# RECITATION 1
# ANALYSIS OF ALGORITHMS II

2014 SPRING

# PROBLEM 1

Suppose you have algorithms with the five running times listed below. (Assume these are the exact running times.) How much slower do each of these algorithms get when you (a) double the input size, or (b) increase the input size by one?

**(a)** $n^2$

**(b)** $n^3$

**(c)** $100n^2$

**(d)** $n \log n$

**(e)** $2^n$

# PROBLEM 1

- When the input size is doubled, the algorithms gets slower by

$(2n)^2$ → the factor of 4 *($n^2$)*

$(2n)^3$ → the factor of 8 *($n^3$)*

$100x(2n)^2$ → the factor of 4 *($100n^2$ )*

2n log2n → a factor of 2, plus an additive 2nlog2 (nlogn)

$2^{2n}$ → the square of previous running time ($2^n$ )

# PROBLEM 1

- When the input size is increased by additive one, the algorithms get slower by

$(n+1)^2 \rightarrow$ an additive $2n+1$ *($n^2$)*

$(n+1)^3 \rightarrow$ an additive $3n^2+3n+1$ *($n^3$)*

$100x(n+1)^2 \rightarrow$ an additive $200n+100$ *($100n^2$ )*

$(n+1)\log(n+1) \rightarrow$

an additive $\log(n+1)+ n[\log(n+1) – \log n]$ (nlogn)

$2^{n+1} \rightarrow$ a factor of 2 ($2^n$ )

# PROBLEM 2

What is the main data structures employed when implementing DFS and BFS?

- ✓ BFS→QUEUE (FIFO Queue can be used)

We trace the graph layer by layer by considering all of the children of a node before starting to trace nodes further away

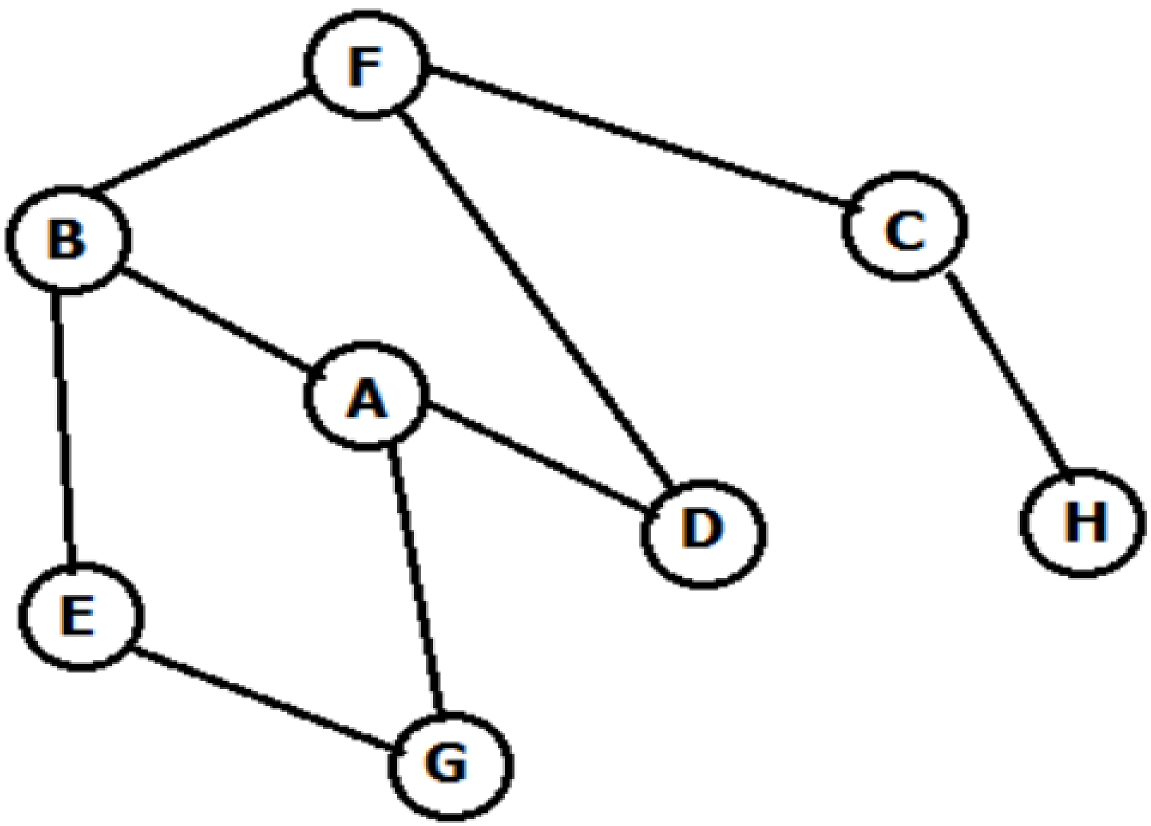
- ✓ DFS → STACK

Algorithm considers the immediate unexplored children before considering the other children of a node while moving from parent node to child node.

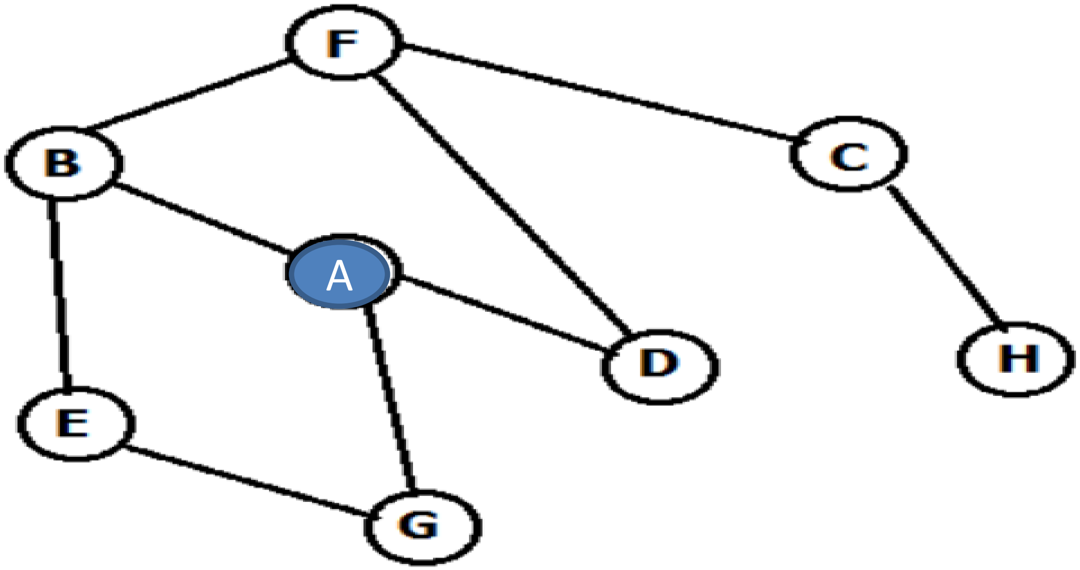
It goes deeper through the branches of the graph, before tracing other branches.

# PROBLEM 3

- Apply BFS and DFS to the following graph



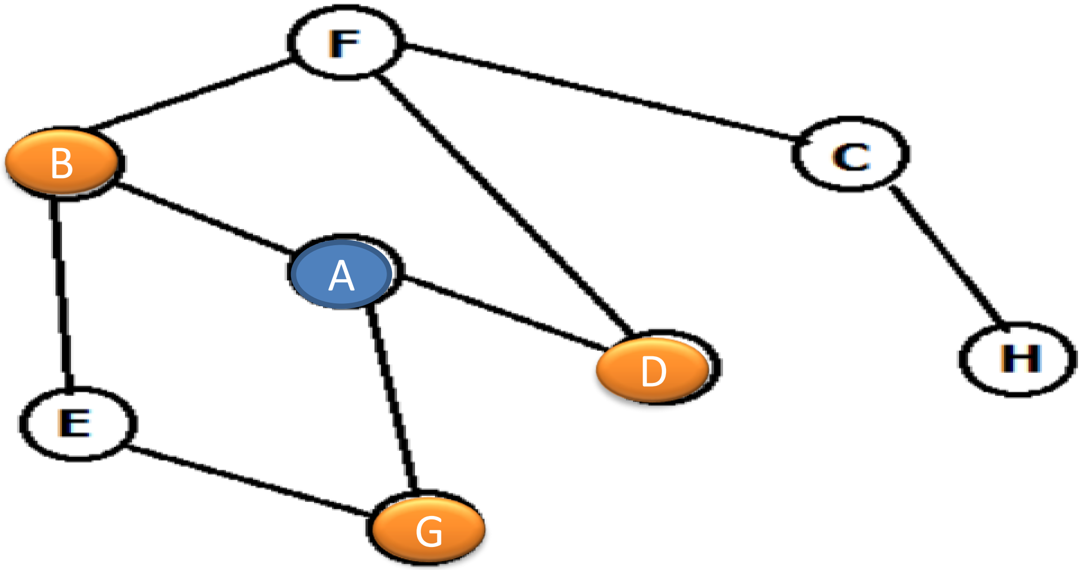
# PROBLEM 3- BFS




BFS Tree= {}

L0={A}

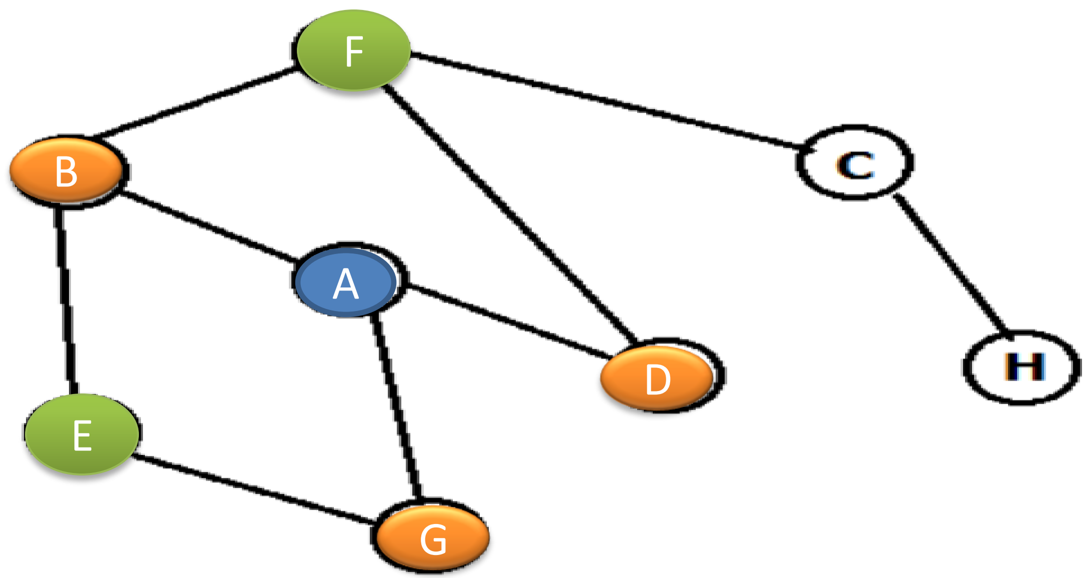
# PROBLEM 3- BFS




BFS Tree= {(A,B),(A,G),(A,D)}

Layers= L0={A}, L1={B,G,D}

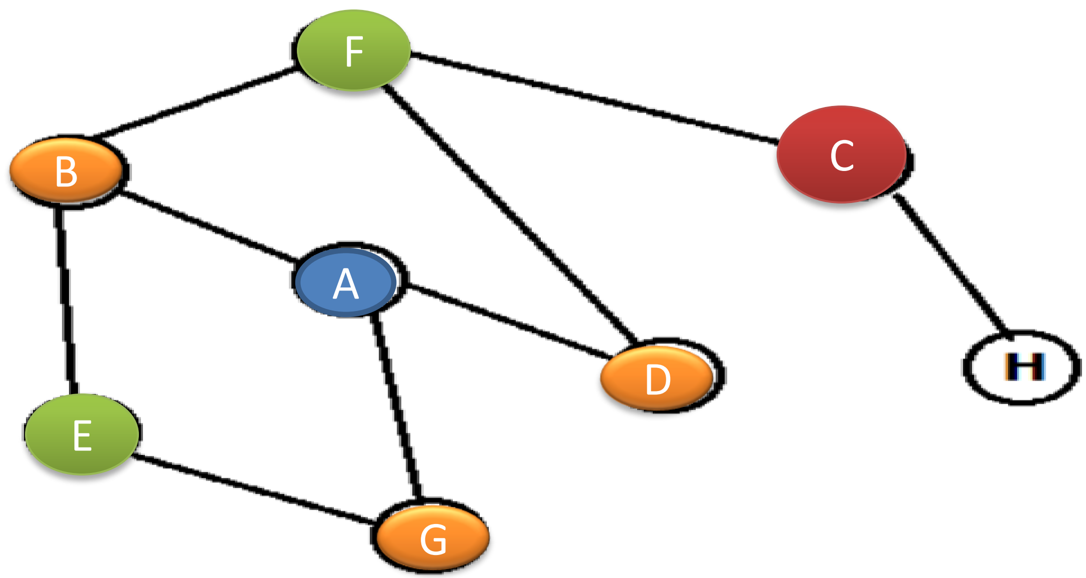
# PROBLEM 3- BFS




BFS Tree= {(A,B),(A,G),(A,D), (B,E),(B,F)}

Layers= L0={A}, L1={B,G,D}, L2={E,F}

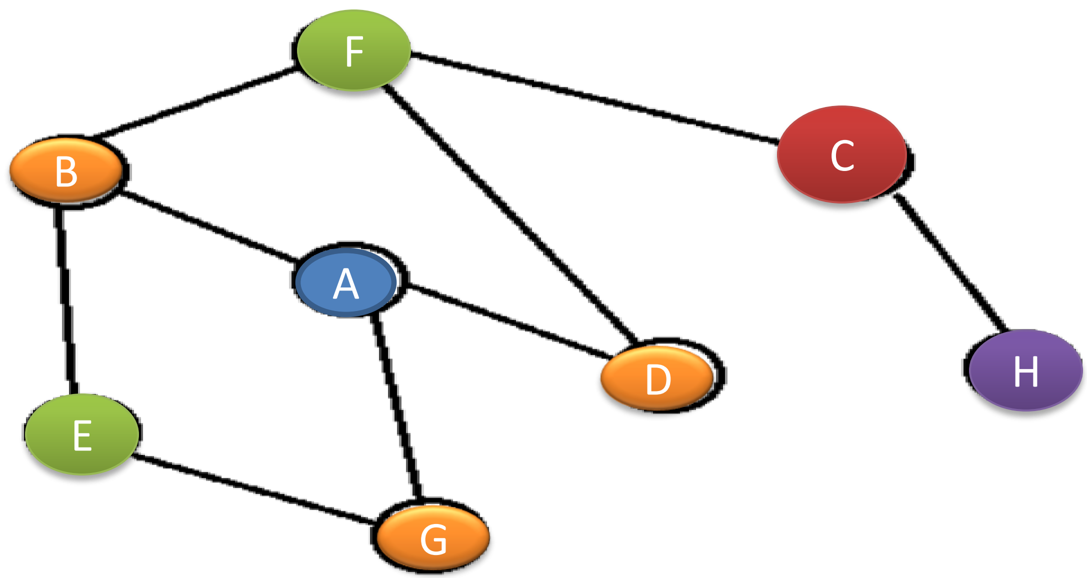
# PROBLEM 3- BFS




BFS Tree= {(A,B),(A,G),(A,D), (B,E),(B,F), (F,C)}

Layers= L0={A}, L1={B,G,D}, L2={E,F}  L3={C}

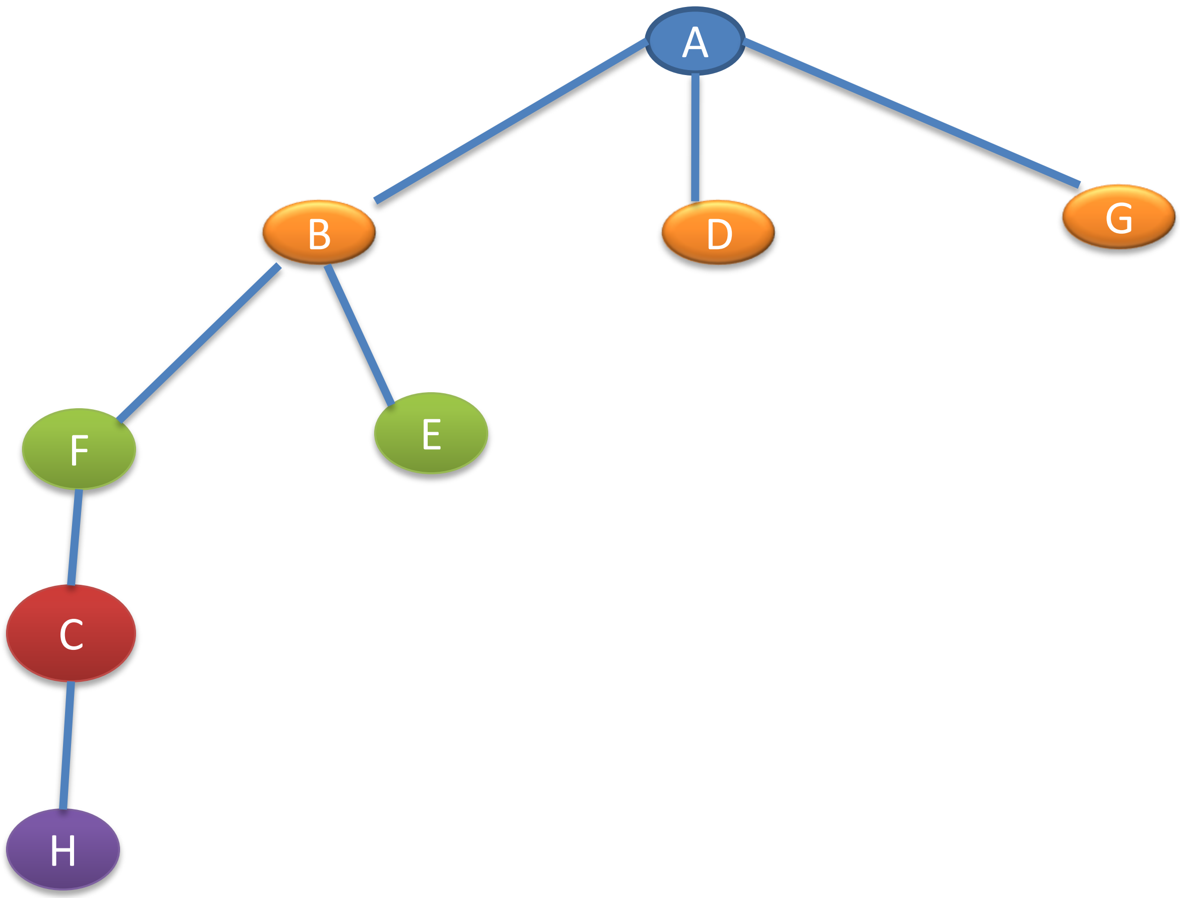
# PROBLEM 3- BFS




BFS Tree= {(A,B),(A,G),(A,D), (B,E),(B,F), (F,C), (C,H)}

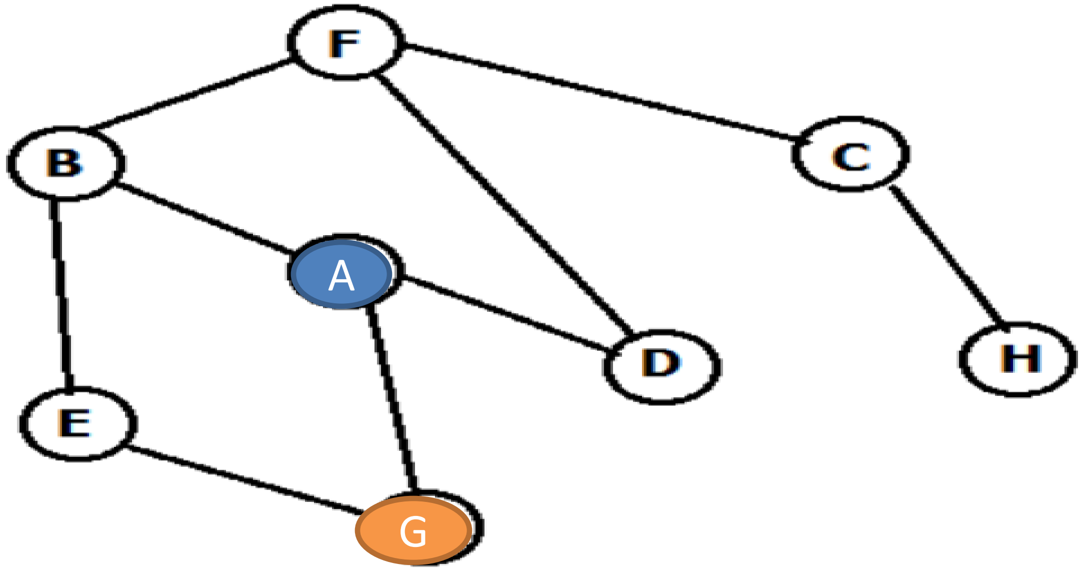
Layers= L0={A}, L1={B,G,D}, L2={E,F} L3={C} L4={H}

# PROBLEM 3- BFS TREE




BFS Tree= {(A,B),(A,G),(A,D), (B,E),(B,F), (F,C), (C,H)}
Layers= L0={A}, L1={B,G,D}, L2={E,F} L3={C} L4={H}

# PROBLEM 3- DFS

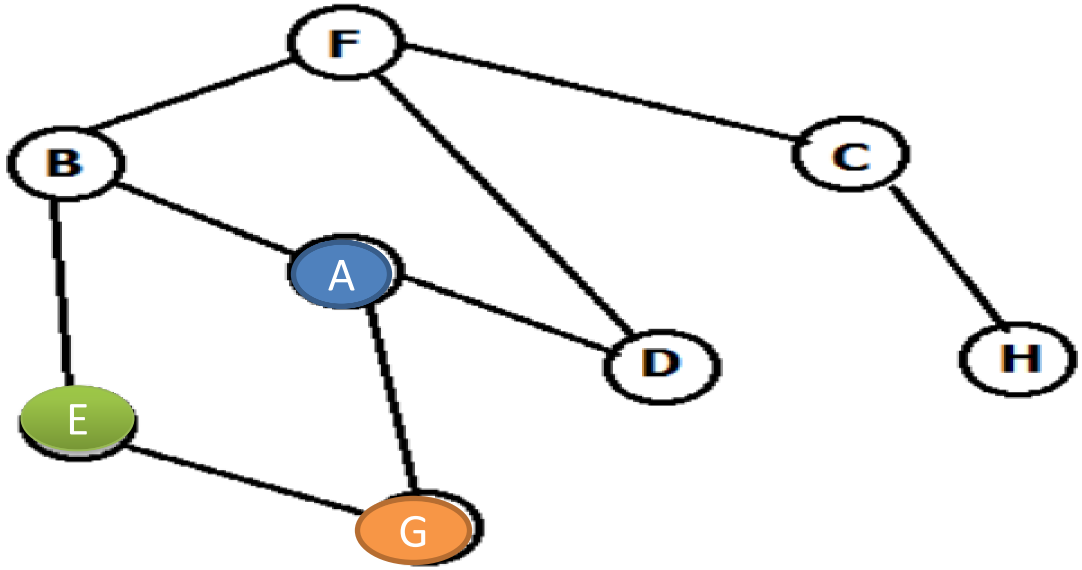



DFS Discovered Order= {A}

Stack={B,D,G}

DFS Tree= {(A,G)}

# PROBLEM 3- DFS

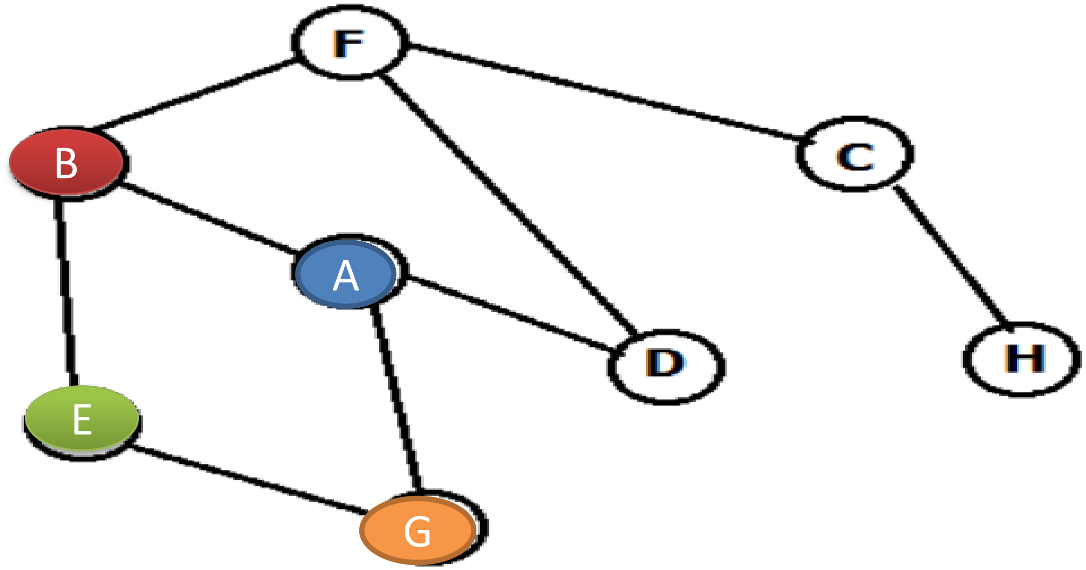



DFS Discovered Order= {A,G}

Stack={B,D,E}

DFS Tree= {(A,G), (G,E)}

# PROBLEM 3- DFS

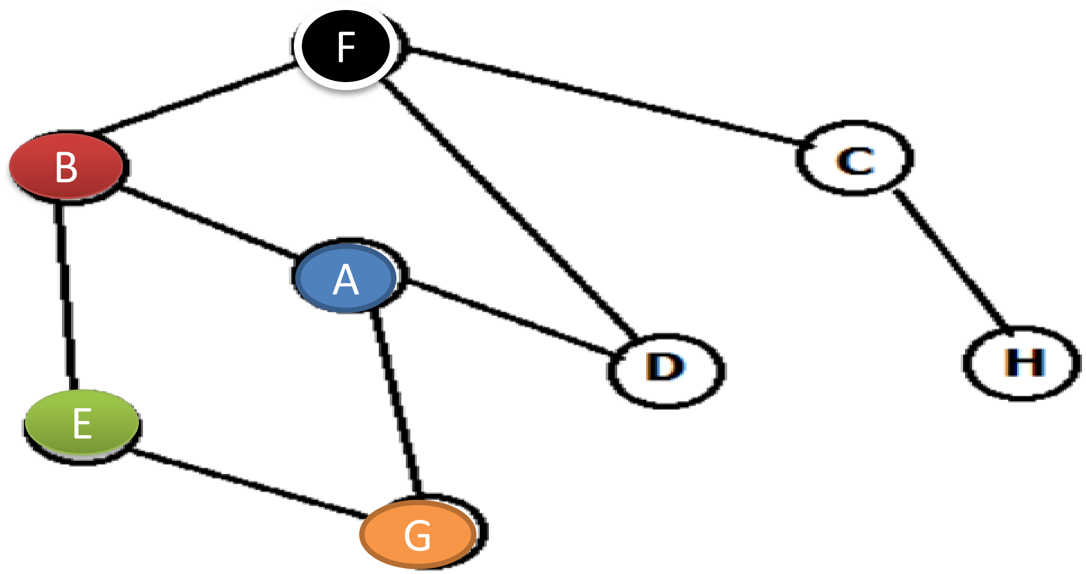



DFS Discovered Order= {A,G,E}

Stack={B,D,B}

DFS Tree= {(A,G), (G,E), (E,B)}

# PROBLEM 3- DFS

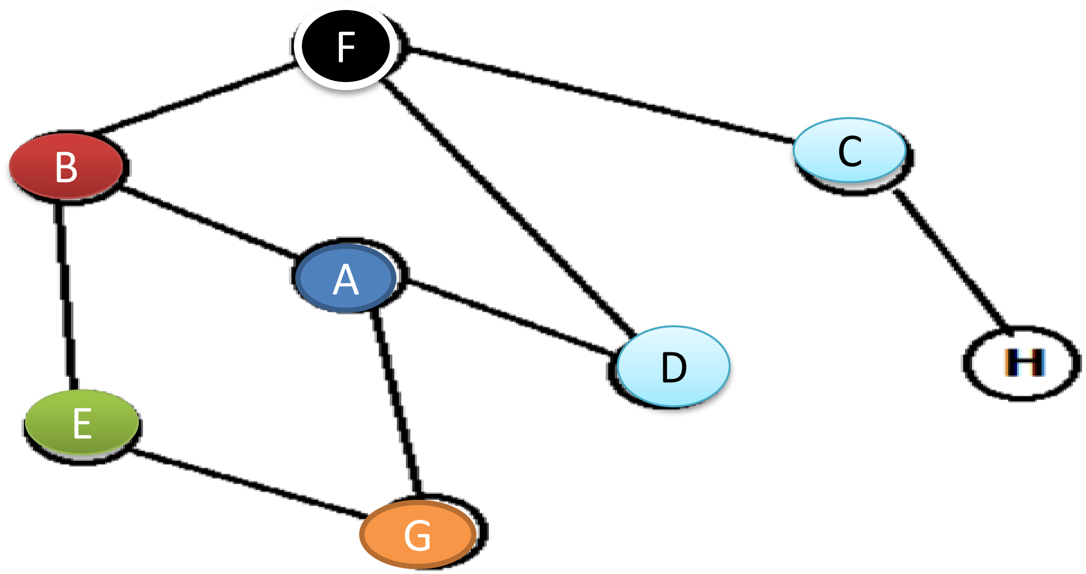



DFS Discovered Order= {A,G,E,B}

Stack={B,D,F}

DFS Tree= {(A,G), (G,E), (E,B), (B,F)}

# PROBLEM 3- DFS

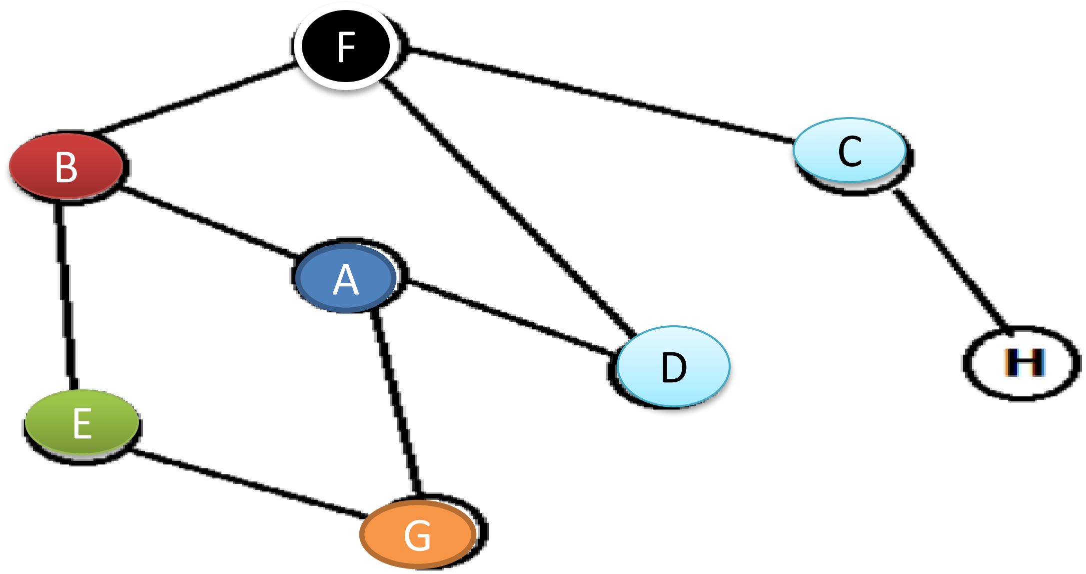



DFS Discovered Order= {A,G,E,B,F, D, C, H}

Stack={B,D,C,D}

DFS Tree= {(A,G), (G,E), (E,B), (B,F), (F,C),(F,D)}

# PROBLEM 3- DFS

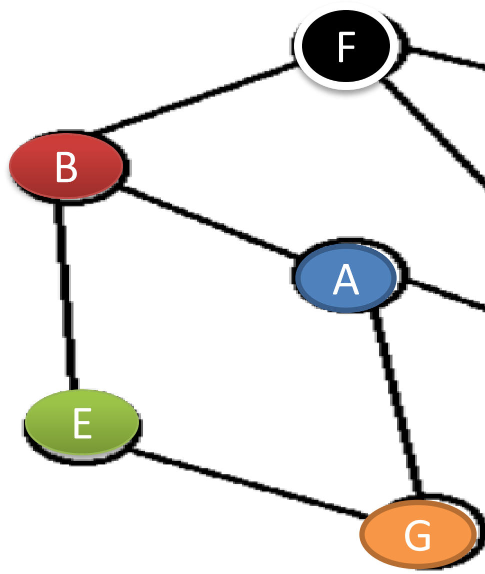



DFS Discovered Order= {A,G,E,B}

Stack={B,D,H}

DFS Tree= {(A,G), (G,E), (E,B), (B,F), (F,C), (F,D)}

# PROBLEM 3- DFS



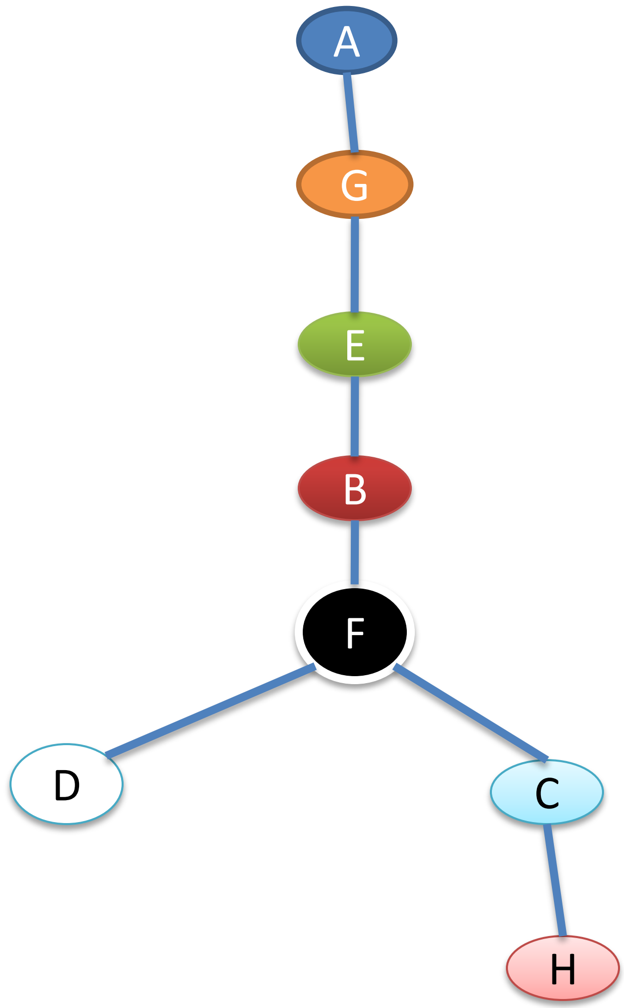

DFS Discovered Order= {A,G,E,B,F}

Stack={B,D,C}

DFS Tree= {(A,G), (G,E), (E,B), (B,F), (F,C),(F,D), (C,H)}

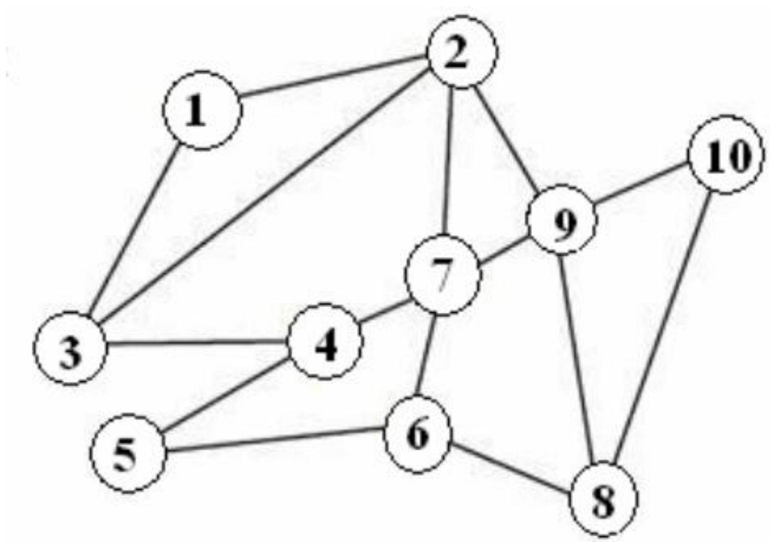
# PROBLEM 3- DFS Tree

DFS Tree= {(A,G), (G,E), (E,B), (B,F), (F,C),(F,D), (C,H)}



# PROBLEM 4

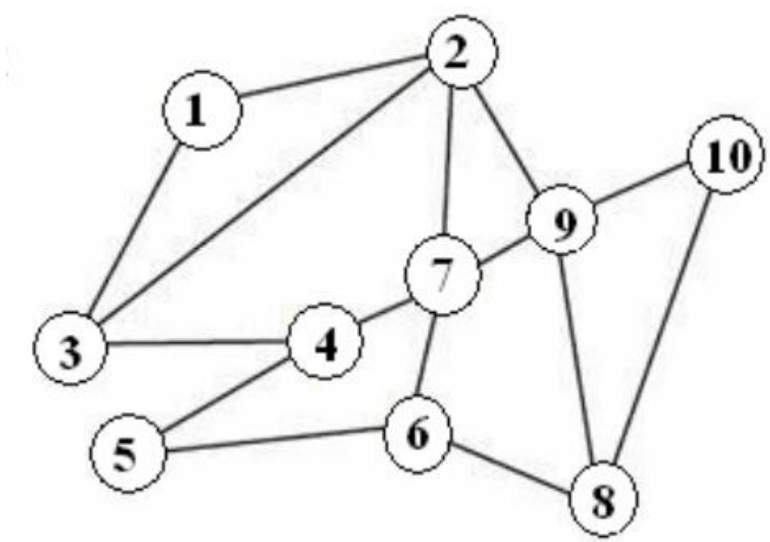
- What is a bipartite graph? Is the following graph is bipartite?




- A bipartite graph is a graph whose vertices can be grouped into two groups such that all the edges are between these two vertex group and there is no edge within a group.

# PROBLEM 4

- What is a bipartite graph? Is the following graph is bipartite?



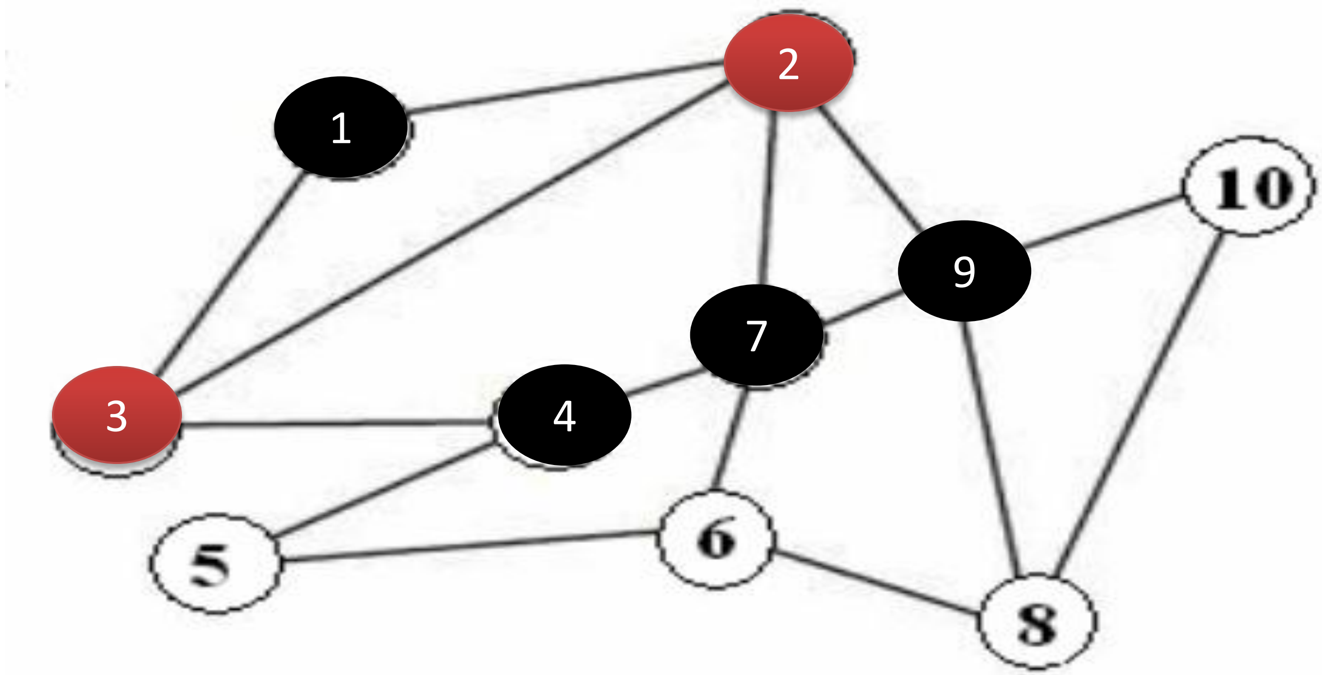

- If a graph can be two colored

OR

- If there is no odd length cycles then the graph is bipartite.

# PROBLEM 4

- What is a bipartite graph? Is the following graph bipartite?




- 4 and 7 has same color. So this graph is not bipartite.
- Also there are many odd length cycles. One of them is (1,2,3)
- **Either 2 color or odd-cycle can be checked.**

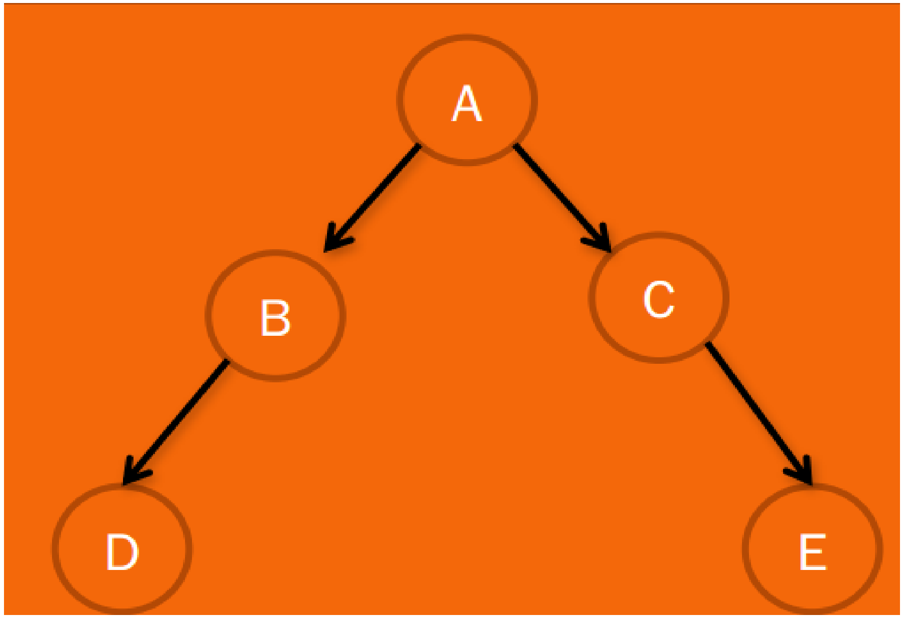
# PROBLEM 5

- The head of the department wants to prepare a prerequisite graph of the courses offered in the department.  What kind of structure should she/he form?

- He / She should construct a Directed Acyclic Graph, whose topological ordering shows the prerequisite relations between the courses.

# PROBLEM 5

- For example; A is the prerequisite of B and C, B is the prerequisite of D, and C is the prerequisite of E, the DAG he should construct is as follows:



# PROBLEM 5

- Considering the structure given in a, how can one list all the courses that are prerequisites of a given course?

  Answer: Suppose that we are trying to find the prerequisites of node (course) u.

```
Add each node v, which has an edge (v,u) to a queue Q.
 While the queue Q is not empty
        Take node X from the head of Q.
        Add X to the list_P
        Add each node v which has an edge (v,x) to the Q.
 End While
 Return list_P
```