Pre-Introduction (Discarded Materials)

Table of Contents

[Types of Graphs 3](#_Toc141041092)

[Complete Graphs 3](#_Toc141041093)

[Path Graph 3](#_Toc141041094)

[Cycle Graphs 4](#_Toc141041095)

[Subgraphs 4](#_Toc141041096)

[Induced Subgraph 5](#_Toc141041097)

[Connected Graph 5](#_Toc141041098)

[Isomorphic Graph 5](#_Toc141041099)

[Acyclic Graph 6](#_Toc141041100)

[Bipartite Graph 7](#_Toc141041101)

[Vertex Cover 8](#_Toc141041102)

[Independent Set 10](#_Toc141041103)

[Matching 12](#_Toc141041104)

[K-Regularity 14](#_Toc141041105)

[Spanning Subgraph 15](#_Toc141041106)

[K-Factor 15](#_Toc141041107)

[Konig’s Theorem 17](#_Toc141041108)

[Hall’s Theorem 17](#_Toc141041109)

[Perfect Matching of K-Regular Bipartite Graph 20](#_Toc141041110)

[Cycle Cover of 2 Regular Graph 20](#_Toc141041111)

[Tutte’s Theorem 20](#_Toc141041112)

[Perfect Matching of Cubic Graphs 21](#_Toc141041113)

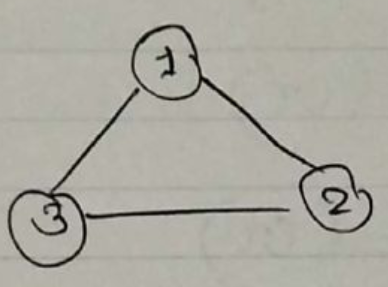
[Edge Cover 21](#_Toc141041114)

A graph, , can be described as a set of **vertices**, , and a set of **edges**, . A **simple graph** is one which has **no loops** and **no multi-edges**. A **finite graph** is one which has a **finite number** of **vertices** and **edges**.

## Types of Graphs

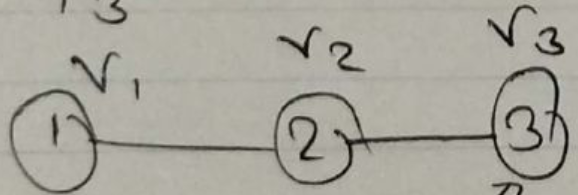
### Complete Graphs

A **complete graph** is one in which all the vertices are connected to each other. For vertices, the complete graph is denoted as . For example, this is the graph for .



### Path Graph

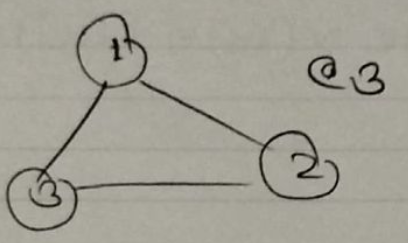
A **path graph** with vertices is denoted as . This is a graph for which there is an edge between every **adjacent pair** of vertices, and . The graph for looks like this:



If , . The two vertices on either end of the path are called **terminal vertices**.

### Cycle Graphs

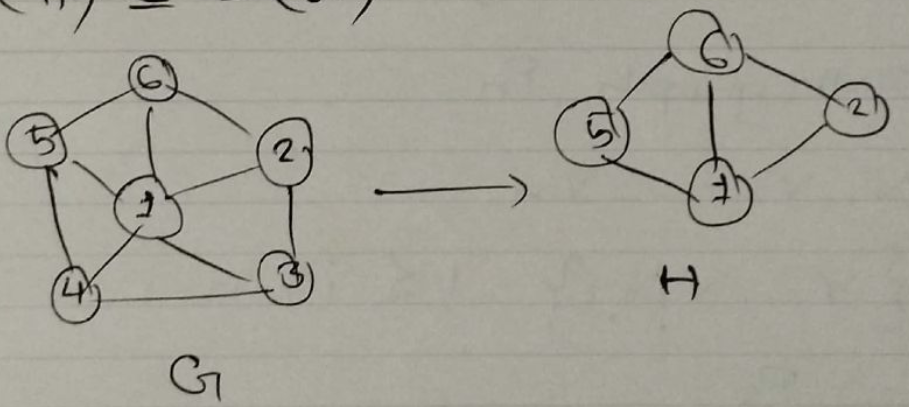
A **cycle graph** has vertices connected in a single cycle and is denoted as . The graph for looks like this:



For cycle graphs, .

### Subgraphs

If is a **subgraph** of , then and .



Note that the **edges** we select for the subgraph must be connected to the **vertices** we select. This may seem obvious for now but will become less obvious when we try to implement this algorithmically.

### Induced Subgraph

An **induced subgraph** is a specific type of subgraph that selects a set of vertices from the original graph as well as all of the vertices that connect the chosen set of vertices.

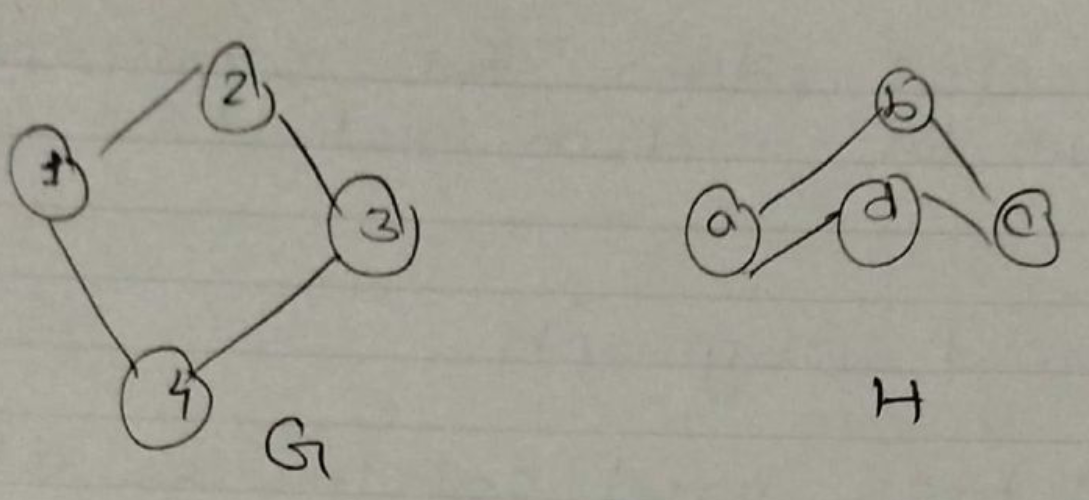
### Connected Graph

A **connected** graph is one for which there is **at least one path** between each of the vertices.

### Isomorphic Graph

It is possible for two separate graphs, and , to have a mapping from to such that a pair exists in if and only if . In such cases, the graphs are called **isomorphic graphs**.

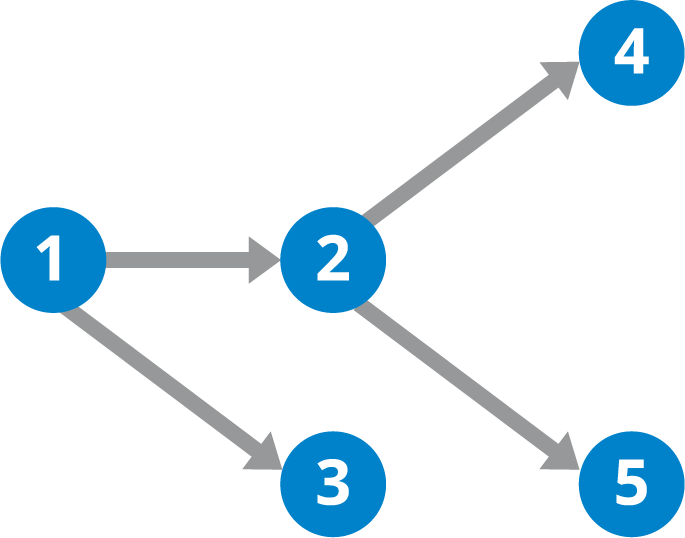
In simpler terms, the graphs are exactly the same.



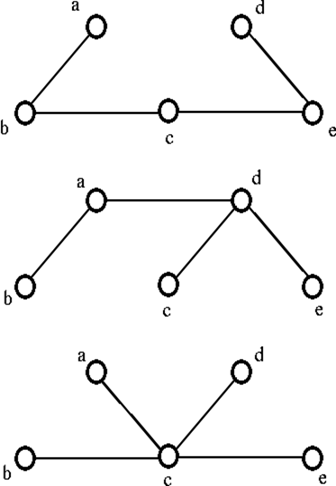
The two graphs above are isomorphic because we have a mapping as:

### Acyclic Graph

An **acyclic graph** has no subgraphs that are isomorphic to a cyclic graph, i.e., the graph has no cycles.

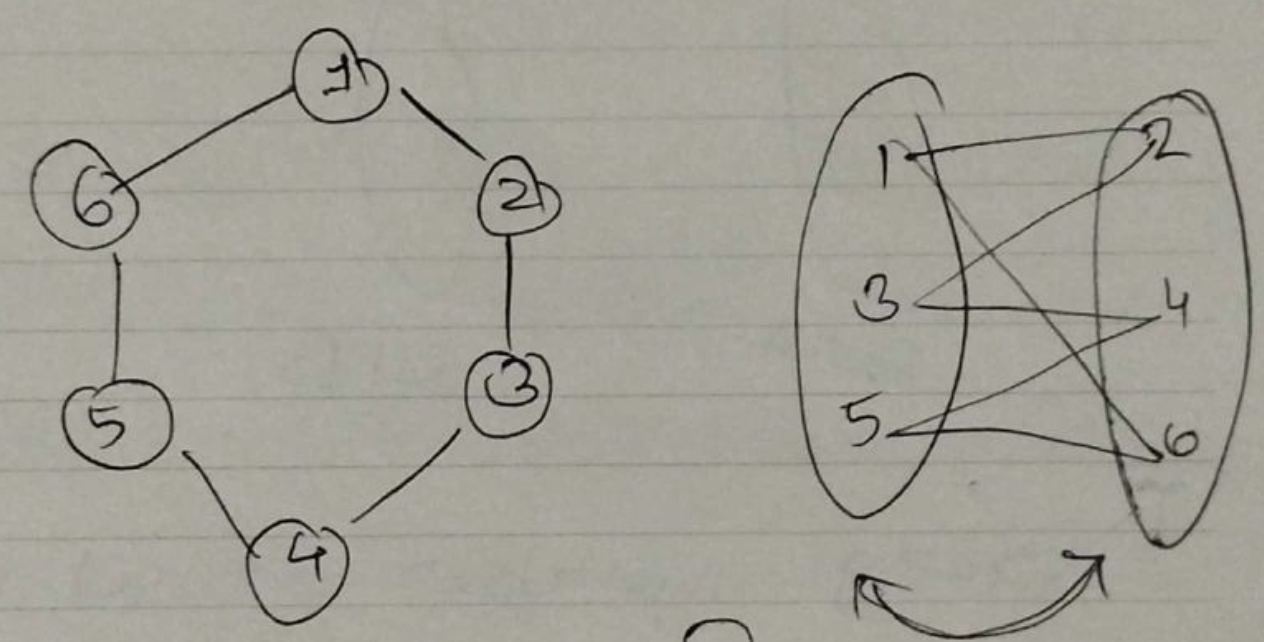


Any acyclic graph that has all its vertices connected is a **tree**. If the vertices are not all connected, it forms a **forest**. A tree also counts as a forest.



### Bipartite Graph

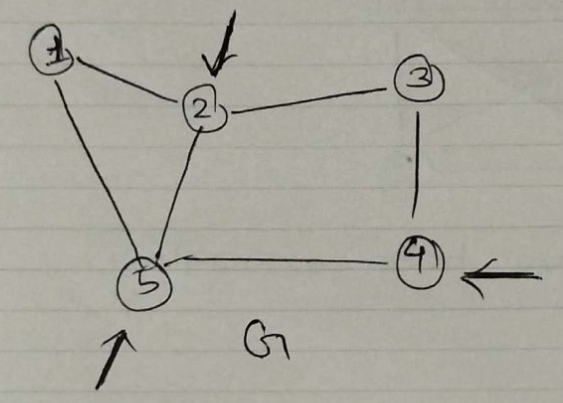
A **bipartite graph** is a one for which the **vertices** can be divided into **two sets** such that the vertices in the same set have **no edges** between them. Each of these sets is called a **partite set**. Bipartite graphs are denoted as .



If every vertex in one of the partite sets is connected to every vertex in the other partite set and vice versa, the graph is called a **complete bipartite graph**.

## Vertex Cover

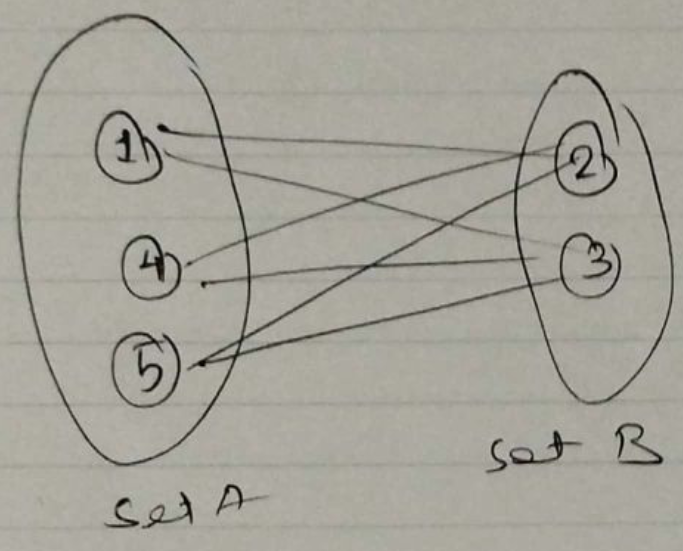
In **vertex cover**, our aim is to **select vertices** from a graph in a way so that **every edge** in the graph is connected to at least one of the selected vertices. We try to select the least number of vertices possible while maintaining this goal. The number of vertices that have been selected is called the **minimum vertex cover**, .



For the graph above, vertex cover can be achieved using , so .

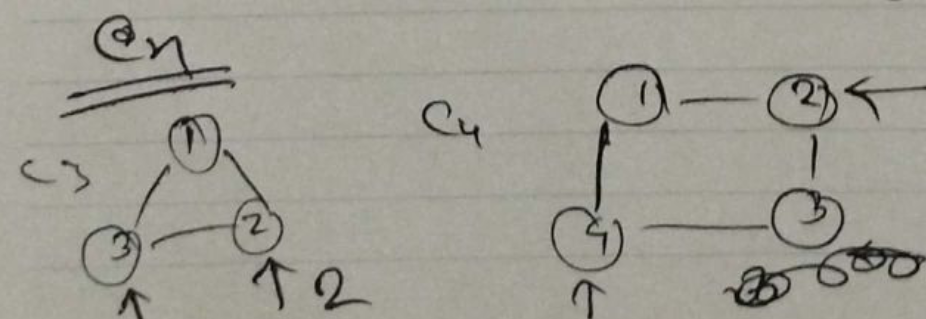


For , . If we choose any value less than this, suppose , we would have to leave out multiple vertices. Suppose we choose to leave out the vertices and . However, is complete, so there must be an edge between which was not covered.



For a complete bipartite graph with two partite sets and , . For the graph above, .

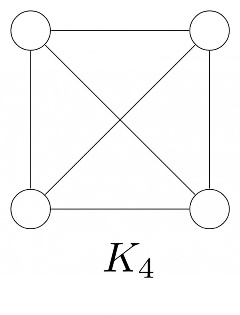
If and we choose as , then we must leave out some vertex . By doing so, we are leaving out all the edges that are connected to only.



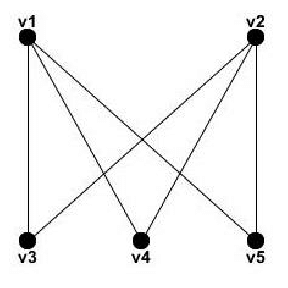
For the two cycle graphs above, we can see that and . Taking any fewer vertices in either case will cause at least one of the edges to remain uncovered. Thus, .

## Independent Set

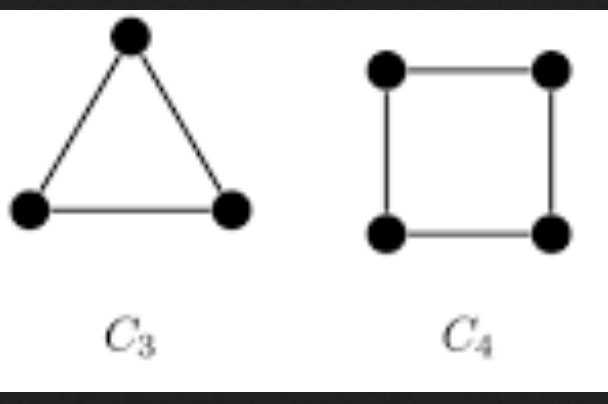
An **independent set** is a set of vertices of a graph which have no edges between them. The number of vertices selected is called the **independence number** and is denoted by . We want to maximize this number.



For a **complete graph**, it is only possible to create an independent set by select any one of the vertices, , , etc., meaning .



For a **bipartite graph**, we can select either of the partite sets. Since we want to maximize the independence number, we should select the larger set, so .



With **circular graphs**, we can see that it is only possible to select **one** vertex for and **two** vertices for . Thus, .



A similar situation exists for **path graphs**, where allows us to select just one vertex while allows us to select two (as long as we pick optimally). Thus, .

There are a few implications of creating an independent set. If we have some set and we remove from it, we get . It is possible that some vertices that were meant to be in were also removed with , so the inequality stands:

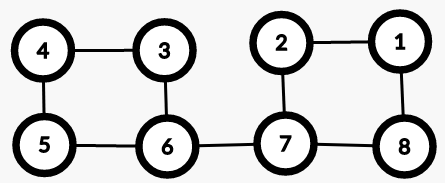
On the flip side if we remove from we should have left. Again, it is possible that some vertices that were meant to be removed along with are left behind with . Thus:

Combining the two equations above:

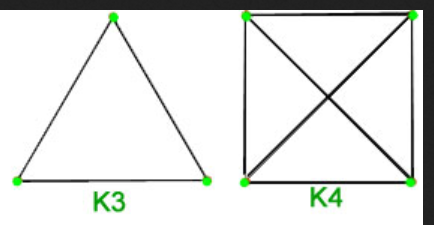
Thus, this inequality must be true in order for both the scenarios above to be true simultaneously.

## Matching

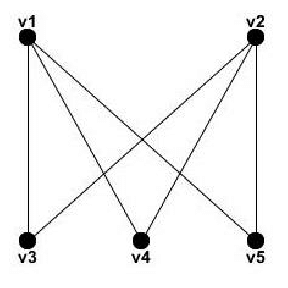
In **matching**, we select edges from a graph which do not share any vertices.



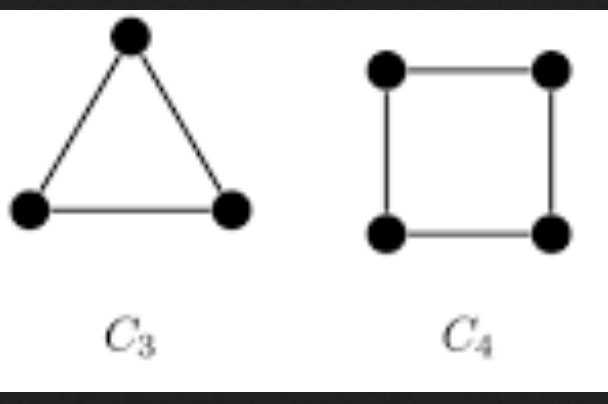
For the graph above, if we take the edge between vertex 1 and 8 and the edge between vertex 4 and 5, these two edges do not share any vertices and are thus a single matching. We want to **maximize** the matching, which can be done in this case by taking the edges 1-8, 2-7, 3-6 and 4-5. The size of the maximum matching is denoted as .



For , the only way to achieve a matching is to select a single edge. For , we can select two edges. Thus, . Notice that has **unmatched vertices**, but does not. Every vertex in a can be matched if the value of is even.



For , the only way to ensure that there are no repeated vertices is to select a single edge from each of the vertices of the **smaller set**. Thus, . In this case, the only way to ensure there are no unmatched vertices is if .



For graphs, we have a similar situation to that of graphs. Thus, .

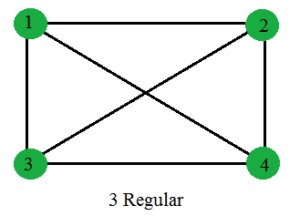


A **path graph** is similar to a circular graph. Again, we can only pick every alternate edge, so .

For every type of graph other than bipartite graphs, we are able to match all the vertices only if the value of is even. This scenario is called **perfect matching**, which is also known as **1-Factor**.

To understand the naming behind 1-Factor, we need to cover two more concepts, K-Regularity and Spanning Subgraphs.

### K-Regularity



The **degree** of a vertex is the number of edges attached to the vertex. The vertices in the graph above are each of degree 3. If every vertex in a graph has a degree of , the graph is said to be -Regular.

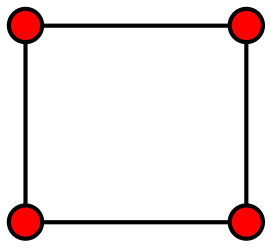
### Spanning Subgraph

A **subgraph** of a graph is one such that:

If , then the subgraph is called a **spanning subgraph**. Note that it is not necessary for to be equal to be .

### K-Factor

A **K-Factor** graph is a **K-Regular Spanning Subgraph**. For example, the 3-regular graph above can be converted to a 2-factor graph if we create the following subgraph:



Based on this information, a **perfect matching** of a graph G can be achieved by taking the **1-Factor Subgraph**.

If we **remove** the **maximum matching** from a graph (both the vertices and the edges), we should only be left with vertices that have no edges between them, i.e., . It is however possible that a few of the vertices that fall into were also a part of . Thus,

We are using because represents the number of edges in the matched graph, when we need the number of vertices.

We previously saw that . Thus,

It can also be argued that taking one vertex from each of the edges of a maximum match should give us the minimum vertex cover. However, it is possible that the minimum vertex cover required more vertices. Thus,

Combining the two inequalities:

## Konig’s Theorem

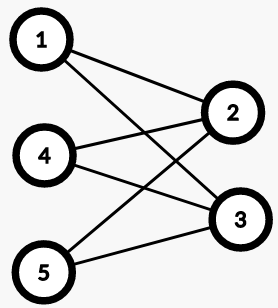
**Konig’s Theorem** states that, for a **bipartite graph**, . This works out because the minimum vertex cover for a bipartite graph will be the set of vertices in the smaller partite set. The edges connected to these vertices will be part of the maximum matching.

## Hall’s Theorem

The **neighborhood** of a vertex is the set of vertices connected to that vertex. For a vertex , the neighborhood is denoted as . For a set of vertices , the neighborhood, , is the set of vertices connected to any vertex in .

**Hall’s Theorem** states that for a **bipartite graph** with two partite sets and , there is a matching of if and only if , .

Consider the bipartite graph below:



If we consider the partite set on the right:

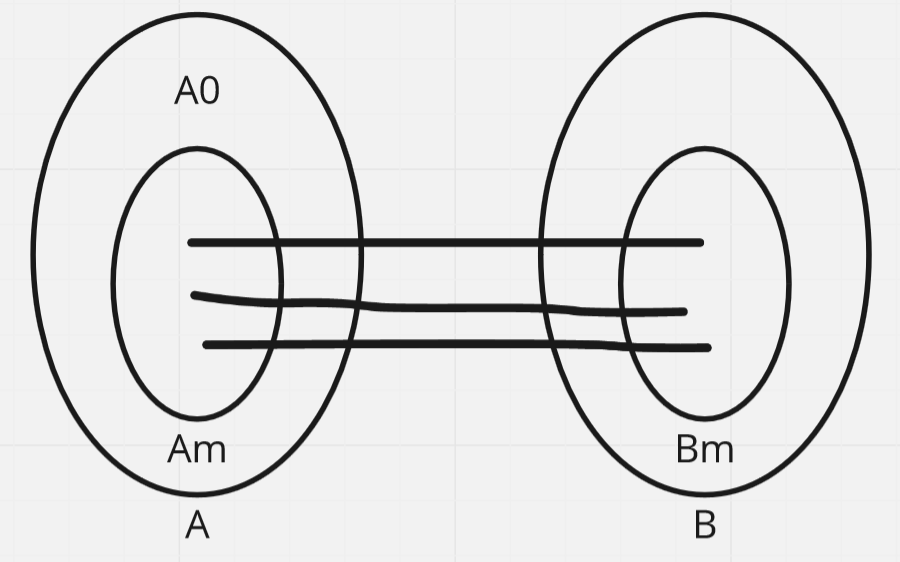
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Thus, for every possible subset, . This means that a matching is possible and will consist of the vertices in this partite set.

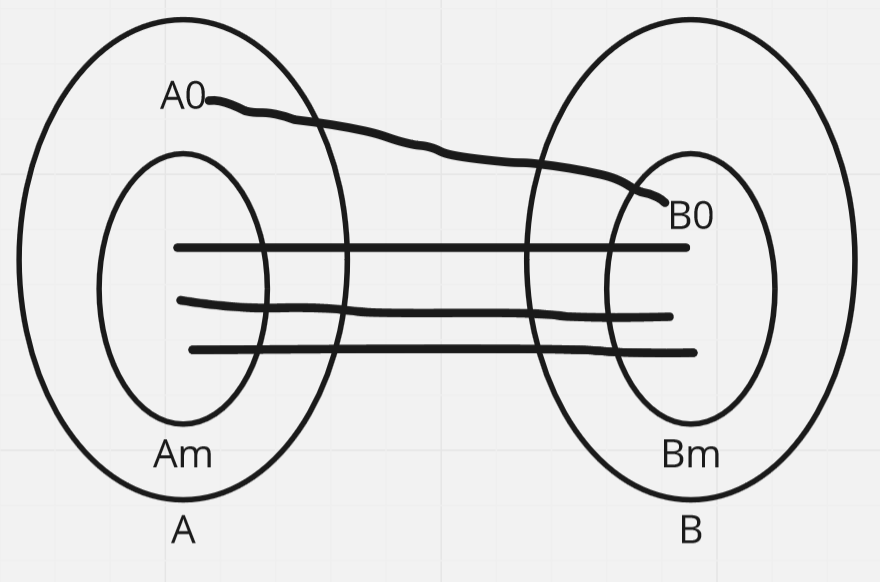
If we consider the other partite set, we will see that this inequality does not hold for all the subsets, meaning that partite set will not give us the maximum matching.

Proof:

To prove Hall’s theorem, we will be using proof by contradiction. Let’s assume that the condition is satisfied but we still have some vertices in the partite set which are **unmatched**. The set of vertices in partite sets and that are already matched are denoted as and respectively.



Suppose a vertex in is unmatched. We can see that for this single vertex, so the condition is clearly satisfied. However, since we are claiming that **maximum matching** is satisfied and the bipartite graph is fully connected, must share an edge with some vertex which is inside . If was not in , then that would mean there is no issue placing and in and respectively.



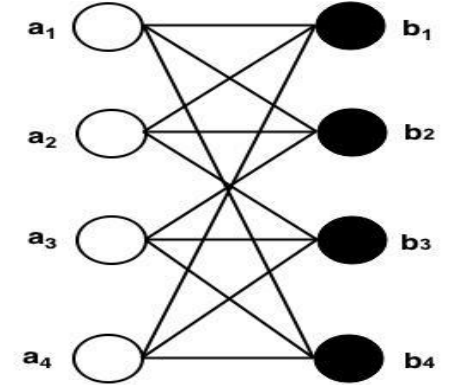
Since we have established that is in but is not, this must mean shares another edge with some vertex, say , which is inside .

Now suppose we take . This creates an augmenting path.

If we keep going, we will find that the same issue keeps repeating, meaning this case cannot exist.

## Perfect Matching of K-Regular Bipartite Graph

If a **bipartite graph** is **k-regular**, then we are guaranteed to have **perfect matching**.



If we pick a subset from one of the partite sets, then we have edges. The neighbors of , , also have edges each making a total of edges. Since every edge connected to must also be connected to , and can have even more edges, . This simplifies to Hall’s condition.

## Cycle Cover of 2 Regular Graph

A 2 regular graph always satisfied **cycle cover**.

## Tutte’s Theorem

**Tutte’s Theorem** states that a graph has **perfect matching** if and only if satisfied **Tutte’s Condition**:

,

A set that does not satisfy **Tutte’s Condition** is called a **bad set**.

## Perfect Matching of Cubic Graphs

A **cubic graph** is a **3-Regular** graph.

A **bridge** is an edge that, if removed, makes the graph **disconnected**.



A **tree graph** is a graph where all the edges are bridges.

A **cycle graph** is a graph which has no bridges.

A **bridgeless cubic graph** is guaranteed to have **perfect matching**.

## Edge Cover

In **edge cover**, we must select edges in a manner such that all the vertices are connected to some edge. The minimum number for which this can be done is called the **minimum edge cover**, .

If the graph has **perfect matching**, the edges involved in the perfect matching create .

One edges can cover vertices, so there are at least edges. However, not all graphs have edge cover. For example, if the graph has an isolated vertex.