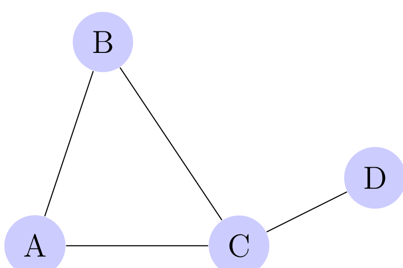
Week 6

Theorems - Graph Theory*

6.0.1 Theorem: If G is a Graph then the diameter of G

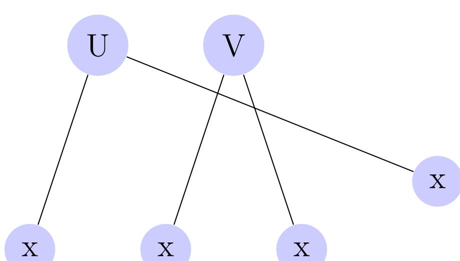
$$diam(G) \geq 3 \Rightarrow diam(G^c) \leq 3 \tag{6.1}$$

Let us consider a Graph as shown below



(i) $\Rightarrow \exists$ u, v $\in$ V(G) such that u, v are not adjacent
(ii) u, v have no common neighbour, if they have a common neighbour then it will be 2 and it will not be a diameter.
$\Rightarrow$ u, v does not exist in G
$\forall$ x $\in$ V(G) - u, v, ux or vx does not exist

G^c will surely have u, v $\Rightarrow$ u,v exist in G^c and ux or vx will also exist in G^c



*Lecturer: Dr. Anand Mishra. Scribe: Abhilash Sharma.

One of U, V is connected to one of x.

$\forall$ x$'^{\in}$ V(G) $\in$ u, v
Therefore diam(G^c) ≤ 3

6.0.2 Stmt: Let G be a simple graph with

diam(G) ≥ 4 Then prove that
diam(G^c) $\leq$ 2
We will use previous theorem diam(G) $\geq 3 \Rightarrow diam(G^c) \leq 2$
We will use technique called Contra +ve

Proof: We have to prove

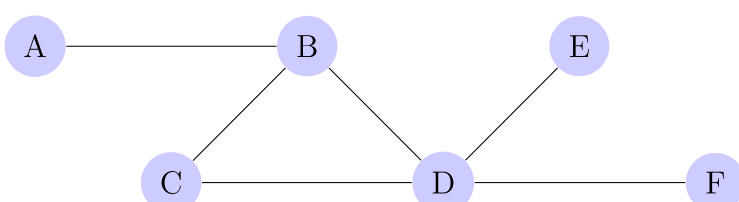
$$P \Rightarrow Q \tag{6.2}$$

where p $\equiv$ diam(G) ≥ 4
$q \equiv$ diam(G^c) ≤ 2
We will prove equivalent to above
$\neg$ Q $\Rightarrow$ $\neg$ P
$\neg$ Q $\equiv$ diam(G^c) ≥ 3
$\neg$ P $\equiv$ diam(G) ≤ 3

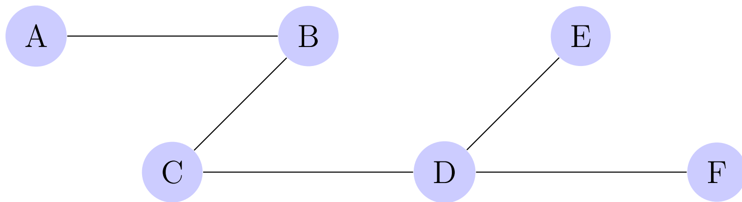
Since diam(G) $\geq 3 \Rightarrow$ diam(G^c) ≤ 3
$Therefore, diam(G^c) \geq 3 \Rightarrow$ diam(G^c)^c ≤ 3
$\Rightarrow$ diam(G^c) ≤ 3

6.1 Spanning Trees

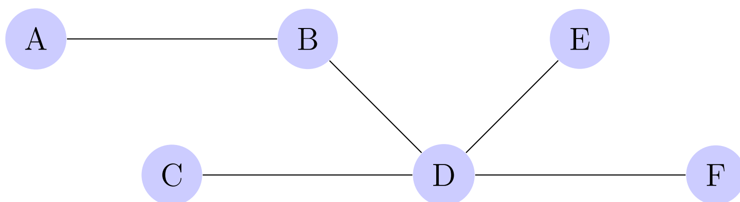
It may or may not include all the edges because it is not a cycle
For example:



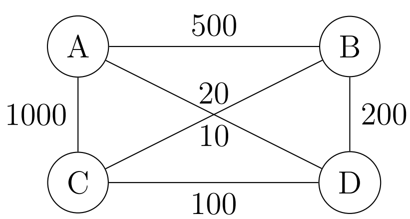
Now without making a cycle, we have to choose vertices and visit all the vertices, so it will look like



Another Spanning tree would look like



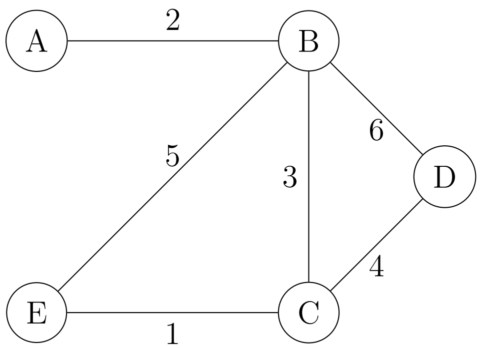
If this is a weighted graph, then it makes minimum Spanning tree



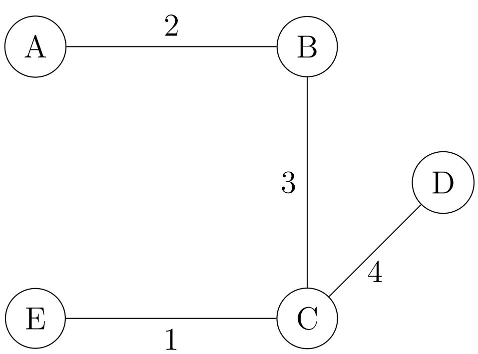
If you have to travel all the cities with minimum cost, then which path would you take.

To answer this we have two minimum Spanning trees -
(i) Prim
(ii) Kruskal

6.2 Prims Algorithm

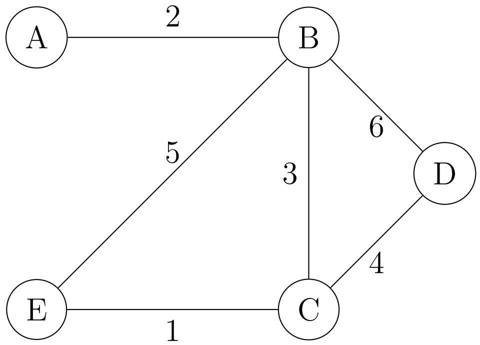


Use Greedy algorithm and find the min cost on the vsisted nodes and travel all the nodes. Start with any node, now from A goto B, so B will be visited, and cost will be 2. From B we can go to C, D, E, so we choose the min, i.e. C and cost is 2+3.
From C we go to E or D, so we visit E with cost 1.
We are left with D, so we can chose from 6 or 4. We select the edge with distance 4.
Visited Nodes: A, B, C, E, D
And sum of visiting: 2+3+1+4 10
And the visited tree will be like



6.3 Kruskal's Algorithm

We take the same Graph as before

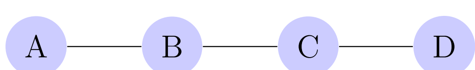


We start with edges instead of vertices, so we sort the edges and start choosing in a way that it does not make a cycle.
We select edge with weight 1 and then 2 and then 3 and then 4.
If there is a cycle on choosing it, we do not consider it.
Complexity will be sorting the weight of edges and will be
10 pow—E—Log—E—
where E is the Edge

An application of it is image segmentation problem (Paper published in 2004)
The task is to separate the image and tree, i.e. foreground and background.
To solve this we create a min spanning tree.

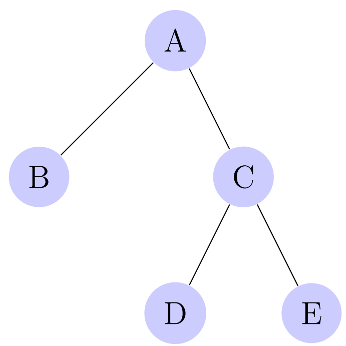
6.4 Matching and Covers

Matching:- Matching is a set of non-loop edges with no sharing points
Lets say the Graph is:



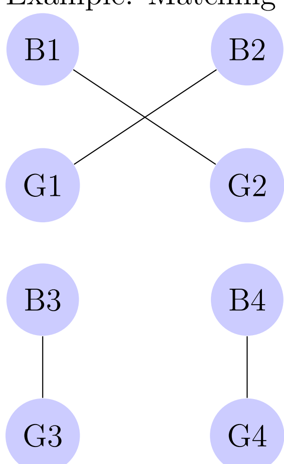
Matching: {AB, CD}
This is more like pairing vertices. Another possibility could be {BC}, this will be a single match, no other matching can happen in this Graph. If something is matched, the same node cannot be matched with other.

Let us consider a Tree

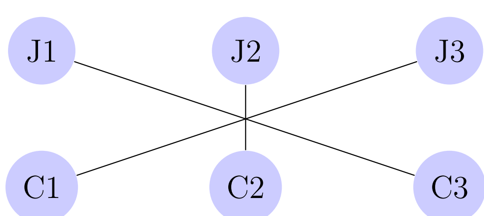


The possible matching sets will be
M1 = {AB, CD}
M2 = {AB, CE}
M3 = {AC}
M4 = {CD}
M5 = {CE}

Perfect Matching: If we have all possible matches in a set then it is perfect matching.
Example: Matching with Boys to Girls



These are perfect matching where every Boy is matched with every Girl.
If we have 3 jobs and 3 CPUs

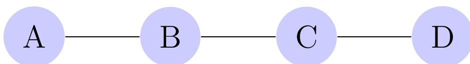


Job1 requires 8MB of memory, Job2 requires 12MB and Job3 requires 20MB of memory

CPU1 supports 20MB, CPU2 supports 20MB and C3 supports 8MB of memory
Matching: {J1C3, J2C2, J3C1} is a perfect matching

6.5 Maximum and Maximal matching

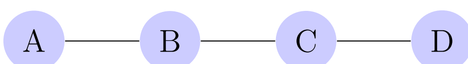
Lets say the Graph is:

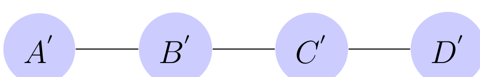


The matching of maximum sites
Max matching: when you cannot anymore match for example {AB, CD}
Maximal matching: Cannot be in last by adding an edge for example {BC}
If we add one edge AB or CD, we cannot add anything else.
{AB, CD} is also maximal matching

Perfect matching: Perfect matching or #PM of components of a Graph = #PM$_{C1}$ x #PM$_{C2}$ x .... x #PM$_{Cn}$

Lets say there are two Graphs as:

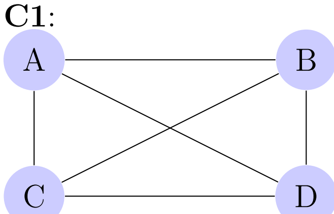
C1: 

C2: 

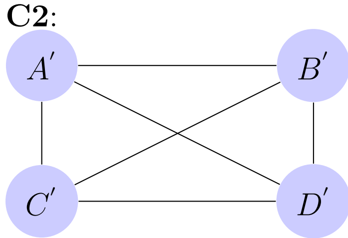
Then the perfect matching in C1 = {AB, CD}
PM in C2 = {A'B$'$, C'D$'$}

Example of more than 1 perfect matching:

C1:



C2:



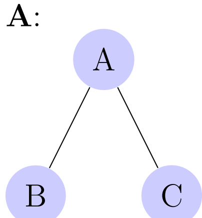
C1 = {AB, CD}
{AD, BC}
{AC, BD}

C2 = $\{A'\text{B}', C'\text{D}'\}$
$\{A'\text{D}', B'\text{C}'\}$
$\{A'\text{C}', B'\text{D}'\}$

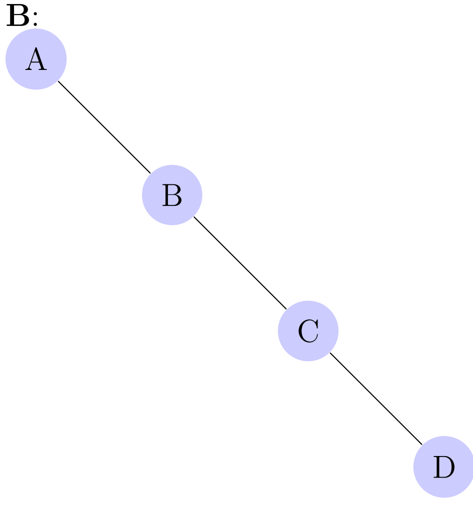
Suppose C1 C2 is a graph then there are 9 perfect matching, because we can choose any 3 in first condition and Any 3 in next condition, so there is a combinations will be multiplication of perfect matching.

Prove that every tree has at most one perfect matching:

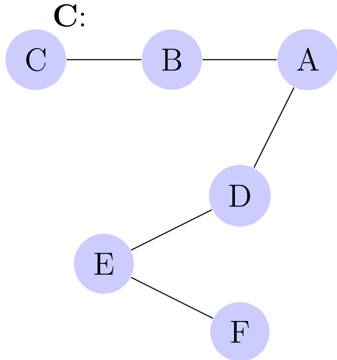
A:



B:



C:



Solution:
Perfect matching in A is Zero, as if we match AB, then C is left

#PM for B = {AB, CD}, so number of matching is one
M2 = {BC} is also a matching (a maximal match) but not a perfect matching.

#PM for C = {BC, AD, EF} that is 1
So the at most matching is coming out to be 1.

Proof:
Proof by induction -
Base condition :- A tree having n=1 node has Zero perfect matching.
For n=2 #PM = 1
Therefore max PM = 1 for tree (base condition)

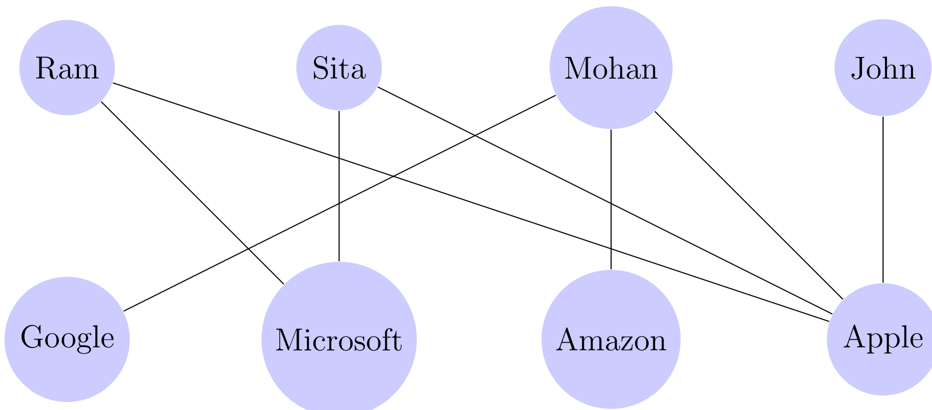
Induction hypothesis: Suppose the theorem is true for $n \leq k$ nodes.
Suppose we have any tree where we have k nodes, so the #PM = 1. If we add one node to the Graph then k+1 node Graph can be constructed in many ways.

We can add it to leaf or end node. There can be two possibilities that the new node is connected to only one node earlier.
We can take out its parent, we can make another graph. We name it T_2 and the rest is T_1.

Therefore #PM in $T_1 \leq 1$
#PM in T_2 can be Zero or 1 i.e. ≤ 1
Number of matches in graph = #PM in T_1 x #PM in T_2
≤ 1 x 1
≤ 1

6.6 Marriage Theorem:-

Let's say there are 4 companies: Google, Microsoft, Amazon, Apple which are coming for placements and lets say there are 4 students eligible for these companies Ram, Sita, Mohan and John



Apple has less constraint so all are eligible for Apple.
Only Mohan for Amazon
Only Mohan for Google
Ram, Sita for Microsoft

Only one position is there for each company, then the question is will every candidate get a job?
Ans: No
If everyone has to get a job then John has to get a job in Apple, Google has only one option Mohan. If Mohan is selected in Google, then a position is left at Amazon.
This theorem is for bi-partite Graph. The theorem is an X-1 bigraph G has a matching that saturates X iff
$\|N(S)\| \geq \|S\| \forall$ S $\subseteq$ X
Cardinality of N(S) $\geq$ Cardinality of S
Here N(S) is a subset of Y is a set of neighbours of elements in S.