Week 6

Spanning Trees, Matching*

6.1 Diameter

Diameter in the graph is the maximum distance d(x,y) over all pairs of vertices. This d(x,y) is the shortest distance between x and y . It is the value of greatest eccentricity.For finding Diameter, find distance between all pairs of vertices and choose maximum among them.

$$\text{Diam(G)} = max\{d(x,y)\} \text{ where } x, y \in V$$

Theorem: If G is simple graph, then diameter(G) ≥ 3 implies that diameter($\overline{G}$) ≤ 3.
With Diameter(G) ≤ 3, it means there exist nodes u and v such that they have no common neighbours. Since, they do not have a common neighbour in G, they'll definitely have a common neighbour in $\overline{G}$. Hence, every $x \in V(G)\{u, v\}$ has at least one of u, v as non-neighbour. So in G , x is adjacent to at least one of u, v. Since $uv \in E(\overline{G})$, for every pair $\{x, y\}$, there is a path of length at most 3 in G through u, v. Hence diameter(G) ≤ 3.

Theorem: If $diam(G) \geq 4$, then diam($\overline{G}$)=2.
Proof: Let's suppose there are four vertices x, y, u, v such that $diam(G) \geq 4$. It signifies that u and v are not connected by path among these 4 vertices x,y,u,v. Now , x or y cannot be adjacent to both u and v otherwise diameter(u,v)=2. So there can be 3 arrangements possible in this graph:

1. Case 1: x and y are connected to one of the vertices either u or v in Figure 6.1.

*Lecturer: Dr. Anand Mishra. Scribe: Sonali.

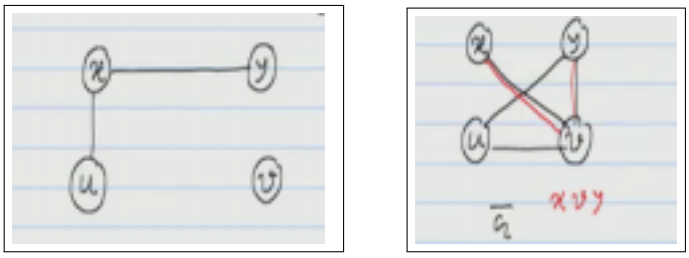

Figure 6.1: Case 1

If we take complement of this graph, then diam($\overline{G}$)=2.

2. Case 2: x and y have edges between them, but neither connected to u nor v in Figure 6.2.

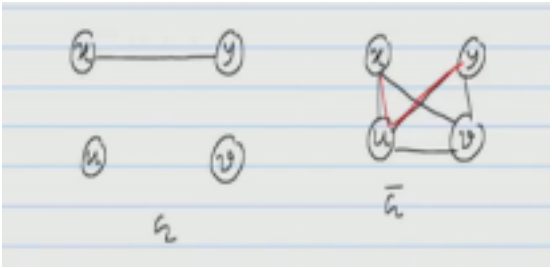

Figure 6.2: Case 2

Then, also diam($\overline{G}$)=2.

3. Case 3: x,y are adjacent to u , v respectively or vice versa in Figure 6.3.

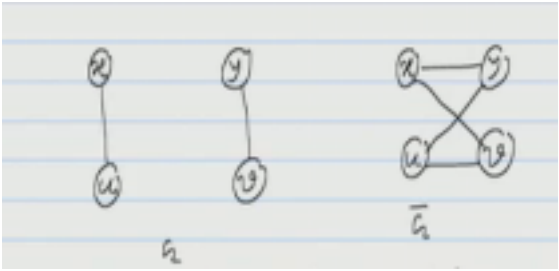

Figure 6.3: Case 3

We assume (x,u) and (y,v) belong to E(G). If (x,y)belongs to E(G) then diameter=3 which contradicts the theorem. Therefore, when (x,y) does not belong to E(G), diameter=1.

6.2 Spanning Trees

6.2.1 Definition

A spanning tree T of an undirected graph G is a subgraph that is a tree that includes all of the vertices of G, with a minimum possible number of edges. For example, in Figure 6.4, a spanning tree can be created using 3 edges. Some of the them are [1-2, 2-3, 3-4], [1-3, 1-2. 1-4], etc.

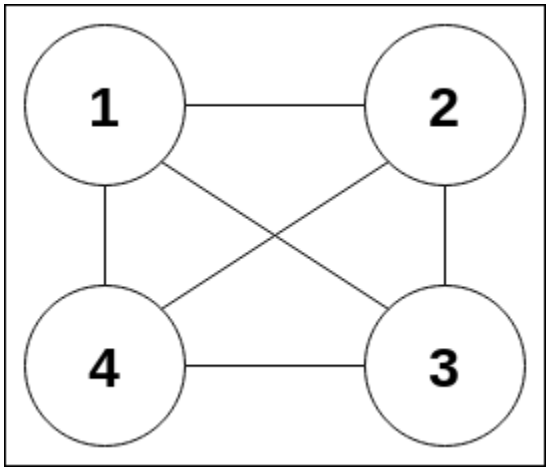

Figure 6.4: A simple graph G

6.2.2 Minimum Spanning Tree

A minimum spanning tree is defined as a spanning tree with minimum cost of edges. For example, in Figure 6.5, the minimum spanning tree is [C1-C4, C2-C3, C3-C4] with costs 10, 20, 100 respectively with a minimum cost of spanning tree as 130.

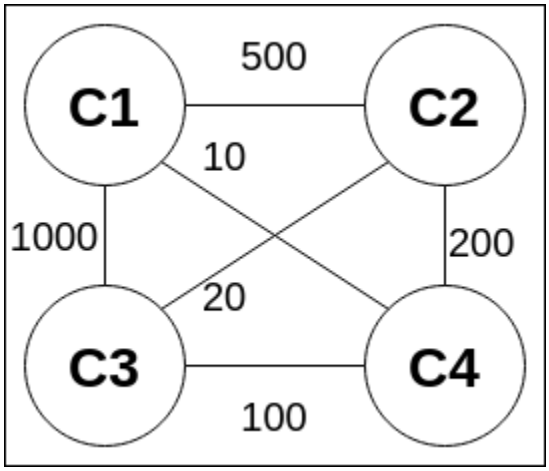

Figure 6.5: A weighted graph G

There are two ways of performing minimum spanning tree - Prim's and Kruskal's algorithm.

Prim's Algorithm

It is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. It finds a subset of the edges that forms a tree that includes every node, where the total weight of all the edges in the tree are minimized. The algorithm was developed in 1930 by Czech mathematician Vojtech Jarnik and later rediscovered and republished by computer scientist Robert Clay Prim in 1957 and Edsger Wybe Dijkstra in 1959. It is also known as DJP algorithm, Jarnik's algorithm, Prim-Jarnik algorithm or Prim-Dijsktra algorithm.

The algorithm is defined in following steps:

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.
2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.
3. Repeat step 2 (until all vertices are in the tree).

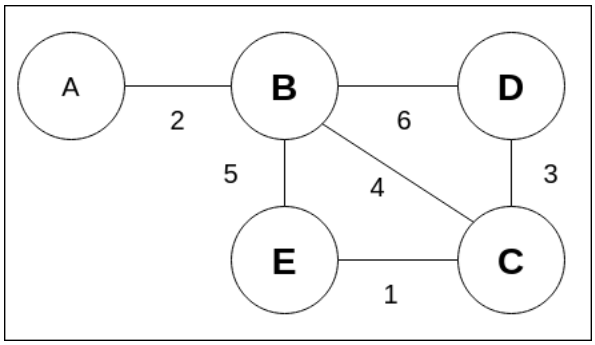

Figure 6.6: A weighted graph G

For example, using Figure 6.6, we compute the minimum spanning tree in the following steps:

1. We need to start an initial node, lets choose the initial node to be C. From C, the reachable nodes are D, B, E with costs 4, 3, 1 respectively. Out of those three, we choose the minimum, i.e. E. MST = CE.
2. After E, the reachable nodes now are B, D with costs min(3, 1+5), 4 respectively. We choose the minimum, i.e. B with cost 3. MST = CE, CB.
3. After B and E, the reachable nodes left are A, D with costs (3+2), 4 respectively. We choose the minimum, i.e. D with cost 4. MST = CE, CB, CD.
4. We are only left with one node A with cost 3+2. Hence, MST = CE, CB, CD, AB.

Kruskal's Algorithm

This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

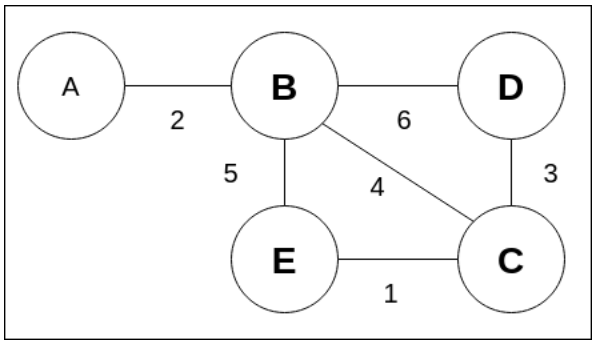

Figure 6.7: A weighted graph G

The algorithm is defined in following steps:

1. Sort all the edges in non-decreasing order of their weight.
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.
3. Repeat step 2 until there are (n-1) edges in the spanning tree.

For example, using Figure 6.7, we compute the minimum spanning tree in the following steps:

1. Sort all the edges and pick the smallest edge i.e. CE with weight 1. MST = CE.
2. The next smallest edge i.e. AB with weight 2. MST = CE, AB.
3. The next smallest edge i.e. BC with weight 3. We need to avoid forming a cycle in the MST. MST = CE, AB, BC.
4. The next smallest edge i.e. CD with weight 4. e need to avoid forming a cycle in the MST. MST = CE, AB, BC, CD. All the vertices have been included and hence we stop the algorithm.

The main difference between Prim's and Kruskal's is in the order in which edges are selected.

Application of Minimum Spanning Trees

1. Designing of efficient network system.
2. Image segmentation.
3. Find airline routes.
4. Designing of efficient routing algorithm

6.3 Matching

6.3.1 Definition

Matching or Independent edge set is a set of non-adjacent edges without common vertices or endpoints in an undirected graph.It does not contain any loop. It can be considered as a subgraph where all nodes have degree 1. Vertex is matched if it has an endpoint of edge present in matching of the graph.It should be matched to exactly one edge by the above definition.This is referred to as saturated vertex.

They have application in assignment problems such as resource allocation ,Internet traffic load balancing and matching such as dating services for finding compatible matches and job allocation according to preference of candidates. We can consider these applications as graphs where each edge denotes compatibility and the objective is maximization of compatible pairs.

For example, in Figure 6.8, we have matching M1 : (AB , CD) and M2 : BD.

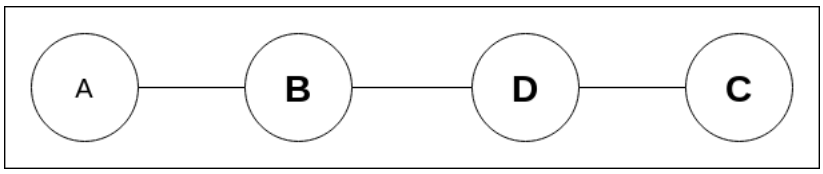

Figure 6.8: A simple graph G

Size of matching = number of edges that appear in matching.

6.3.2 Perfect Matching

A perfect matching is defined as a matching where every vertex is saturated. For example, in Figure 6.9, every boy is perfectly matched with a girl, where, X represents set of boys (B1,B2,B3,B4) and Y represents set of girls (G1,G2,G3,G4).

Following are the properties of perfect matching:

1. The size of perfect matching in graph = n/2.
2. It is possible only when graph has an even number of vertices but not a sufficient condition as shown in Figure 6.10.

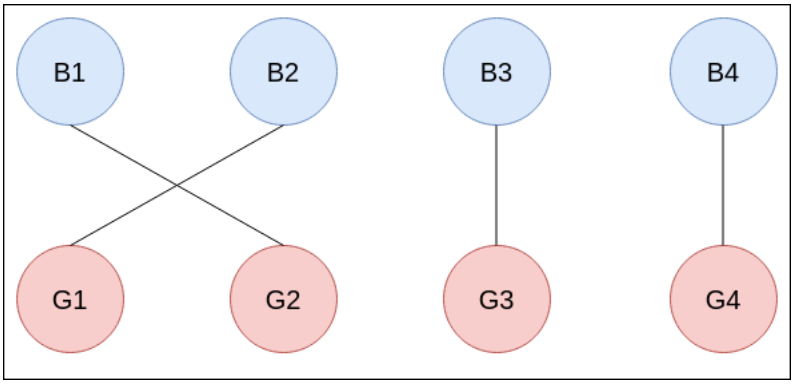

Figure 6.9: Perfect Matching Example

Figure 6.10: Not a perfect matching even with even number of vertices

3. Perfect matching is maximum hence maximal.

4. It is also called as complete matching.

5. It is also a minimum size edge cover.

6. A graph may have more than one perfect matching.

All perfect matching are equally desirable . So we often give weights according to desirability called Maximum weight matching.

Near perfect matching Maximum matching with an odd number of vertices and one vertex is left out.

Proof: Perfect matching is (2m)!/(2^m * m!) for a complete graph of K_n.
Soln: Perfect matching hold only for even no of vertices.For odd, it is 0.
We take n as even vertices that can be written as n =2m for some m. In a perfect graph every vertex is connected with all other vertices so degree 2m-1.From these 2m-1 edges we will select 1 edge in 2m-1 ways and cover two vertices.
So, 2m-1 vertices and (2m-3) edges are left.
Number of ways to select 2m-3 edges =2m-3.
Other edges can be selected in same manner using product stated below:

$=[(2m-1)*(2m-3)*.....*3*1][(2m)*(2m-2)*....4*2]/[(2m)*(2m-2)*....4*2]$
$= 2m!/2^m * m... * 1$
$= 2m!/2^m * m!$

6.3.3 Maximum and Maximal Matching

A matching is said to be **maximum** which cannot accommodate more edges. For example, in Figure 6.8, we have matching as BD. In this, we cannot add any more edges.
A matching is said to be **maximal** that has maximum cardinality or maximum number of edges. For example, in Figure 6.8, we have matching as M1 : (AB , CD).

Notes:

1. Every maximum matching is maximal but vice versa does not hold.
2. Every perfect matching is maximum but the reverse is not true.
3. More than one maximum matching is possible in a Bipartite graph and solved using flow graph. (covered in further lectures)

Lemma 1 : Every tree has at most one perfect matching.

Proof (by contradiction) Let M1 and M2 be the perfect matchings in a tree. Therefore, the symmetric difference of the edge sets, $M1 \triangle M2$. Since, these matchings are perfect matchings, each vertex has either degree 0 or 2 in the symmetric difference, hence, every component is an isolated vertex or a cycle. Since the tree has no cycle, every vertex must have degree 0 in the symmetric difference, which means that the two matchings are the same.

Proof (by induction) The base condition is: a tree having only 1 node has PM = 0, and with n = 2, it has PM = 1. If we consider that the number of vertices in the tree is 2 or more, then, it has a leaf. Leaf is the end-nodes of the tree. Les pick a leaf, then, the edge from this leaf to its neighbor must be a part of any perfect matching. Now, if we remove the edge and both its vertices from the graph, then, This reduces our tree into zero or more smaller trees. Hence, together with the rest of the original forest these form a smaller forest that has at most one perfect matching.

6.4 Hall's Marriage Theorem

In Figure 6.11, Google company is interested to hire Aman, Apple is interested to hire Ram and Sita, Meta is interested to hire Aman and Amazon wants to hire Ram, Sita, Aman and Viru. Each company has only one position. Every position will fill at the end. Now the question is **Will every position be filled?**. Therefore for the given example, answer is **NO**.

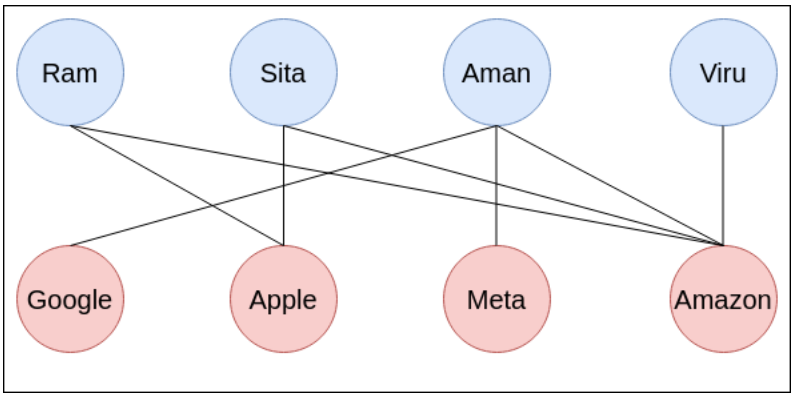

Figure 6.11: Hall's Marriage Theorem

We can solve this problem using Hall's Marriage Theorem.
A X - Y bipartite graph G has a matching that saturates X if and only if

$$|N(S)| \geq |S| \ \forall S \subseteq X$$

Here, $N(S) \subseteq Y$ is set of neighbors of elements in S.

6.4.1 Problems:

1. Perfect matching in K_6 and K_{253}?
   Soln : 6! / 3!*2^3 =15 and 0.

2. Prove maximal matching size is at least half size of maximum matching . Let maximum matching = M^* and $|M^*|$=k and cover 2k vertices.
   Soln : By contradiction, if M (maximal matching) covers $< k$ vertices and at least one endpoint is left of all k edges in the graph. It means that one edge still remains and can be added in M. But that contradicts the fact that it is maximal matching which cannot accommodate more edges.

3. Which matching is near perfect matching and perfect matching in Figure 6.12?

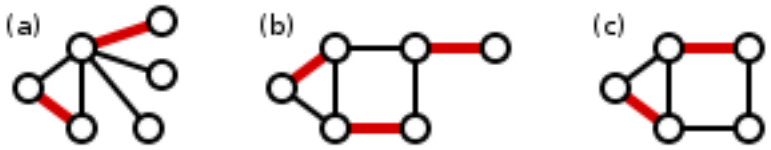

Figure 6.12: Graphs

Soln : In c, only one vertex is left out and b is perfectly matched. Graph a is neither of both.