implementation, and testing; Chen has been trained to do requirements, architecture, and implementation; and Davis has only been trained to do requirements.

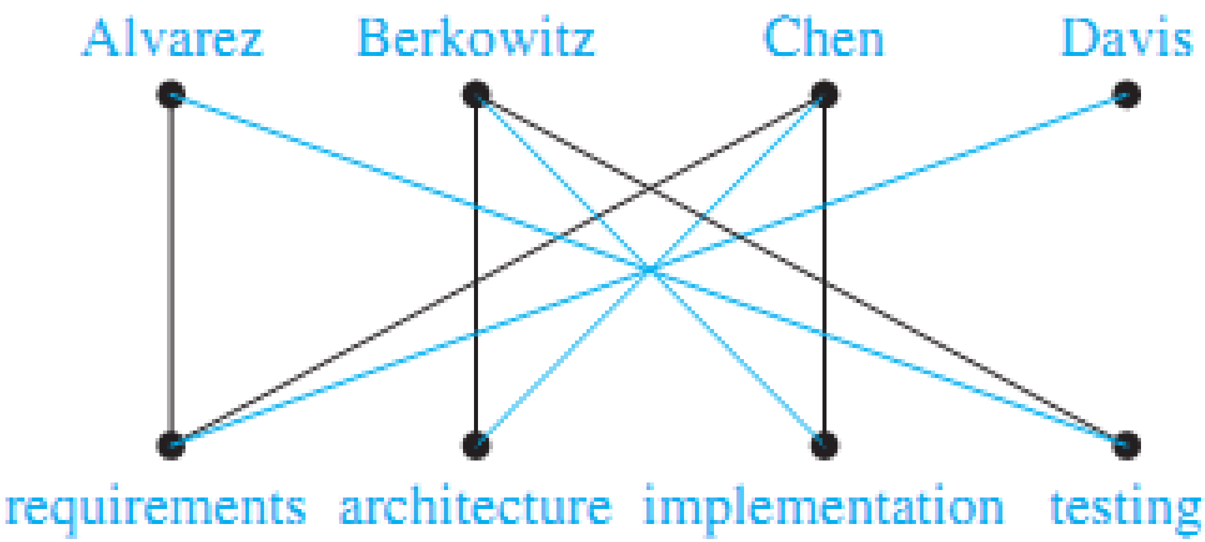
We model these employee capabilities using the bipartite graph bellow.



(a)

To complete Project 1, we must assign an employee to each job so that every job has an employee assigned to it, and so that no employee is assigned more than one job.

We can do this by assigning Alvarez to testing, Berkowitz to implementation, Chen to architecture, and Davis to requirements, as shown bellow (blue lines show this assignment of jobs).



(a)

Complete matching

We say that a matching M in a bipartite graph $G = (V, E)$ with bipartition (X, Y) is a complete matching from X to Y if every vertex in X is the endpoint of an edge in the matching, or equivalently, if $|M|=|X|$.

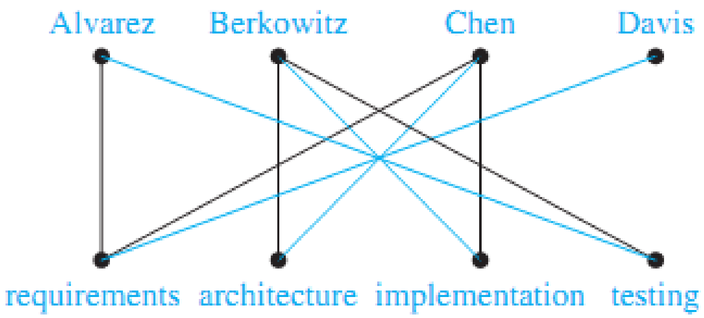
For example, to assign jobs to employees so that the largest number of jobs are assigned employees, we seek a maximum matching in the graph that models employee capabilities.

To assign employees to all jobs we seek a complete matching from the set of jobs to the set of employees.

In Example 3, we found a complete matching from the set of jobs to the set of employees for Project 1, and this matching is a maximum matching.

We now give an example of how matchings can be used to model marriages.

X = {requirements, architecture, implementation, testing}



(a)

EXAMPLE 4 Marriages on an Island

Suppose that there are m men and n women on an island.

Each person has a list of members of the opposite gender acceptable as a spouse.

We construct a bipartite graph $G = (X, Y)$ where X is the set of men and Y is the set of women so that there is an edge between a man and a woman if they find each other acceptable as a spouse.

A matching in this graph consists of a set of edges, where each pair of endpoints of an edge is a husband-wife pair.

A maximum matching is a largest possible set of married couples, and a complete matching of X is a set of married couples where every man is married, but possibly not all women.

Suppose that a group has second group also has 4 employees: Washington, Xuan, Ybarra, and Ziegler; and suppose that the same 4 jobs need to be done to complete Project 2.

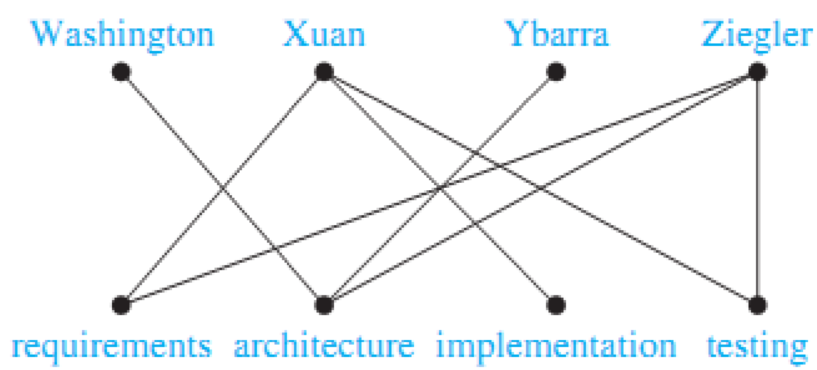
Suppose that Washington has been trained to do architecture; Xuan has been trained to do requirements, implementation, and testing;

Ybarra has been trained to do architecture;

Ziegler has been trained to do requirements, architecture and testing.

We model these employee capabilities using the bipartite graph bellow.

X = {requirements, architecture, implementation, testing}



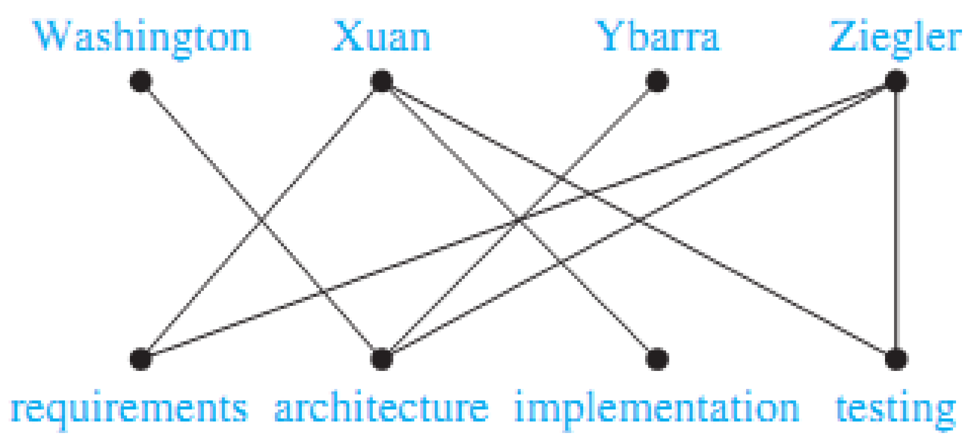
(b)

To complete Project 2, we must also assign an employee to each job so that every job has an employee assigned to it and no employee is assigned more than one job.

However, this is impossible because there are only 2 employees, Xuan and Ziegler, who have been trained for at least one of the 3 jobs of requirements, implementation, and testing.

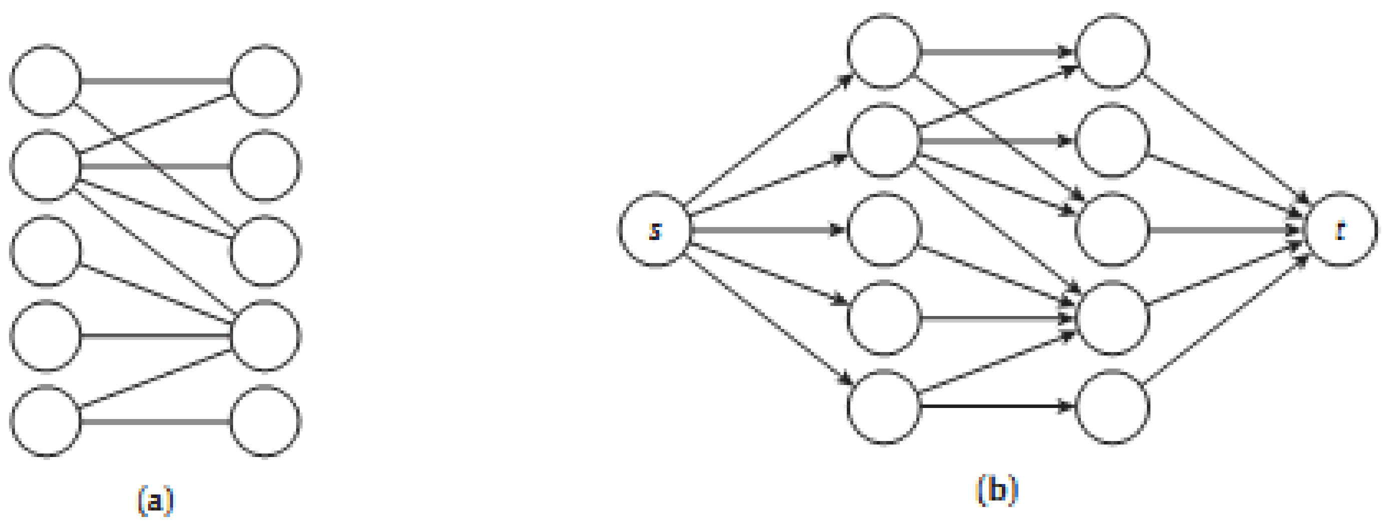
Consequently, there is no way to assign 3 different employees to these 3 job so that each job is assigned an employee with the appropriate training.

X = {requirements, architecture, implementation, testing}

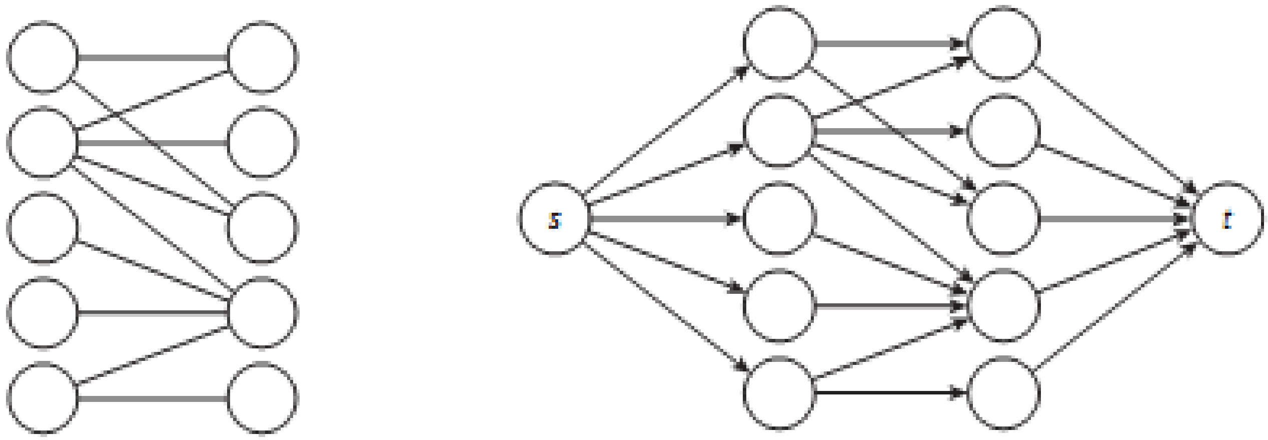


Using an algorithm for the Maximum- Flow Problem to find a maximum matching.

Beginning with the graph *G* in an instance of the Bipartite Matching Problem, we construct a flow network *G'* as shown bellow.



(a) A bipartite graph *G*. (b) The corresponding flow network *G'*, with all capacities equal to 1.



(a) (b)

1) direct all edges in *G* from *X* to *Y*.
2) add a node *s*, and an edge (*s, x*) from *s* to each node in *X*.
3) add a node *t*, and an edge (*y, t*) from each node in *Y* to *t*.
4) give each edge in *G'* a capacity of 1.
5) compute a maximum *s*-*t* flow in this network *G'*.

Analyzing the Algorithm

We will show that the value of this maximum flow in *G'* is equal to the size of the maximum matching in *G*.

Moreover, our analysis will show how one can use the flow itself to recover the matching.

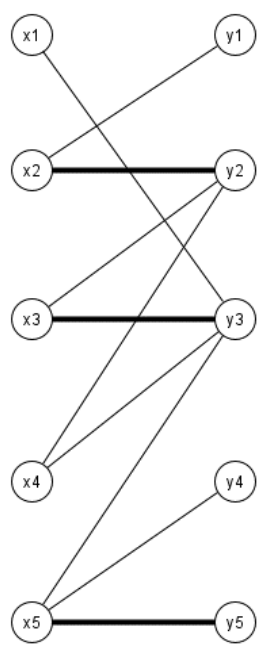
The analysis is based on showing that integer-valued flows in *G'* encode matchings in *G* in a fairly transparent fashion.

1) Suppose there is a matching M in G consisting of k edges $(x_{i1}, y_{i1}), \ldots, (x_{ik}, y_{ik})$. $|M| = k$

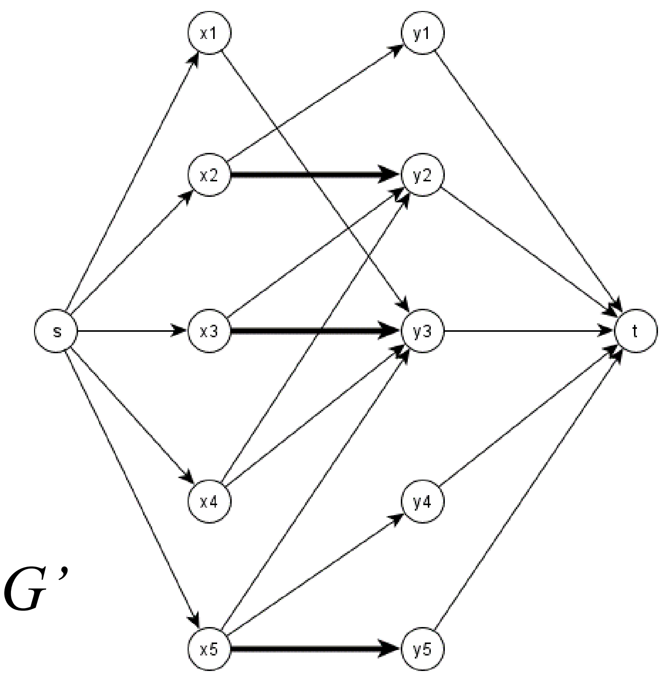
Then consider the flow f that sends 1 unit along each path of the form <s -> x_{ij}-> y_{ij}-> t> .

That is, $f(e) = 1$ for each edge on one of these paths.

One can verify easily that the capacity and conservation conditions are indeed met and that f is an s-t flow of value k.



G



G'

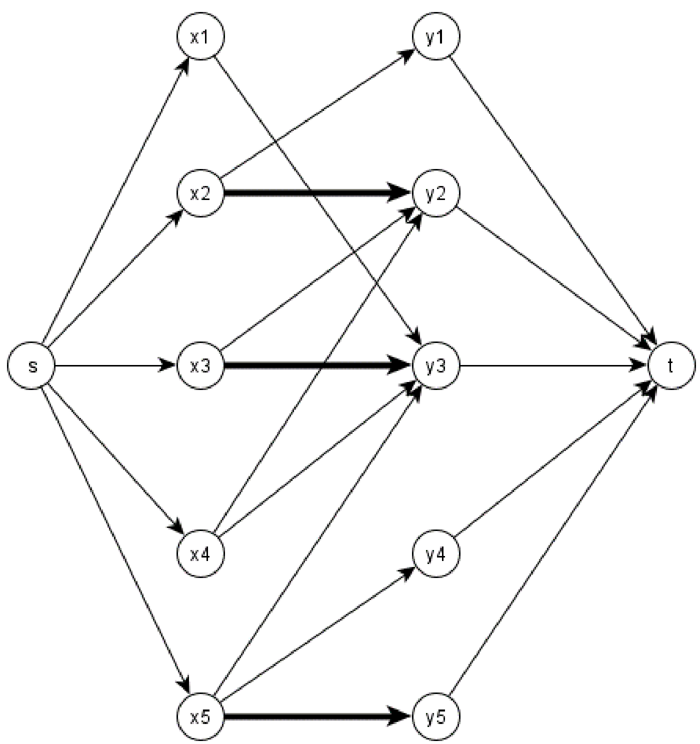
2) Conversely, suppose there is a flow *f* in *G'* of value *k*.

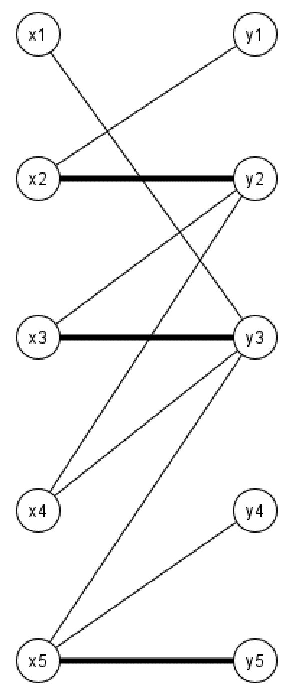
We know there is an integer-valued flow *f* of value *k*;

and since all capacities are 1, this means that *f*(*e*) is equal to either 0 or 1 for each edge *e*.

Now, consider the set *M* of edges of the form (*x, y*) on which the flow value is 1. We will show that M is a matching

We will prove 3 simple facts about the set *M*.





Lemma 1 M contains k edges.

Proof. To prove this, consider the cut (A, B) in G' with

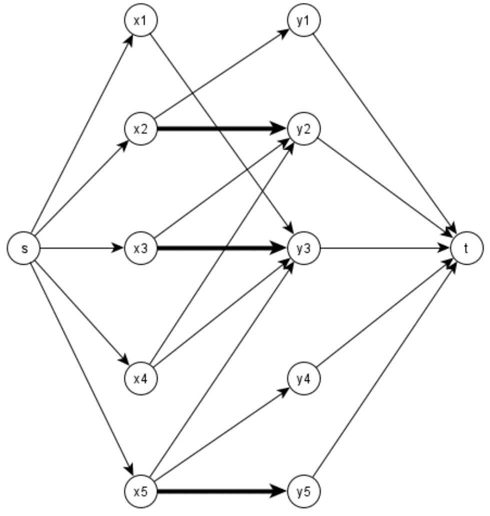
$A = \{s\} \cup X$.

The value of the flow is the total flow leaving A, minus the total flow entering A.

The first of these terms is simply $|M|$, since these are the edges leaving A that carry flow, and each carries exactly 1 unit of flow.

The second of these terms is 0, since there are no edges entering A.

Thus, M contains k edges.



Lemma 2 Each node in X is the tail ≤ 1 edge in M.

Proof. To prove this, suppose $x \in X$ were the tail of ≥ 2 edges in M.

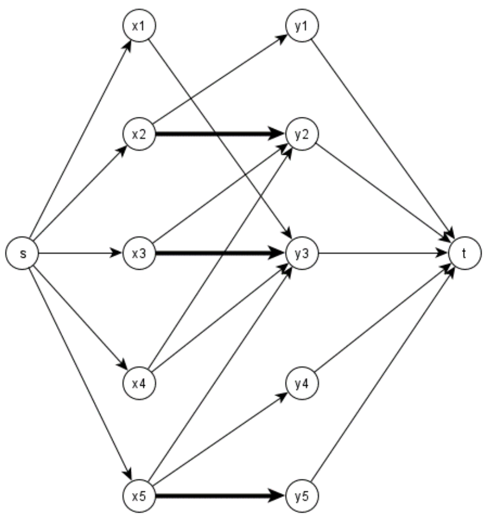
Since our flow is integer-valued, this means that ≥ 2 units of flow leave from x.

By conservation of flow, ≥ 2 units of flow would have to come into x—but this is not possible, since only a single edge of capacity 1 enters x.

Thus x is the tail of ≤ 1 edge in M.

By the same reasoning, we can show

Lemma 3 Each node in Y is the head of ≤ 1 edge in M.



Combining these facts, we see that if we view M as a set of edges in the original bipartite graph G, we get a matching of size k.

In summary, we have proved the following fact.

Lemma 4 The size of the maximum matching in G is equal to the value of the maximum flow in G';

the edges in such a matching in G are the edges that carry flow from X to Y in G'.

Bounding the Running Time

Now let's consider how quickly we can compute a maximum matching in *G*.

Let $n = |X| = |Y|$, and let m be the number of edges of *G*.

We'll assume that there is ≥ 1 edge incident to each node in the original problem, and hence $m \geq n/2$.

The time to compute a maximum matching is dominated by the time to compute an integer-valued maximum flow in *G'*, since converting this to a matching in *G* is simple.

For this flow problem, we have an estimation of the number of iterations:

$$C = \sum_{e \text{ out of } s} c_e = |X| = n$$

as *s* has an edge of capacity 1 to each node of X.

Thus, by using the $O(mC)$ bound in Lemma 5 from the Lecture on Flow Networks, we get the following Lemma:

Lemma 5 The Ford-Fulkerson Algorithm can be used to find a maximum matching in a bipartite graph in $O(mn)$ time.

The meaning of the augmenting paths in the network G'.

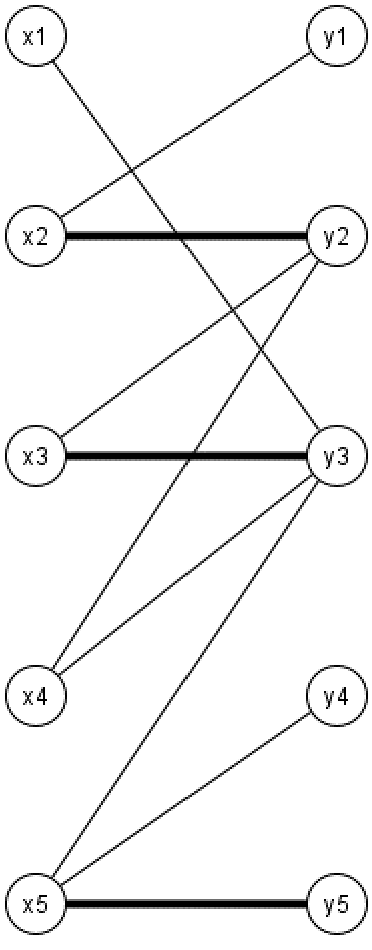
It is worthwhile to consider what the augmenting paths mean in the network G'.

Consider the matching M consisting of edges (x_2, y_2), (x_3, y_3), and (x_5, y_5) in the bipartite graph in Figure on the right.

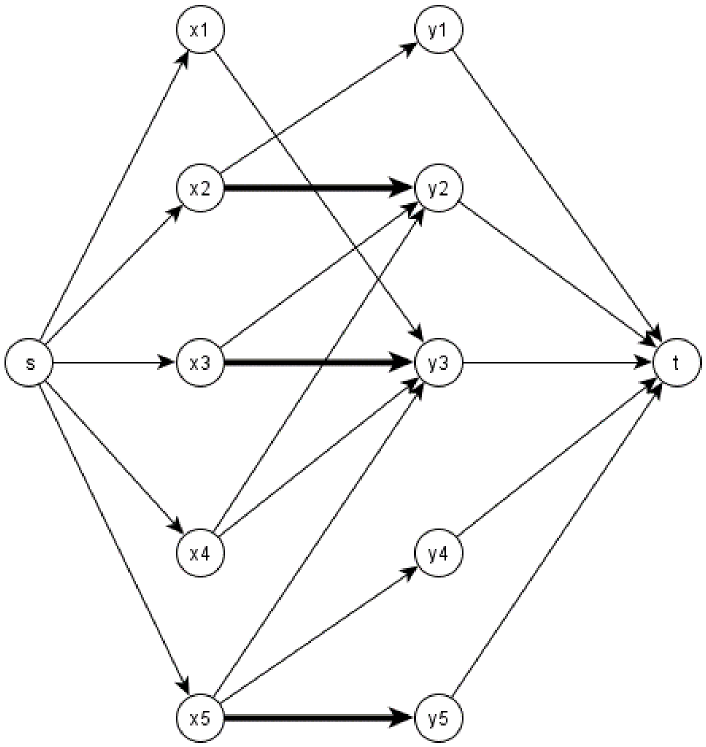
Let f be the corresponding flow in G'.

This matching is not maximum, so f is not a maximum s-t flow, and hence

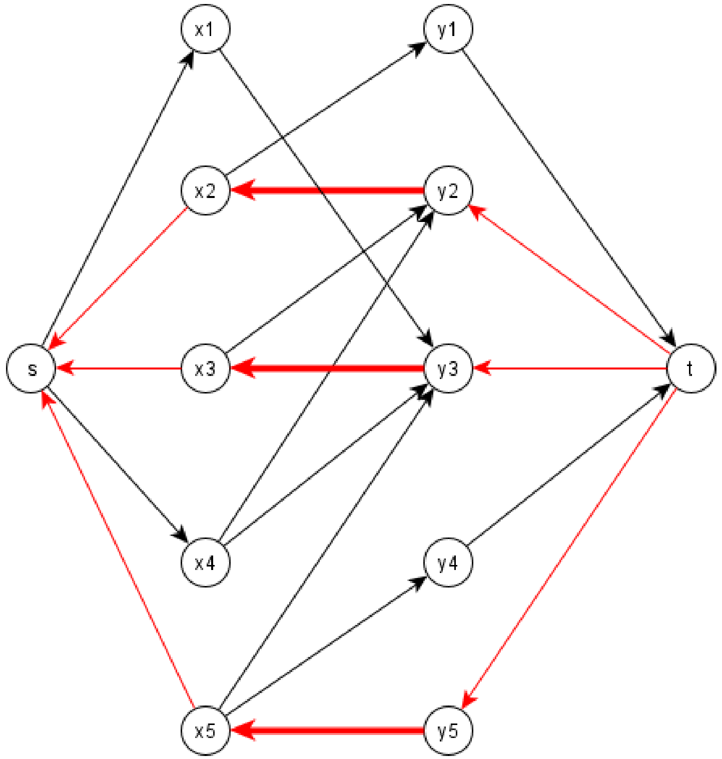
there is an augmenting path in the residual graph G'_f.



The flow network G' and the residual graph G'_f for the matching of the previous slide



G'



G'_f

One such augmenting path is shown on the right.

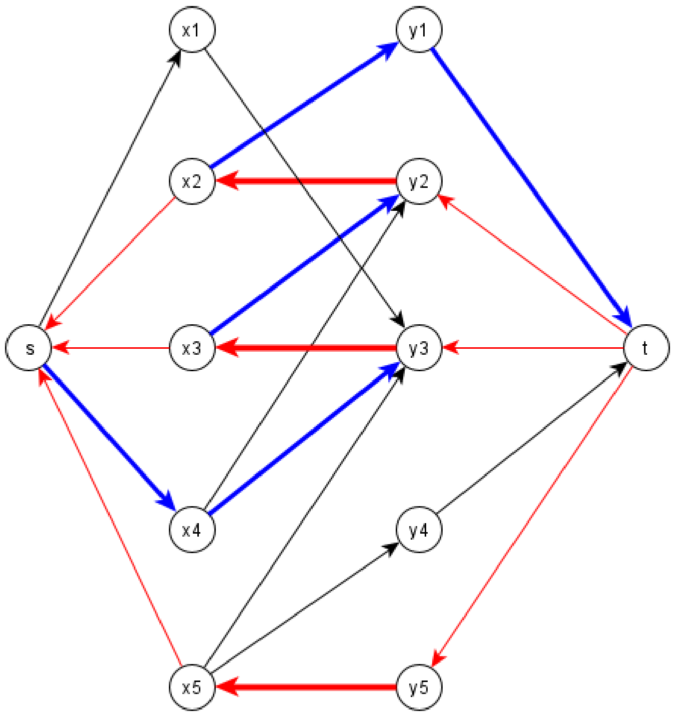
Note that the edges (x_2, y_2) and (x_3, y_3) are used backward, and all other edges are used forward.

All augmenting paths must alternate between edges used backward and forward, as all edges of the graph G' go from X to Y.

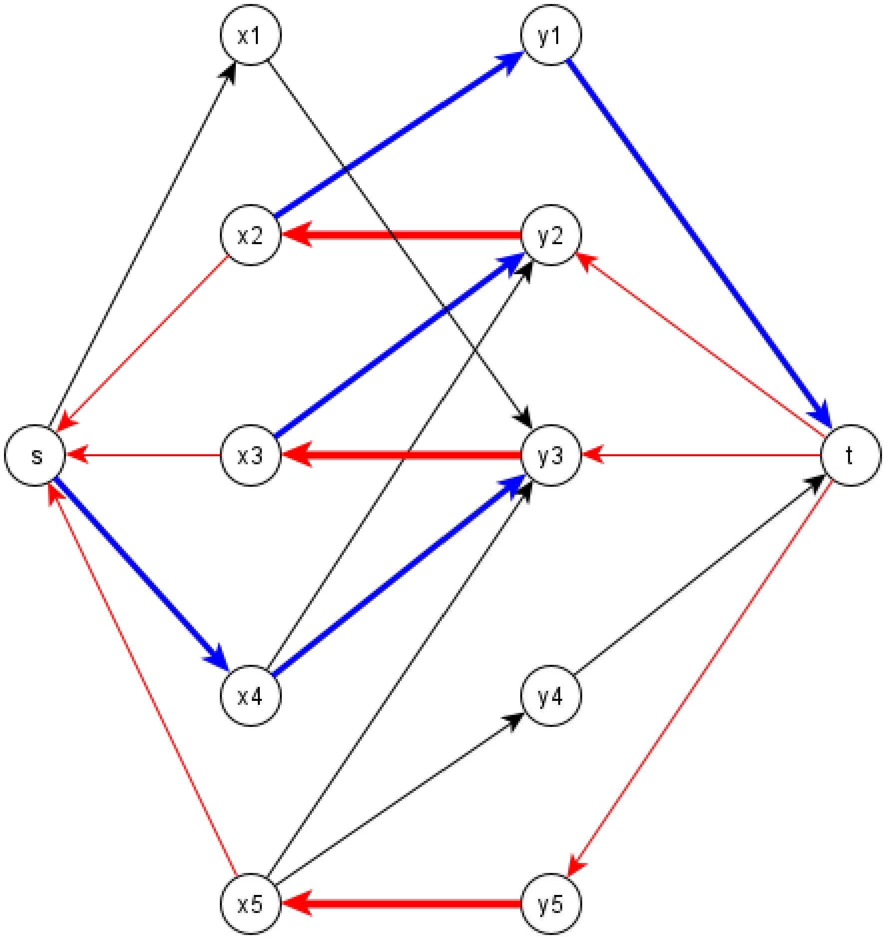
Augmenting paths are therefore also called alternating paths (чередующиеся пути, альтернирующие пути) in the context of finding a maximum matching.

The effect of this augmentation is to take the edges used backward out of the matching, and replace them with the edges going forward.

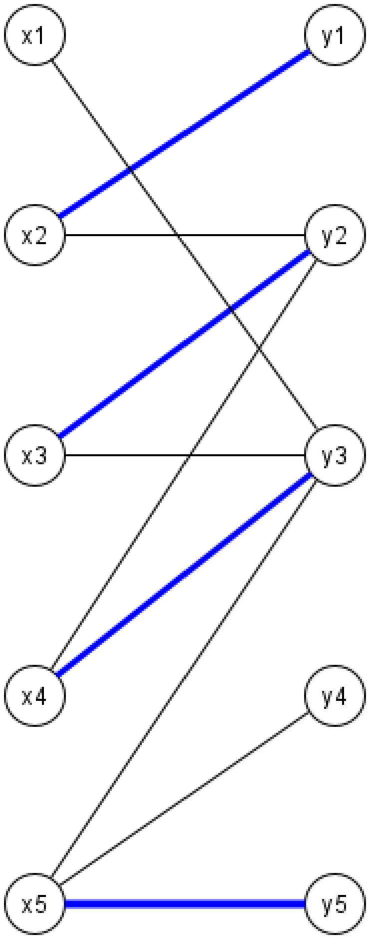
Because the augmenting path goes from s to t, there is one more forward edge than backward edge; thus the size of the matching increases by one.



G'_f



G'_f



G: $|M| = 4$

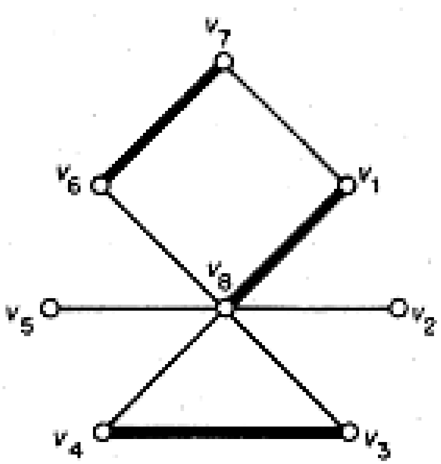
M -alternating path

Let M be a matching in G.

An M -alternating path (M- альтернирующий, M- чередующийся) in G is a path whose edges are alternately in $E \backslash M$ and M.

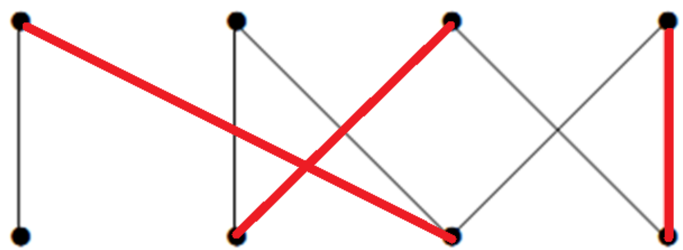
For example, the path <v_5, v_8, v_1, v_7, v_6> in the graph bellow is an M -alternating path.

An M -augmenting path is an M -alternating path whose origin and terminus are not covered by M.

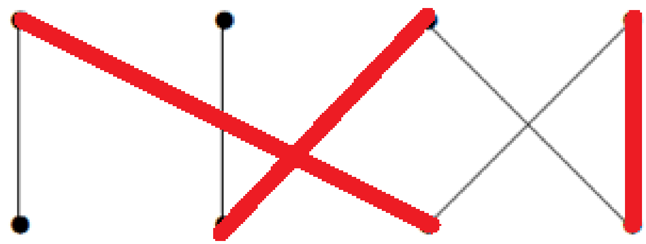


(a)

Example of an *M*-augmenting path in a bipartite graph



(a) Matching *M*



(b) An M- augmenting path

Theorem 6 (Berge, 1957) A matching M in G is a maximum matching $\Leftrightarrow G$ contains no M -augmenting path.

Proof

(=>) Let M be a matching in G, and suppose that G contains an M-augmenting path $\langle v_0, v_1, \ldots, v_{2m+1}\rangle$.

Define $M' \subseteq E$ by $M' = (M \setminus \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{2m-1}, v_{2m}\}\})$ $\cup \{\{v_0, v_1\}, \{v_2, v_3\}, \ldots, \{v_{2m}, v_{2m+1}\}\}$

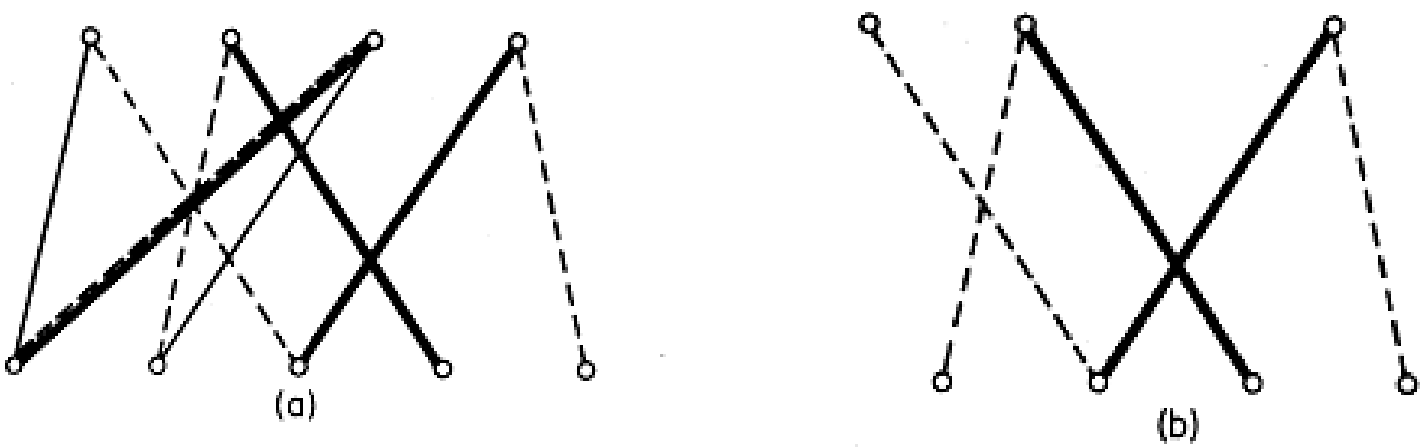
Then M' is a matching in G, and $|M'| = |M| + 1$.

=> M is not a maximum matching.

(<=) Suppose that M is not a maximum matching, and let M' be a maximum matching in G.

Then $|M'| > |M|$

Set $H = G[M \Delta M']$), where $M \Delta M'$ denotes the symmetric difference (симметрическая разность) of M and M'



(a) G with M heavy and M' broken; (b) $G[M \Delta M']$

Each vertex of H has degree either 1 or 2 in H, since it can be incident with $\leq$ 1 edge of M and 1 edge of M'.

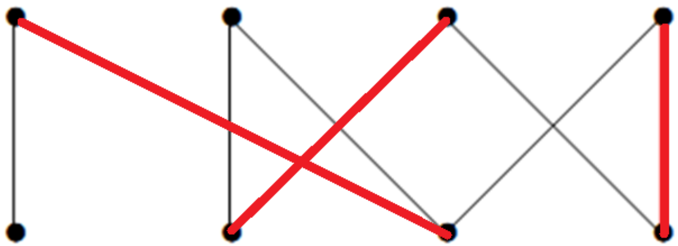
Thus each component of H is either an even cycle with edges alternately in M and M', or else a path with edges alternately in M and M'.

Since $|M'| > |M|$, H contains more edges of M' than of M, and therefore some path component P of H must start and end with edges of M'.

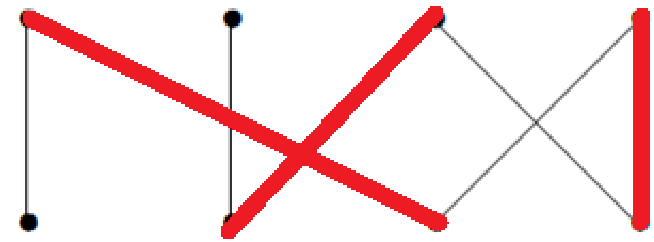
The origin and terminus of P, being covered with M' in H, are not coverred with M in G.

Thus P is an M -augmenting path in G.

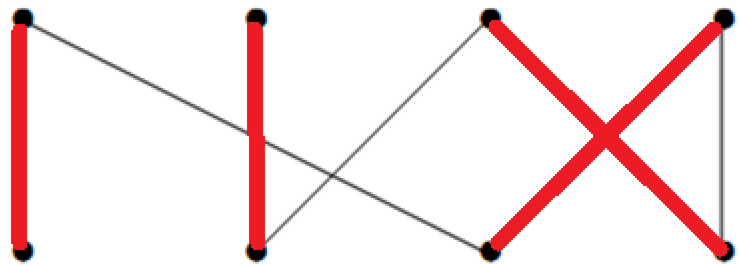
Example of using an augmenting path



(a) $|M| = 3$



(b) An augmenting path



(c) $|M'| = 4$

Complete Matchings in Bipartite Graphs

For any set S of vertices in G, we define the neighbour set of S in G to be the set of all vertices adjacent to vertices in S; this set is denoted by $N(S)$.

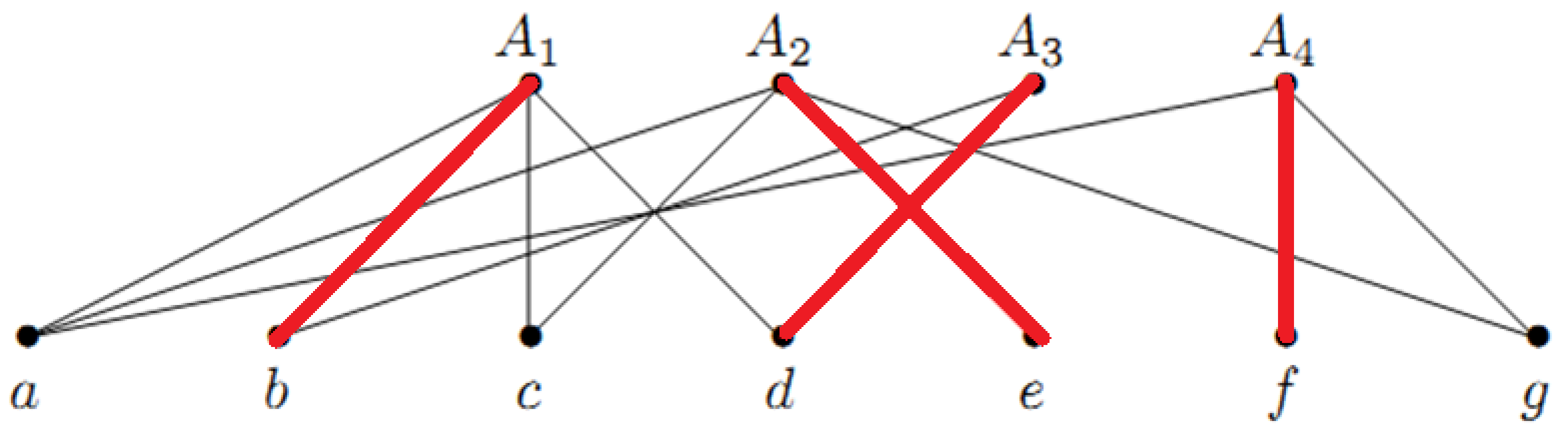
Suppose, now, that G is a bipartite graph with bipartition (X, Y).

In many applications one wishes to find a *complete matching* of G that covers *every vertex* in X.

Necessary and sufficient conditions for the existence of such a matching were first given by Hall (1935).

EXAMPLE of a complete matching

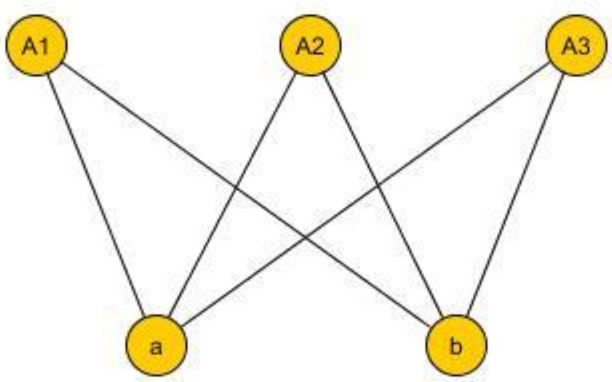
This is a largest possible matching, since it contains edges incident with all 4 of the top vertices, and it thus corresponds to a complete matching.



$N(A1) = \{a, b\}$, $N(A2) = \{a, b\}$, $N(A3) = \{a, b\}$.

$|N(X)| = 2$, while $|X| = 3$.

so a complete matching cannot be found.



Necessary and sufficient conditions for Complete Matchings

Theorem 7 (Hall, 1935) Let G be a bipartite graph with bipartition (X, Y). Then G contains a matching that covers every vertex in X ⇔

$$|N(S)| \geq |S| \text{ for all } S \subseteq X \qquad (2)$$

Proof

(=>) Suppose that G contains a matching M which covers every vertex in X, and let S be a subset of X.

Since the vertices in S are matched under M with distinct vertices in $N(S)$, we clearly have

$$|N(S)| \geq |S|.$$

(<=) Suppose that G is a bipartite graph satisfying (2), but that G contains no matching covering all the vertices in X.

We shall obtain a contradiction.

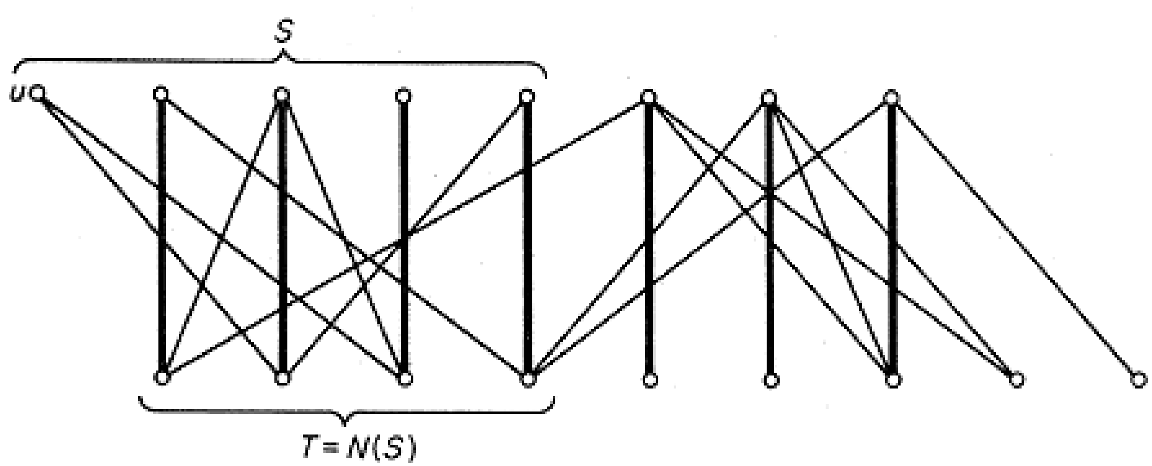
Let M^* be a maximum matching in G.

By our supposition, M^* does not cover all vertices in X.

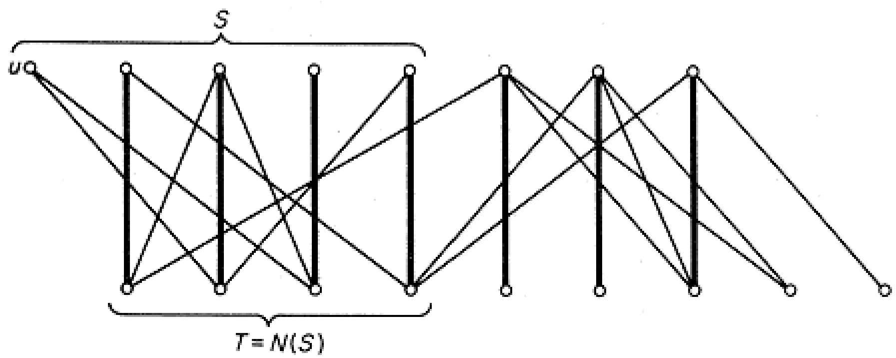
Let u be a vertex that is not covered with M^* in X, and let Z denote the set of all vertices connected to u by M^*-alternating paths.

Since M^* is a maximum matching, it follows from theorem 6 that u is the only M^*- unsaturated vertex in Z.

Set $S = Z \cap X$ and $T = Z \cap Y$ (see bellow).



Clearly, the vertices in $S \setminus \{u\}$ are matched under M^* with the vertices in T.



Therefore ·

$|T| = |S|-1$ (3)

and $N(S) \in T$.

$N(S) = T$ (4)

In fact, we have $N(S) = T$ $\qquad (4)$

since every vertex in N(S) is connected to u by an M^*-alternating path.

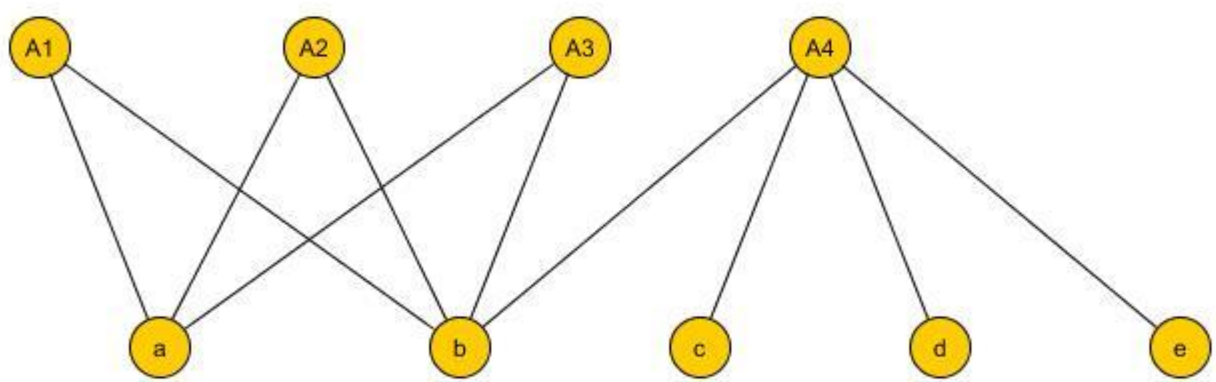
But (3) and (4) imply that $|N(S)| = |S| - 1 < |S|$ contradicting assumption (2)

EXAMPLE

$N(A1) = \{a, b\}$, $N(A2) = \{a, b\}$, $N(A3) = \{a, b\}$, $N(A4) = \{b, c, d, e\}$.

Now the total number of $N(X) = 5$, and we only need 4.

Nevertheless, complete matching impossible, because the first 3 sets have now $|N(A1, A2, A3)| = 2$.



The foregoing discussion suggests the following general scheme for designing a bipartite maximum matching algorithm.

Algorithm 1 Iterative scheme for computing a maximum matching

1: Initialize $M = \varnothing$;

2: **repeat**

3: Find an augmenting path P with respect to M.

4: $M' = M \Delta P$

5: until there is no augmenting with respect to M.

Modification of the max-flow algorithm

Given a bipartite graph $G = (X \cup Y, E)$ and matching M.

We will construct $G_{\mathrm{M}} = (V_M, E_M)$ as follows:

$V_M = X \cup Y \cup \{s, t\}$

$E_M = \{(x_i, y_{\mathrm{j}}) : (x_i, y_j) \in E\text{-}M, x_i \in X; y_j \in Y\} \cup$

$\{(y_j, x_i) : (y_j, x_i) \in M, y_j \in Y, x_{\mathrm{i}} \in X\} \cup$

$\{(s, x_l) : x_l \in X' \} \cup$

$\{(y_m, t) : y_m \in Y'\}$

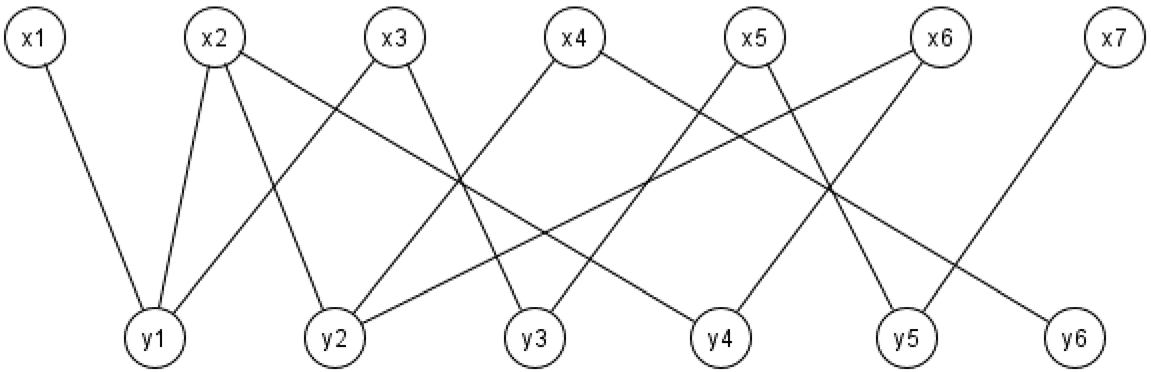
Where

$X' = \{x \in X : x$ is not covered by $M\}$;

$Y' = \{y \in Y : y$ is not covered by $M\}$:

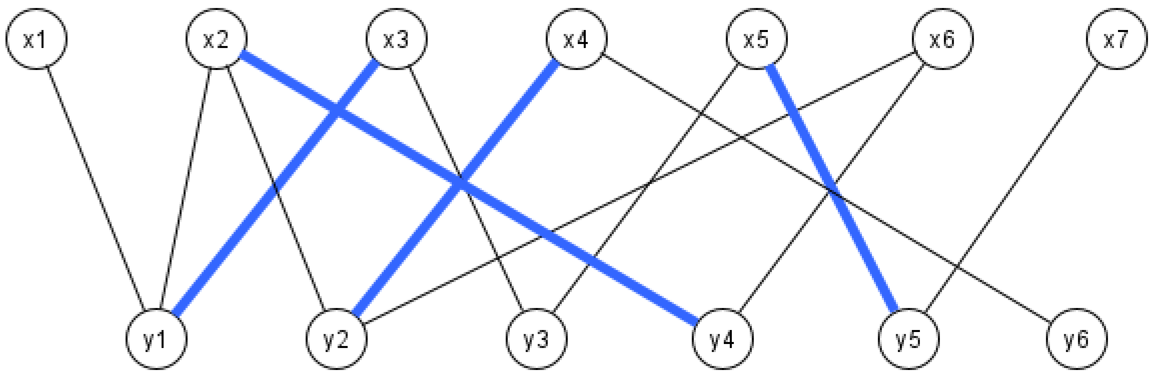
$s\sim > t$ paths in G_M correspond to the augmenting paths in G.

Example of finding the maximum matching in a bipartite graph

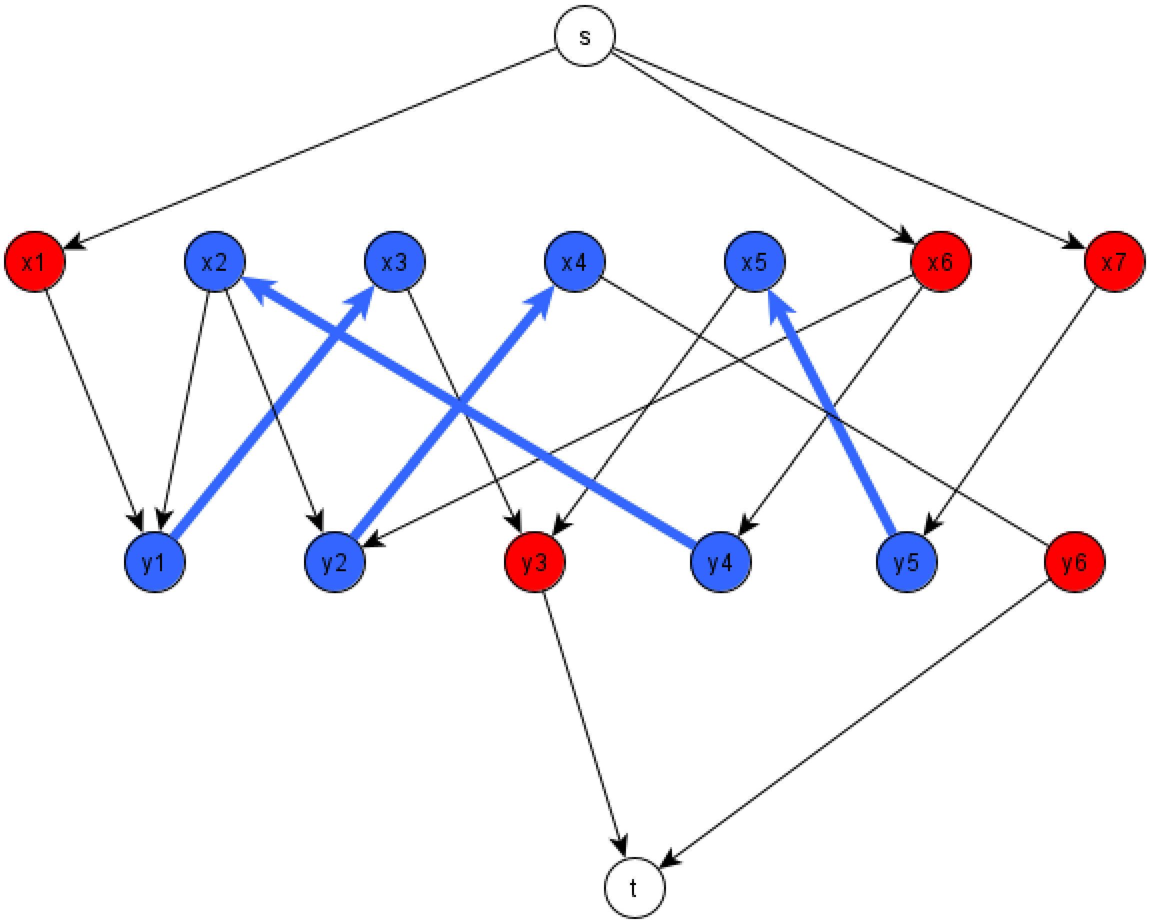


[This example is described in the Lecture by Ivan Takhonov on matchings: https://classroom.google.com/w/MjgzNjg4OTMzMjRa/tc/MzY2NTQwNDI2NTNa](https://classroom.google.com/w/MjgzNjg4OTMzMjRa/tc/MzY2NTQwNDI2NTNa)

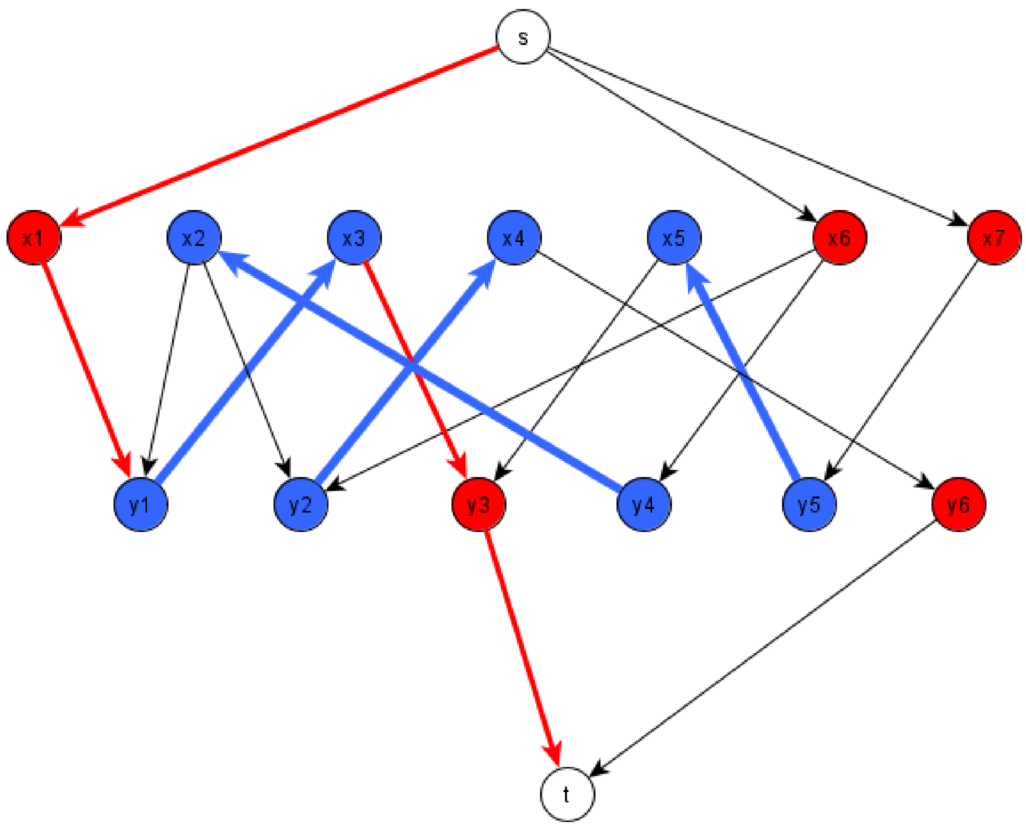
Construct some matching



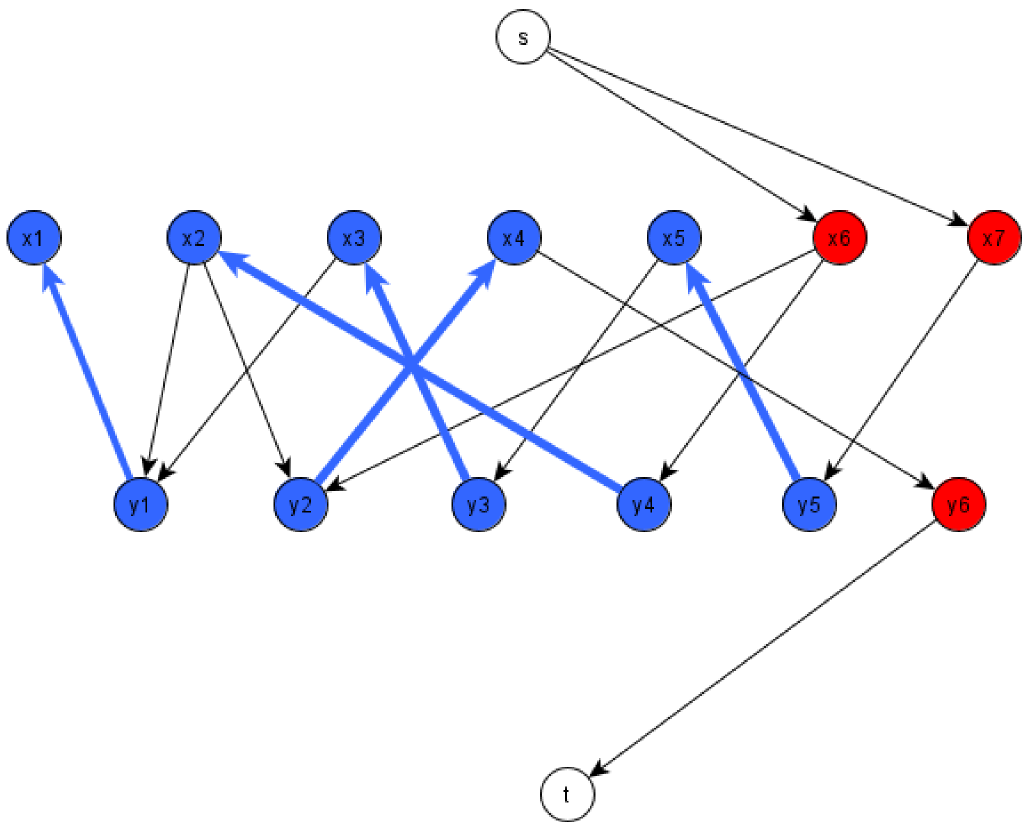
Construct an st-graph



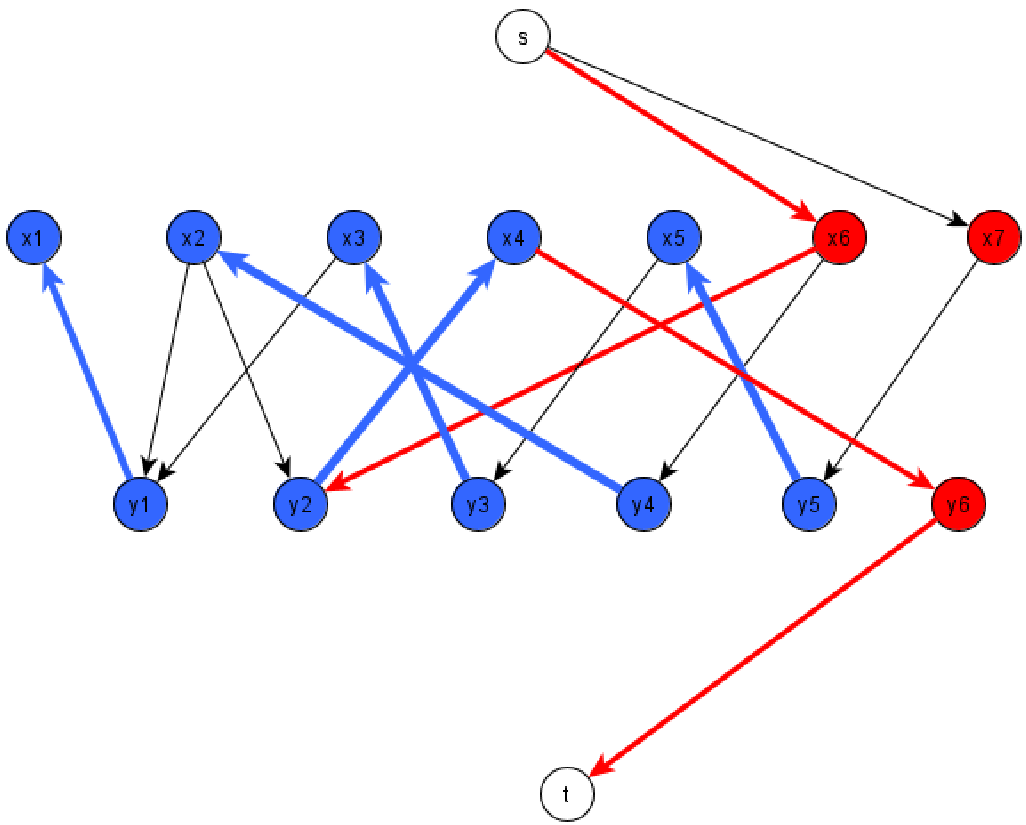
Find an agmenting path in the st-graph



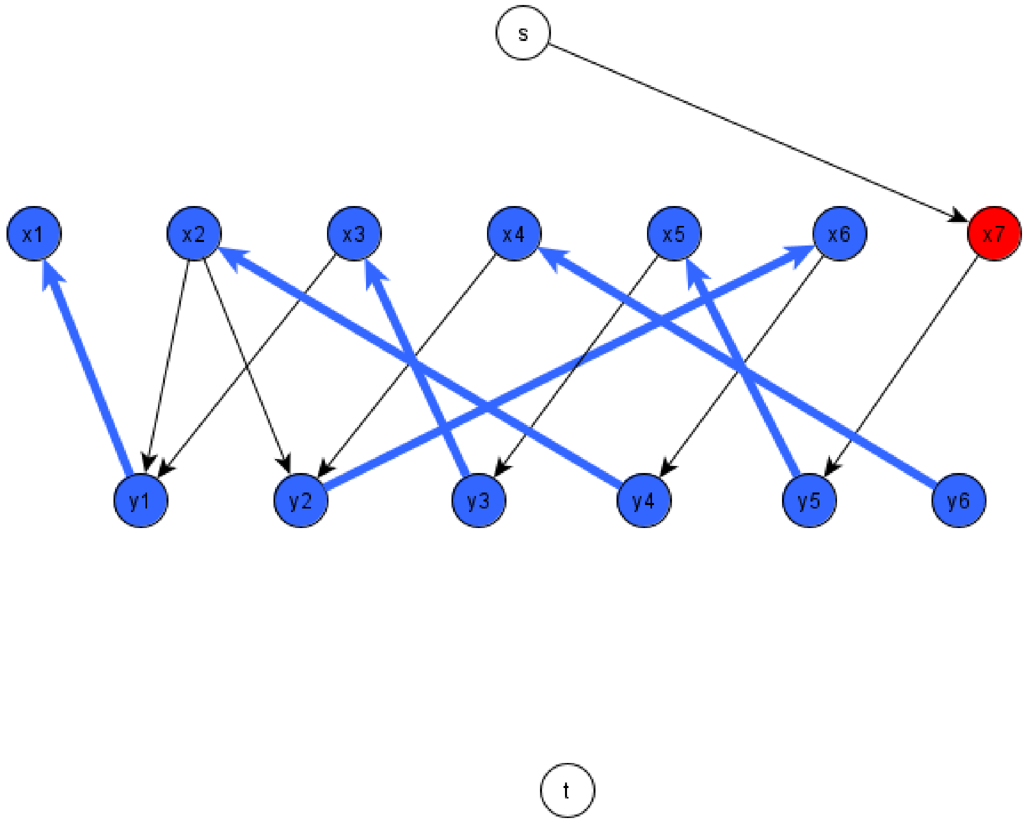
Construct a new matching and a new st-graph



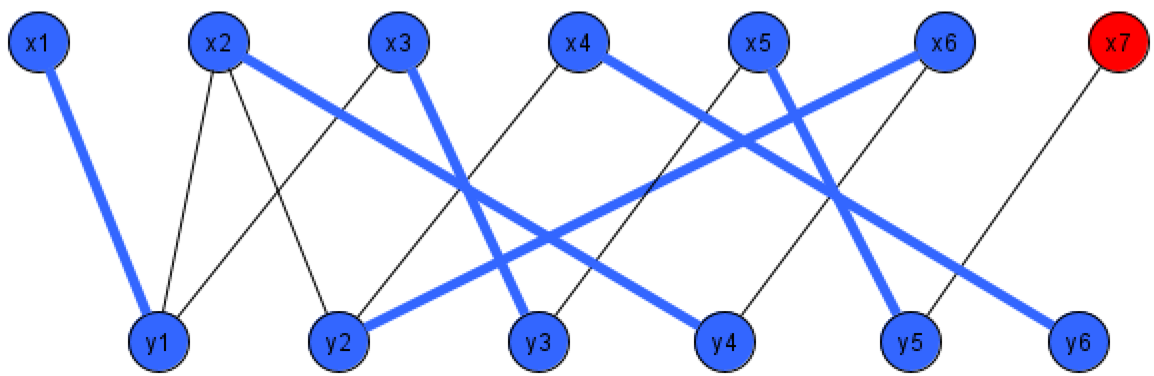
Find the next augmenting path



Construct a larger matching, now vertex t is unreachable



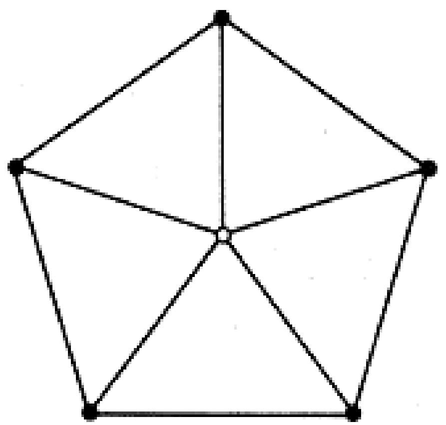
The maximum matching is found



Matchings and Coverings in Bipartite graphs

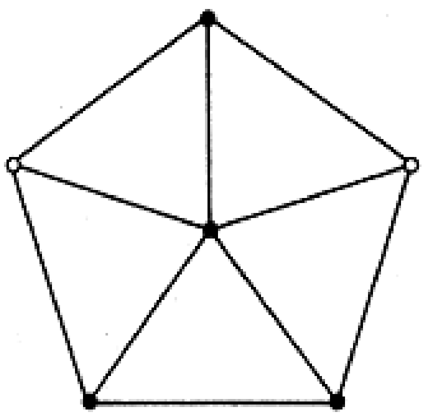
A covering, vertex covering, vertex cover (покрытие, вершинное покрытие) of a graph G is a subset K of V such that every edge of G has at least one end in K.

A covering K^ is a minimum covering if G has no covering K' with $|K'| < |K\text{^}|$.



(a)

A covering, $|K| = 5$



(b)

A minimum covering $|K\text{^}| = 4$

If K is a covering of G, and M is a matching of G, then K contains at least one end of each of the edges in M.

Thus, for any matching M and any covering K, $|M| \leq |K|$.

Indeed, if M^* is a maximum matching and K* is a minimum covering, then

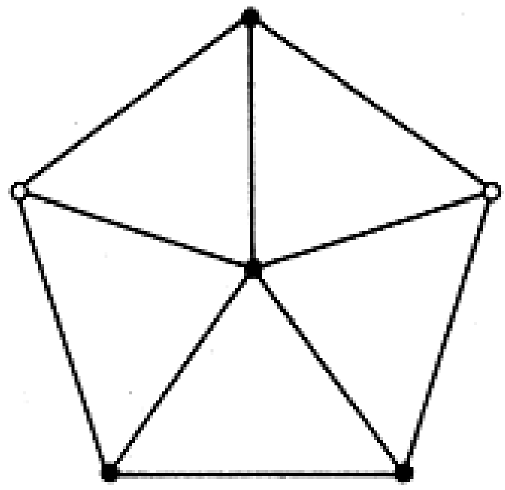
$$|M^*| \leq |K^\wedge| \qquad (5)$$

In general, *equality* in (5) does not hold .

For example,

$|K^\wedge| = 4$,

$|M^*| = 3$



(b)

However, if G is bipartite we do have $|M^*| = |K^\wedge|$.

This result, due to Konig (1931), is closely related to Hall's theorem.

Before presenting its proof, we make a simple, but important, observation.

Lemma 7 Let M be a matching and K be a covering such that $|M| = |K|$.

Then M is a maximum matching and K is a minimum covering.

Proof If M^* is a maximum matching and K^* is a minimum covering then, by (5),

$|M| \leq |M^*| \leq |K\^| \leq |K|$

Since $|M| = |K|$, it follows that $|M| = |M^*|$ and $|K| = |K\^|$

Theorem 8 (König) In a bipartite graph, the number of edges in a maximum matching is equal to the number of vertices in a minimum covering.

Proof Let G be a bipartite graph with bipartition (X, Y), and let M^* be a maximum matching of G.

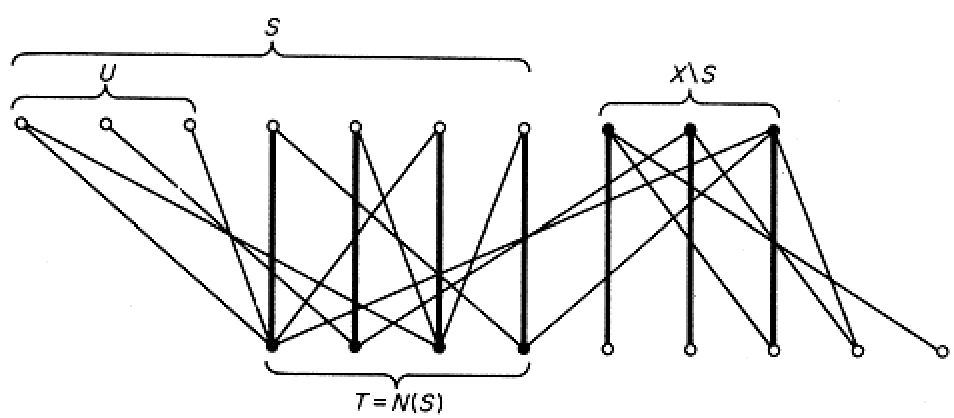
Denote by U the set of vertices that are not covered by M^* in X, and by Z the set of all vertices connected by

M^* -alternating paths to vertices of U.

Set $S = Z \cap X$ and $T = Z \cap Y$.

Then, as in the proof of theorem 6, we have that every vertex in T is covered by M^* and $N(S) = T$.

Define $K\hat{} = (X \setminus S) \cup T$ (see bellow)



Every edge of G must have at least one of its ends in K^.

For, otherwise, there would be an edge with one end in S and one end in $Y \setminus T$, contradicting $N(S) = T$;

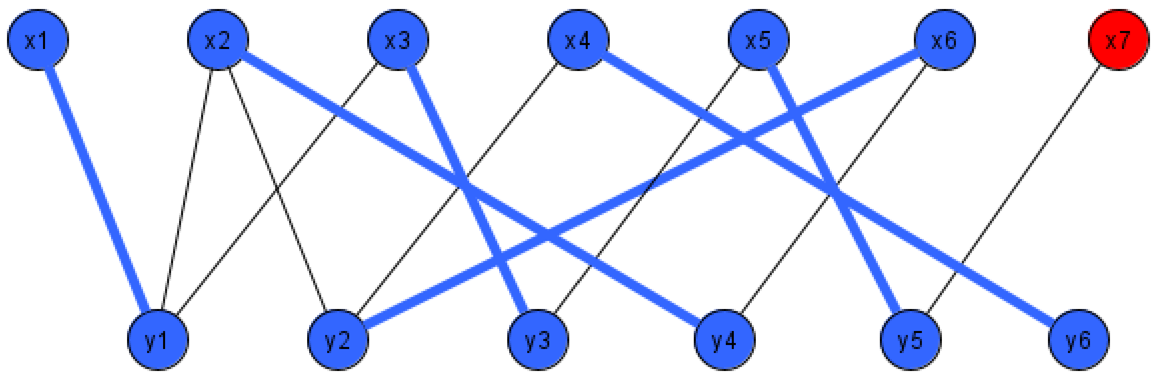
Thus K^ is a covering of G and clearly

$|M^*| = |K\text{\^{}}|$

By lemma 7, K^ is a minimum covering, and the theorem follows

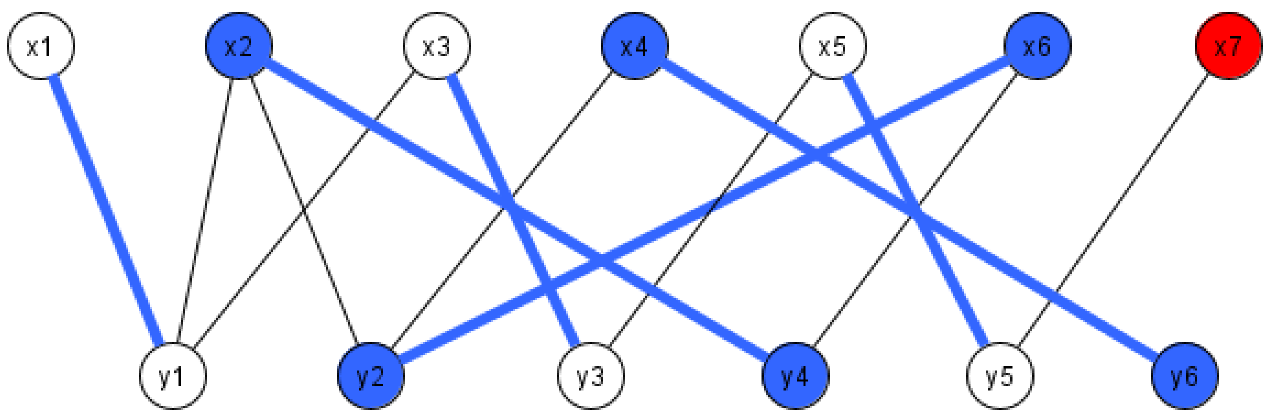
Example of finding a minimum covering

Given a maximum matching



[This example is described in the Lecture by Ivan Takhonov on matchings https://classroom.google.com/w/MjgzNjg4OTMzMjRa/tc/MzY2NTQwNDI2NTNa](https://classroom.google.com/w/MjgzNjg4OTMzMjRa/tc/MzY2NTQwNDI2NTNa)

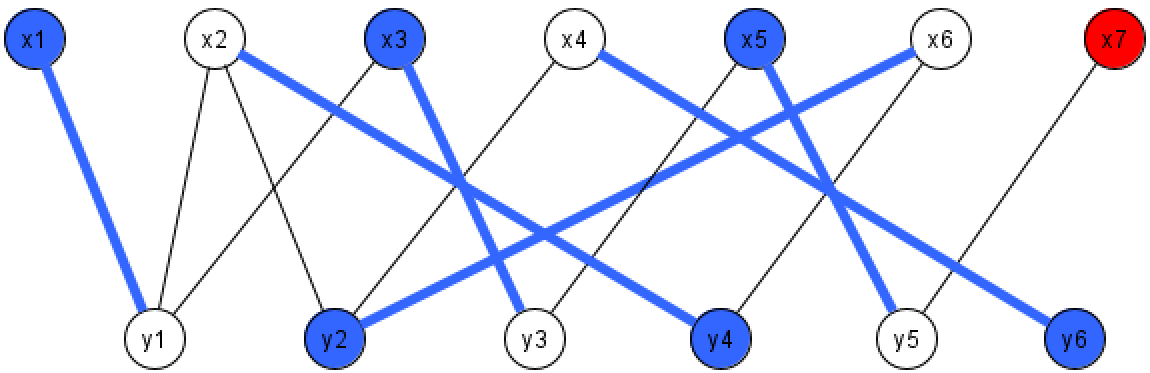
Find vertices that are reachable from the uncovered vertex x_7



$Z = \{x_7, y_5, x_5, y_3, x_3, y_1, x_1\}$
$S = \{x_7, x_5, x_3, x_1\}, T = \{y_5, y_3, y_1\}$

Then $K^\wedge = (X \setminus S) \cup T$

The minimum vertex covering is found



K^ = $\{x_2, x_4, x_6, y_1, y_3, y_5\}$

Independent set

Let G be a graph.

A set of vertices I is called an independent set (независимое множество вершин) if no 2 vertices in I are adjacent.

An independent set is also called a stable set (внутренне устойчивое множество) .

Any singleton set is an independent set.

So one is interested to find a largest independent set.

The parameter $\alpha_0(G) = \max\{|I| : I$ is an independent set in $G\}$ is called the vertex-independence number of G (*число вершинной независимости, числом внутренней устойчивости*).

Any independent set I with $|I| = \alpha_0(G)$ is called a maximum independent set (наибольшее независимое множество) .

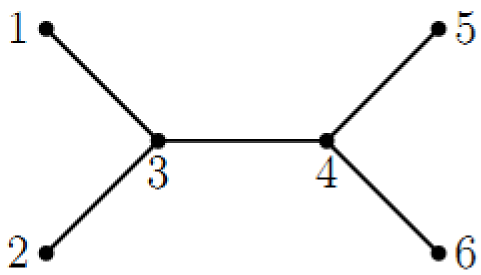
An independent set I is called a maximal independent set (максимальное независимое множество) if there is no independent set which properly contains I.

Clearly, any maximum independent set is a maximal independent set but a maximal independent set need not be a maximum independent set.

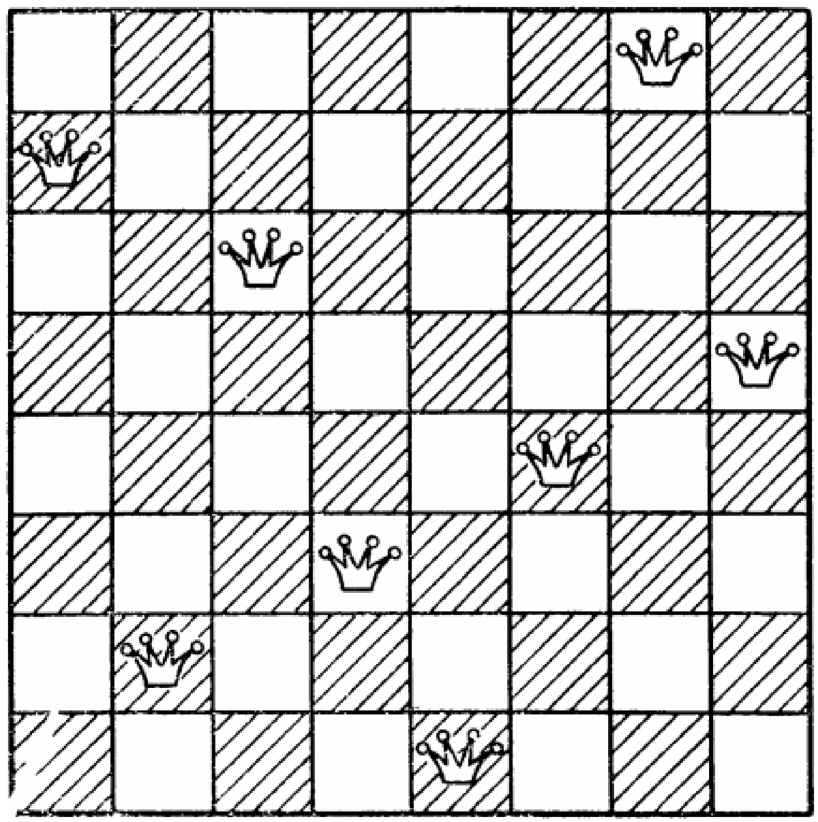
Example

In the graph shown bellow {1, 2, 4}, {3, 5, 6} are maximal independent sets, but they are not maximum independent sets;

{1, 2, 5, 6} is the maximum independent set



Example of independent set problem



Vertex-covering number of *G*

If $e(u, v)$ is an edge in G, then e is said to cover u and v and vice versa.

A set of vertices K is called a vertex-cover, vertex-covering of G if every edge in G is covered by a vertex in K.

That is, if every edge in G has at least 1 of its end vertices in K.

Clearly, $V(G)$ is a vertex-cover of G.

So one is interested to find a smallest vertex cover.

The parameter $\beta_0(G) = \min\{|K| : K \text{ is a vertex cover of } G\}$ is called the vertex-covering number of G (число вершинного покрытия) .

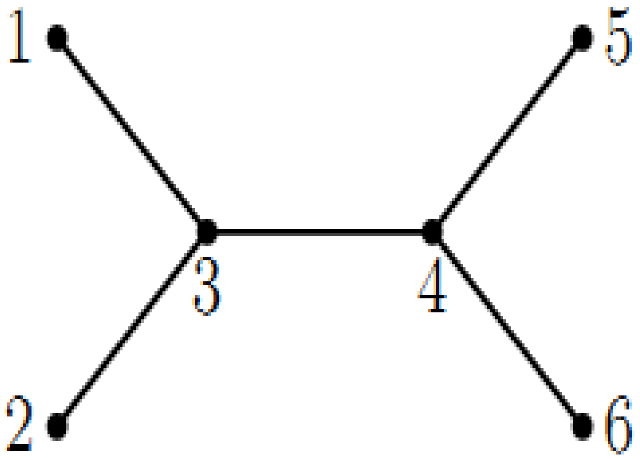
Any vertex-cover K with $|K| = \beta_0(G)$ is called a minimum vertex-cover.

A vertex-cover K is called a minimal vertex-cover if there is no vertex-cover which is properly contained in K.

Clearly, every minimum vertex-cover is a minimal vertex-cover but a minimal vertex-cover need not be a minimum vertex-cover.

Example

The vertex subset (1, 2, 4} is a minimal vertex-cover but it is not a minimum vertex-cover;
{3, 4} is a minimum vertex-cover.
This example illustrates that a graph may contain many minimal and minimum vertex-covers.



Independence numbers and covering numbers

	Pn	***Cn*** **n**	***Kn***	***Km,n***
α_0	$\left\lceil \frac{n}{2} \right\rceil$	$\left\lfloor \frac{n}{2} \right\rfloor$	1	Max{m, n}
β_0	$\left\lfloor \frac{n}{2} \right\rfloor$	$\left\lceil \frac{n}{2} \right\rceil$	n-1	Min{m, n}

It is possible to notice, that for every graph G shown in the table

$$\alpha_0(G) + \beta_0(G) = |V(G)|.$$

In fact, this equation holds for any arbitrary graph *G*.

Before proving this claim, we observe a stronger statement.

Theorem 9. I is an independent set in G $\Leftrightarrow$ $V(G) - I$ is a vertex-cover of G.

Proof. I is an independent set

<=> no two vertices in I are adjacent

<=> no edge has both of its end vertices in I

<=> every edge in G has an end vertex in $V(G) - I$

<=> $V(G) - I$ is a vertex-cover of G.

Corollary. For any graph G,
$\alpha_0(G) + \beta_0(G) = n(G)$

Proof. Let I be a maximum independent set.

By the above theorem, $V(G) - I$ is a vertex cover of G.

Hence, $\beta_0(G) \leq |V(G)-I| = n-\alpha_0(G)$;

that is $\alpha_0(G)+\beta_0(G) \leq n$.

Let K be a minimum vertex-cover of G.

By the above theorem, $V(G) - K$ is an independent set of vertices.

Hence, $\alpha_0(G) \geq |V(G) - K| = n(G) - \beta_0(G)$; that is
$\alpha_0(G) + \beta_0(G) \geq n(G)$:

The two inequalities imply the corollary.

Independent set of edges (matchings)

We now define the edge analogues of independent sets of vertices and vertex-covers.

A subset of edges M in G is called an independent set of edges if no 2 edges in M are adjacent.

An independent set of edges is more often called as a matching.

Clearly, if M is a singleton, then it is a matching.

So one is interested to find a largest matching.

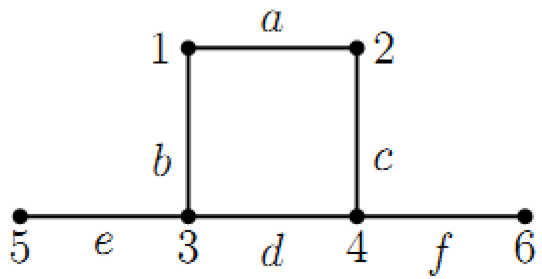
The parameter $\alpha_1(G) = \max\{|M| : M \text{ is a matching in } G\}$ is called the matching number of G (число паросочетания графа G.

Any matching M with $|M| = \alpha_1(G)$ is called a maximum matching

A matching M is called a maximal matching if there is no matching which properly contains M.

In the graph bellow {*a*, *d*} and {*b*, *c*} are maximal matchings but they are not maximum matchings ;

{*a, e, f*} is a maximum matching.



Edge-cover of G

A set of edges F is called an edge-cover (реберное покрытие) of G if every vertex in G is incident with an edge in G.

A graph G need not have an edge cover;

for example, $K_2 \cup K_1$ has no edge-cover.

In fact, a graph G has an edge cover $\Leftrightarrow$ $\delta(G) > 0$:

If $\delta(G) > 0$; then $E(G)$ is an edge-cover.

So one is interested to find a smallest edge-cover.

If $\delta(G) > 0$, then the parameter $\beta_1(G) = \min\{|F| : F$ is an edge cover of $G\}$

is called the edge covering number (число вершинного покрытия) of G.

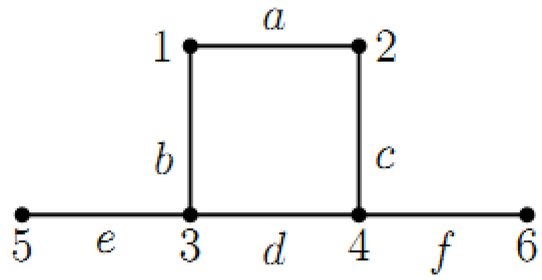
Any edge-cover F with $|F| = \beta_1(G)$ is called a minimum edge cover.

An edge-cover F is called a minimal edge-cover if there is no edge-cover which is properly contained in F.

Example

{b, c, e, f} is a minimal edge-cover but it is not a minimum edge-cover.

- {a, e, f} is a minimum edge-cover.



Again we observe that a graph G may contain many maximal and maximum matchings.

It also may contain many minimal and minimum edge covers.

But $\alpha_1(G)$ and $\beta_1(G)$ are unique.

The following table shows the matching-number and edge-covering number of some standard graphs.

	Pn	***Cn***	***Kn***	***Km,n***
α_1	$\left\lfloor \frac{n}{2} \right\rfloor$	$\left\lfloor \frac{n}{2} \right\rfloor$	$\left\lfloor \frac{n}{2} \right\rfloor$	Min{n, m}
β_1	$\left\lceil \frac{n}{2} \right\rceil$	$\left\lceil \frac{n}{2} \right\rceil$	$\left\lceil \frac{n}{2} \right\rceil$	Max{n, m}

Again notice that for every graph G shown in the above table $\alpha_1(G)+\beta_1(G) = n(G)$.

However, the edge analogue of Theorem 9 does not hold.

That is, if M is a matching in G, then $E(G) \setminus M$ need not be an edge cover of G.

And if F is an edge cover of G, then $E(G) \setminus F$ need not be a matching.

However,

$\alpha_1(G) + \beta_1(G) = n(G)$, for any graph G with $\delta(G) > 0$.