Graph (II)

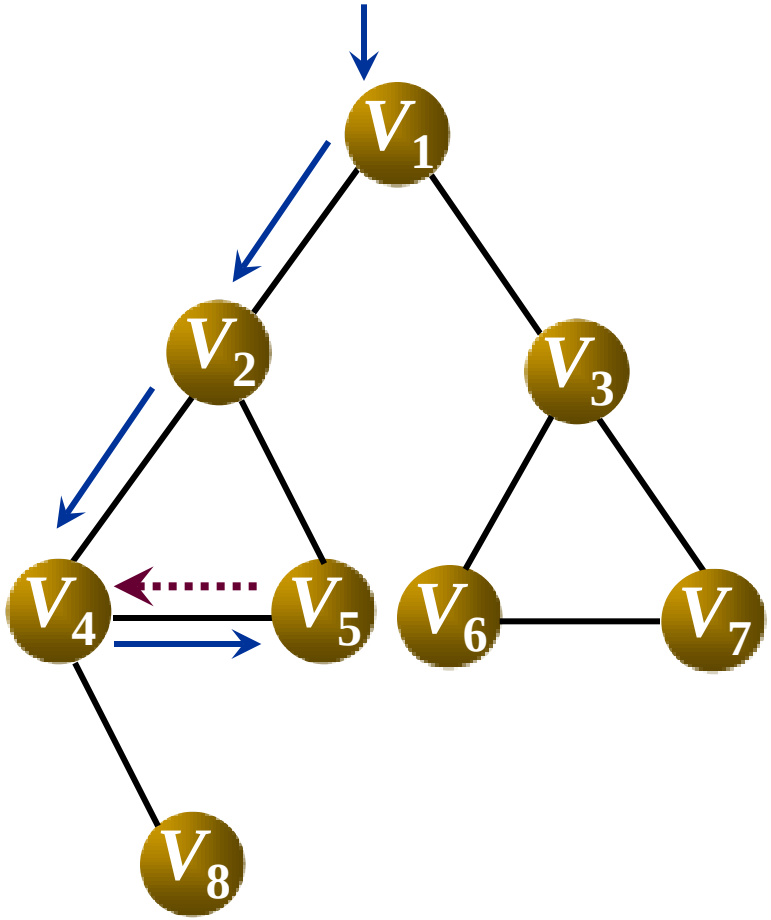
Bruce Nan

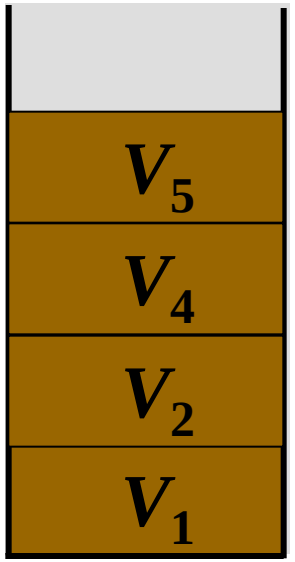
Depth Fist Search

- Depth-first search uses the same idea as backtracking.
  - Exhaustively searching all possibilities by advancing if it is possible;
  - Backing up as soon as there is no unexplored possibility for further advancement.
- Recursive algorithms.

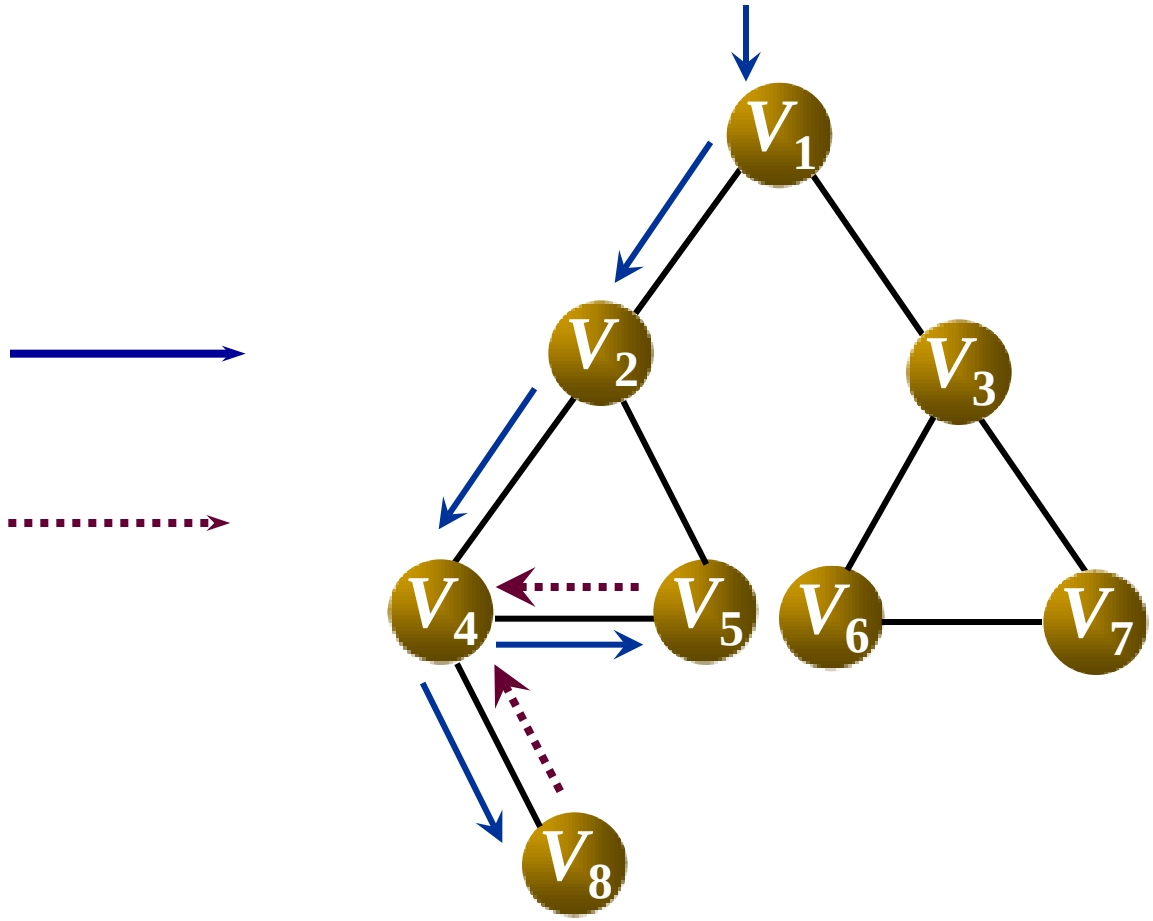
Depth First Search (DFS)

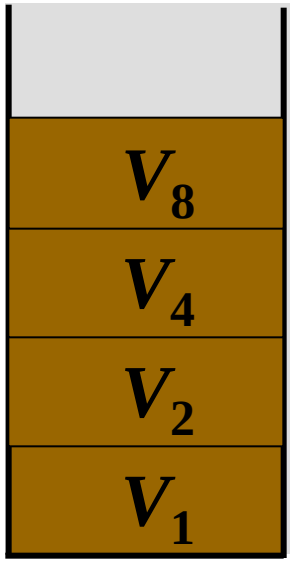
- A **depth-first search (DFS)** in an undirected graph G is like wandering in a labyrinth with a string and a can of red paint without getting lost.
- We start at vertex ***s***, tying the end of our string to the point and painting ***s*** "visited". Next we label ***s*** as our current vertex called ***u***.
- Now we travel along an arbitrary edge (***u***, ***v***).
- If edge (***u***, ***v***) leads us to an already visited vertex ***v*** we return to ***u***.
- If vertex ***v*** is unvisited, we unroll our string and move to ***v***, paint ***v*** "visited", set ***v*** as our current vertex, and repeat the previous steps.



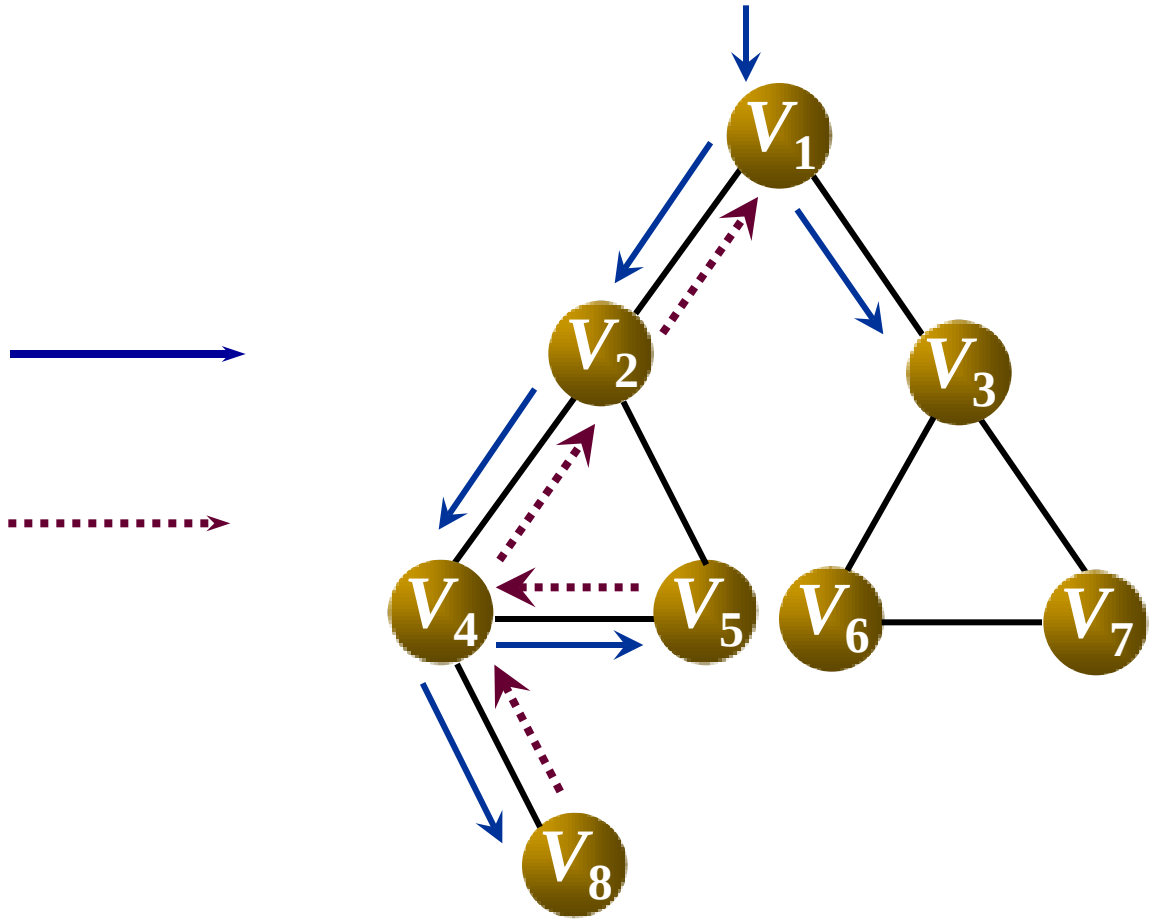


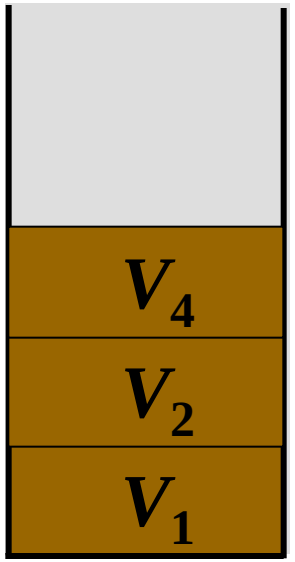
Series： V_1 V_2 V_4 V_5



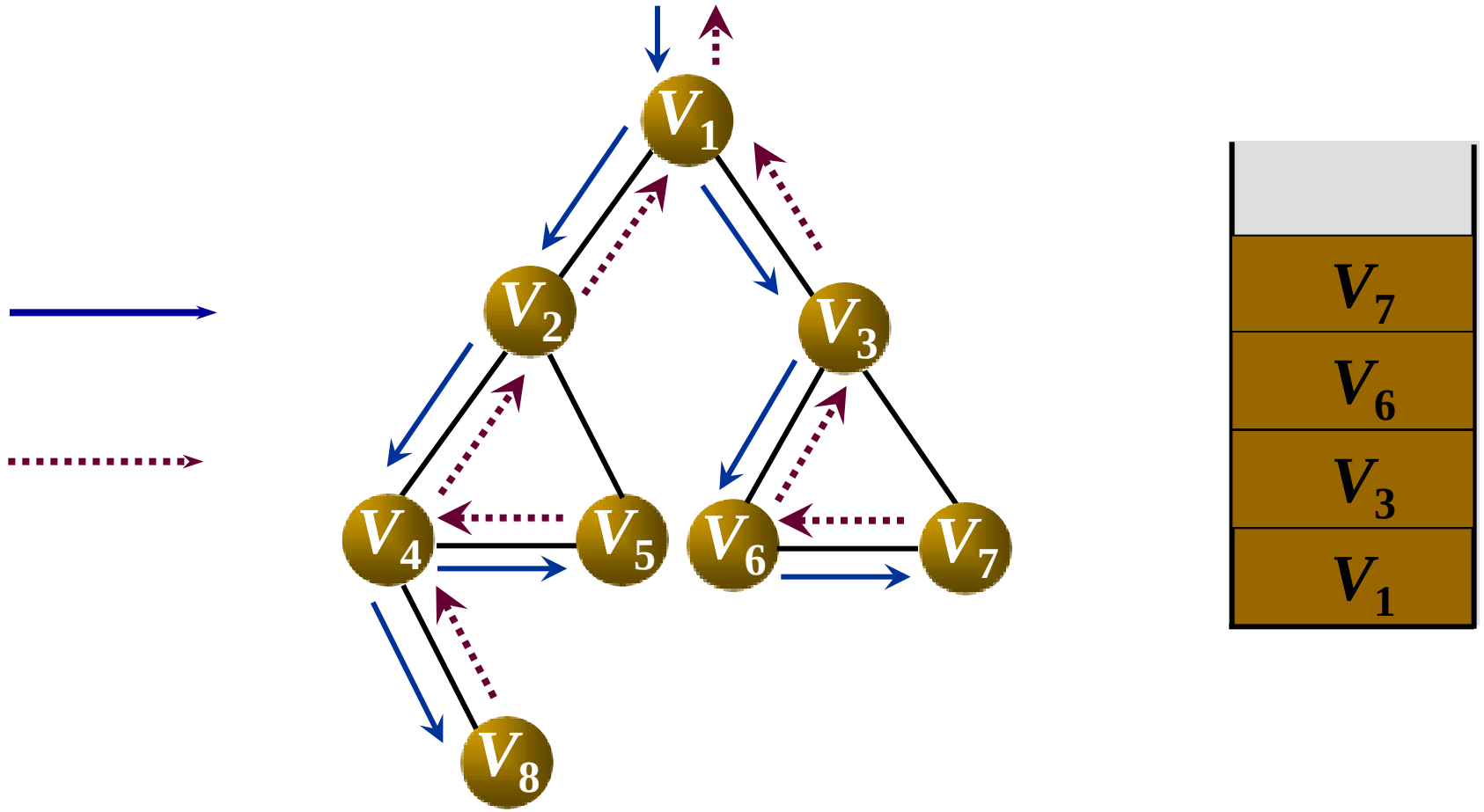


Series： V_1 V_2 V_4 V_5 V_8



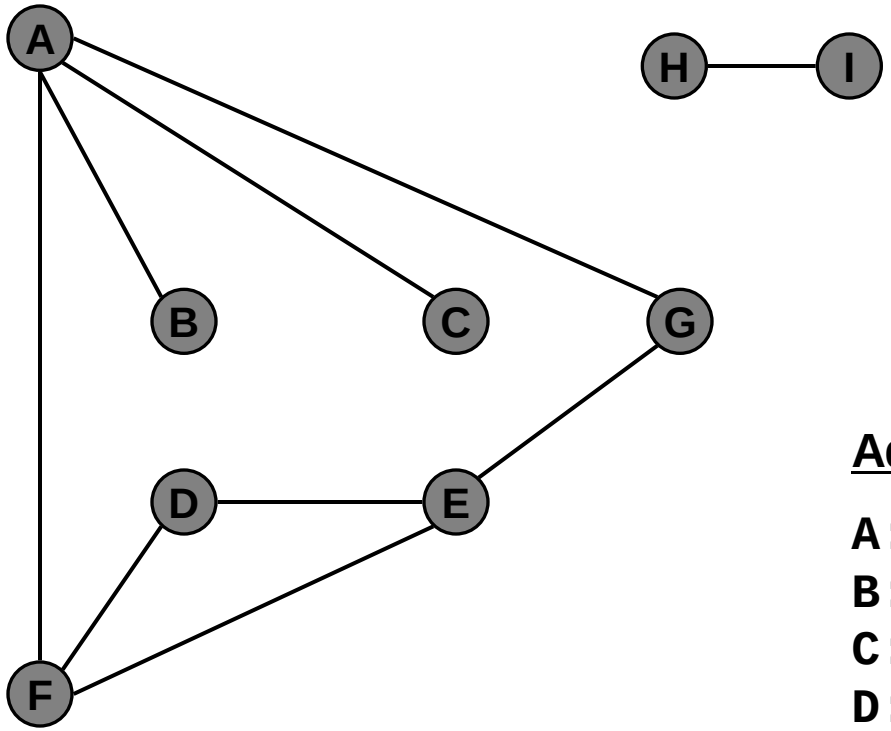


Series： V_1 V_2 V_4 V_5 V_8



Series： V_1 V_2 V_4 V_5 V_8 V_3 V_6 V_7

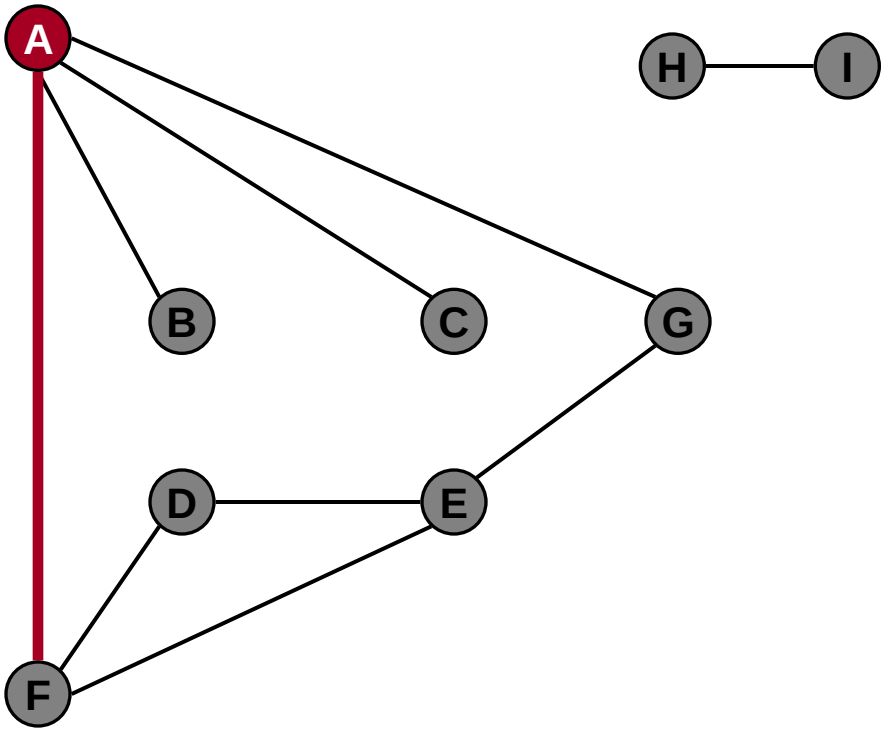
Undirected Depth First Search



Adjacency Lists

```
A:  F C B G
B:  A
C:  A
D:  F E
E:  G F D
F:  A E D
G:  E A
H:  I
I:  H
```

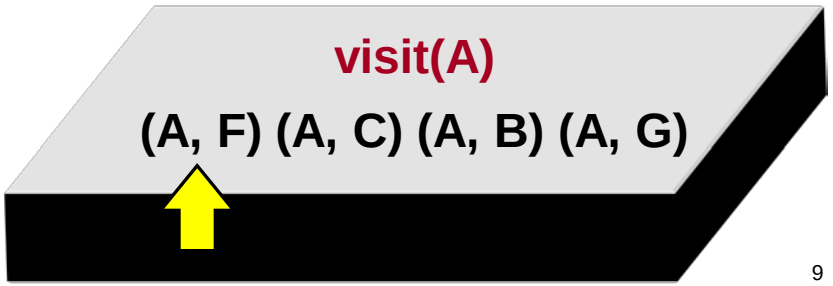
Undirected Depth First Search



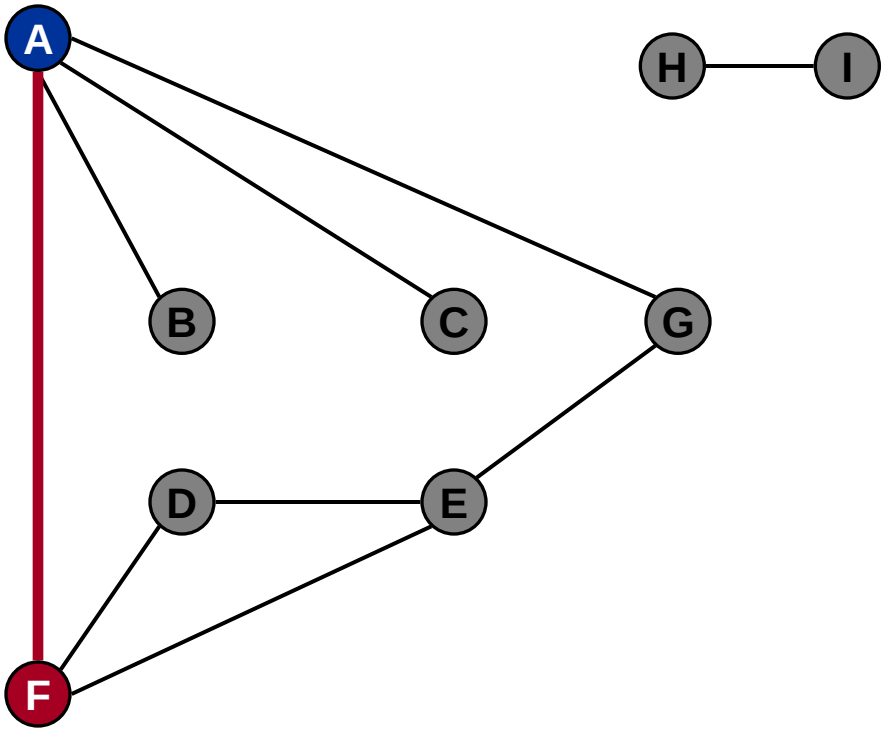
F newly discovered

Undiscovered
Marked
Active
Finished

Stack

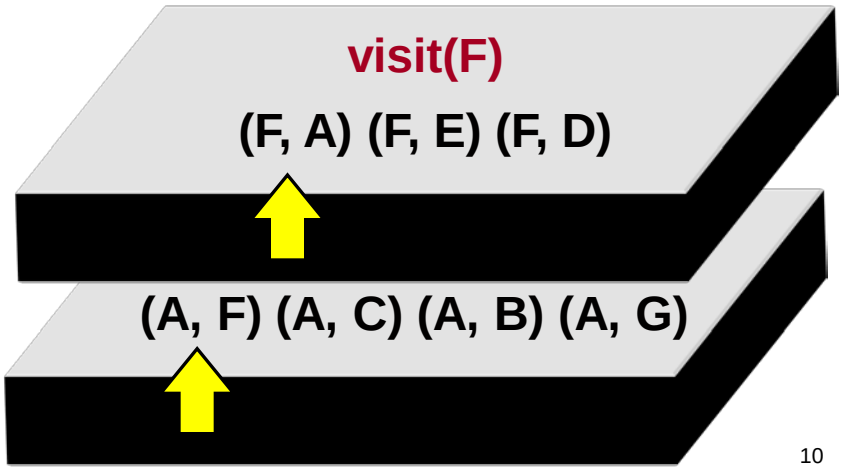


Undirected Depth First Search



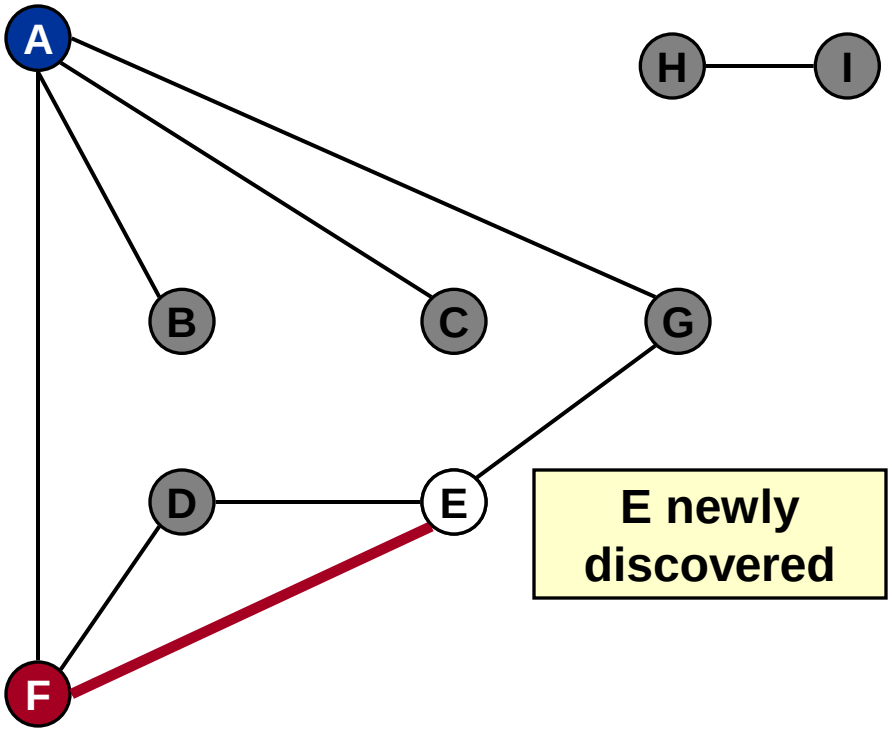
A already marked

Undiscovered
Marked
Active
Finished



Stack

Undirected Depth First Search

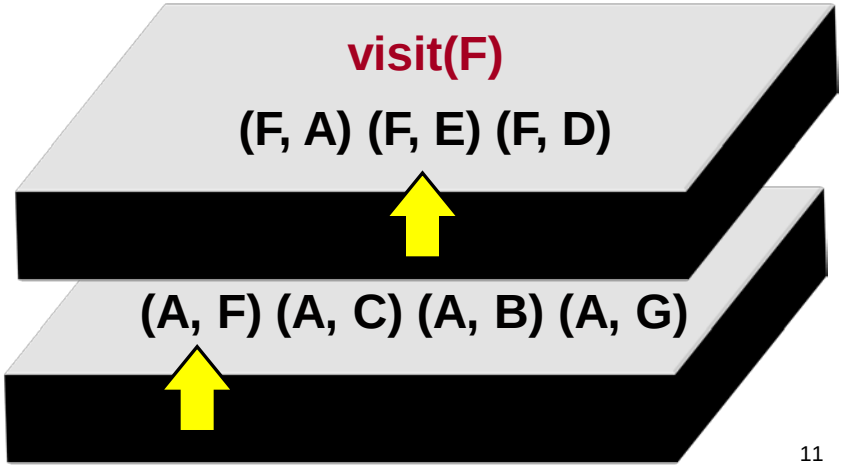


Undiscovered

Marked

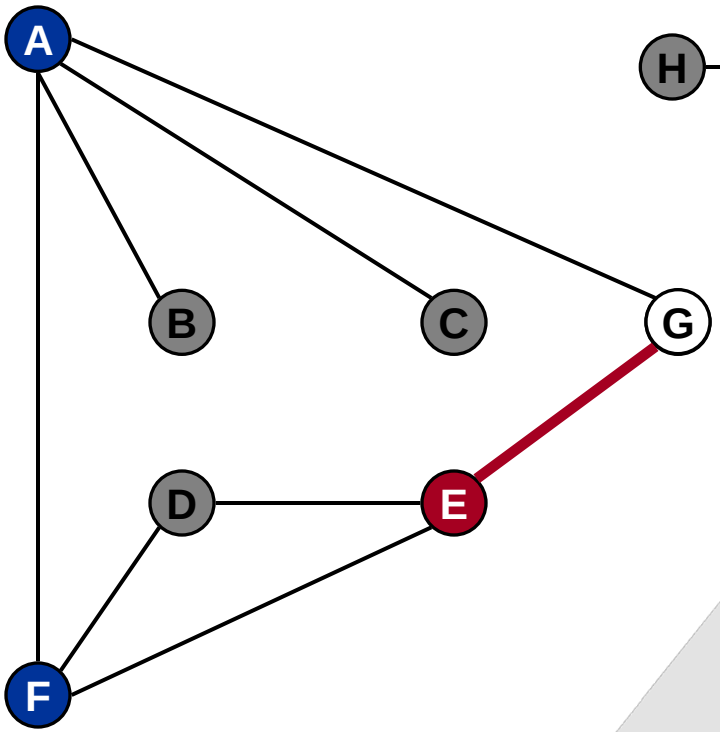
Active

Finished

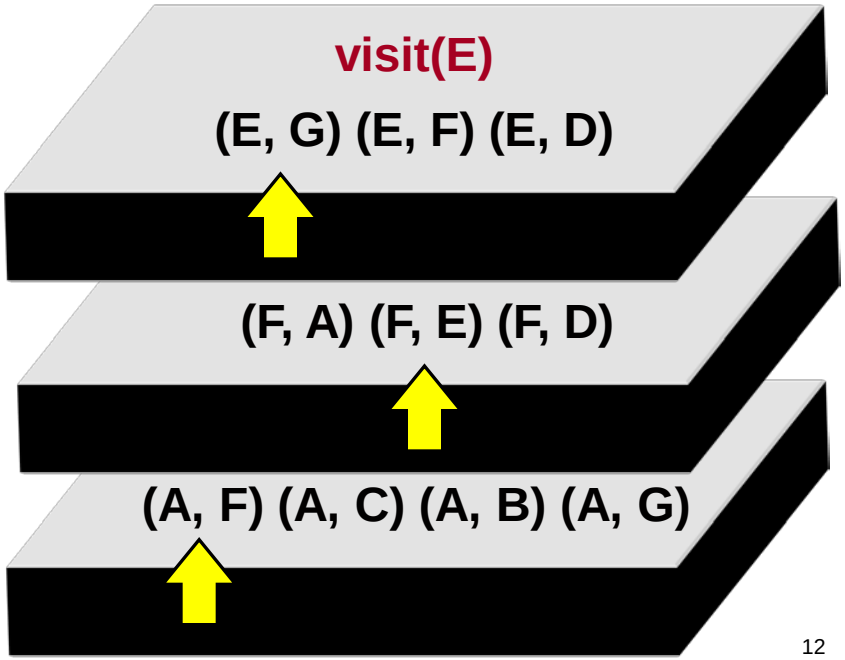


Stack

Undirected Depth First Search

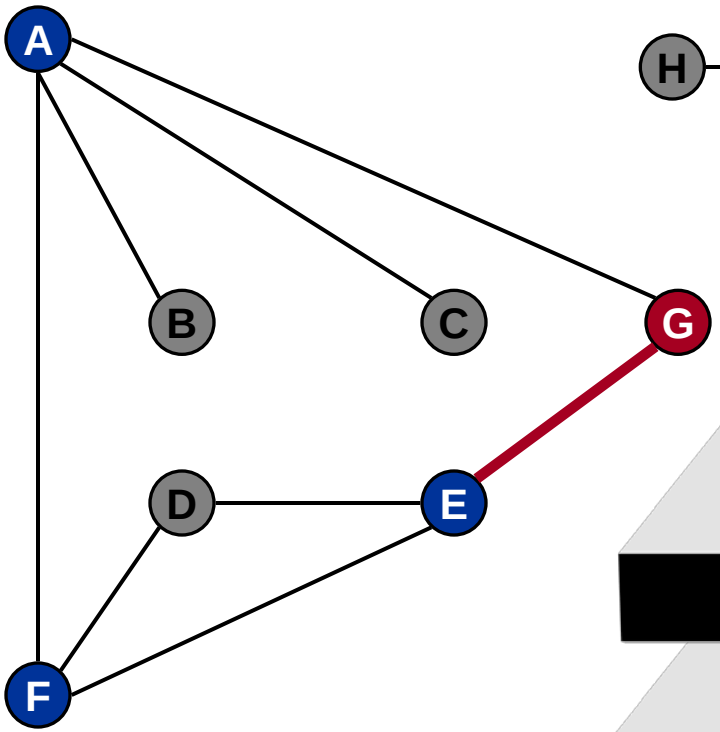


G newly discovered



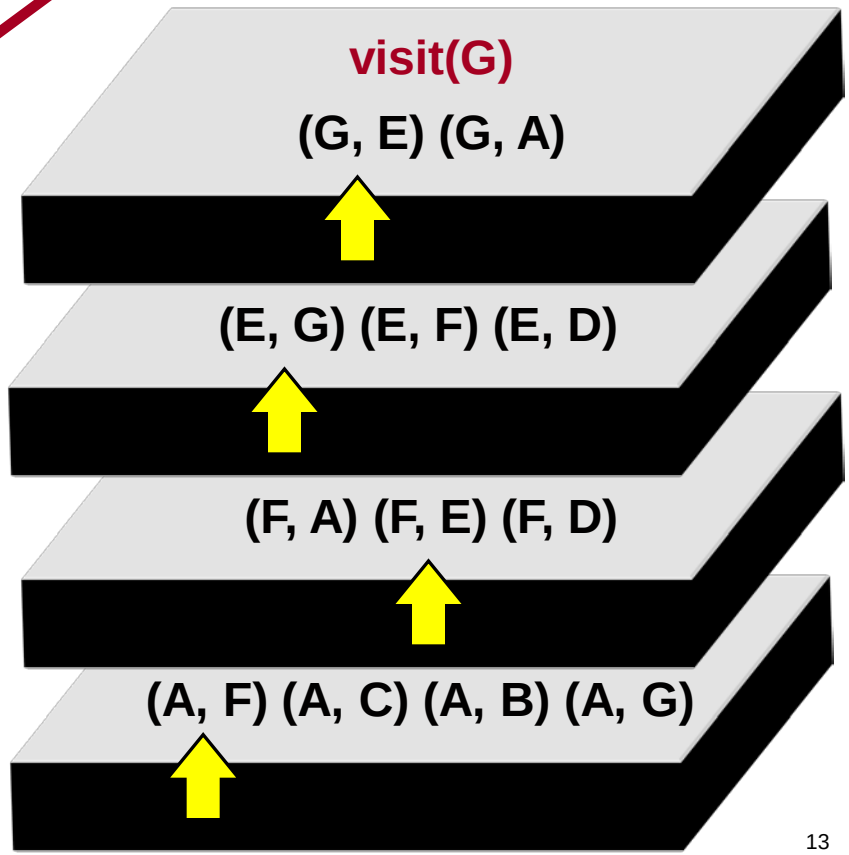
Stack

Undirected Depth First Search



E already marked

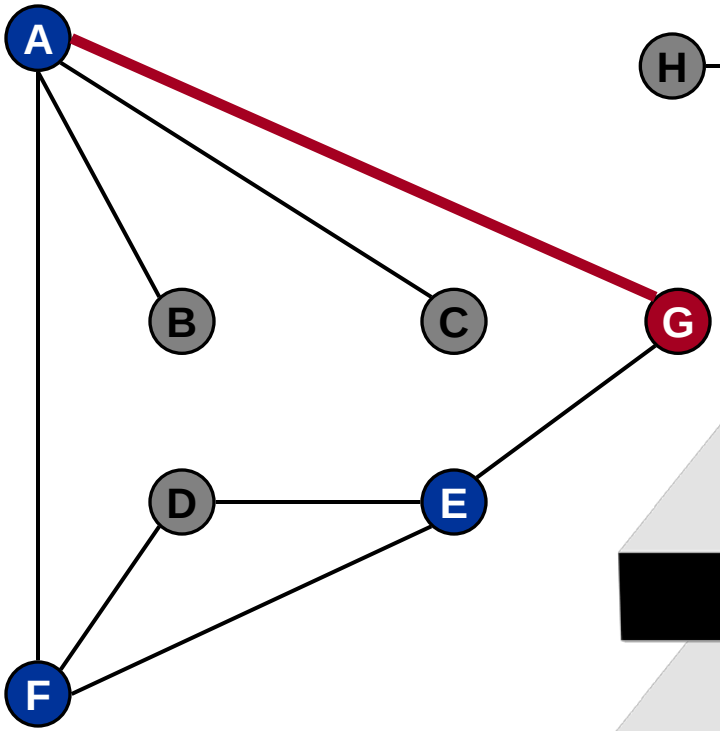
Undiscovered
Marked
Active
Finished



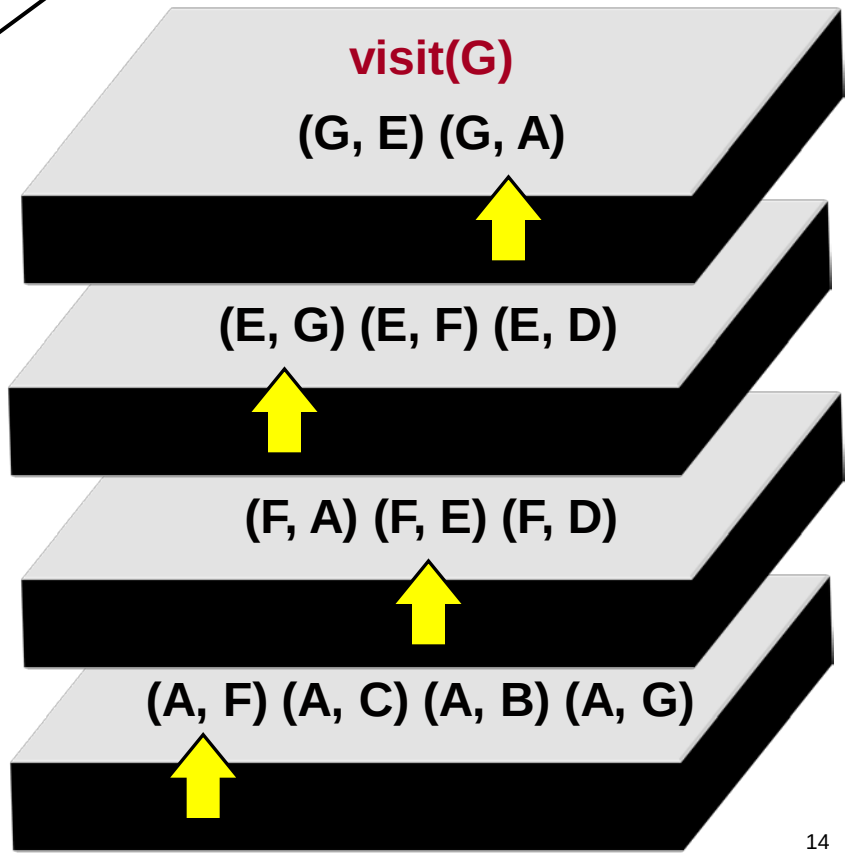
Stack

Undirected Depth First Search

A already marked

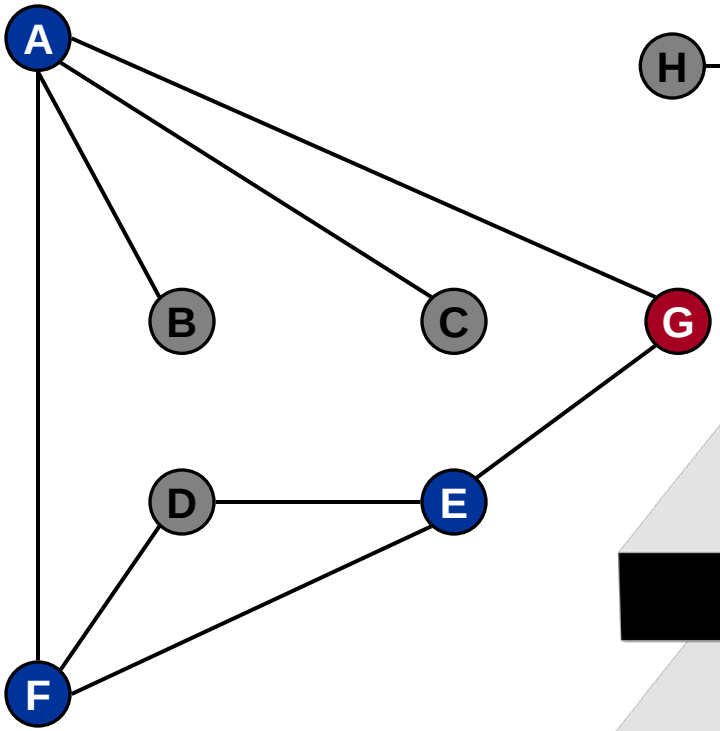


Undiscovered
Marked
Active
Finished

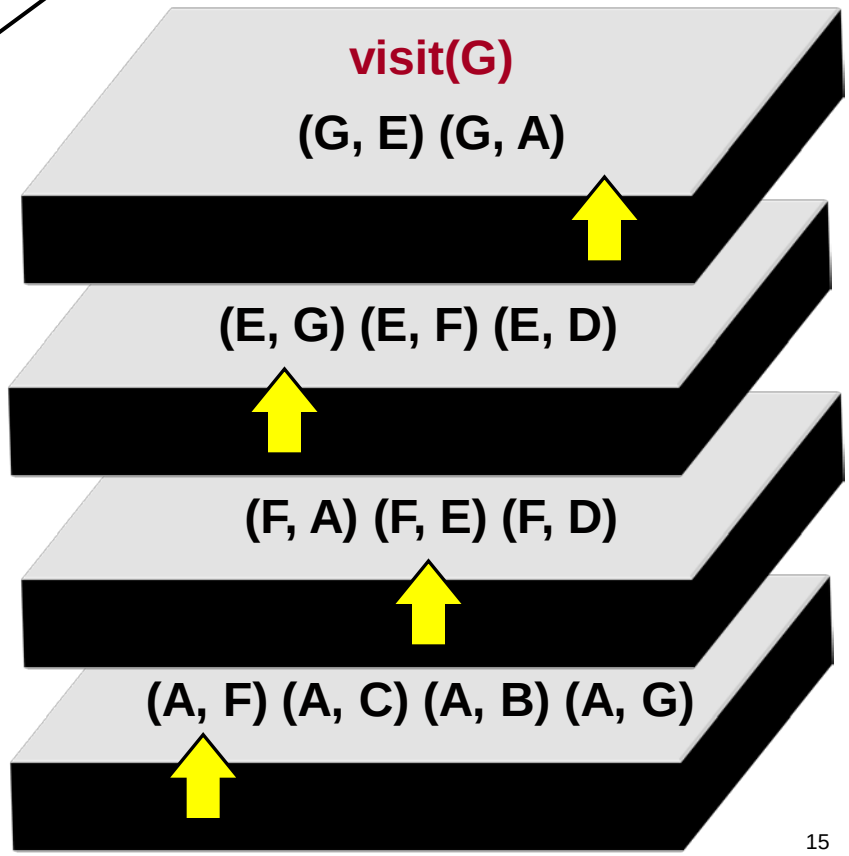


Stack

Undirected Depth First Search



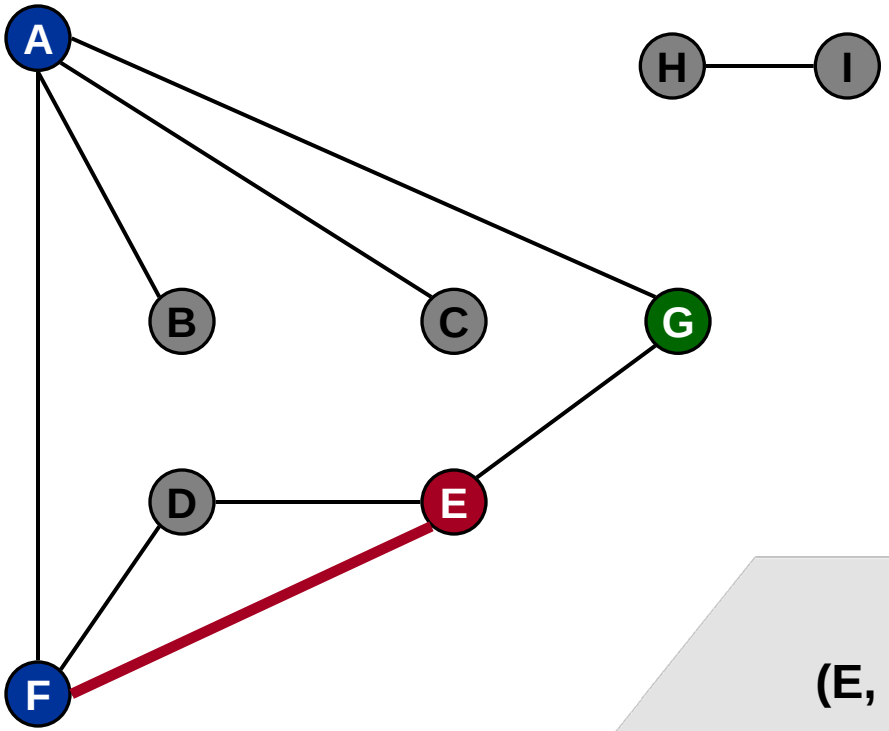
Finished G



Undiscovered
Marked
Active
Finished

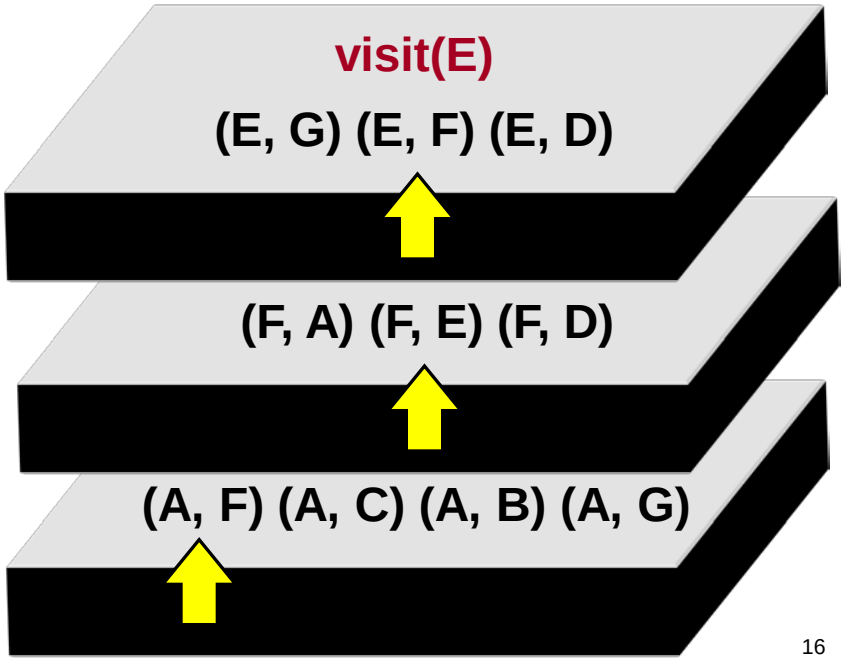
Stack

Undirected Depth First Search



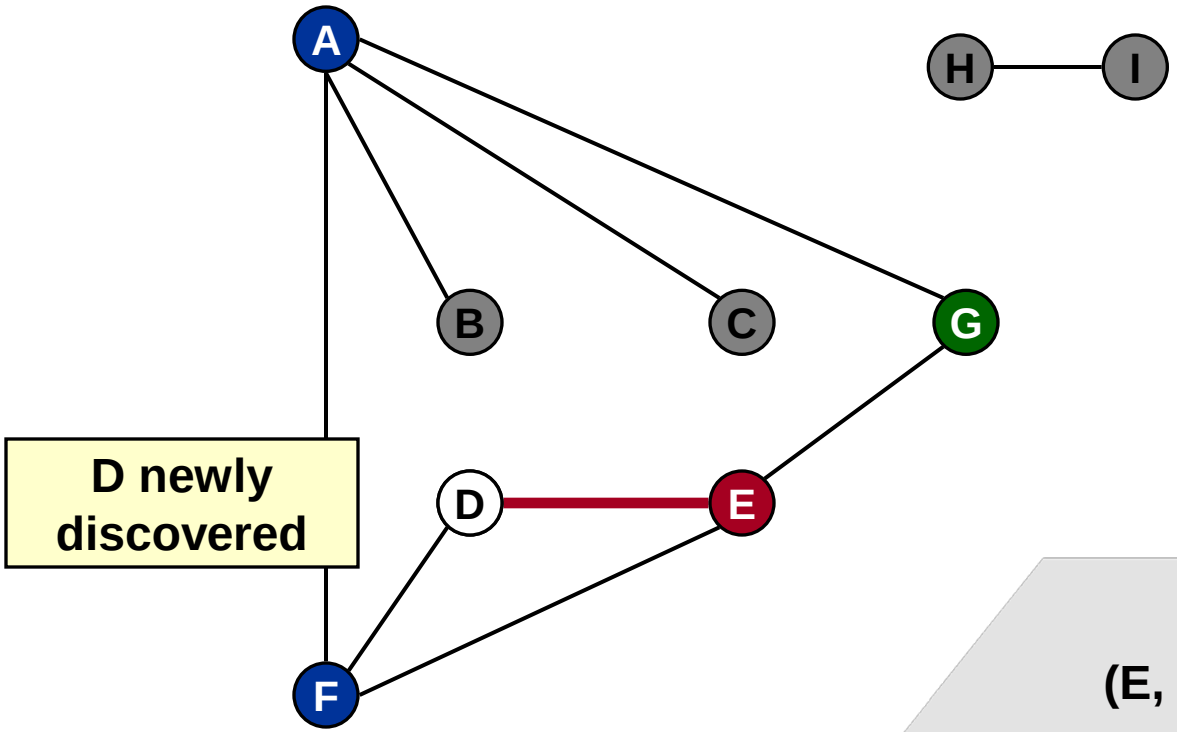
F already marked

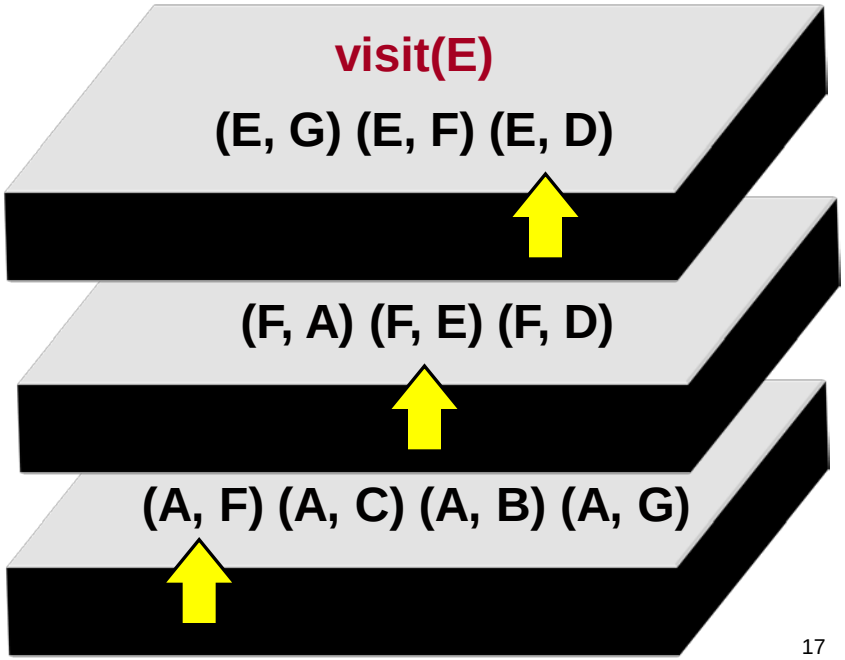
Undiscovered
Marked
Active
Finished



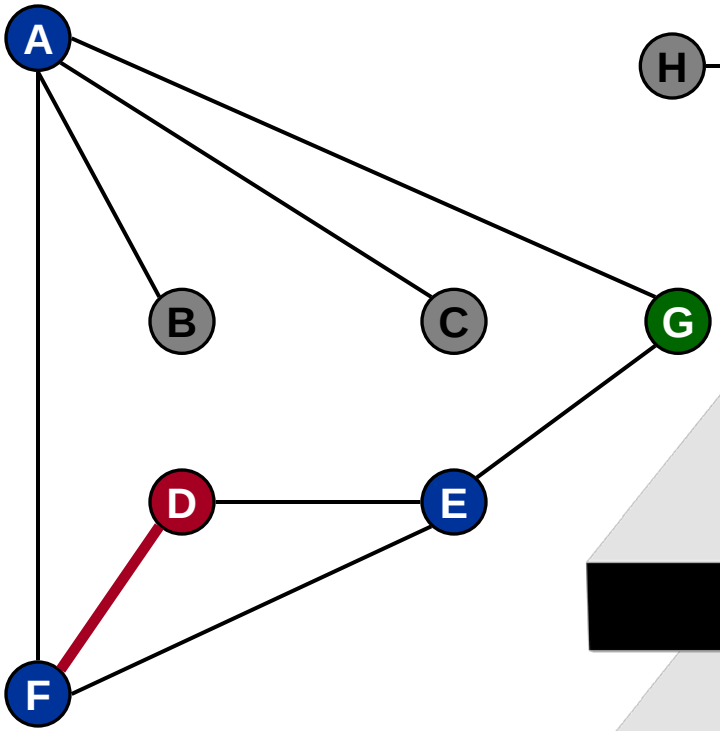
Stack

Undirected Depth First Search

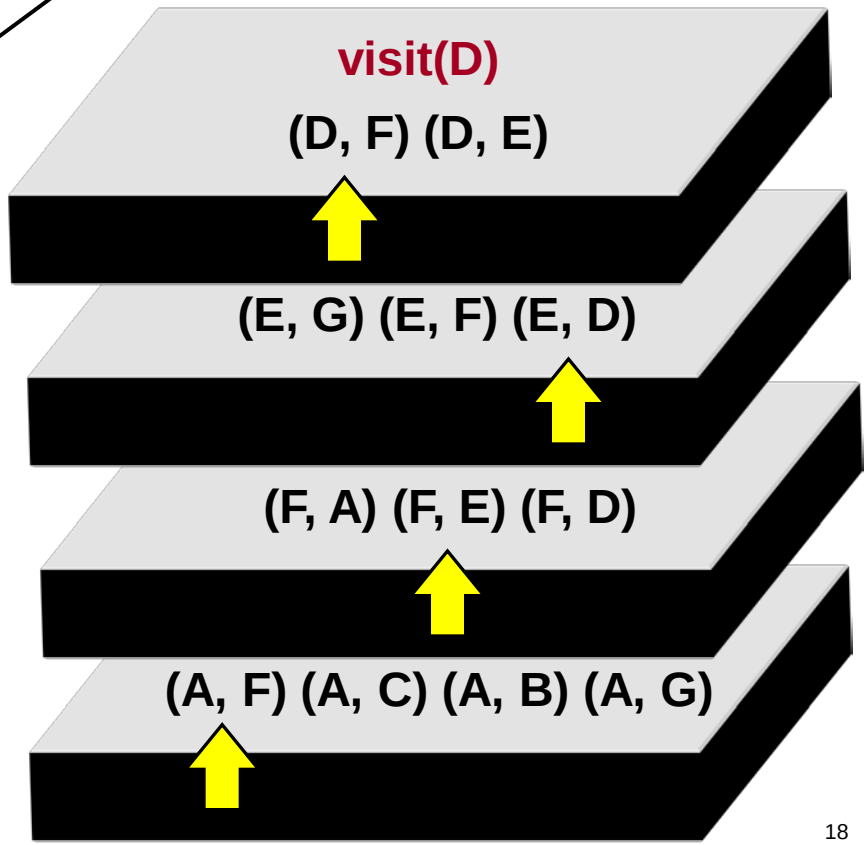




Undirected Depth First Search



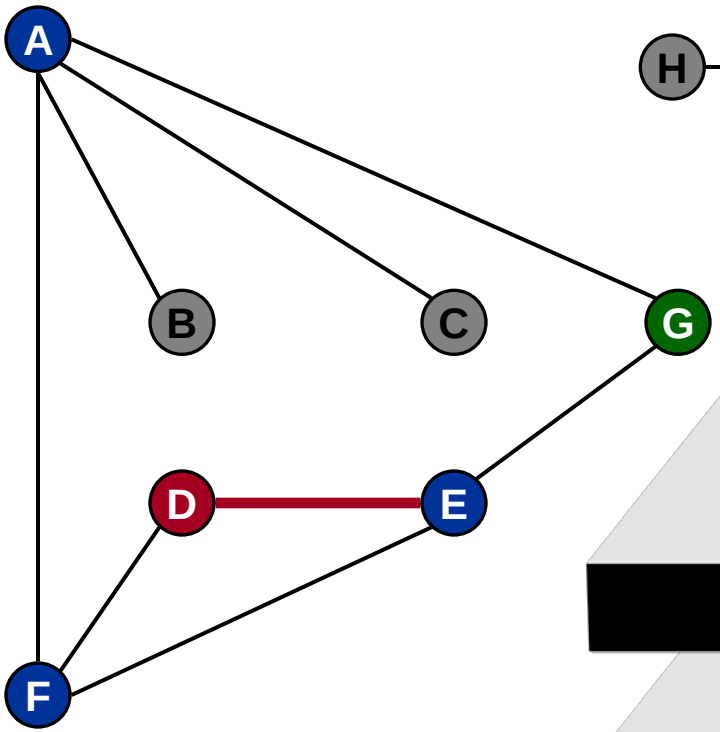
F already marked



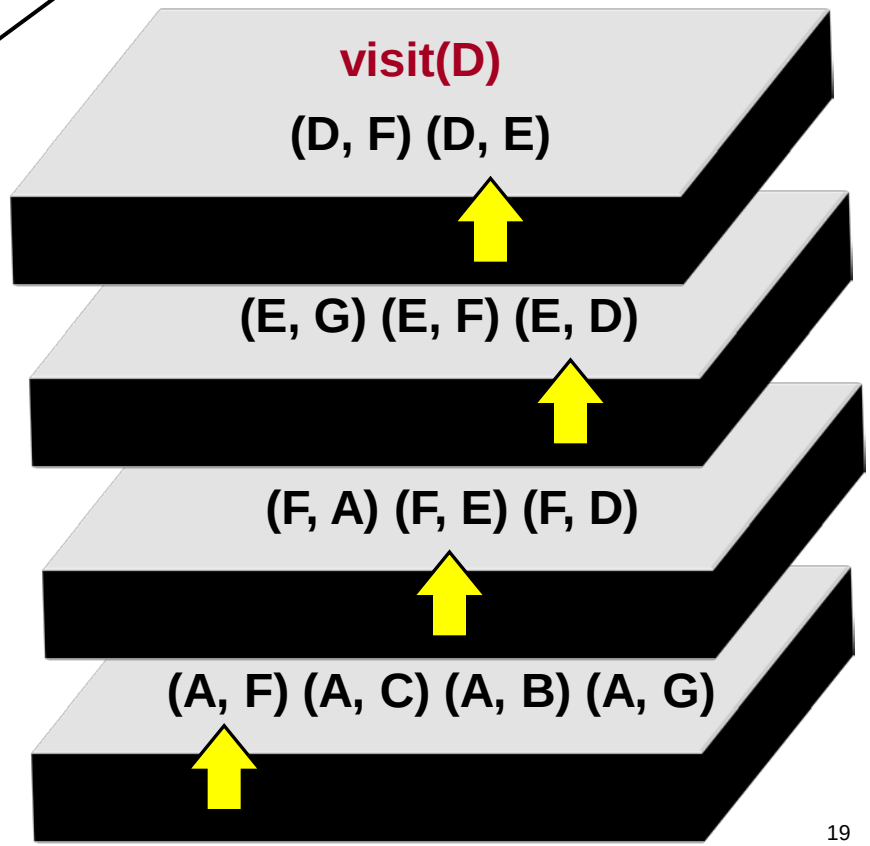
Stack

Undiscovered
Marked
Active
Finished

Undirected Depth First Search

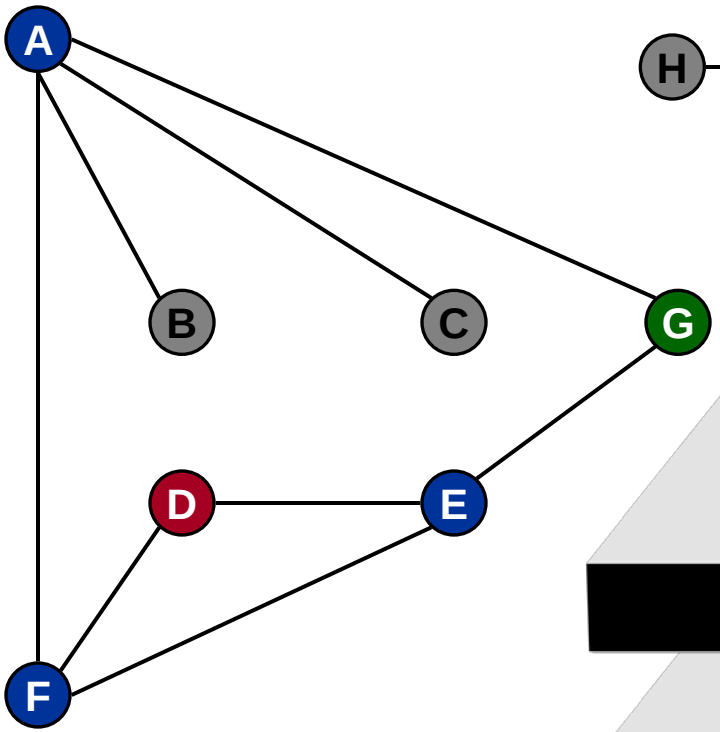


E already marked



Stack

Undirected Depth First Search



Finished D

visit(D)

(D, F) (D, E)

(E, G) (E, F) (E, D)

(F, A) (F, E) (F, D)

(A, F) (A, C) (A, B) (A, G)

Stack

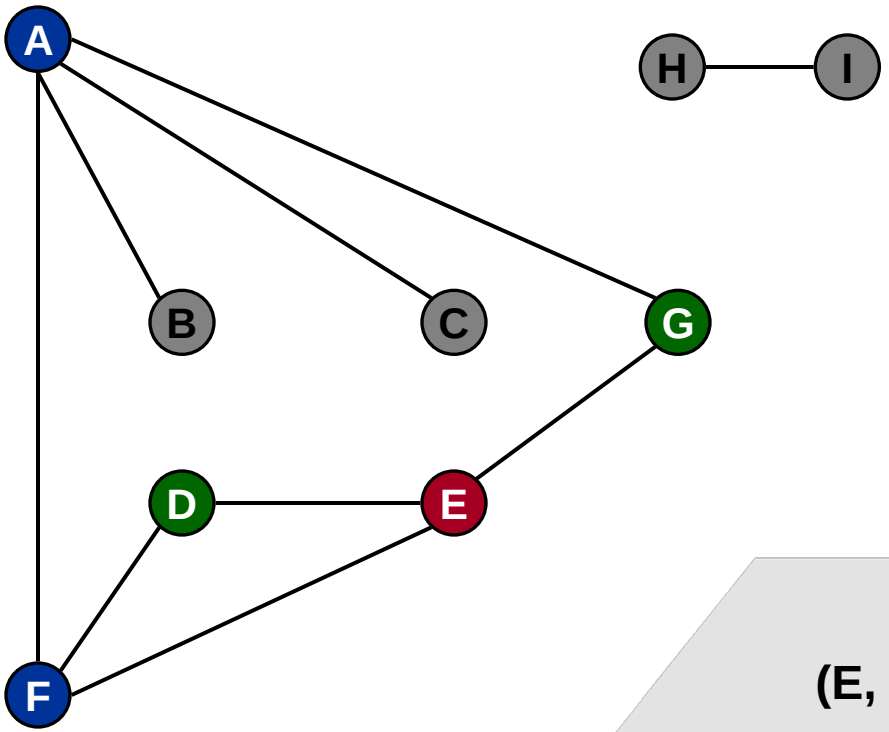
Undiscovered

Marked

Active

Finished

Undirected Depth First Search



Finished E

visit(E)

(E, G) (E, F) (E, D)

(F, A) (F, E) (F, D)

(A, F) (A, C) (A, B) (A, G)

Undiscovered

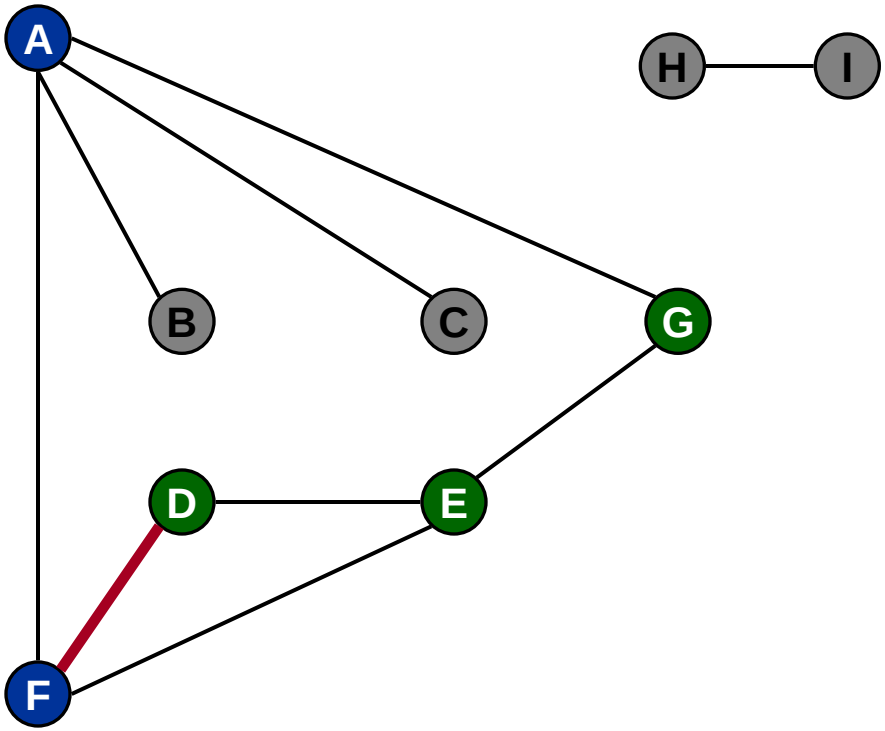
Marked

Active

Finished

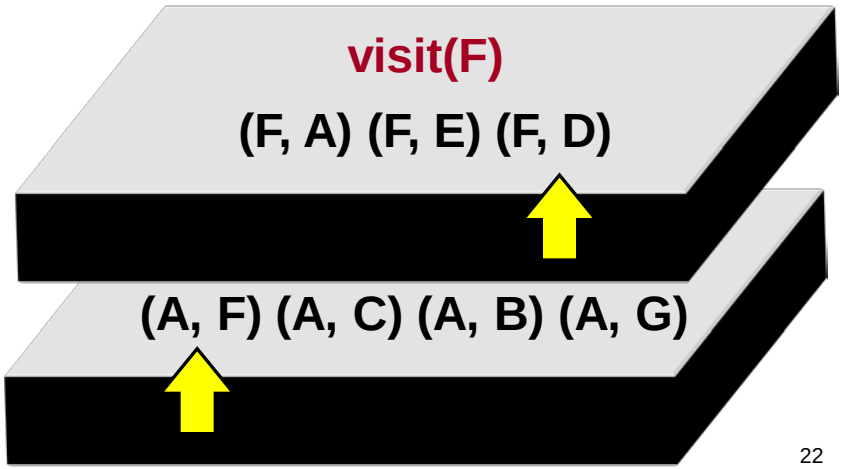
Stack

Undirected Depth First Search



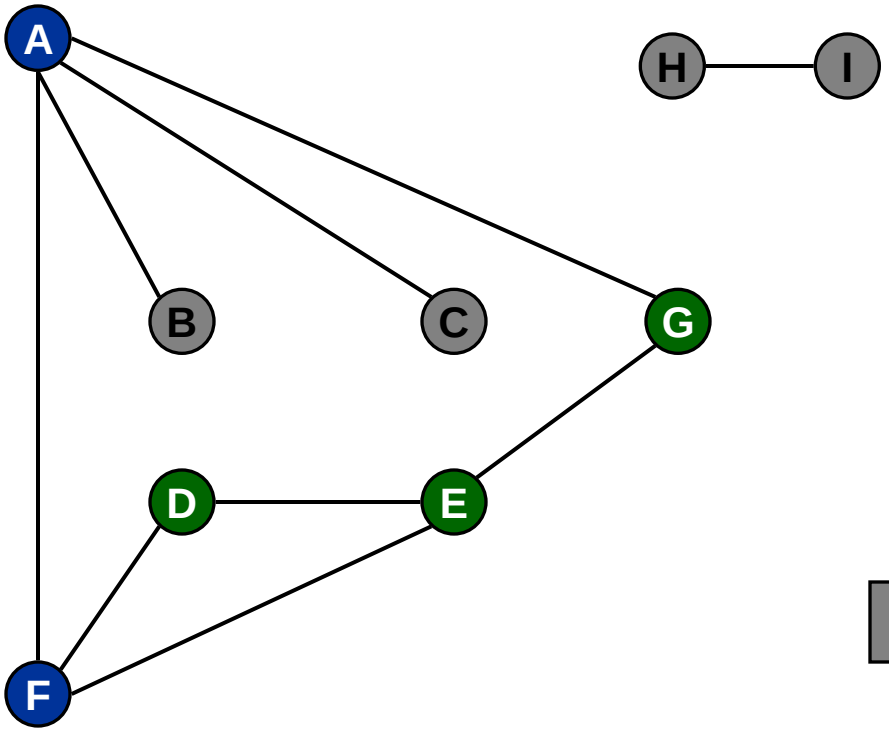
D already marked

Undiscovered
Marked
Active
Finished



Stack

Undirected Depth First Search



Undiscovered

Marked

Active

Finished

Finished F

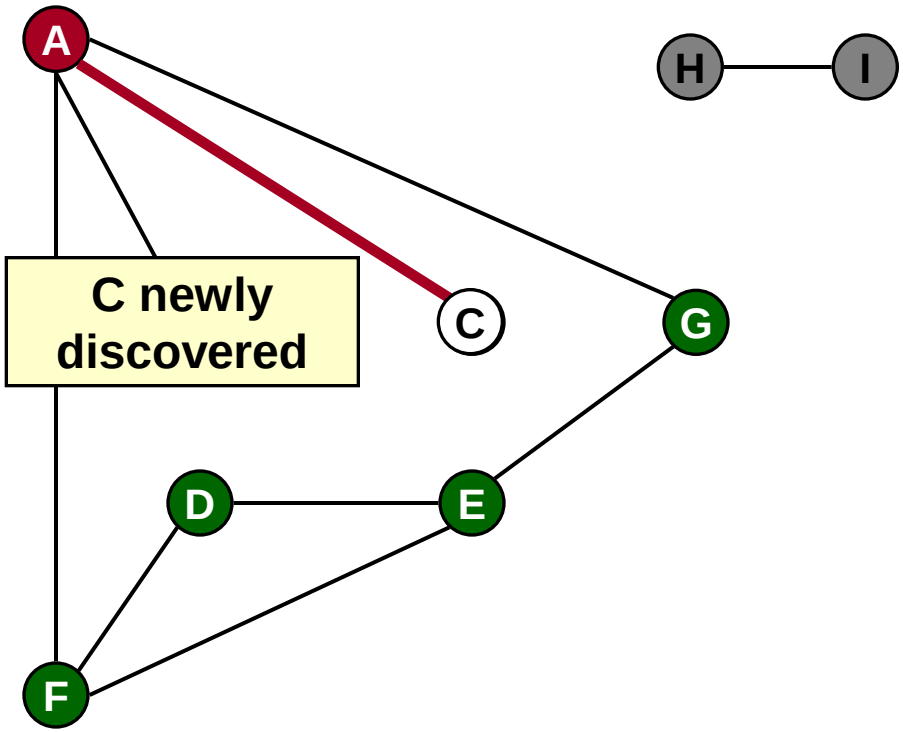
visit(F)

(F, A) (F, E) (F, D)

(A, F) (A, C) (A, B) (A, G)

Stack

Undirected Depth First Search



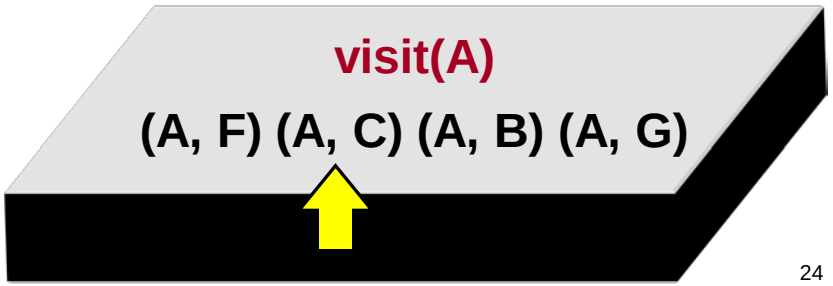
Undiscovered

Marked

Active

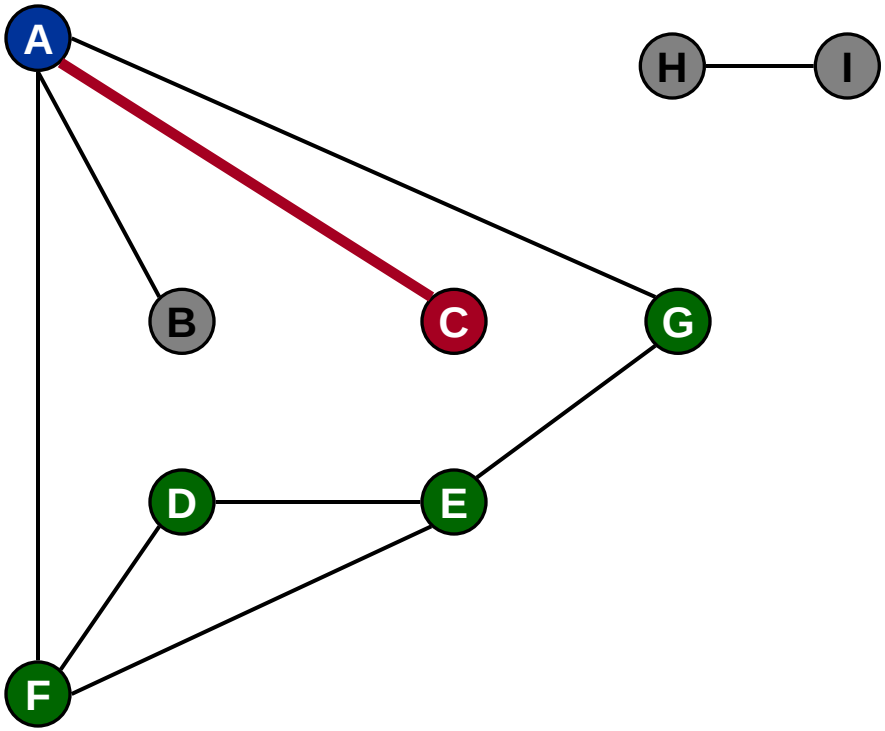
Finished

Stack

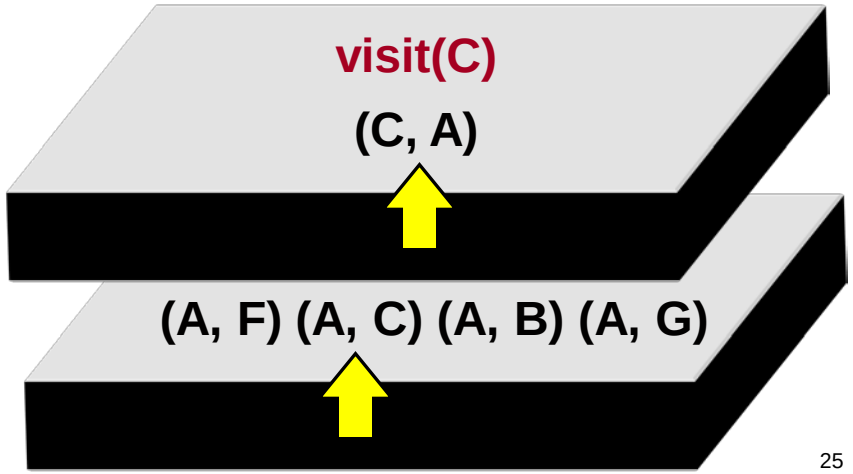


Undirected Depth First Search

A already marked

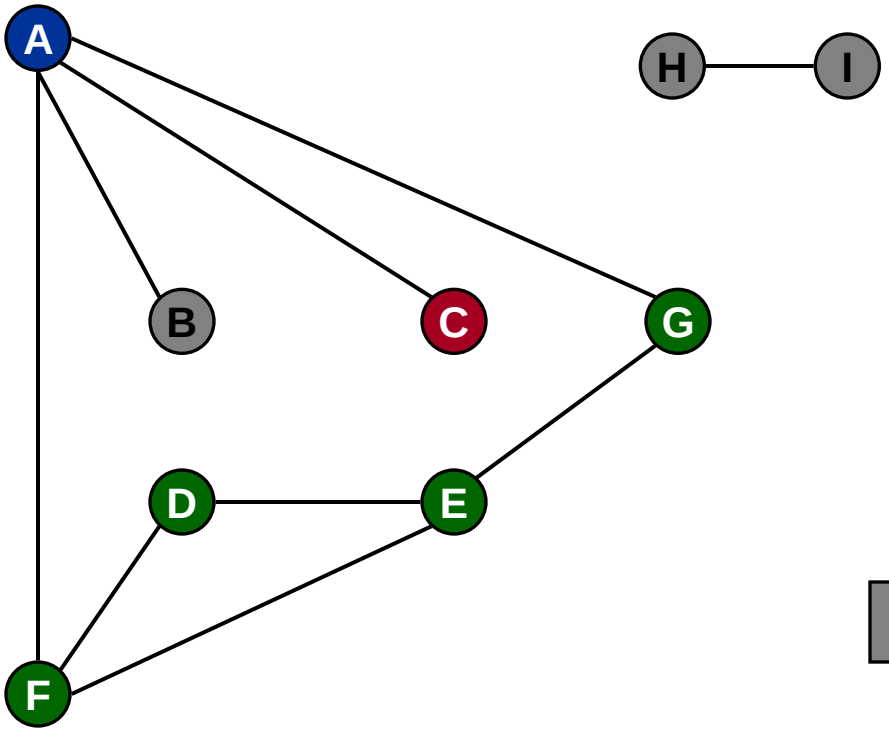


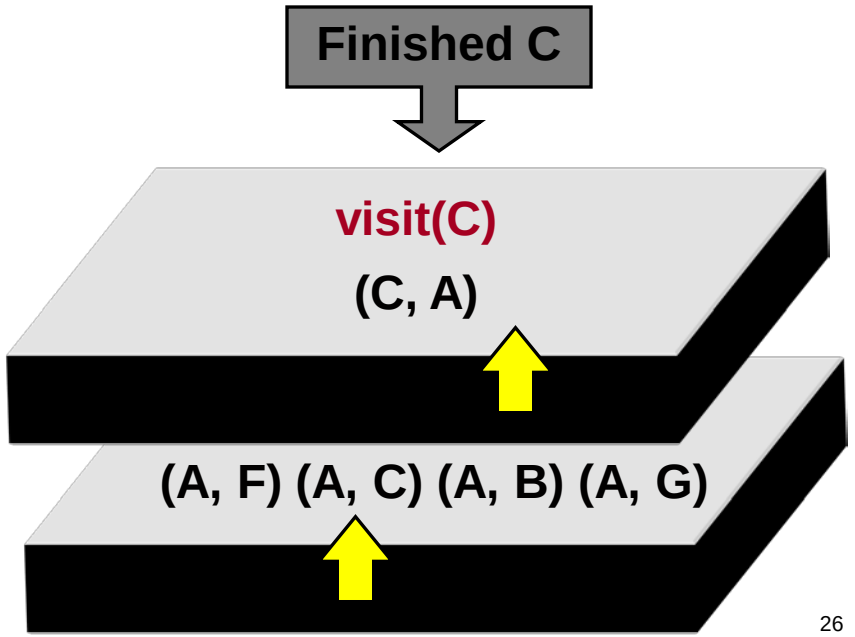
Undiscovered
Marked
Active
Finished



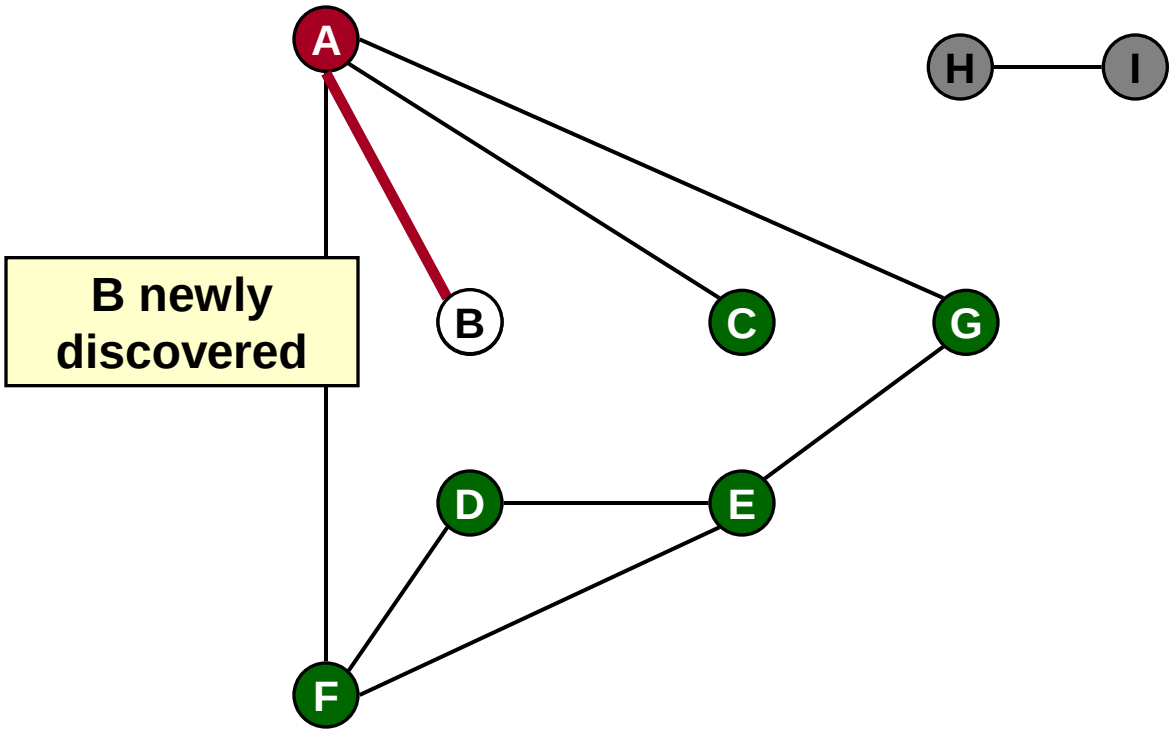
Stack

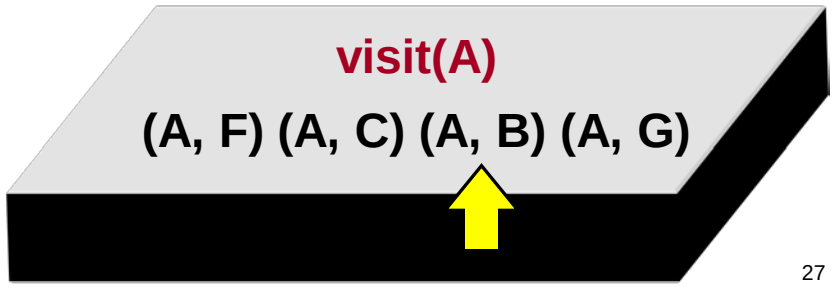
Undirected Depth First Search



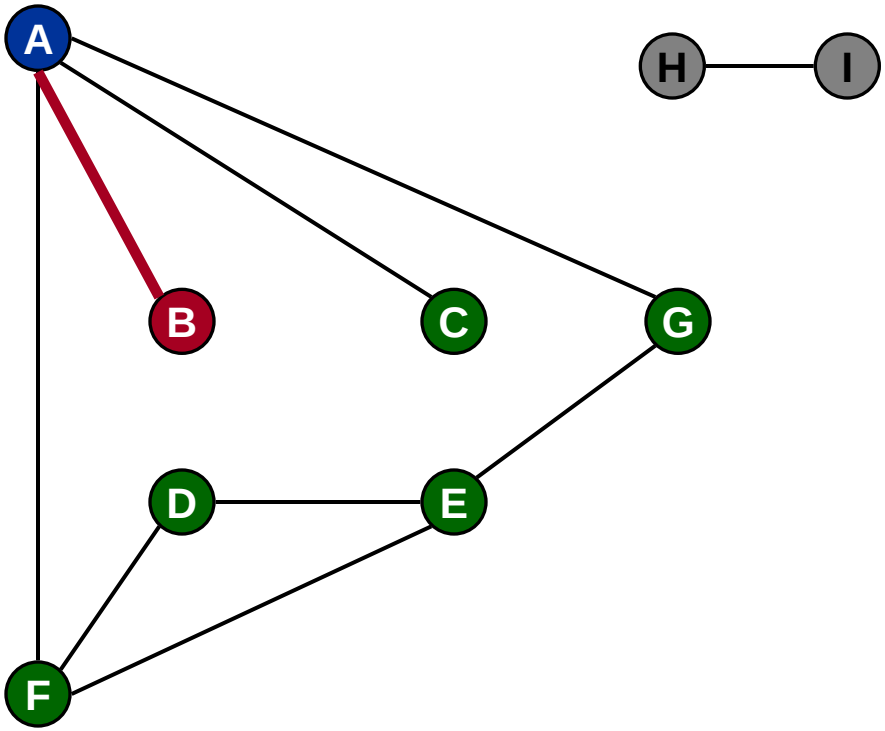


Undirected Depth First Search



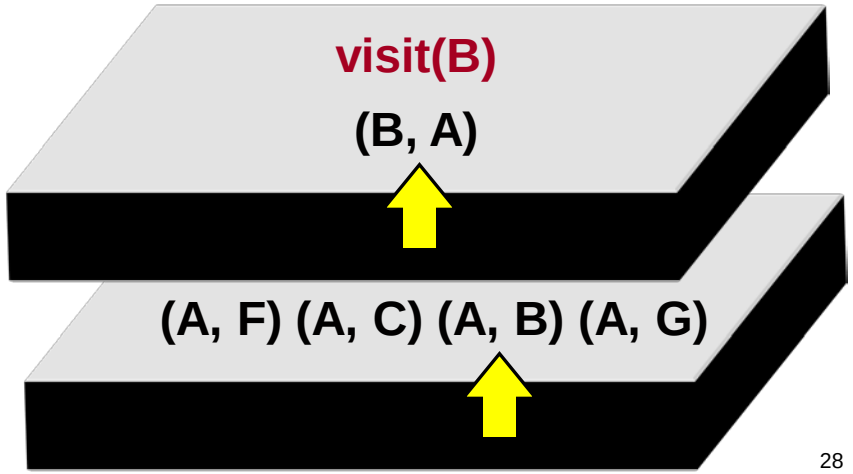


Undirected Depth First Search



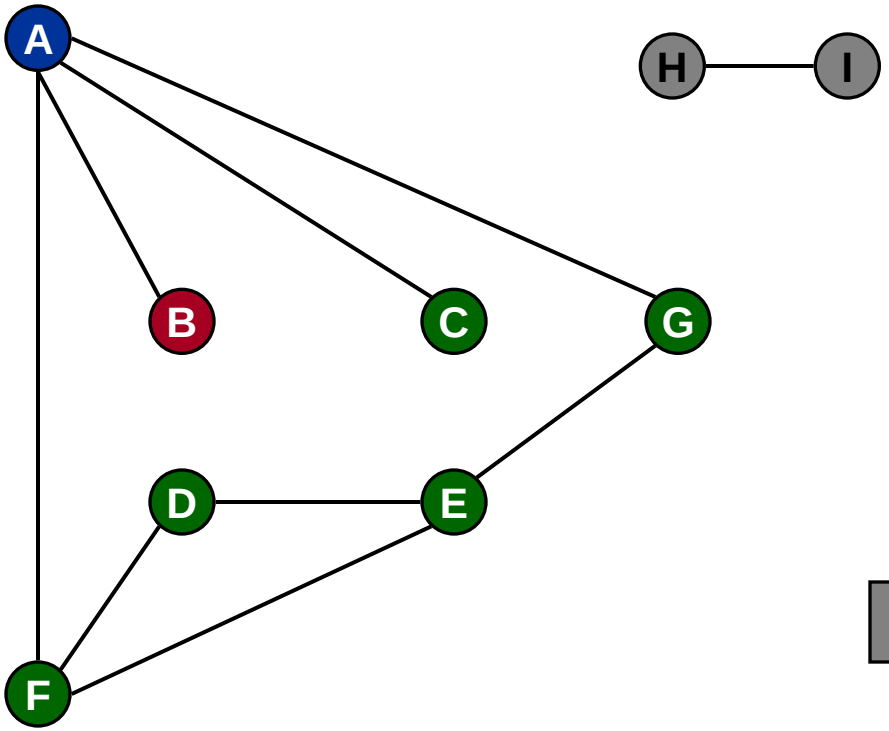
A already marked

Undiscovered
Marked
Active
Finished



Stack

Undirected Depth First Search



Finished B

visit(B)

(B, A)

(A, F) (A, C) (A, B) (A, G)

Stack

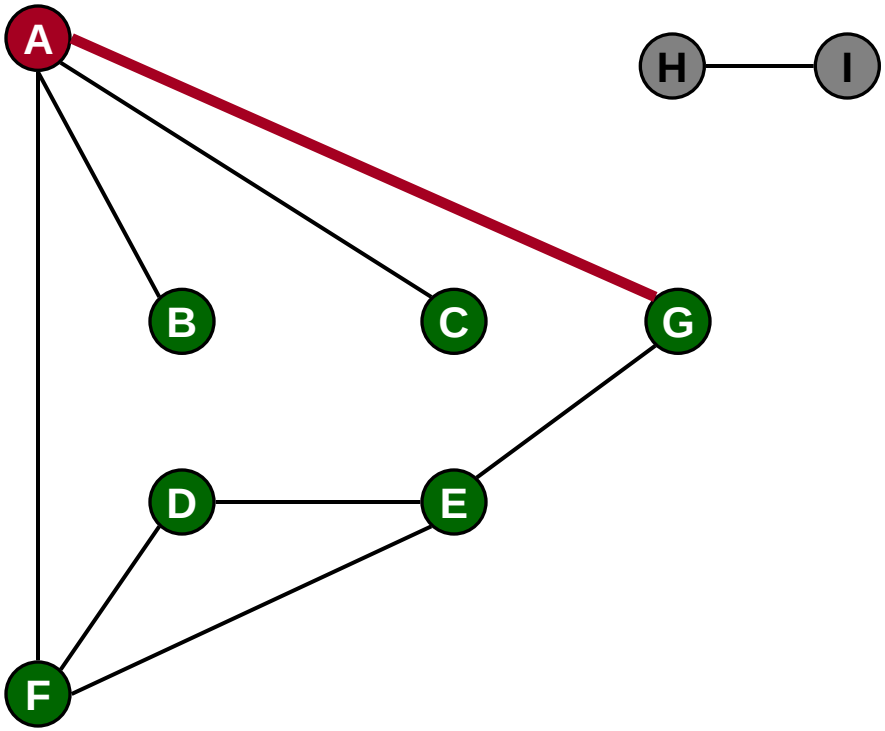
Undiscovered

Marked

Active

Finished

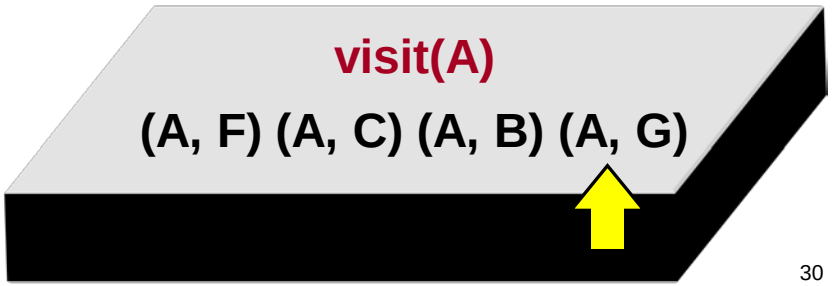
Undirected Depth First Search



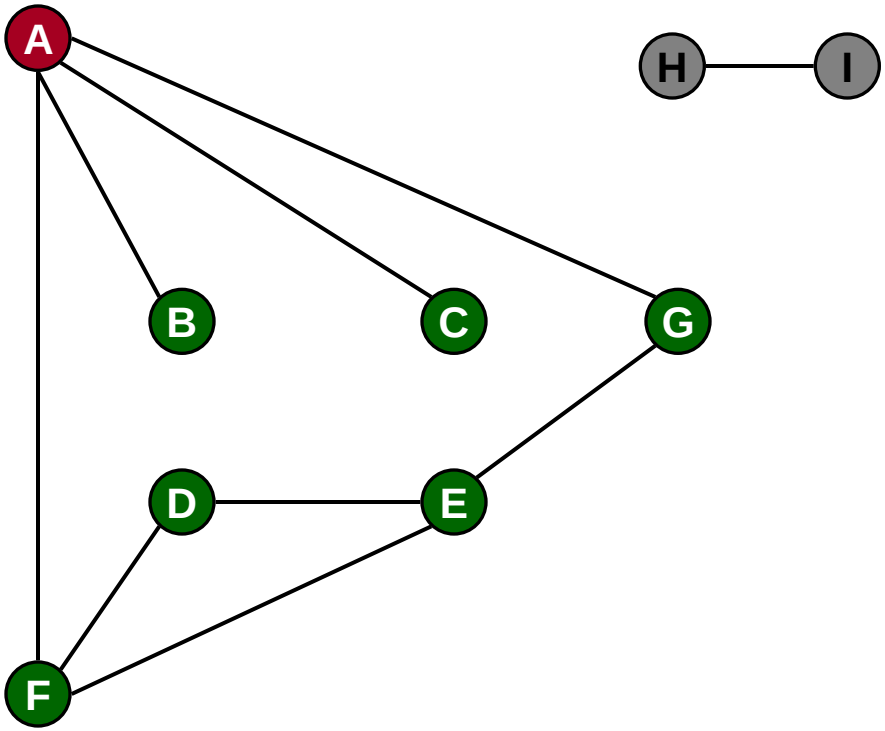
G already finished

Undiscovered
Marked
Active
Finished

Stack



Undirected Depth First Search



Undiscovered

Marked

Active

Finished

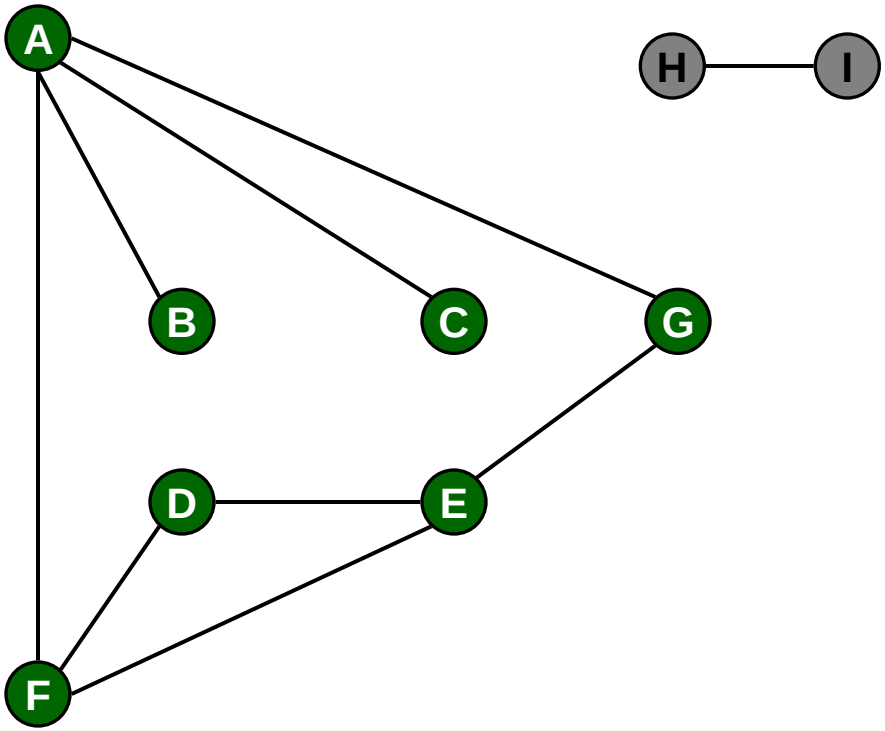
Finished A

visit(A)

(A, F) (A, C) (A, B) (A, G)

Stack

Undirected Depth First Search



Undiscovered

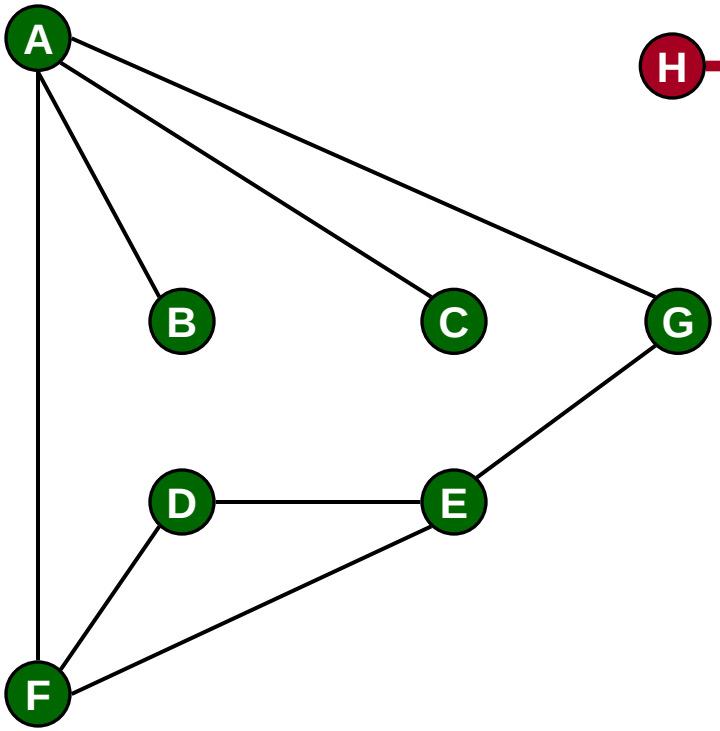
Marked

Active

Finished

Stack

Undirected Depth First Search



I newly discovered

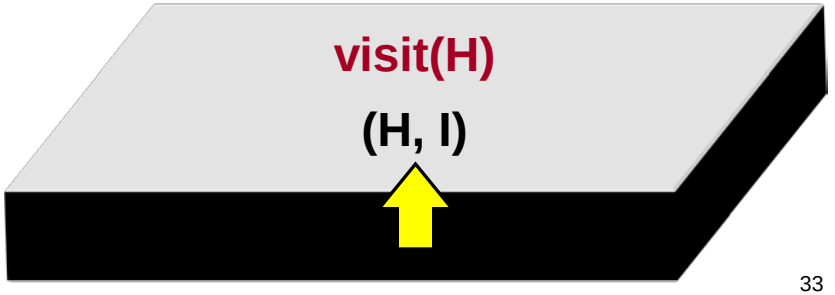
Undiscovered

Marked

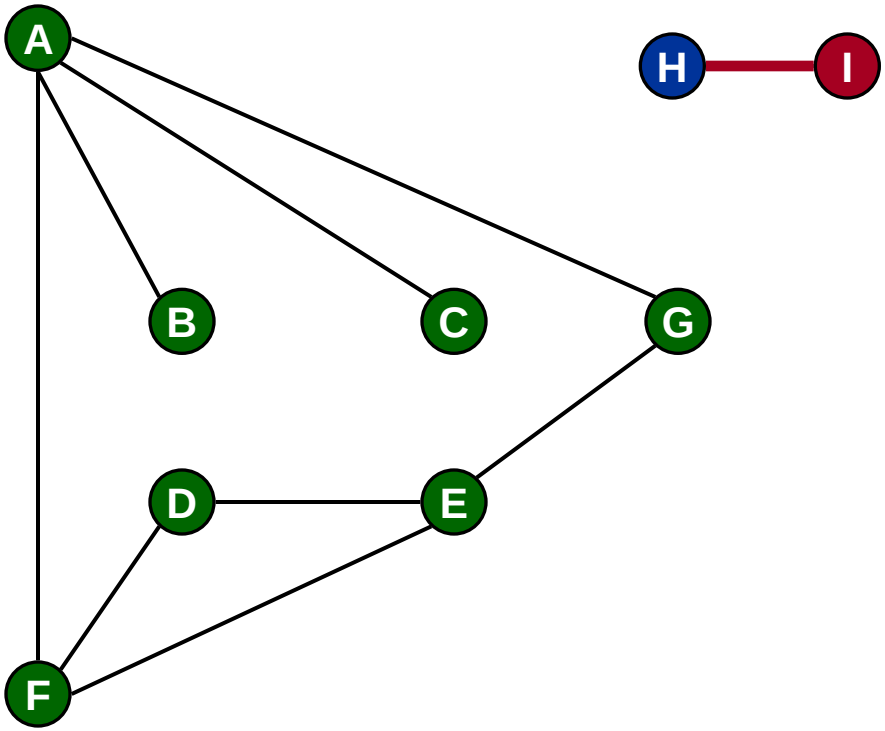
Active

Finished

Stack



Undirected Depth First Search

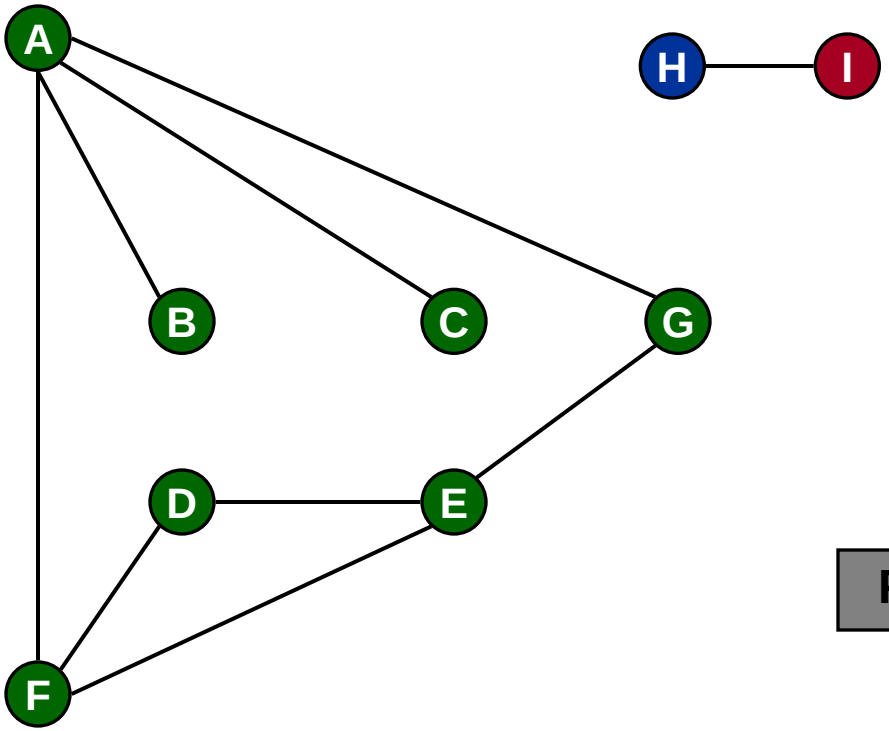


H already marked

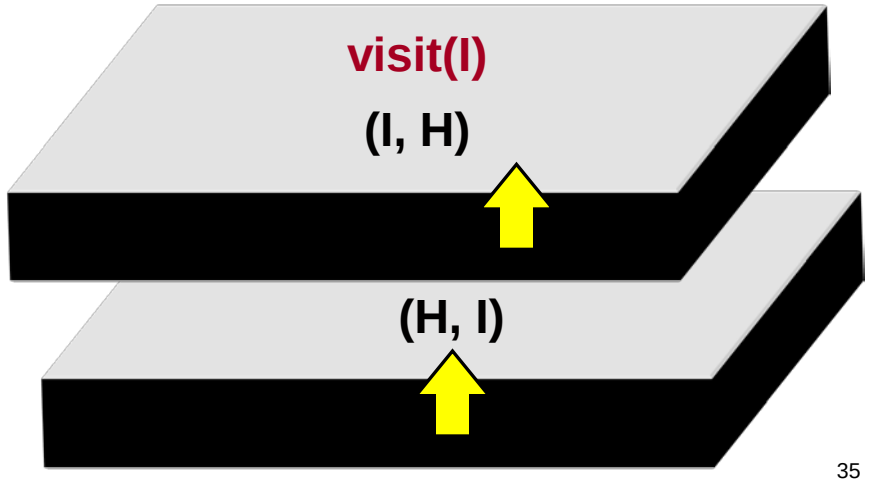
Undiscovered
Marked
Active
Finished

Stack

Undirected Depth First Search



Finished I



Undiscovered

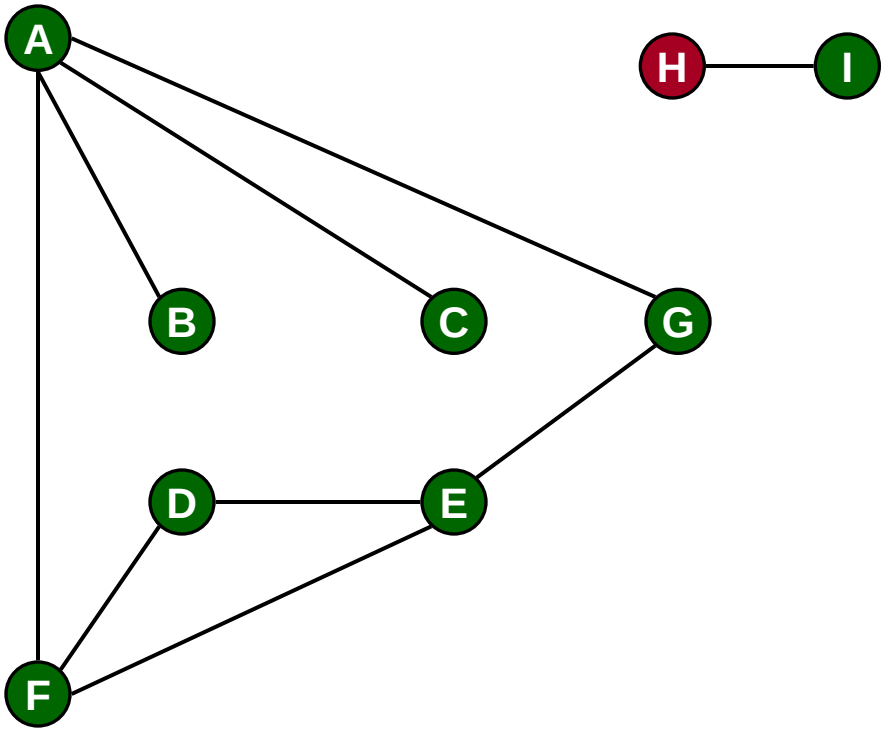
Marked

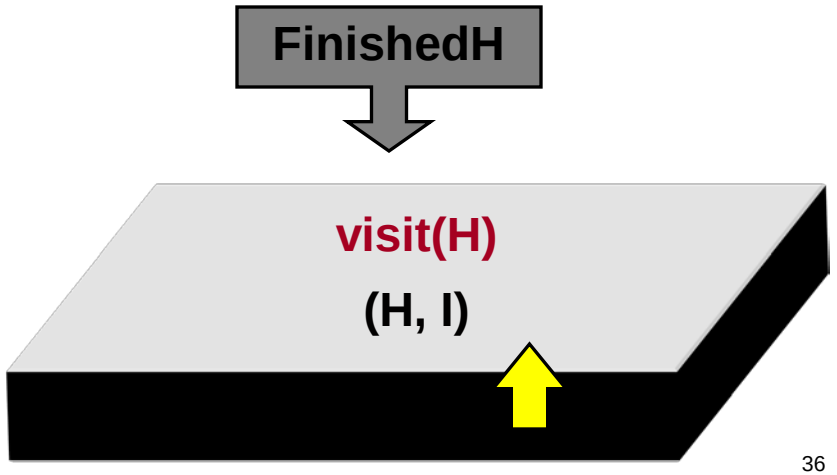
Active

Finished

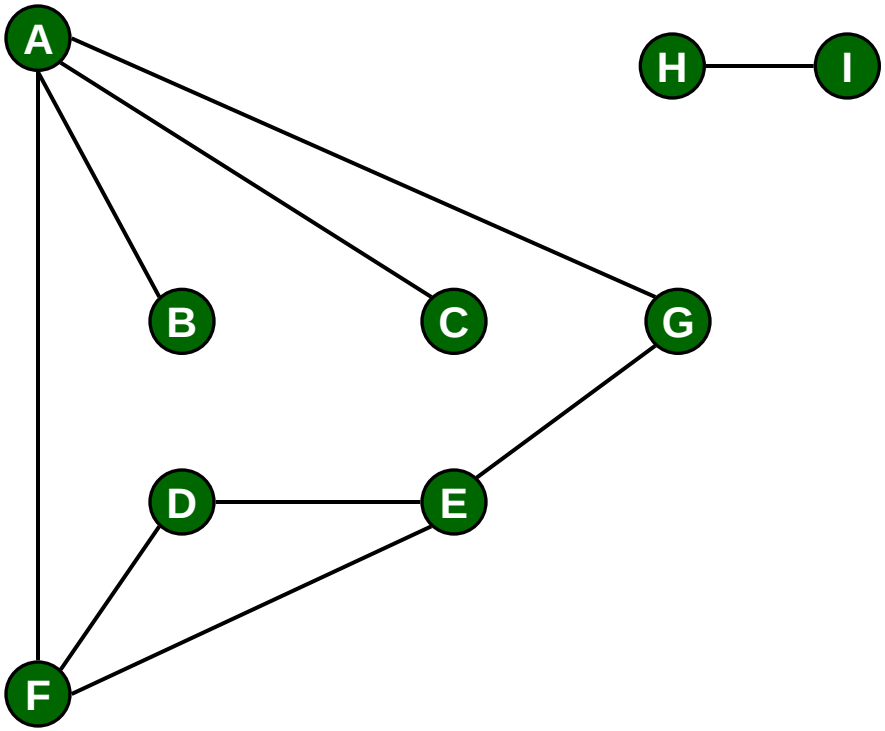
Stack

Undirected Depth First Search





Undirected Depth First Search



Undiscovered

Marked

Active

Finished

Stack

DFS Implementation

- BFS: Queue
- DFS: Stack
- The beauty of implementing dfs recursively is that recursion eliminates the need to keep an explicit stack

Depth-First Search Algorithm

- The beauty of implementing dfs recursively is that recursion eliminates the need to keep an explicit stack

Algorithm DFS(v); **Input**: A vertex v in a graph

Output: A labeling of the edges as "discovery" edges and "backedges"

for each edge *e* incident on *v* **do**

if edge *e* is unexplored **then** let *w* be the other endpoint of *e*

if vertex w is unexplored **then** label *e* as a discovery edge

recursively call **DFS(*w*)**

else label *e* as a backedge

DFS—Using Adjacent Matrix

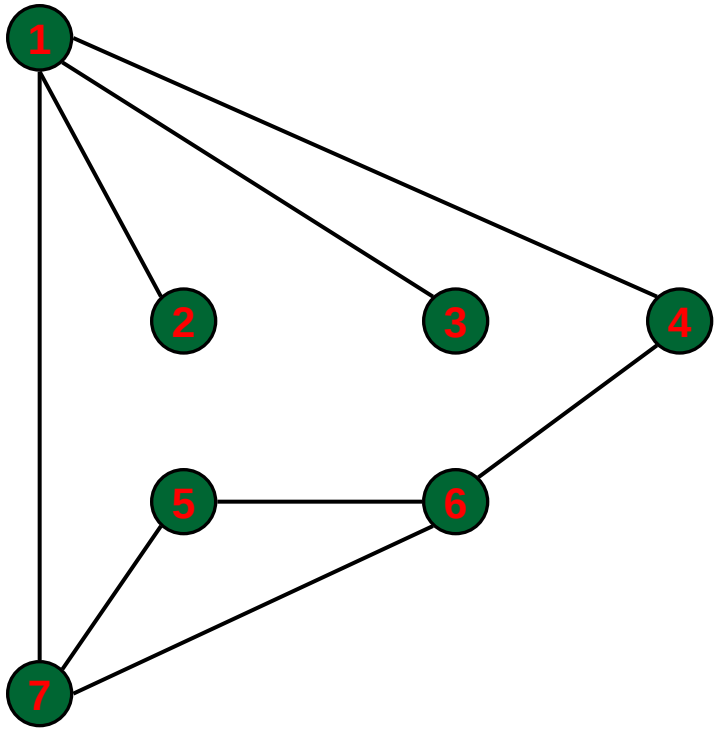
```
void DFSTraverse（int v）
{
    cout<<vertex[v]; visited [v]=1;
    for（j=0; j<vertexNum; j++）
        if（arc[v][j]==1 && visited[j]==0）
            DFSTraverse（ j ）;
}
```

Thinking problem

- Finding cycle in an undirected graph

  Given an undirected graph, how to find whether there is a cycle? Design your algorithm and evaluate its time complexity.

Practice



Input:
7 //number of vertices
8 //number of edges
1 2 //edge 1—2
1 3
1 4
1 7
4 6
5 6
5 7
6 7

Output:
Yes //there is cycle

Thank You