Math 239 - Introduction to Combinatorics	**Spring 2017**

Lecture 24: June 24th, 2017

Lecturer: Alan Arroyo Guevara *Notes By: Harsh Mistry*

Missed This Lecture, Notes Taken By Xyan Bhatnagar

Theorem 24.1 *If T is a spanning tree of G and $e \in E(G) \setminus E(T)$ then $T + e$ has a unique cycle c. Moreover for every $e' \in E(C) \setminus \{e\}, T' := (T + e) - e'$ is a spanning tree*

Notation : $T + e = (V(T), E(T) \cup \{e\})$

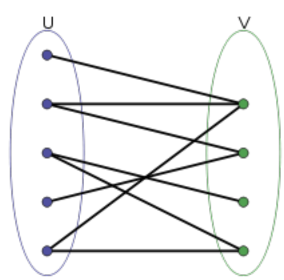
Proof: Let e = xy and Let P be a unique xy-path in T.
As every cycle in $T + e$ contains e. $C = P + e$ is the unique cycle in $T + e$

Let $e' \in E(P) \subseteq E(C)$. As e' is not a bridge in $T + e$. Then $T' = (T + e) - e'$ is connected

$$| E(T') | = | E(T) | = | V(T) | -1 = | V(T') | -1$$

By previous theorem, T' is a tree and $| V(T') | = | V(T) | = | V(G) |$. So it is a spanning tree ■

24.1 Characterizing Bipartite Graphs



G is bipartite $\iff$ V(G) can be coloured using 2 colours so that no 2 adjacent vertices have the same color

Observations

- A graph that contains a cycle of off length (odd cycle) as a sub graph is **not bipartite**.
- Trees are bipartite

Theorem 24.2 *A graph is bipartite if and only if it has no odd cycles as a sub-graph*

Proof:

→ Use previous observations

← Suppose G has no odd cycles. Assume that G is connect (or we treat components separately). Let T be a spanning tree of G and Let $A \cup V = V(T)$ be a bipartition of T.

We claim that $A \cup B$ is also a bipartition of G. Suppose not, then there exists $e = uv \in E(G) \setminus E(T)$ where u and v both belong to A or B.

Let $u, v \in A$ and Let P be the uv-path in T.
Length(P) is even because $u, v \in A \implies p + e$ is an odd cycle. **Contradiction**

■

24.2 Breadth-First Search Trees (BFS)

Definition 24.3 *A queue is an ordered list*

$$Q : q_1, q_2, q_3, \ldots, q_n$$

where every new element added to Q is added at the end. The first element in Q is the ***active element***

BFS Algorithm

Input: Graph G and $r \in V(G)$

Output: Spanning Tree T or that G is not connected

Process:

1. Initialization
   $T = (\{r\}, \phi)$ r = vertices and ϕ = edges
   $P : V(G) \to V(G) \cup \{null\}$
   $P(v) \to null$
   $Q = r$
2. Operations

```
While Q != phi
  Let u be the active vertex in Q
  While u has a neighbour not in V(T)
    Add v and uv to T
    Let P(v) = u
    Add v to Q
  Remove u from Q
```

3. Output (T,P)