CSE 421: Introduction to Algorithms

Bipartiteness - DFS

Shayan Oveis Gharan

Bipartite Graphs

Definition: An undirected graph G=(V,E) is bipartite

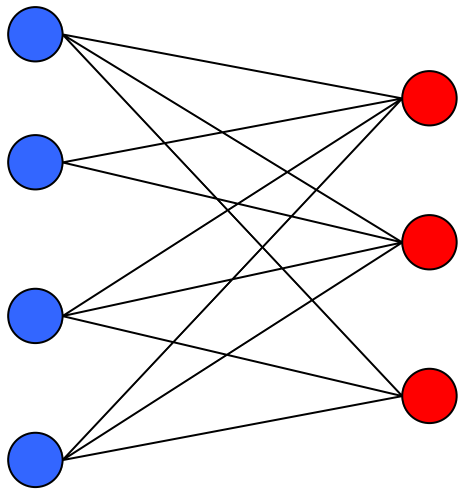
if you can partition the node set into 2 parts (say, blue/red or left/right) so that

all edges join nodes in different parts

i.e., no edge has both ends in the same part.

Application:

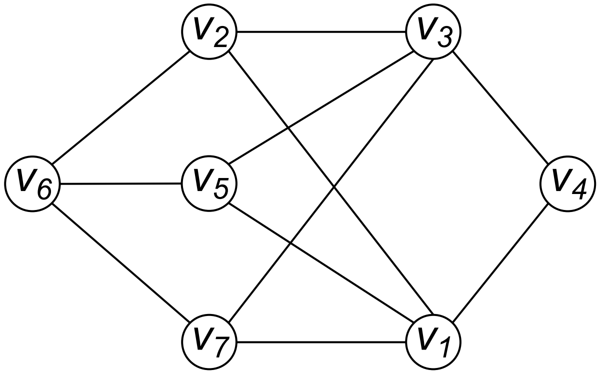
- Scheduling: machine=red, jobs=blue
- Stable Matching: men=blue, wom=red



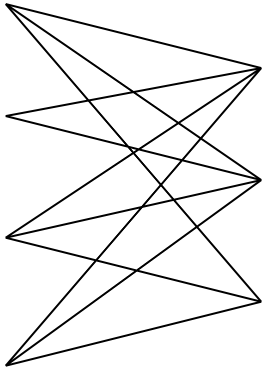
a bipartite graph

Testing Bipartiteness

Problem: Given a graph G, is it bipartite?



a bipartite graph G



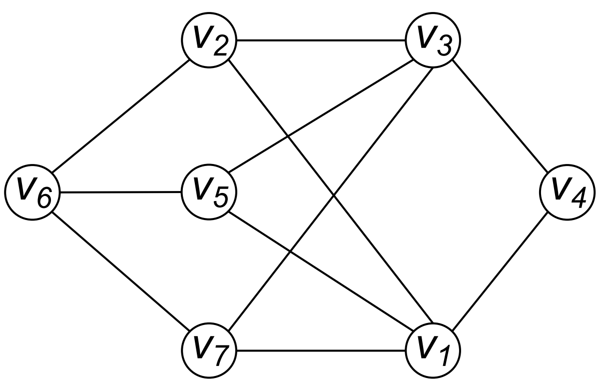
Testing Bipartiteness

Problem: Given a graph G, is it bipartite?

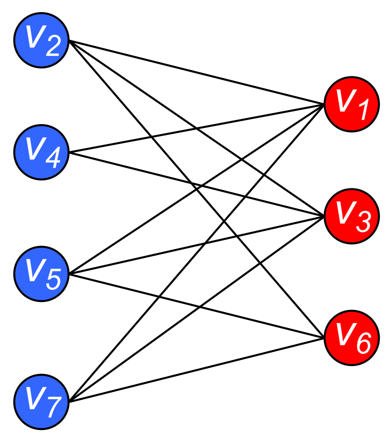
Many graph problems become:

- Easier if the underlying graph is bipartite (matching)
- Tractable if the underlying graph is bipartite (independent set)

Before attempting to design an algorithm, we need to understand structure of bipartite graphs.



a bipartite graph G

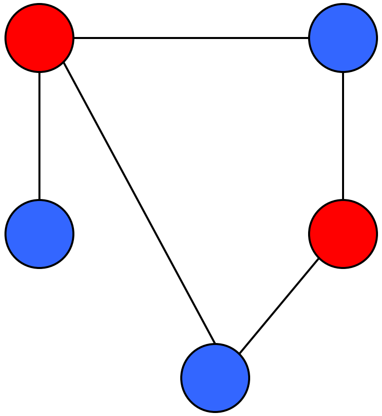


another drawing of G

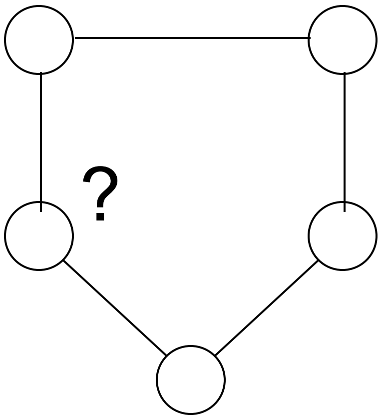
An Obstruction to Bipartiteness

Lemma: If G is bipartite, then it does not contain an odd length cycle.

Pf: We cannot 2-color an odd cycle, let alone G.



bipartite
(2-colorable)



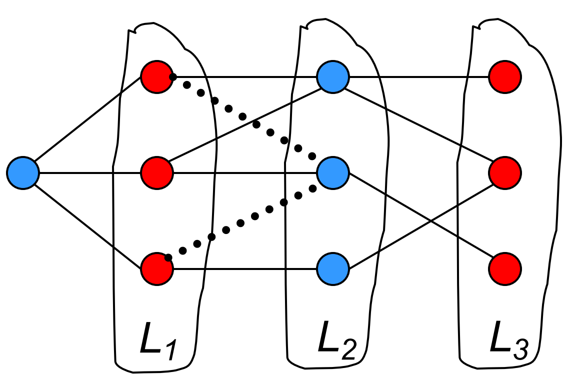
not bipartite
(not 2-colorable)

A Characterization of Bipartite Graphs

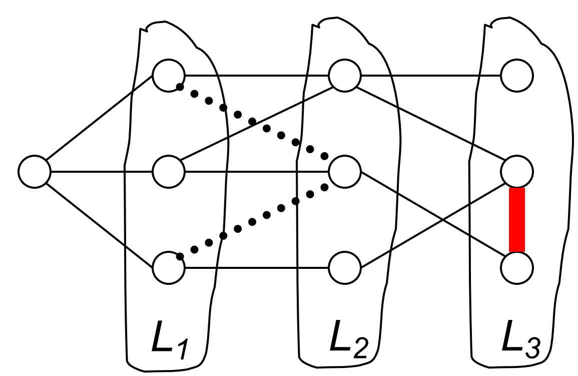
Lemma: Let G be a connected graph, and let $L_0, \ldots, L_k$ be the layers produced by BFS(s). Exactly one of the following holds.

(i) No edge of G joins two nodes of the same layer, and G is bipartite.

(ii) An edge of G joins two nodes of the same layer, and G contains an odd-length cycle (and hence is not bipartite).



Case (i)



Case (ii)

A Characterization of Bipartite Graphs

Lemma: Let G be a connected graph, and let $L_0, \ldots, L_k$ be the layers produced by BFS(s). Exactly one of the following holds.

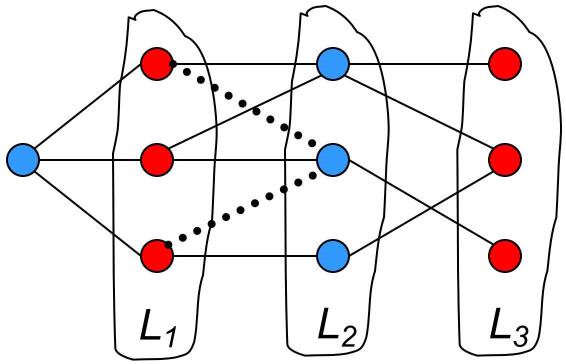
(i) No edge of G joins two nodes of the same layer, and G is bipartite.

(ii) An edge of G joins two nodes of the same layer, and G contains an odd-length cycle (and hence is not bipartite).

Pf. (i)

Suppose no edge joins two nodes in the same layer.

By previous lemma, all edges join nodes on adjacent levels.



Case (i)

Bipartition:

blue = nodes on odd levels,

red = nodes on even levels.

A Characterization of Bipartite Graphs

Lemma: Let G be a connected graph, and let $L_0, \ldots, L_k$ be the layers produced by BFS(s). Exactly one of the following holds.

(i) No edge of G joins two nodes of the same layer, and G is bipartite.

(ii) An edge of G joins two nodes of the same layer, and G contains an odd-length cycle (and hence is not bipartite).

Pf. (ii)

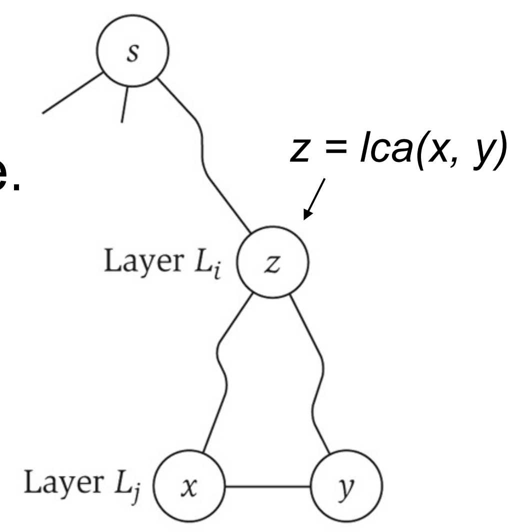
Suppose (x, y) is an edge & x, y in same level L_j.

Let z = their lowest common ancestor in BFS tree.

Let L_i be level containing z.

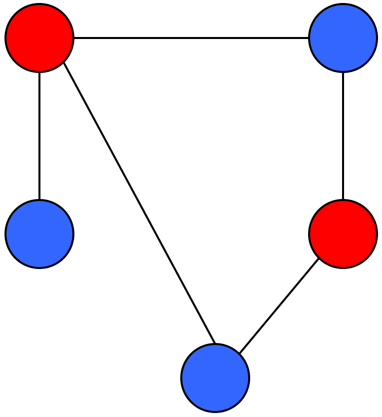
Consider cycle that takes edge from x to y, then tree from y to z, then tree from z to x.

Its length is 1 + (j-i) + (j-i), which is odd.

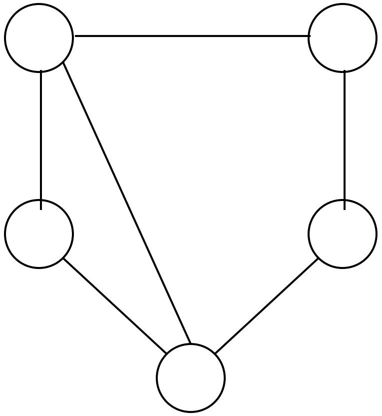


Obstruction to Bipartiteness

Cor: A graph G is bipartite iff it contains no odd length cycles.



bipartite
(2-colorable)



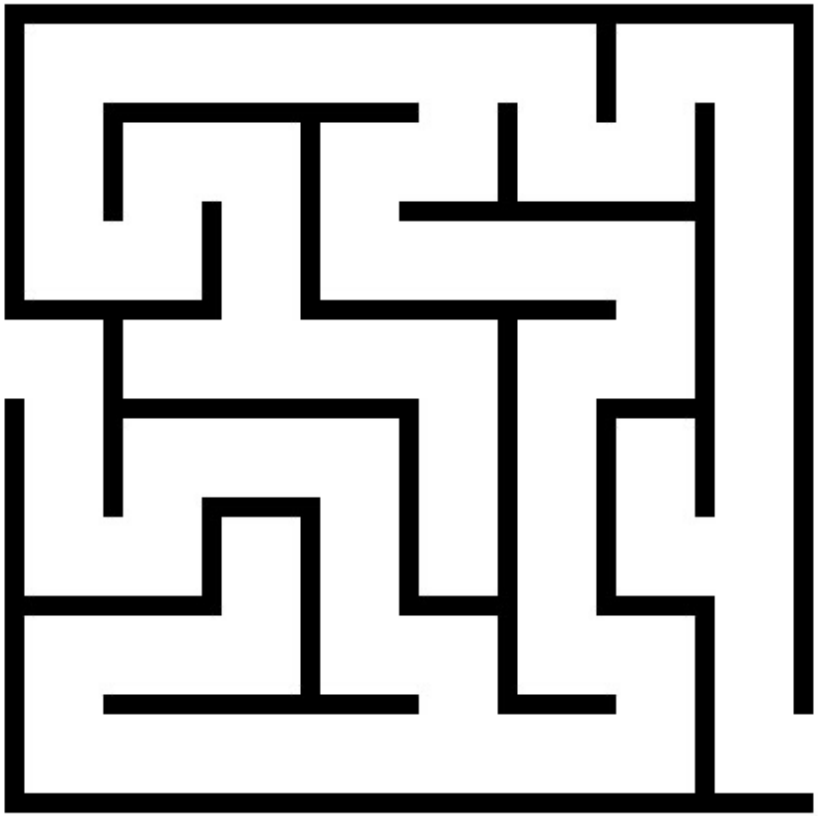
not bipartite
(not 2-colorable)

In class Exercise

Let G be a graph with n vertices and at least n edges. Show that G has a cycle.

Depth First Search

Follow the first path you find as far as you can go; back up to last unexplored edge when you reach a dead end, then go as far you can



Naturally implemented using recursive calls or a stack

DFS(s) – Recursive version

Global Initialization: mark all vertices undiscovered

DFS(v)

    Mark v discovered

    for each edge {v,x}
        if (x is undiscovered)
            Mark x discovered
            DFS(x)

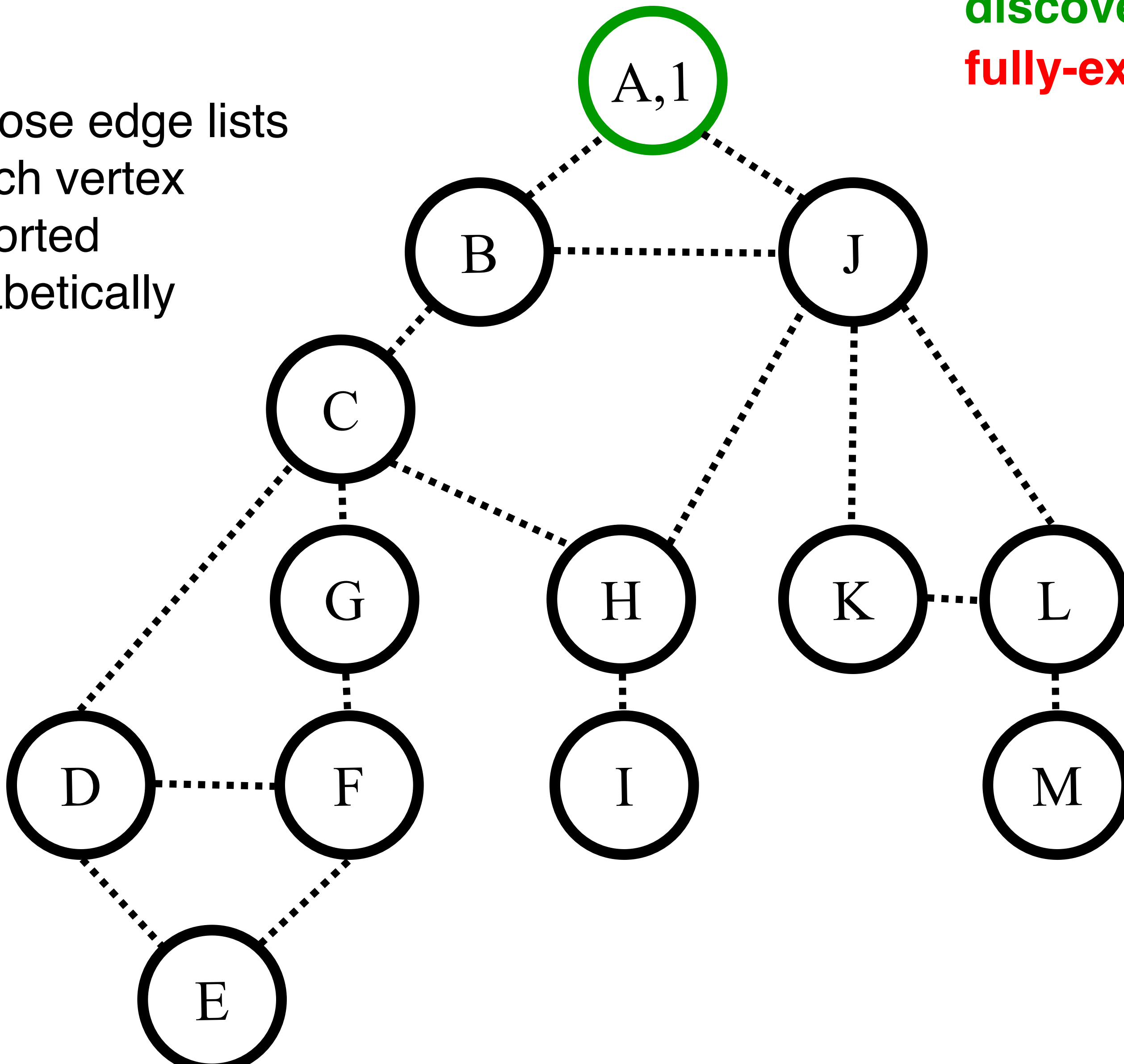
    Mark v full-discovered

DFS(A)

Color code:

undiscovered
discovered
fully-explored

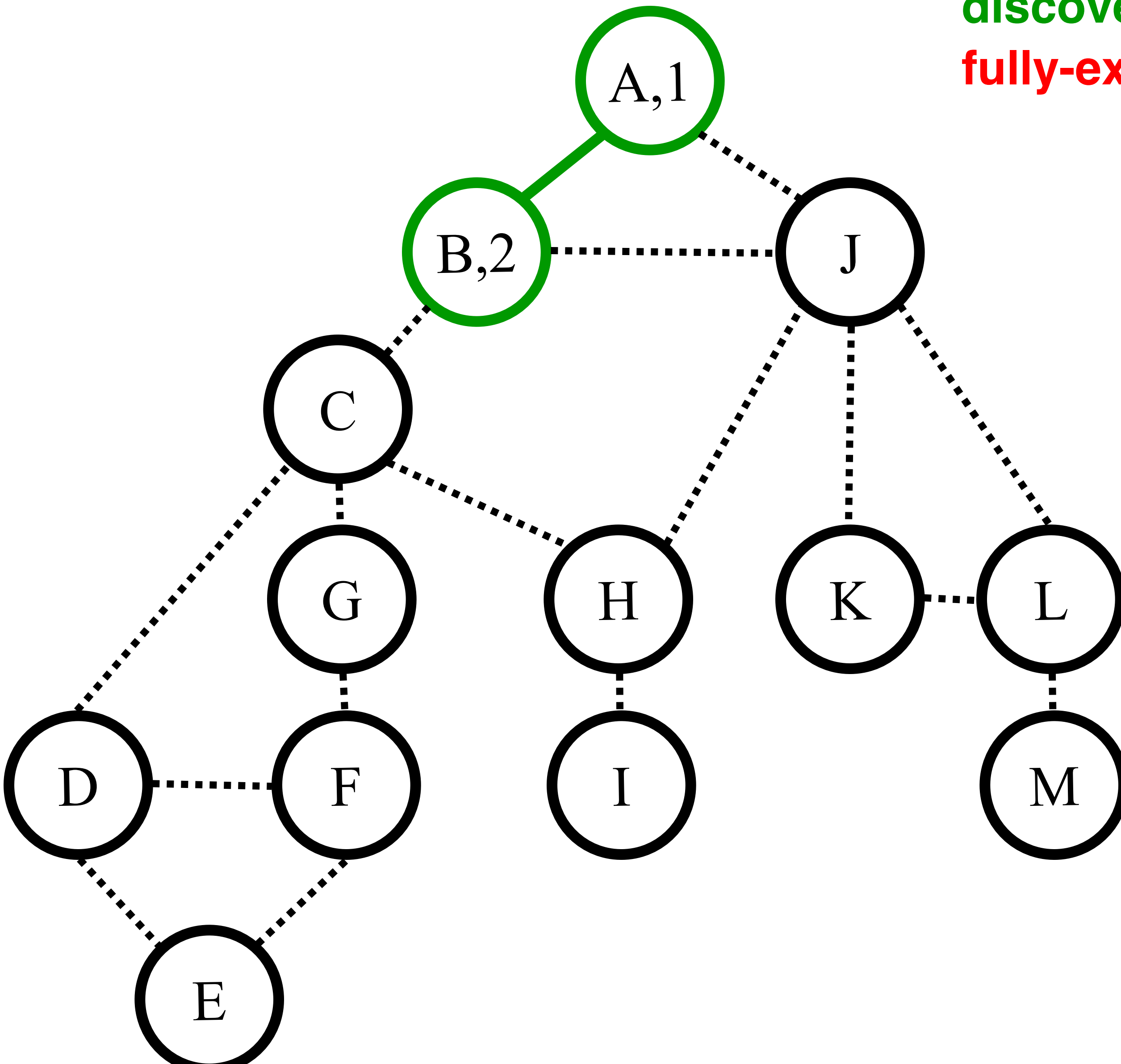
Suppose edge lists at each vertex are sorted alphabetically



Call Stack (Edge list):
A (B,J)

st[] = {1}

DFS(A)



Color code:

undiscovered

discovered

fully-explored

Call Stack:
(Edge list)

A (~~B~~,J)
B (A,C,J)

st[] =
{1,2}

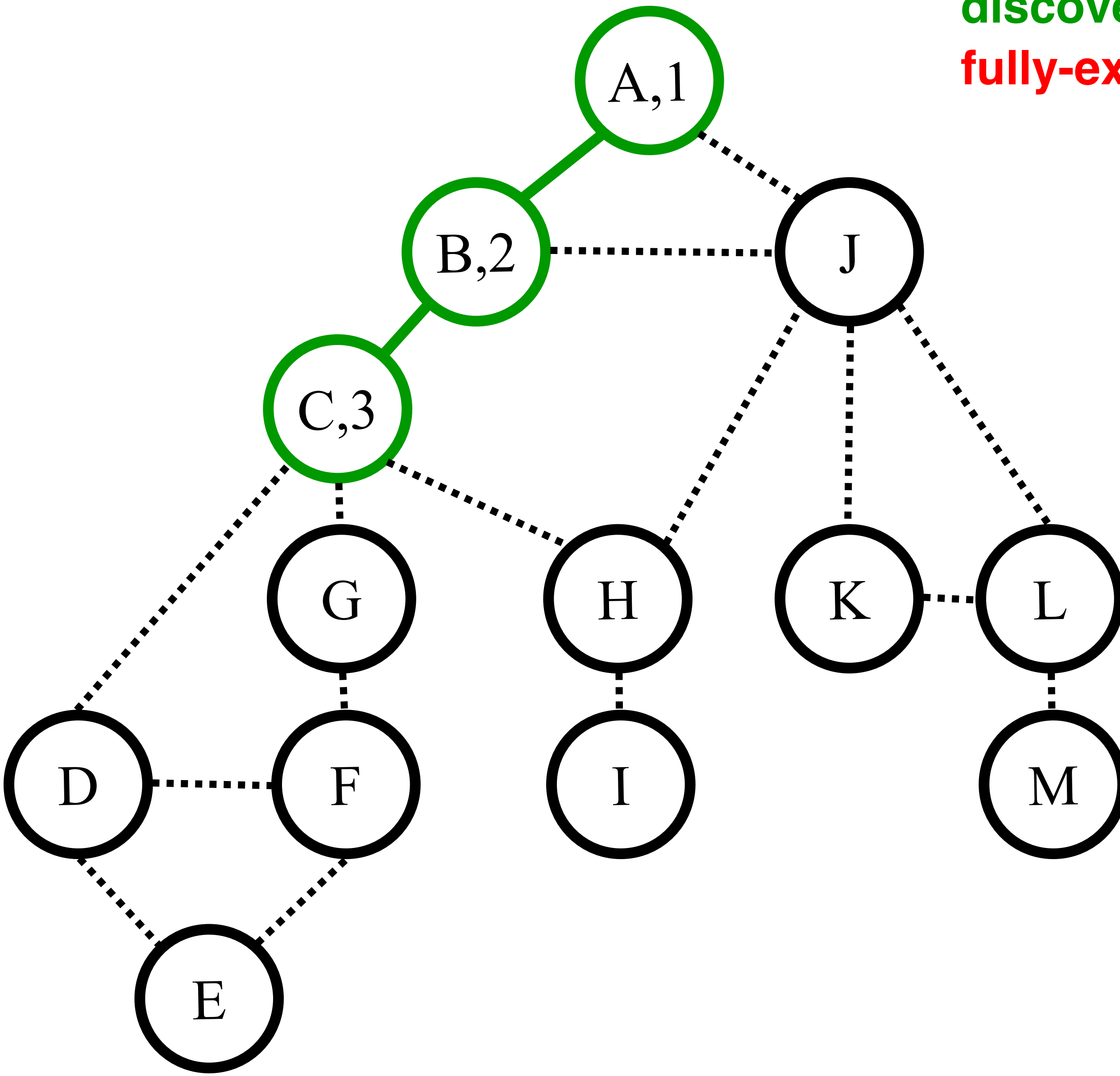
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (B,D,G,H)

st[] =
{1,2,3}

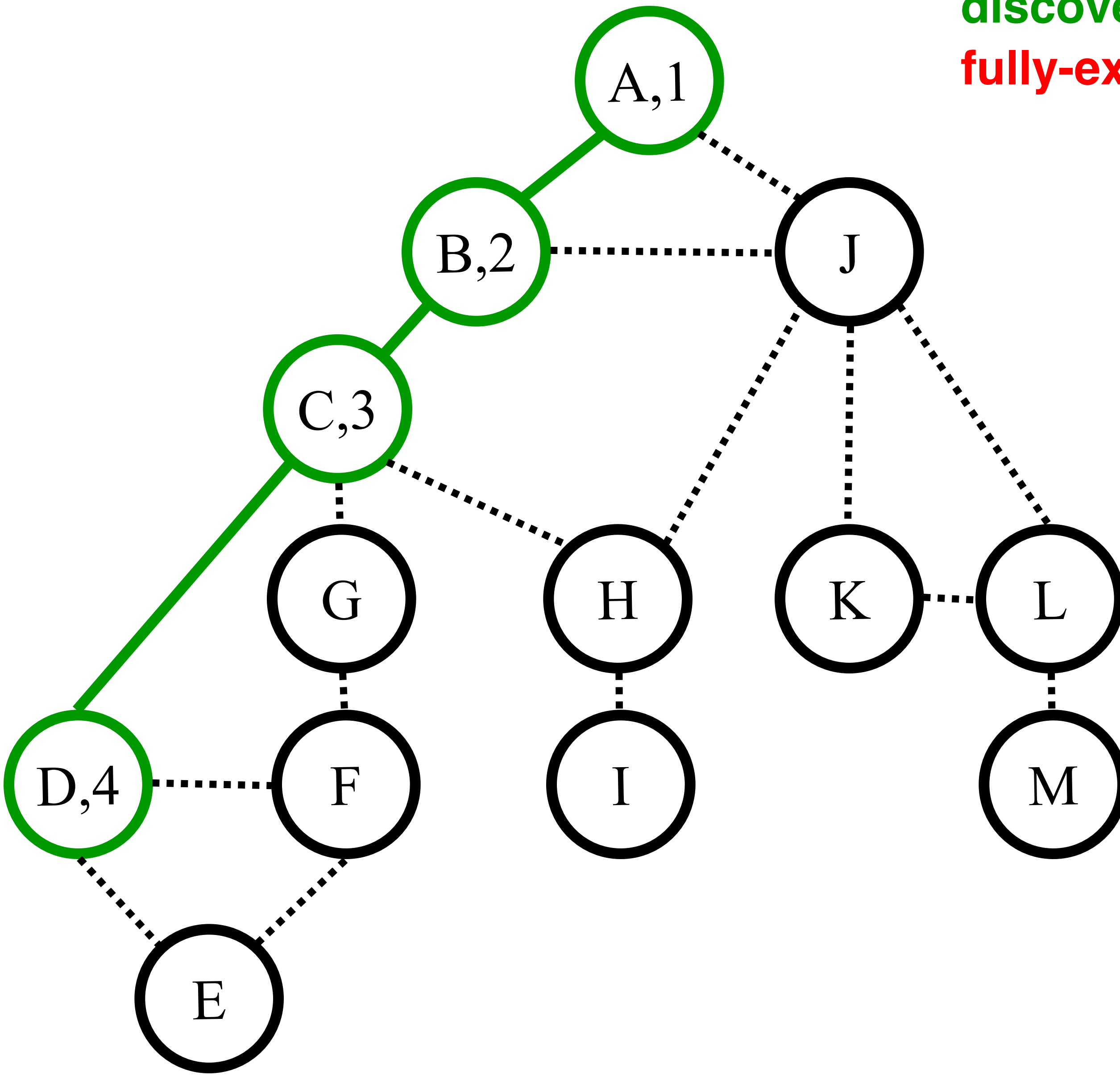
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,G,H)
D (C,E,F)

st[] =
{1,2,3,4}

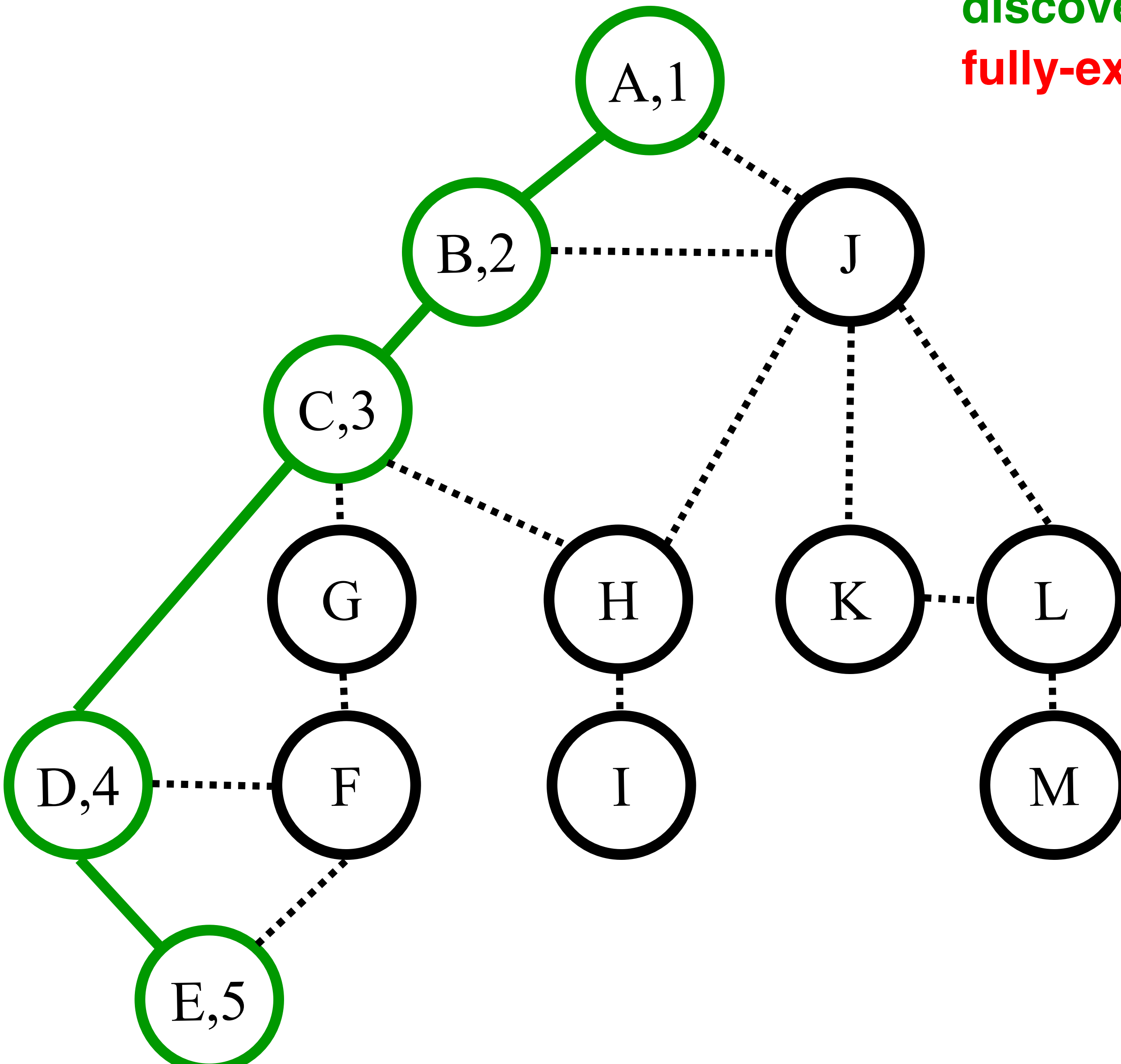
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,G,H)
D (~~C~~,~~E~~,F)
E (D,F)

st[] =
{1,2,3,4,5}

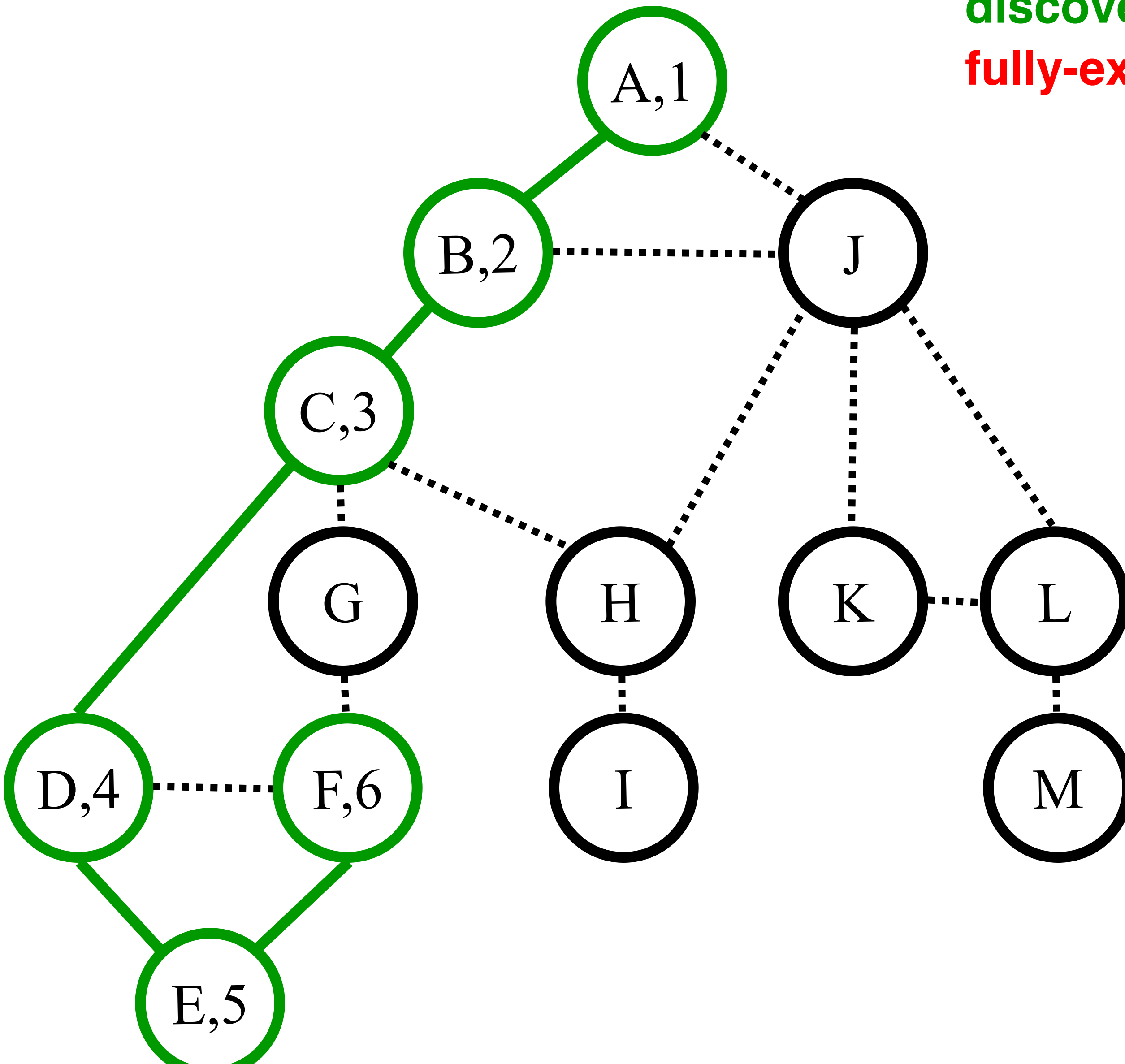
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,G,H)
D (~~C~~,~~E~~,F)
E (~~D~~,~~F~~)
F (D,E,G)

st[] = {1,2,3,4,5,6}

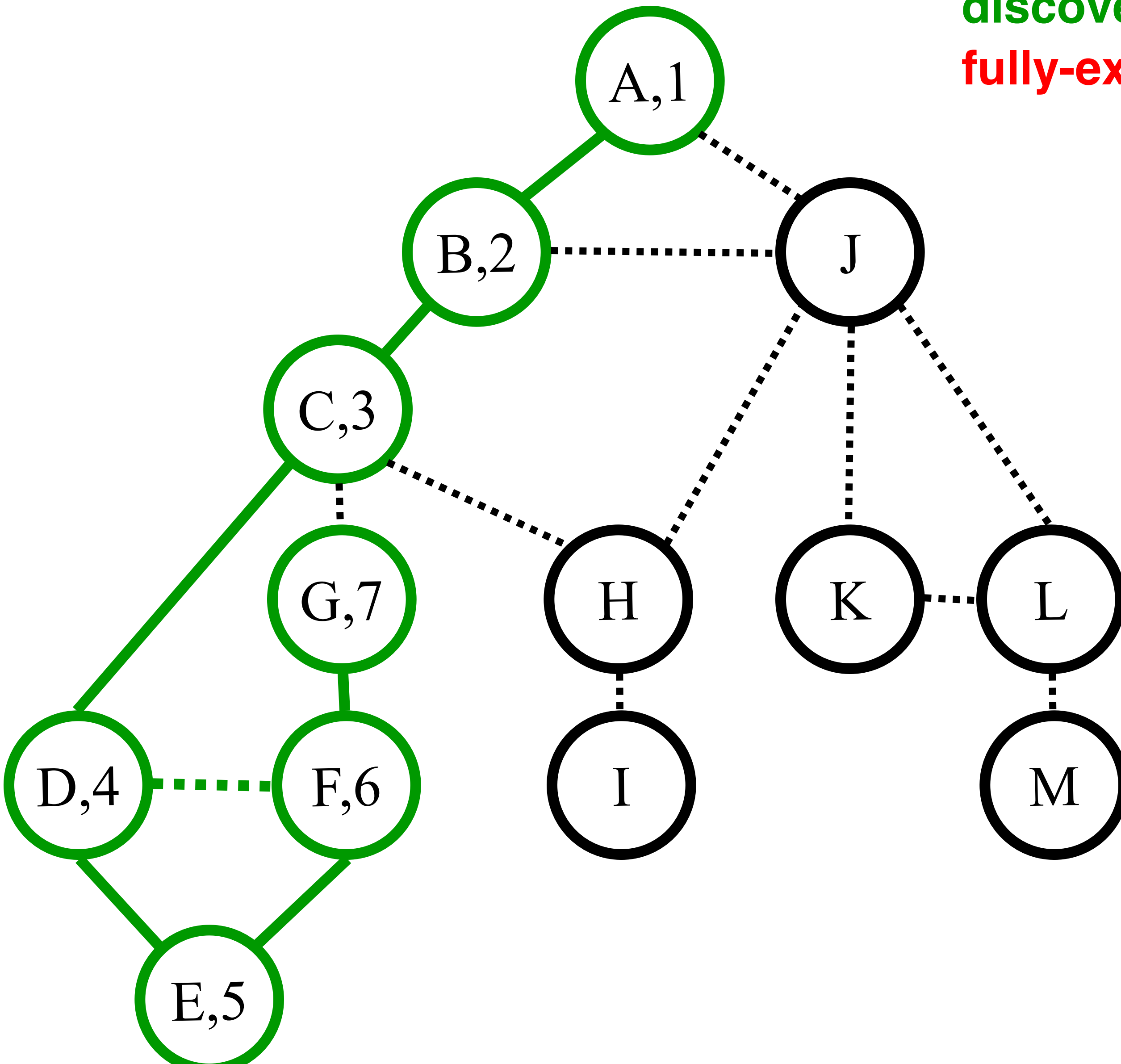
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,G,H)
D (~~C~~,~~E~~,F)
E (~~D~~,~~F~~)
F (~~D~~,~~E~~,~~G~~)
G (C,F)

st[] =
{1,2,3,4,5,
6,7}

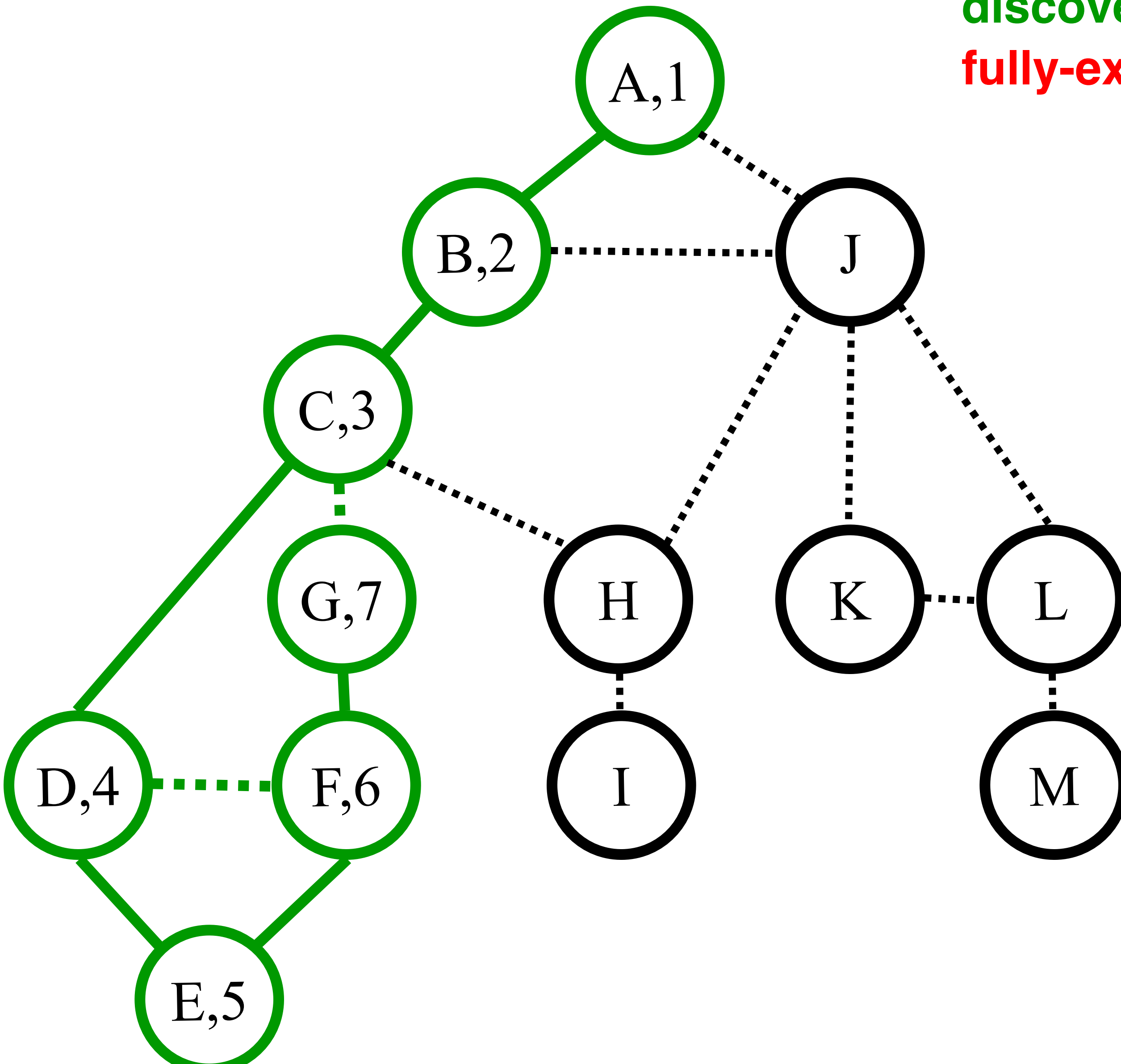
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,G,H)
D (~~C~~,~~E~~,F)
E (~~D~~,~~F~~)
F (~~D~~,~~E~~,~~G~~)
G (~~C~~,~~F~~)

st[] =
{1,2,3,4,5,
6,7}

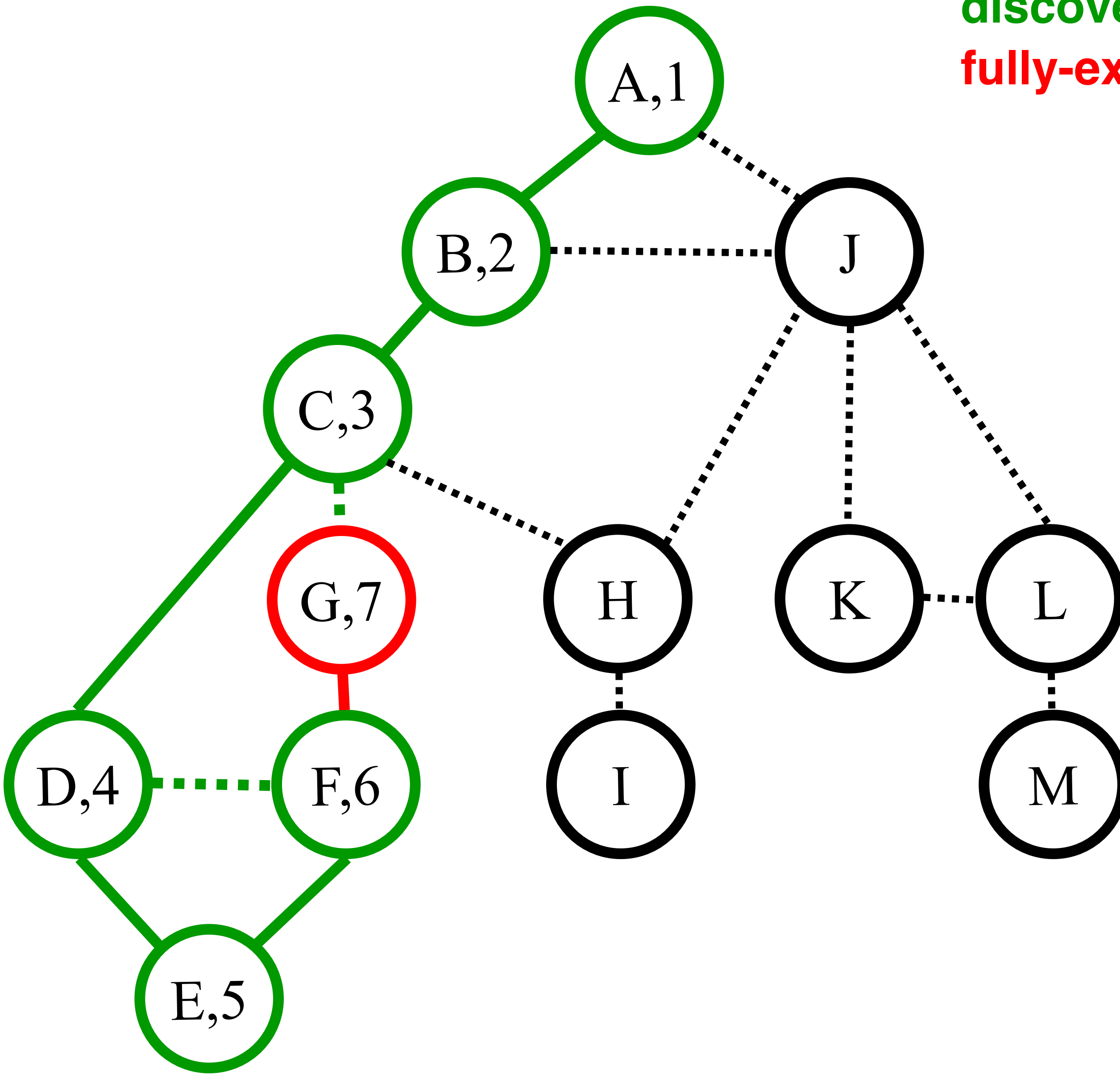
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,G,H)
D (~~C~~,~~E~~,F)
E (~~D~~,~~F~~)
F (~~D~~,~~E~~,~~G~~)

st[] =
{1,2,3,4,5,
6}

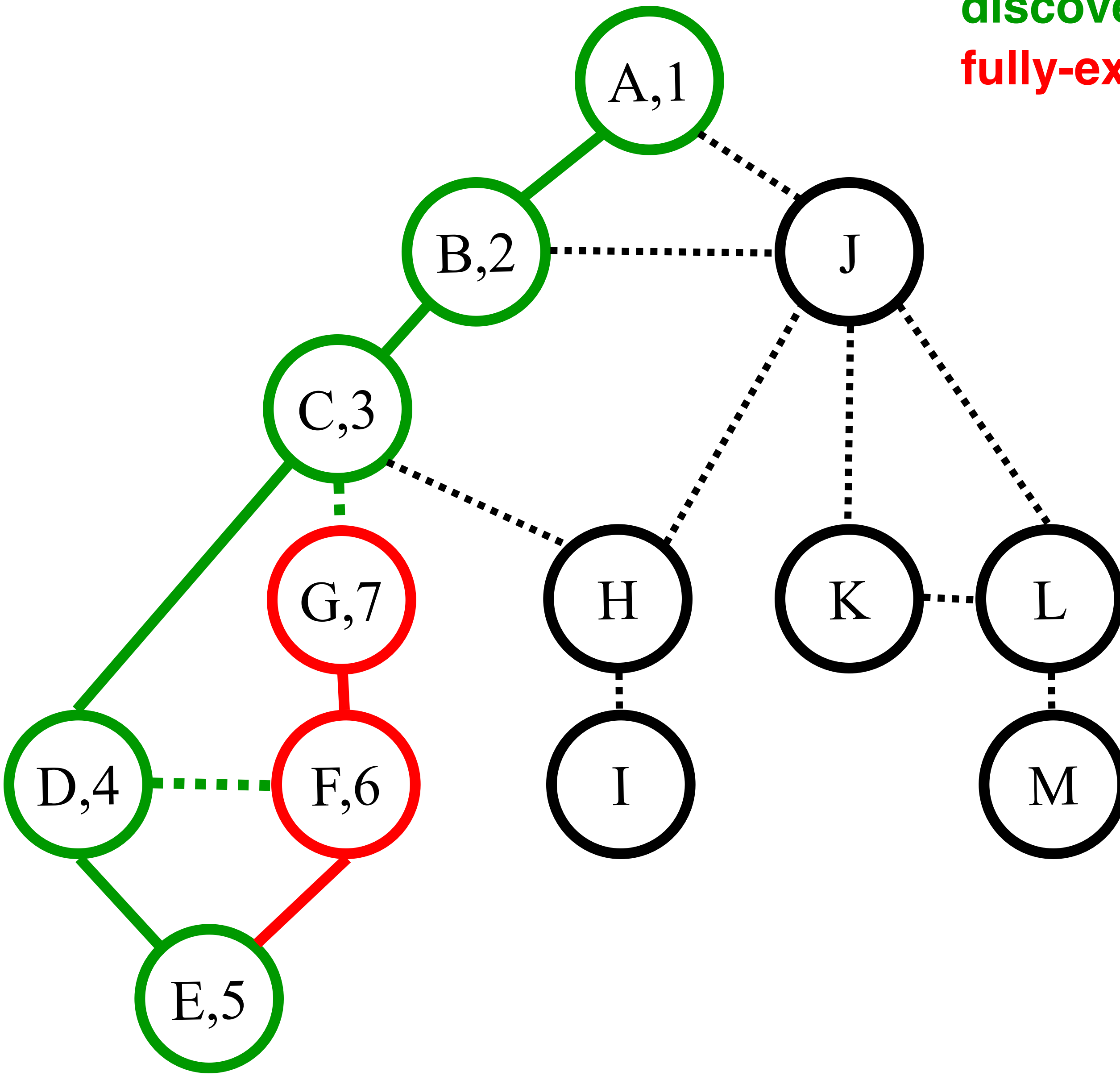
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,G,H)
D (~~C~~,~~E~~,F)
E (~~D~~,~~F~~)

st[] =
{1,2,3,4,5}

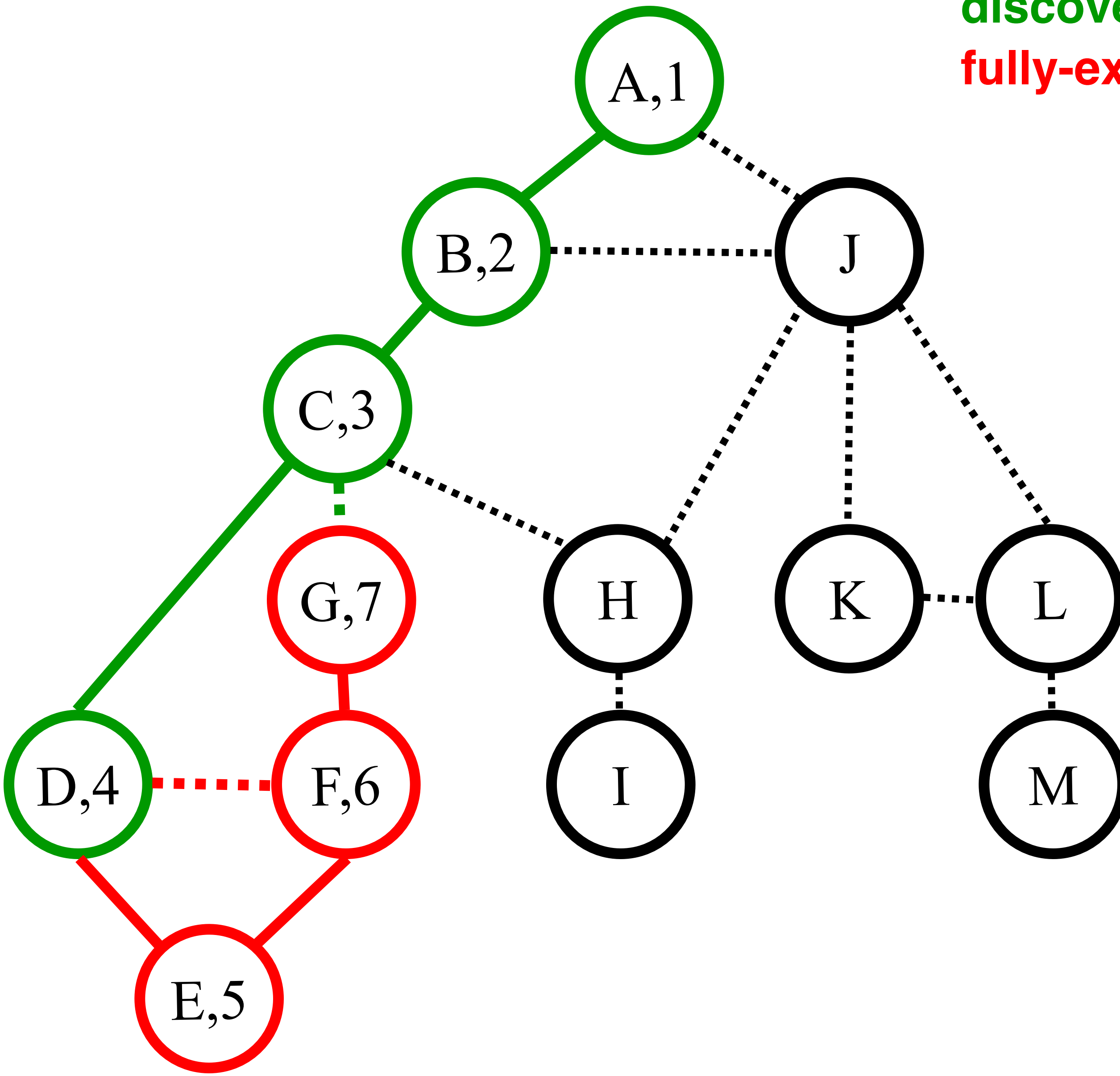
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,G,H)
D (~~C~~,~~E~~,~~F~~)

st[] =
{1,2,3,4}

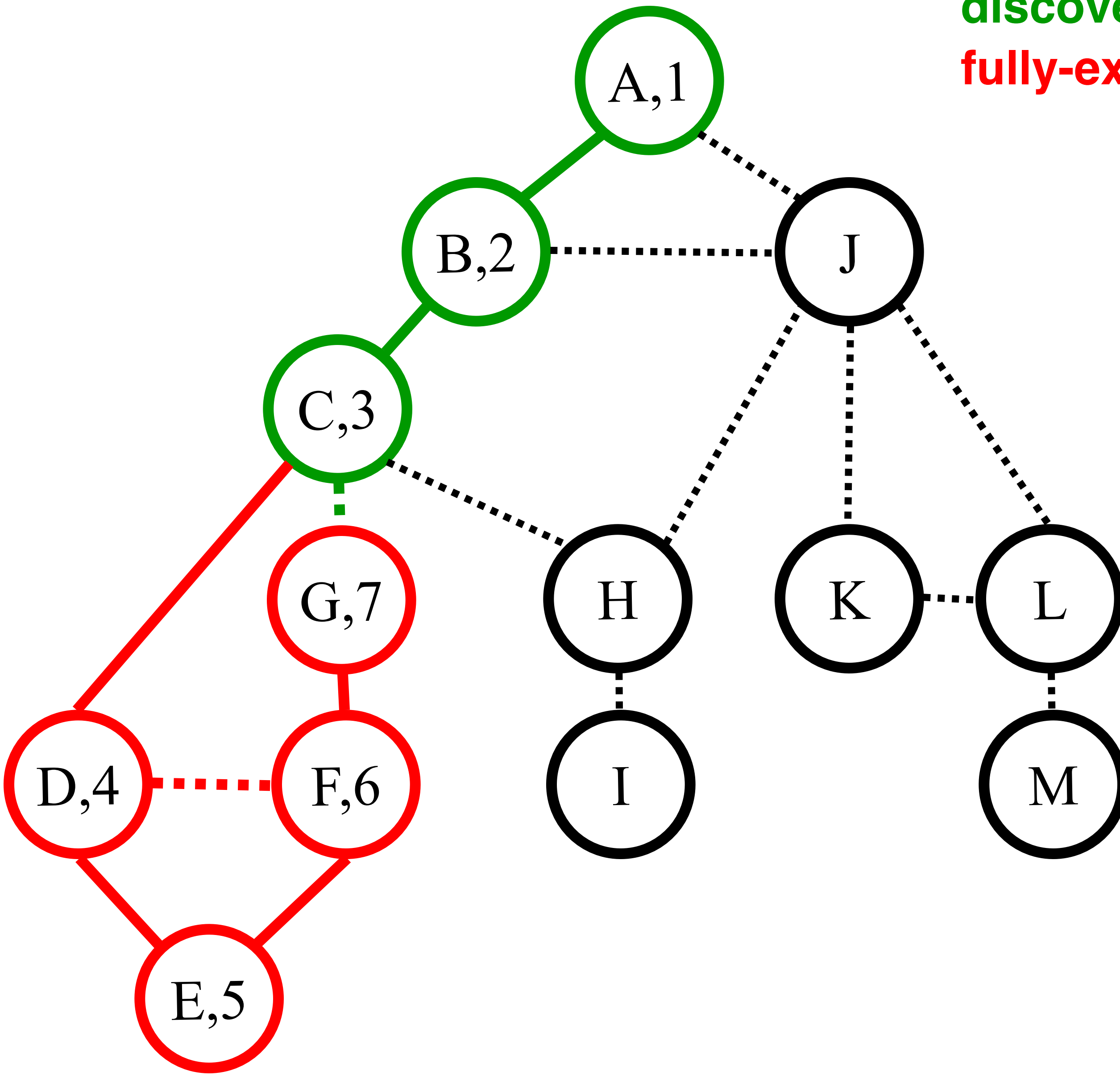
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

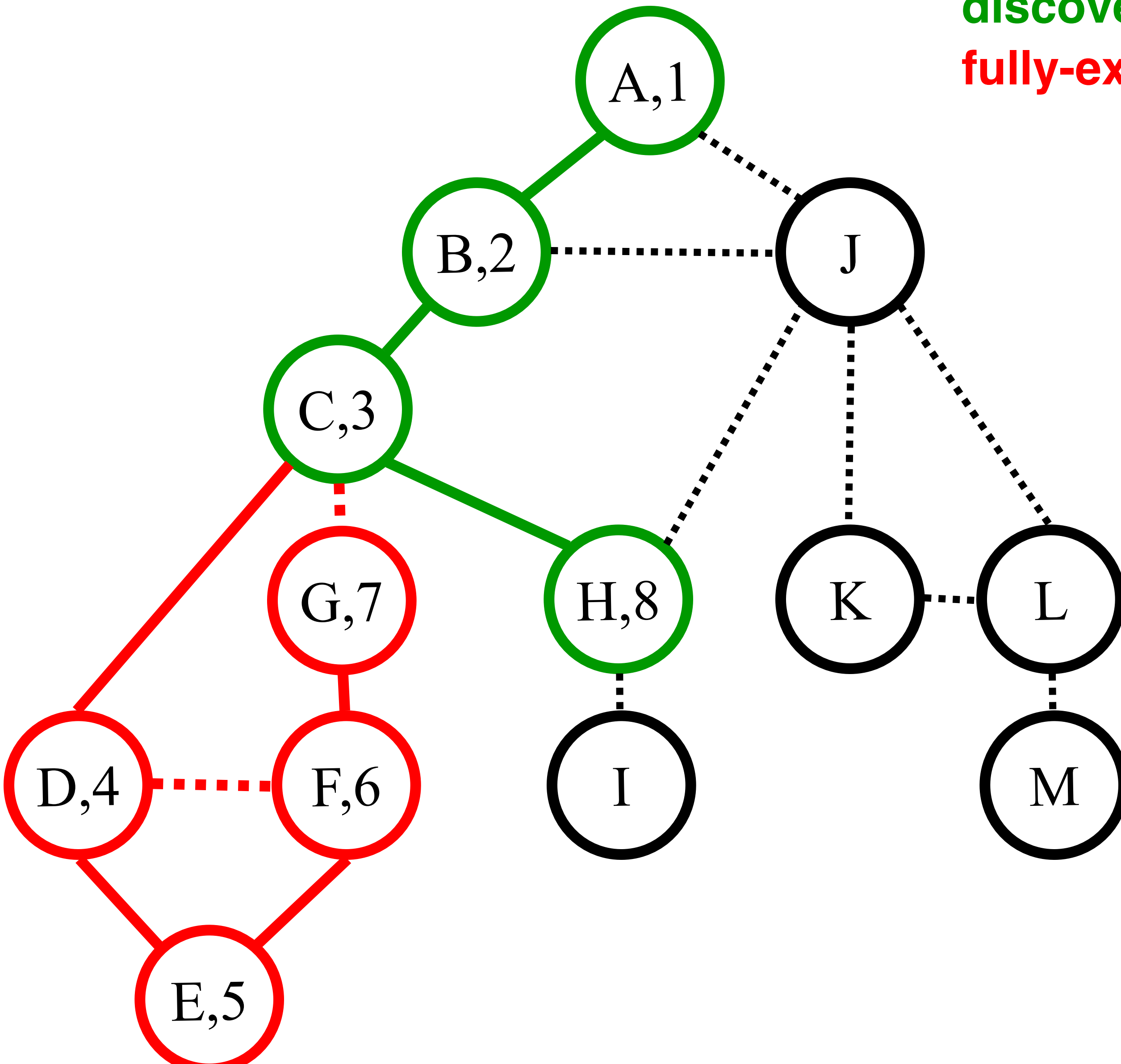
A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,G,H)

st[] =
{1,2,3}

DFS(A)

Color code:

undiscovered
discovered
fully-explored



Call Stack:
(Edge list)

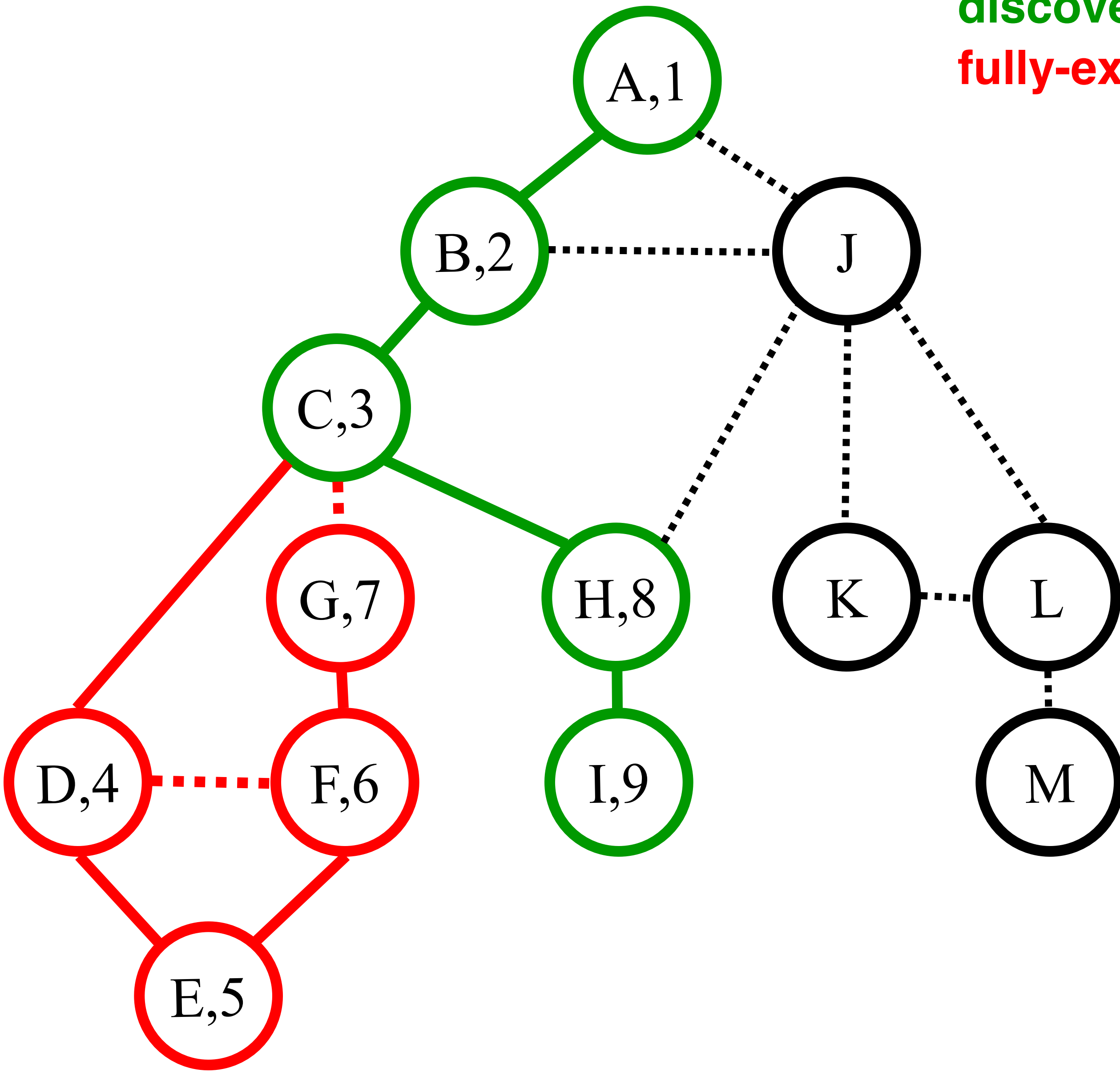
A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (C,I,J)

st[] =
{1,2,3,8}

DFS(A)

Color code:

undiscovered
discovered
fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,J)
I (H)

st[] =
{1,2,3,8,9}

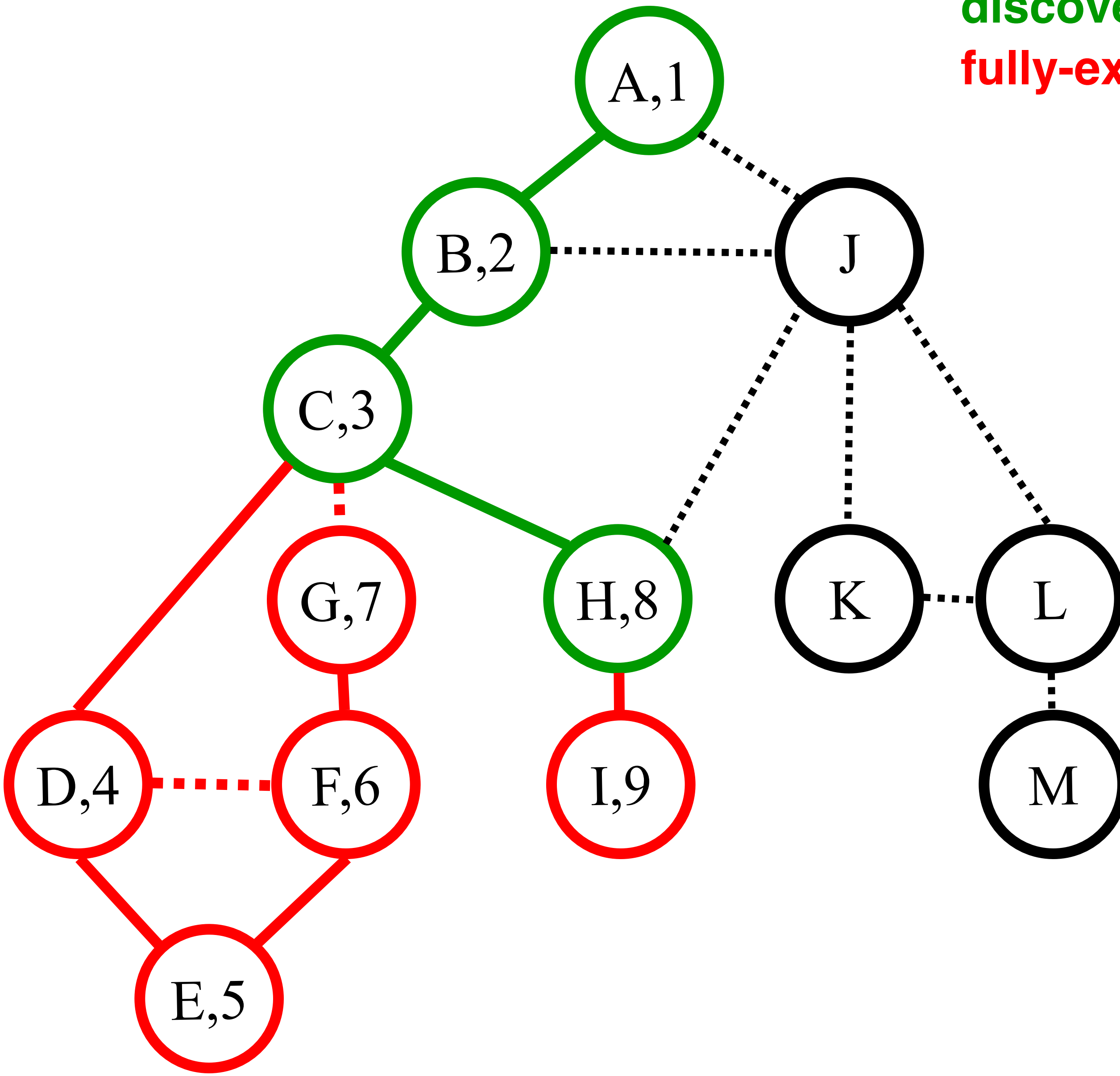
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,J)

st[] =
{1,2,3,8}

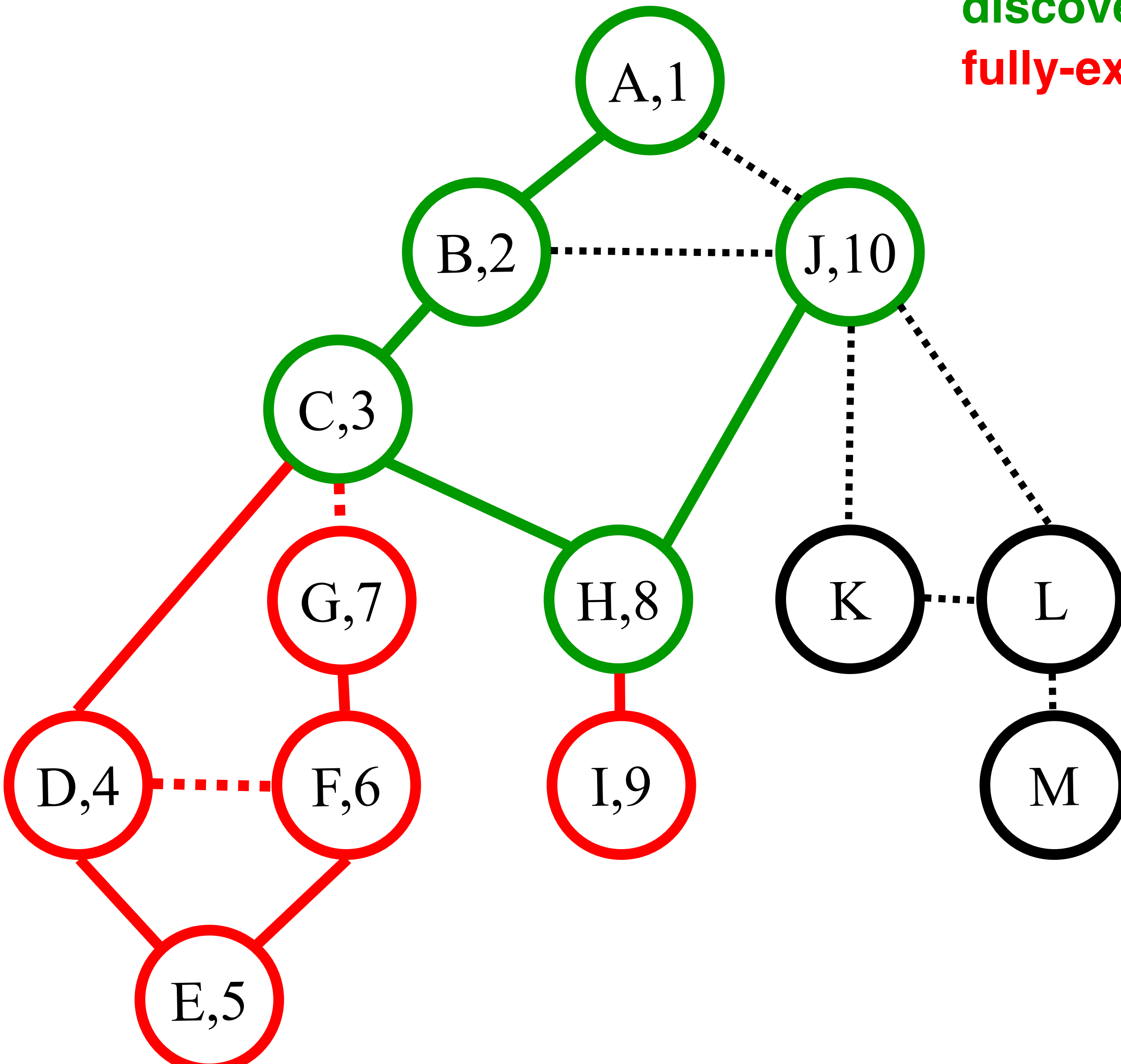
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

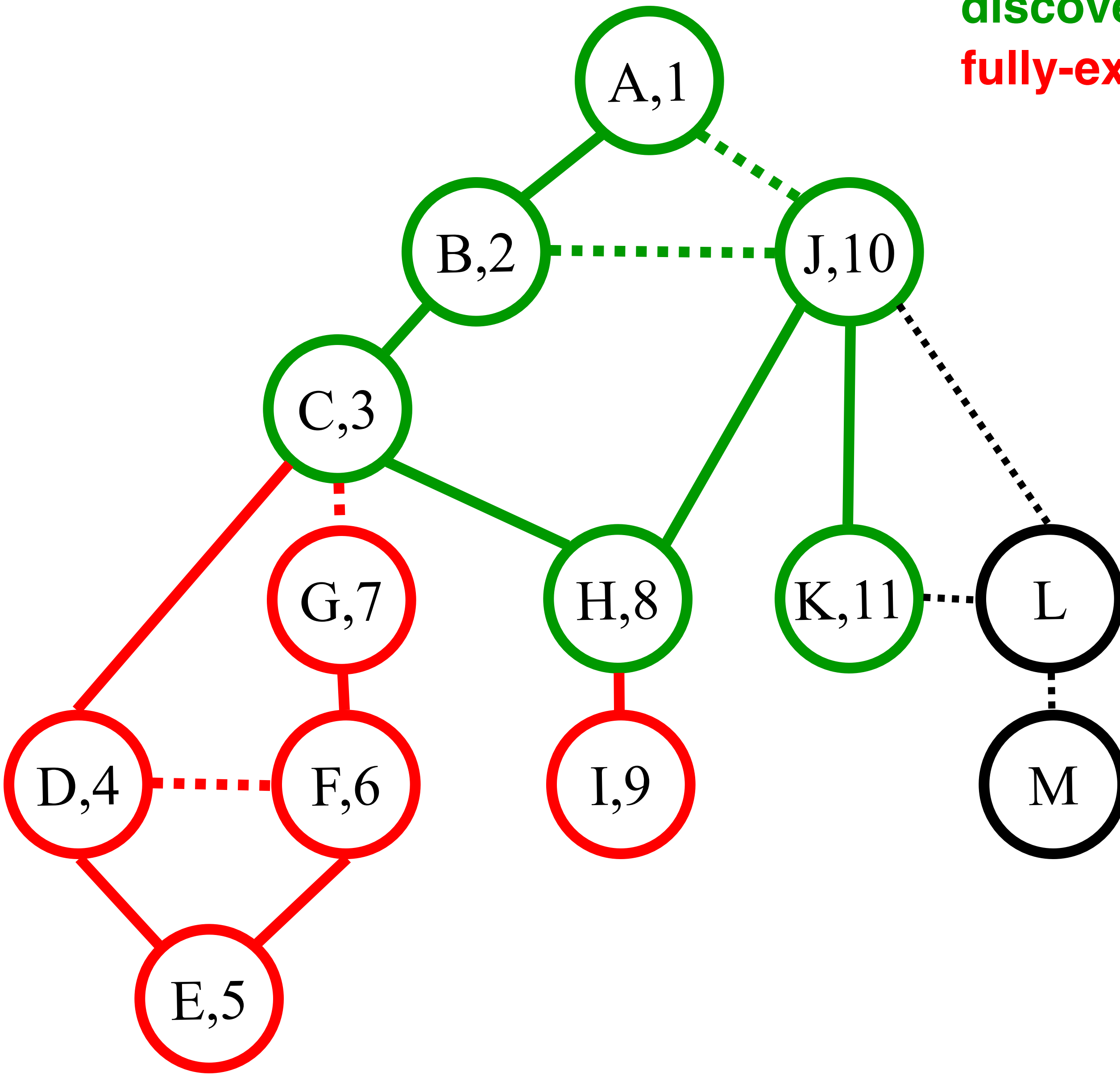
A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,~~J~~)
J (A,B,H,K,L)

st[] =
{1,2,3,8,
10}

DFS(A)

Color code:

undiscovered
discovered
fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,~~J~~)
J (~~A~~,~~B~~,~~H~~,~~K~~,L)
K (J,L)

st[] = {1,2,3,8,10,11}

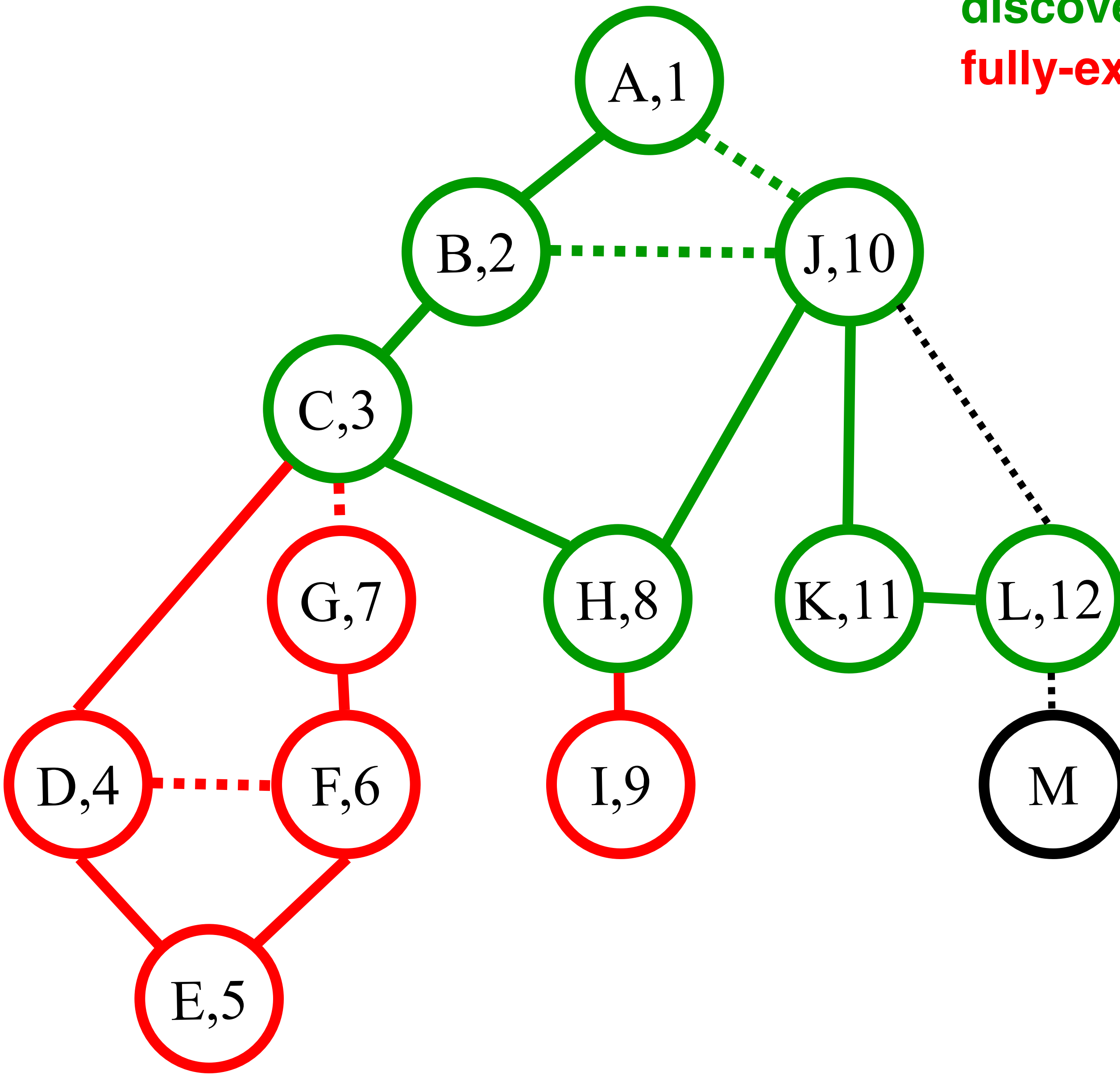
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

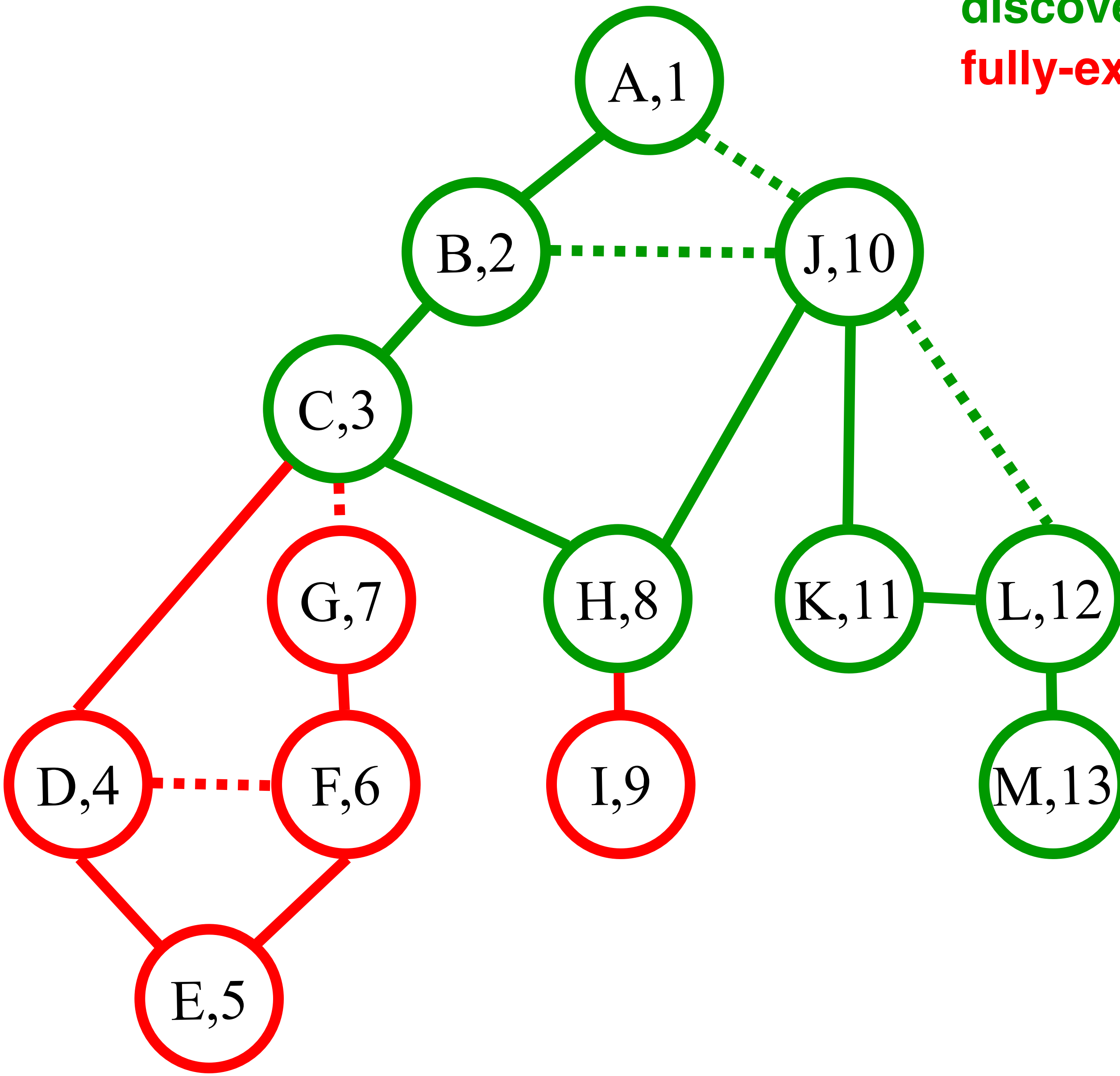
A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,~~J~~)
J (~~A~~,~~B~~,~~H~~,~~K~~,L)
K (~~J~~,~~L~~)
L (J,K,M)

st[] = {1,2,3,8,10,11,12}

DFS(A)

Color code:

undiscovered
discovered
fully-explored



Call Stack:
(Edge list)

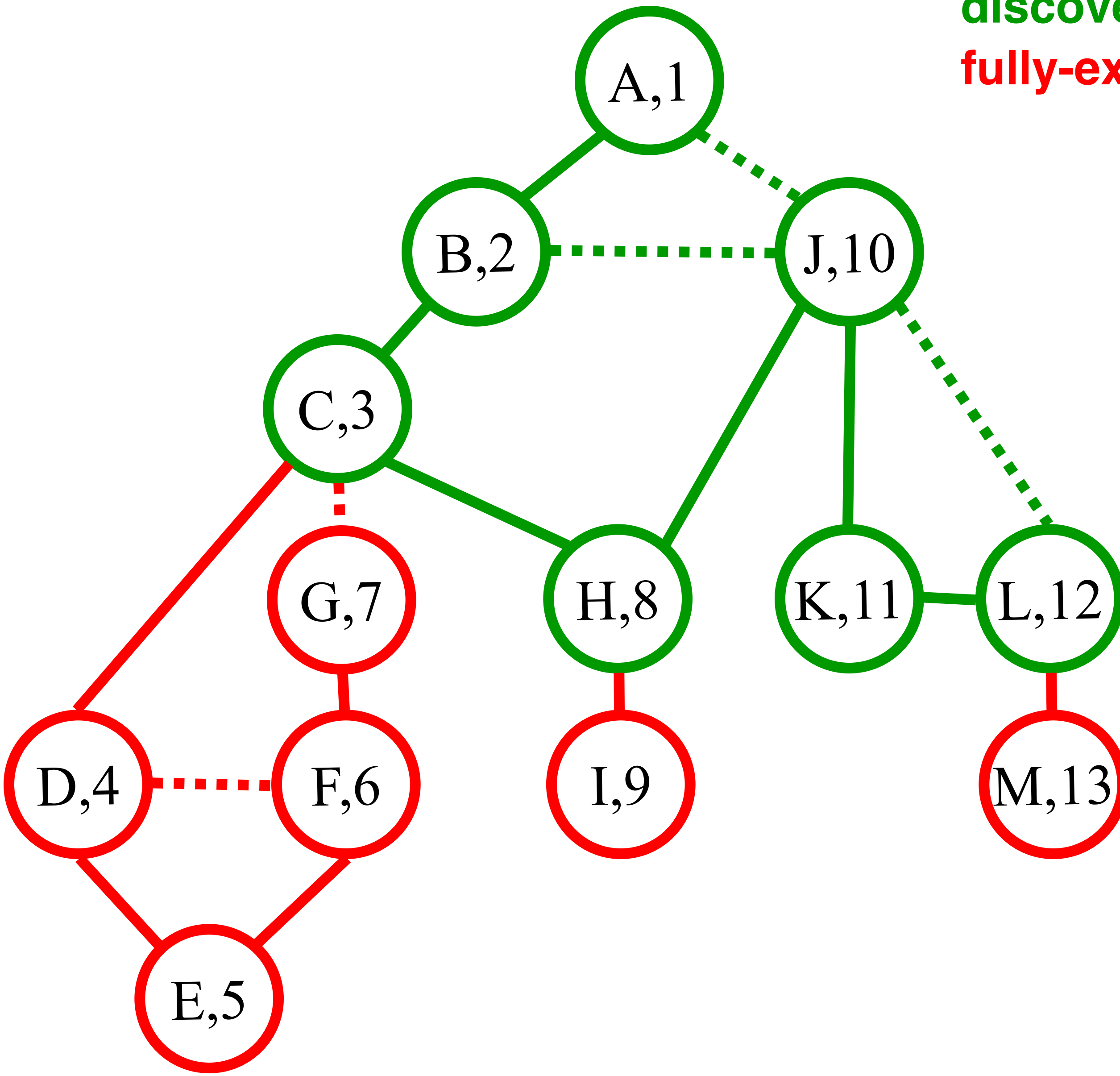
A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,~~J~~)
J (~~A~~,~~B~~,~~H~~,~~K~~,L)
K (~~J~~,~~L~~)
L (~~J~~,~~K~~,~~M~~)
M (L)

st[] =
{1,2,3,8,10
,11,12,13}

DFS(A)

Color code:

undiscovered
discovered
fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,~~J~~)
J (~~A~~,~~B~~,~~H~~,~~K~~,L)
K (~~J~~,~~L~~)
L (~~J~~,~~K~~,~~M~~)

st[] = {1,2,3,8,10,11,12}

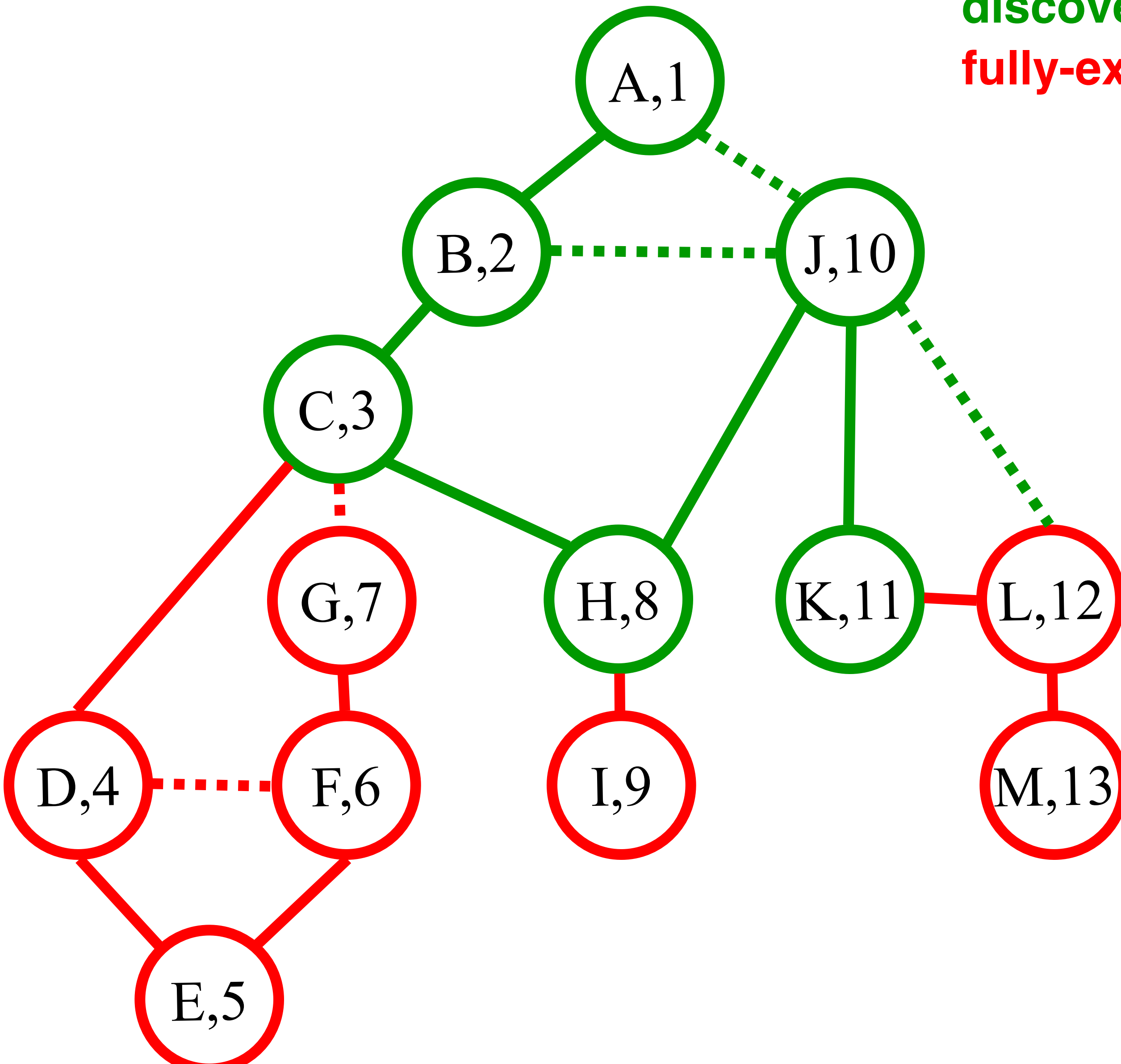
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,~~J~~)
J (~~A~~,~~B~~,~~H~~,~~K~~,L)
K (~~J~~,~~L~~)

st[] = {1,2,3,8,10,11}

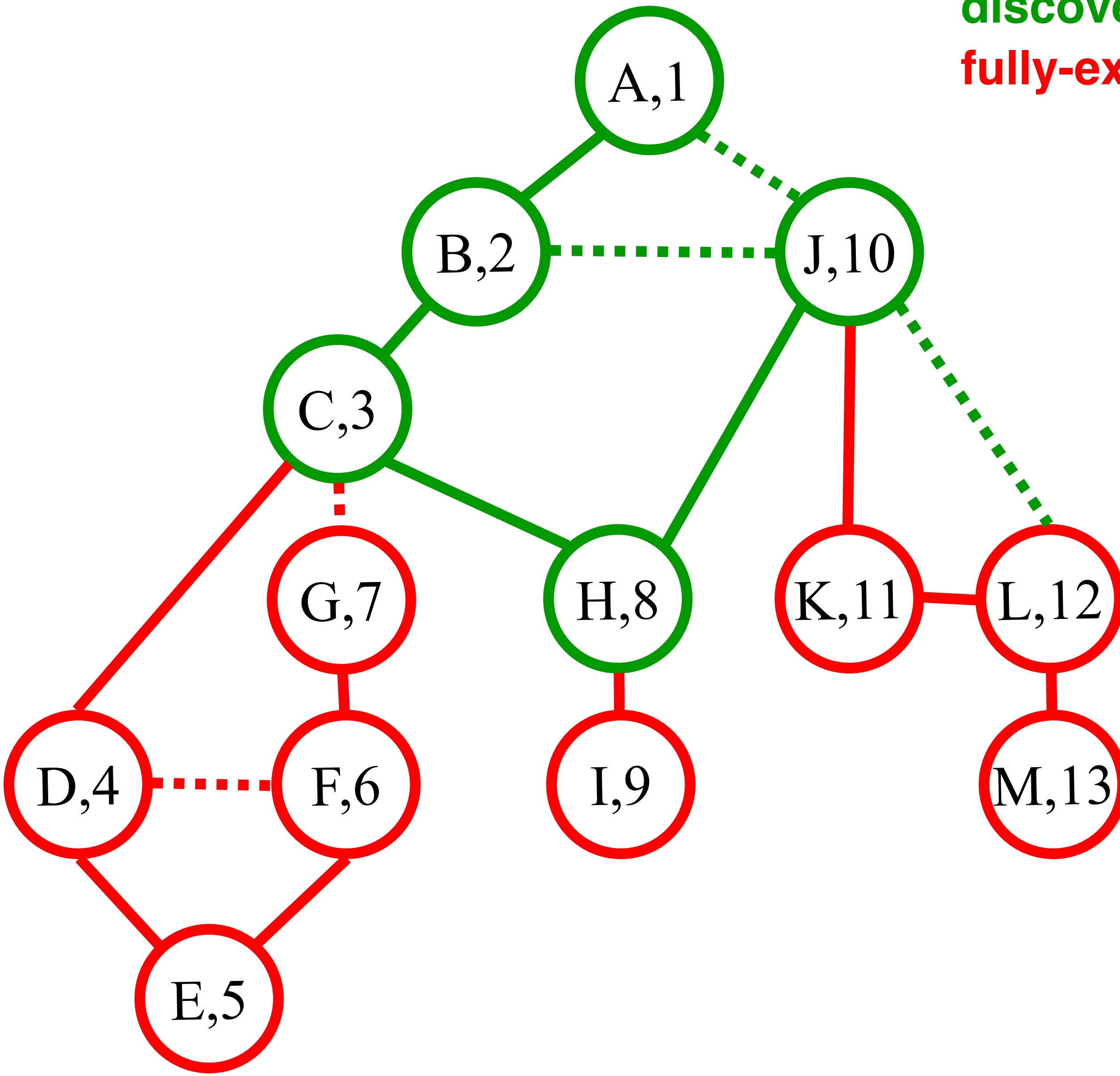
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,~~J~~)
J (~~A~~,~~B~~,~~H~~,~~K~~,L)

st[] =
{1,2,3,8,
10}

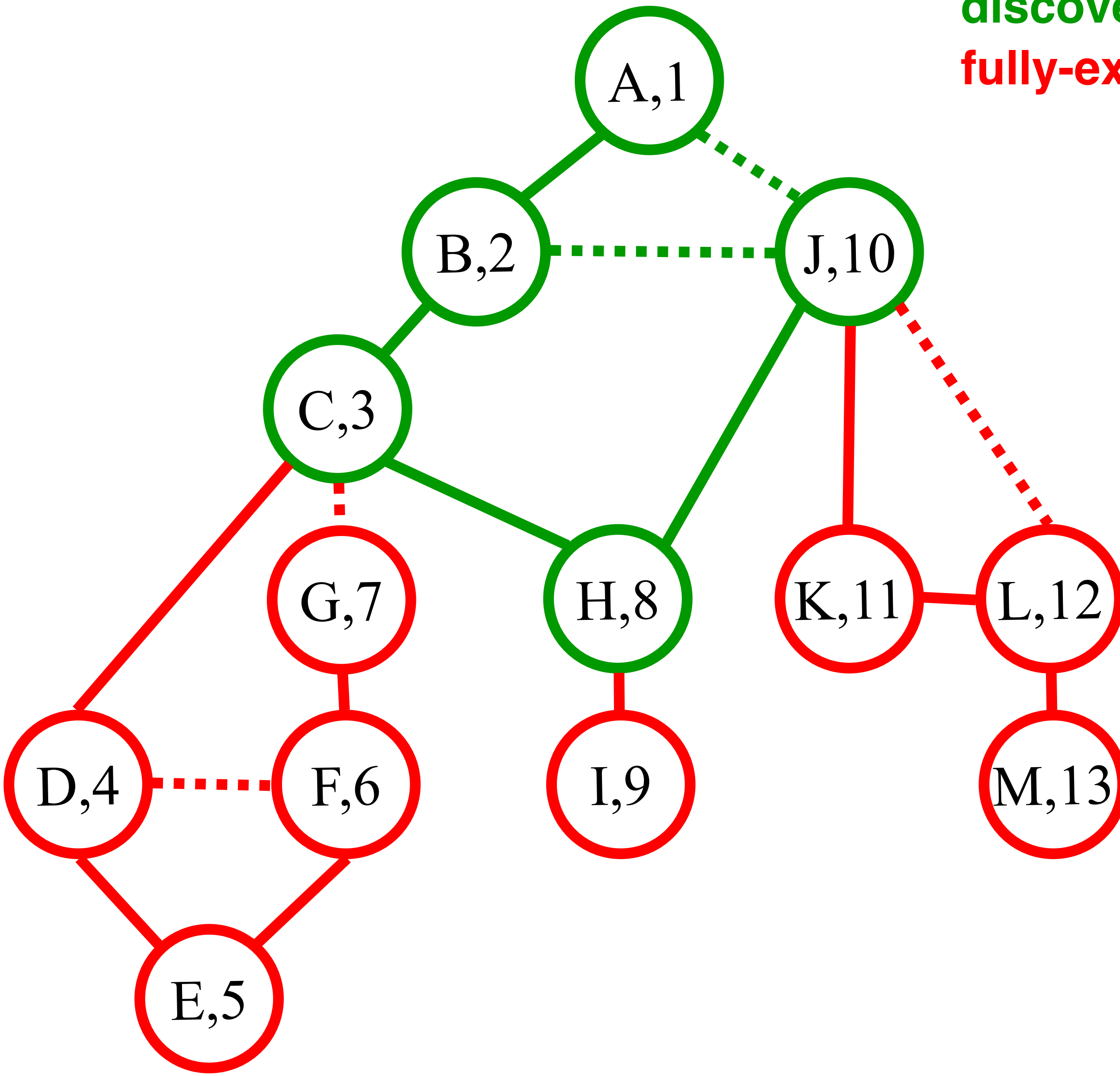
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

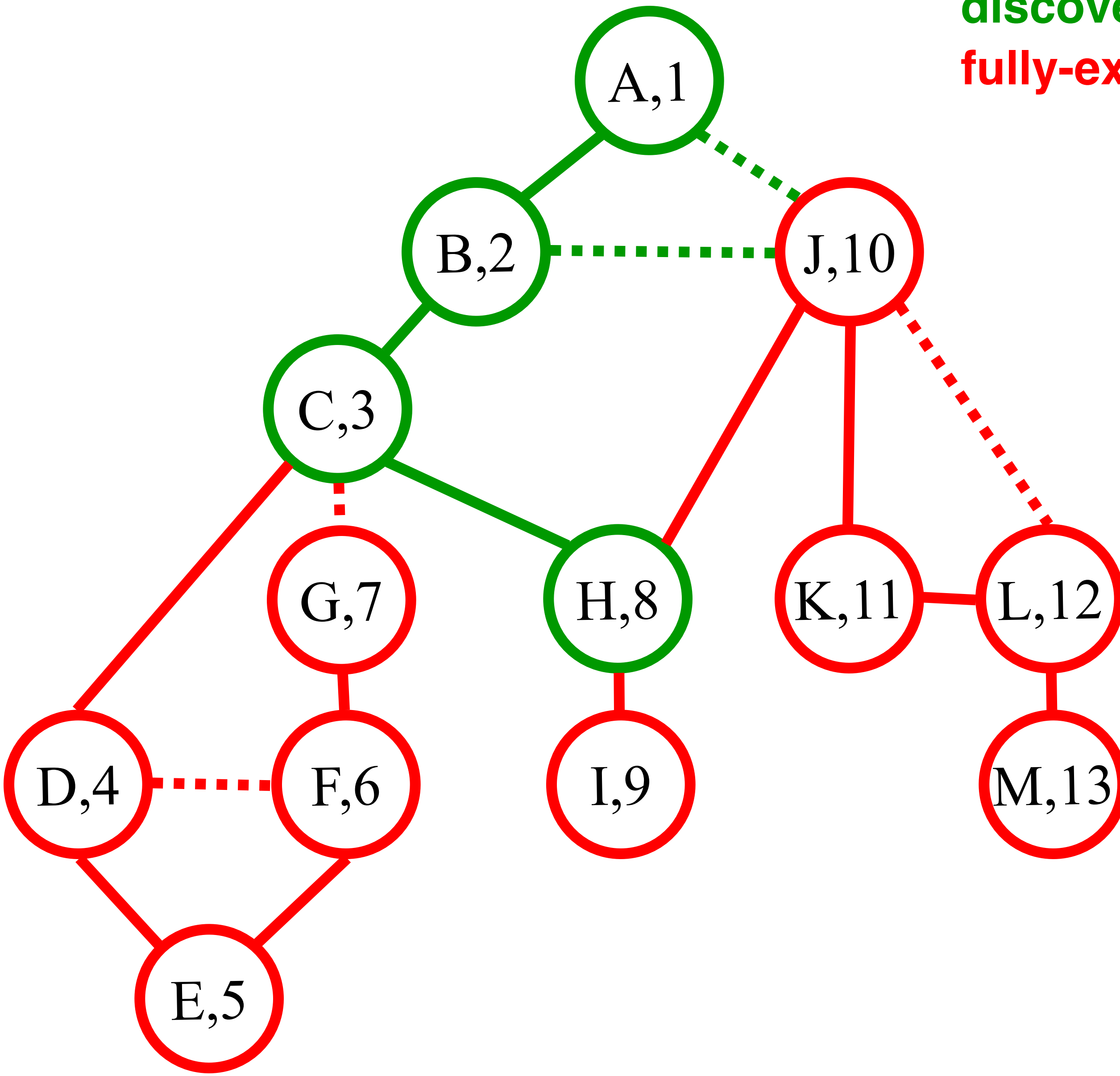
A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,~~J~~)
J (~~A~~,~~B~~,~~H~~,~~K~~,~~L~~)

st[] =
{1,2,3,8,
10}

DFS(A)

Color code:

undiscovered
discovered
fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)
H (~~C~~,~~I~~,~~J~~)

st[] =
{1,2,3,8}

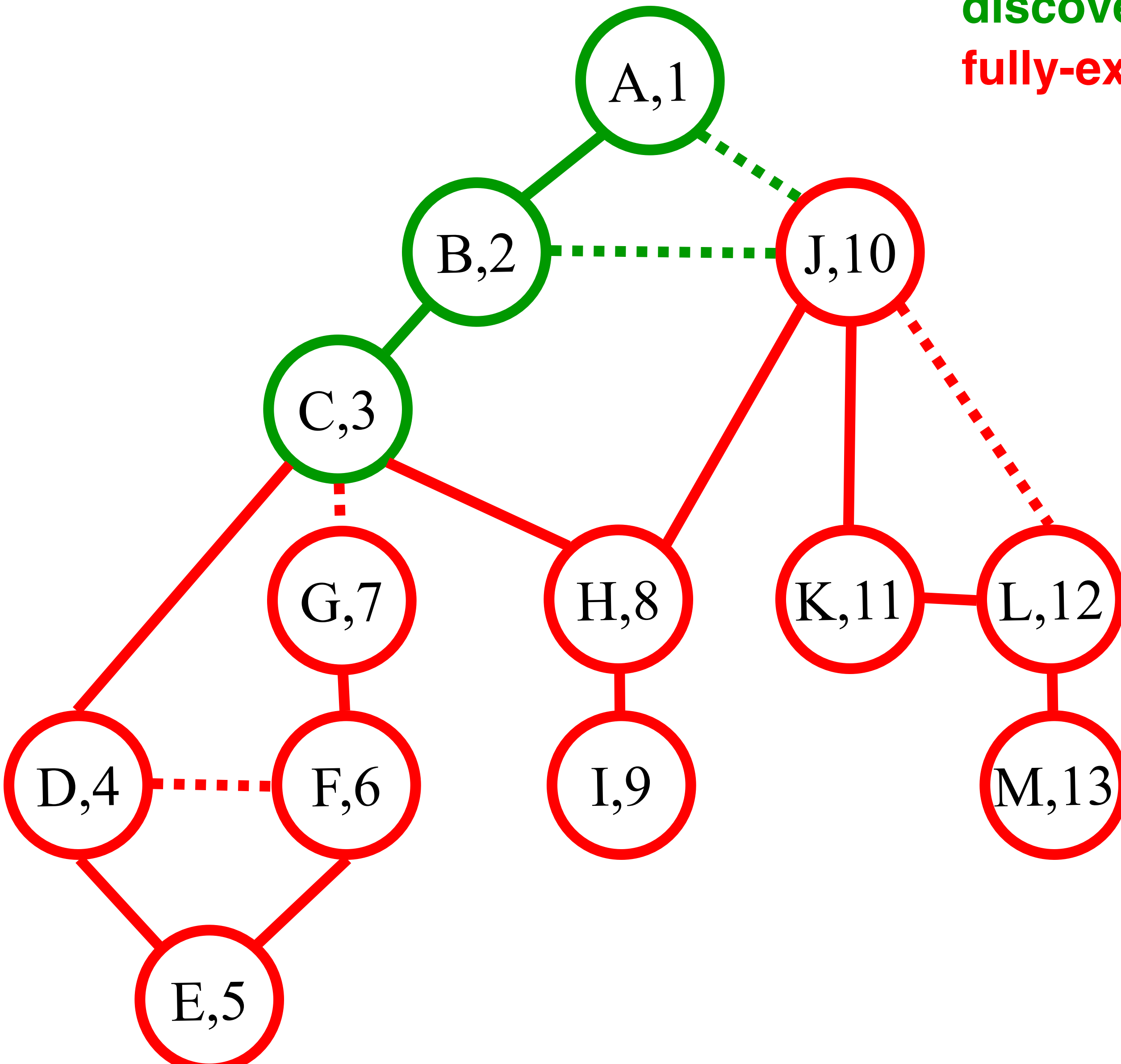
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)
C (~~B~~,~~D~~,~~G~~,~~H~~)

st[] =
{1,2,3}

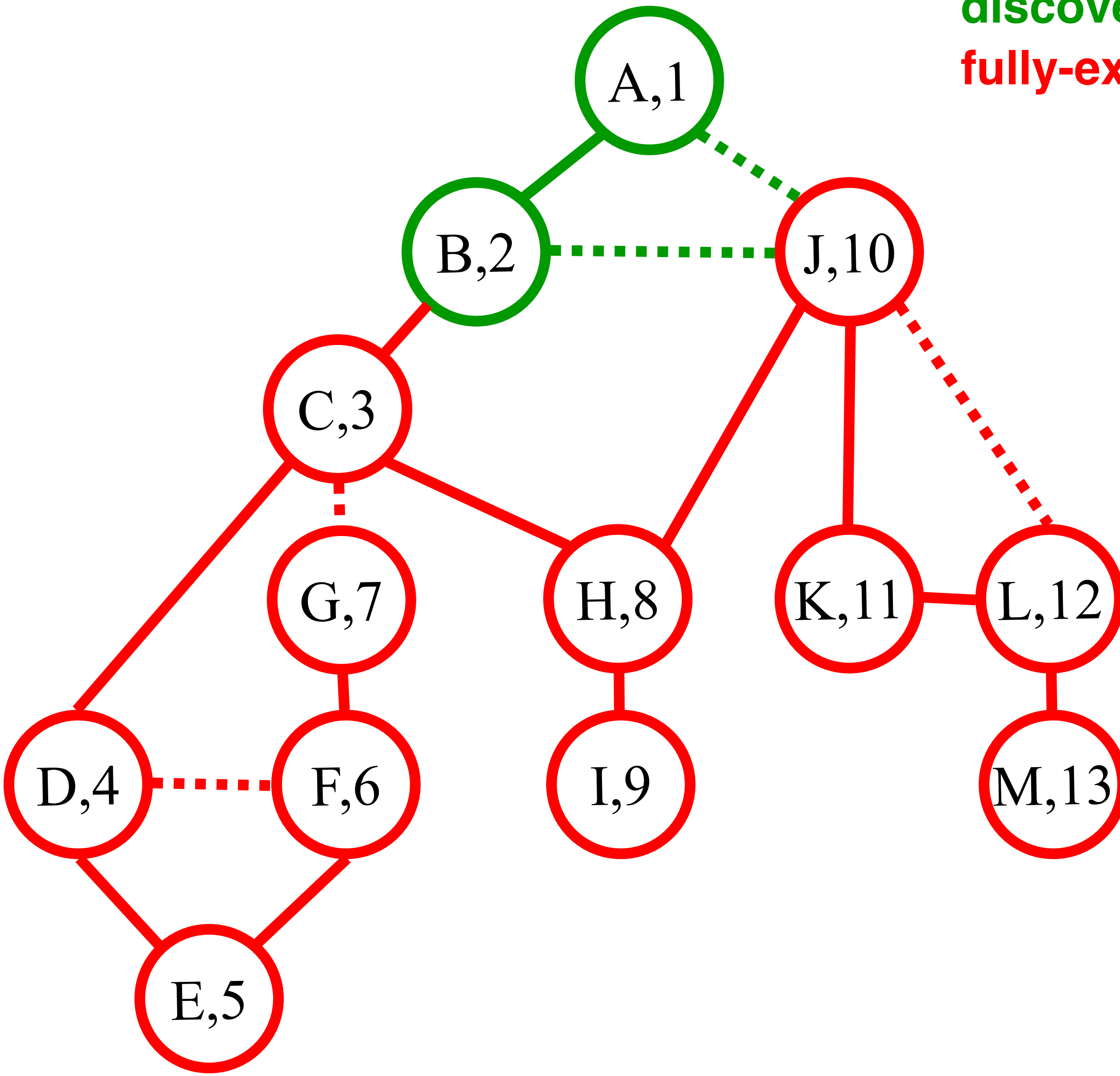
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)
B (~~A~~,~~C~~,J)

st[] =
{1,2}

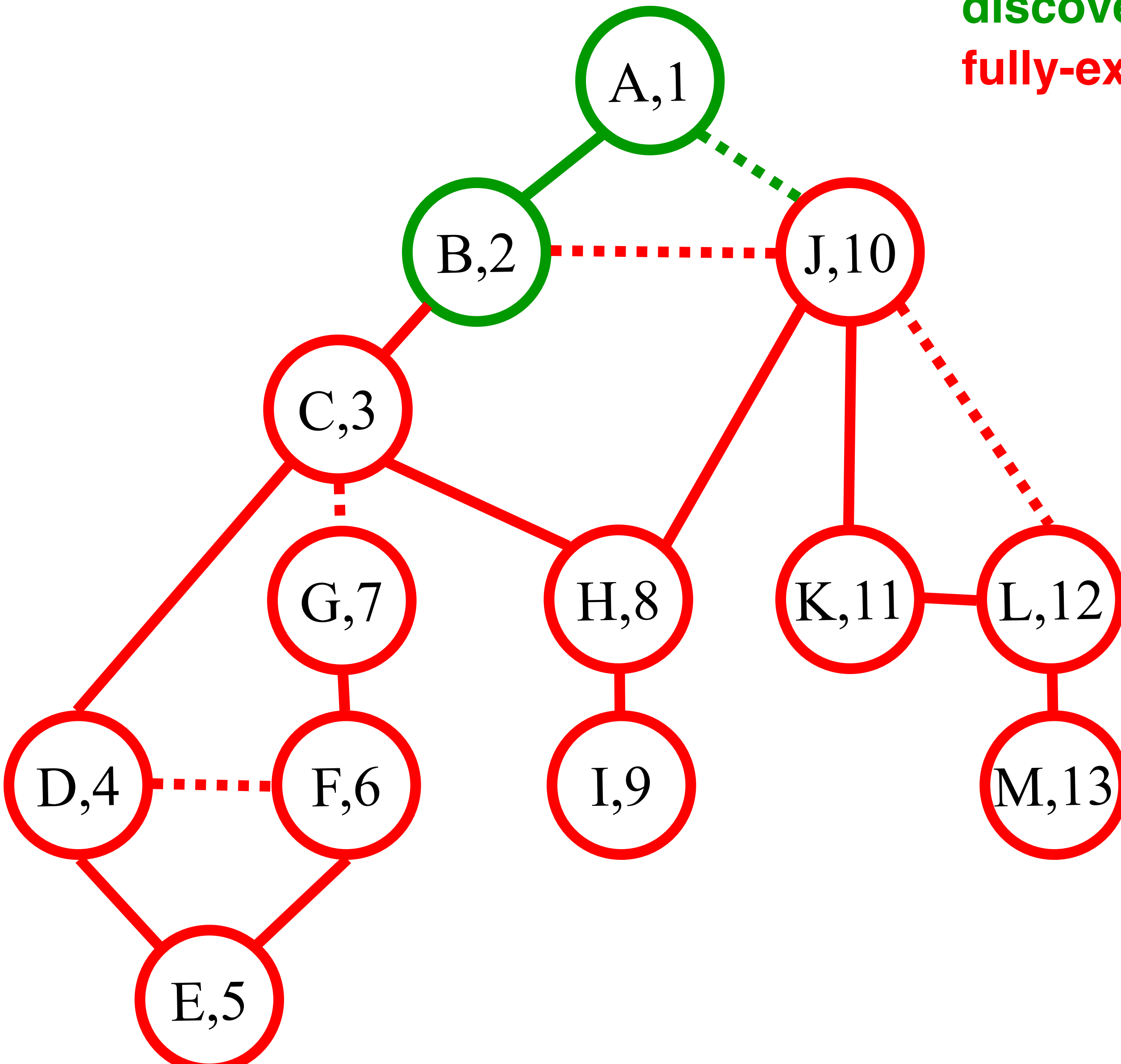
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

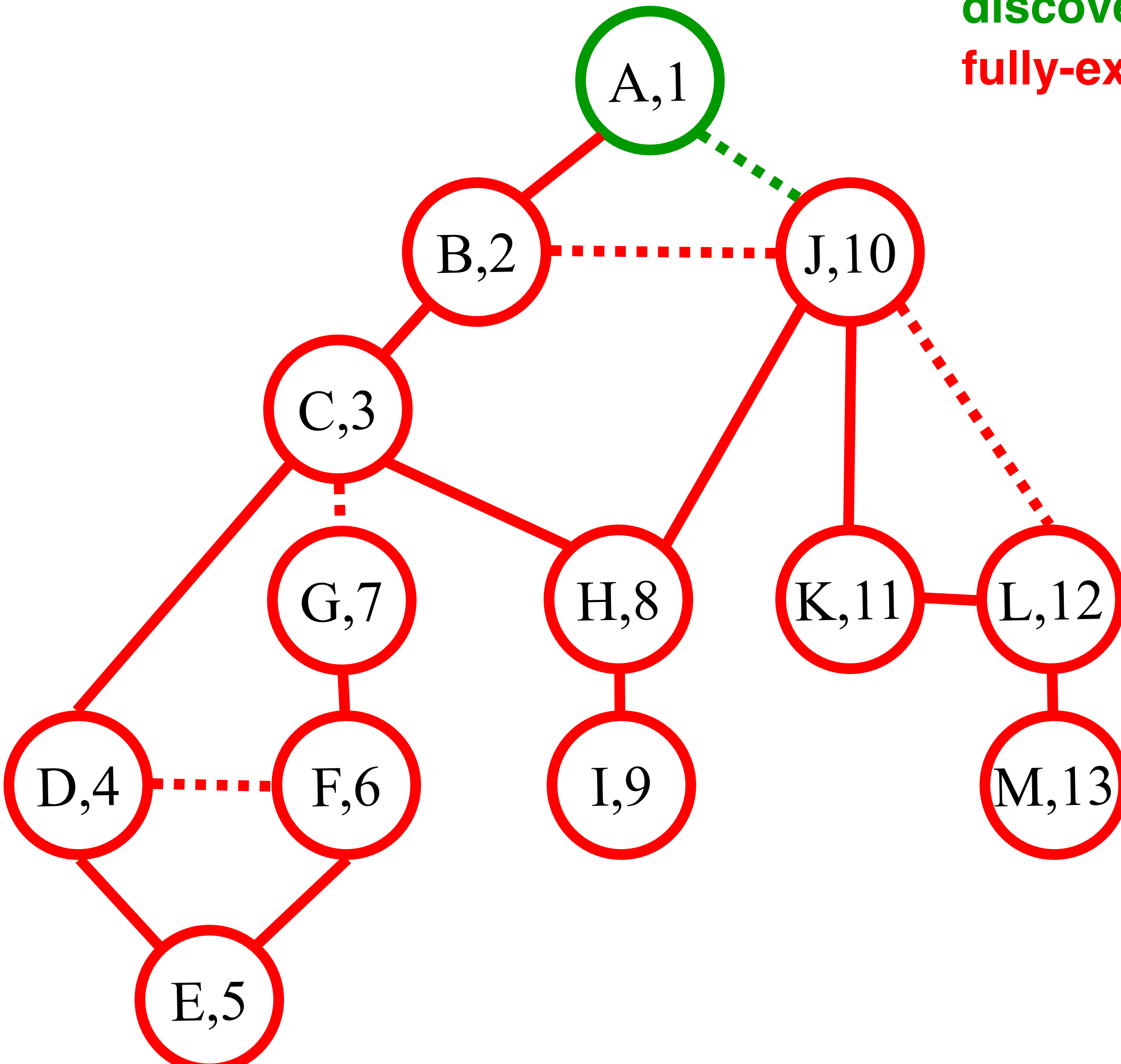
A (~~B~~,J)
B (~~A~~,~~C~~,~~J~~)

st[] =
{1,2}

DFS(A)

Color code:

undiscovered
discovered
fully-explored



Call Stack:
(Edge list)

A (~~B~~,J)

st[] =
{1}

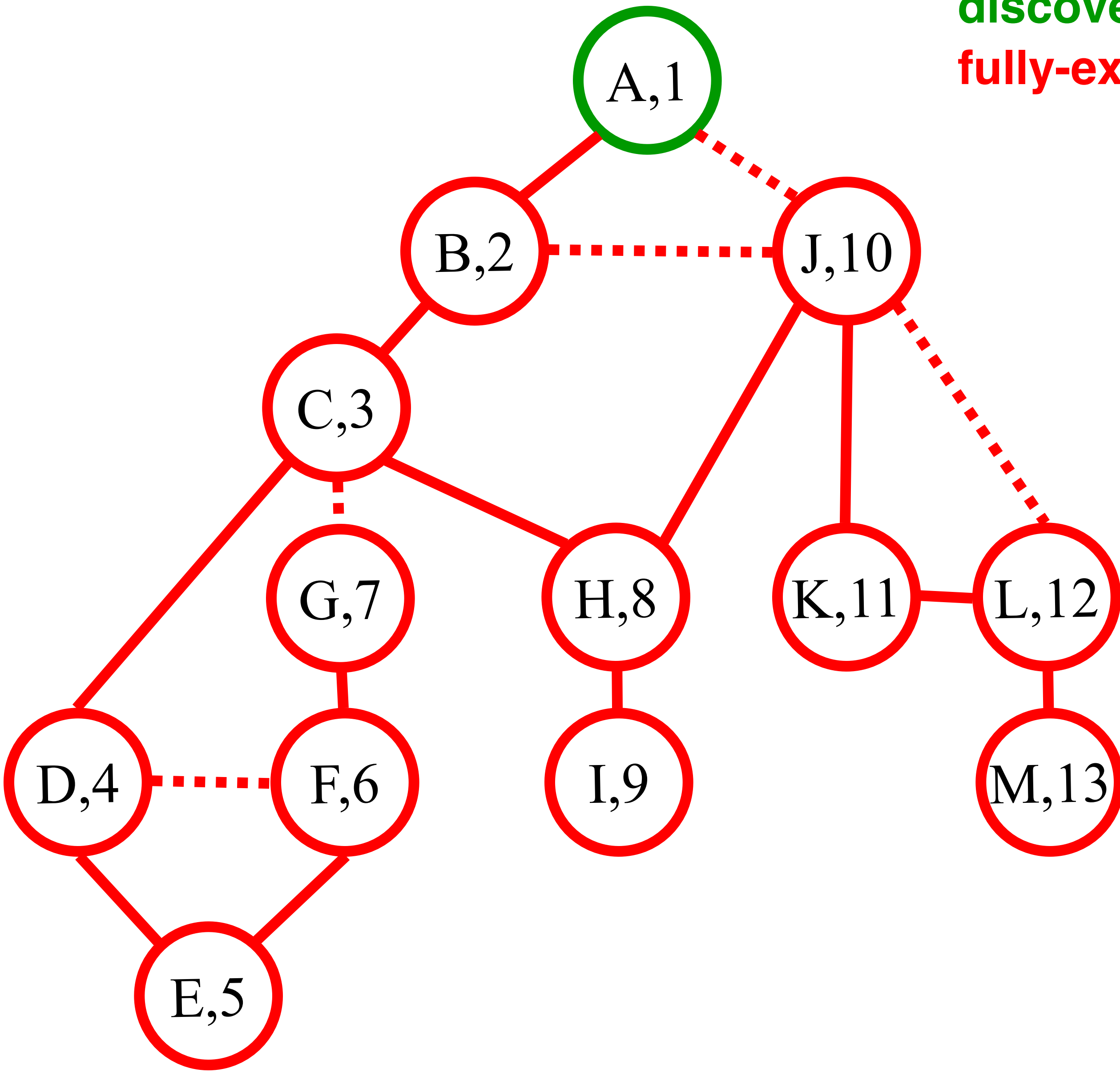
DFS(A)

Color code:

undiscovered

discovered

fully-explored



Call Stack:
(Edge list)

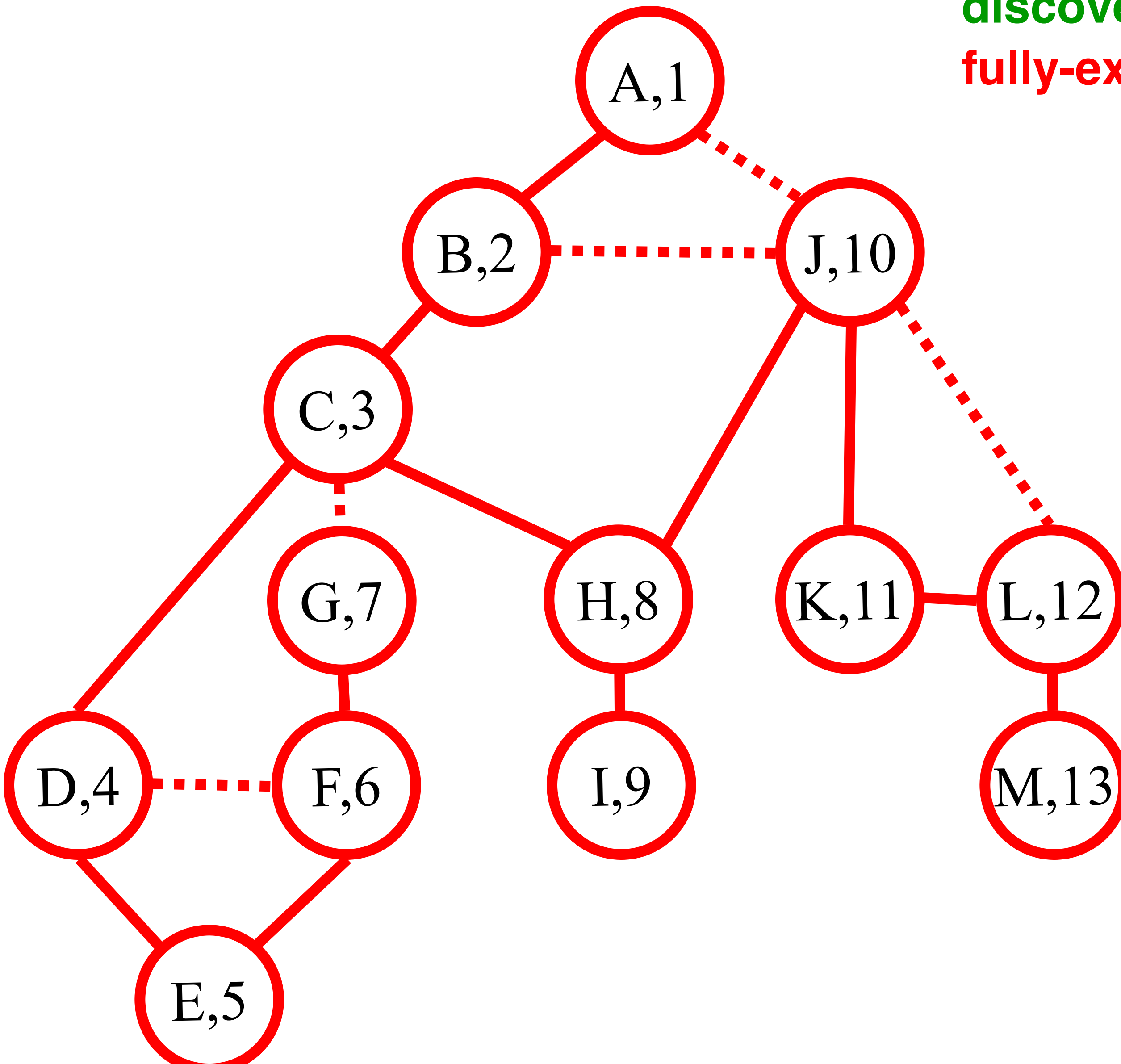
A (~~B~~,~~J~~)

st[] =
{1}

DFS(A)

Color code:

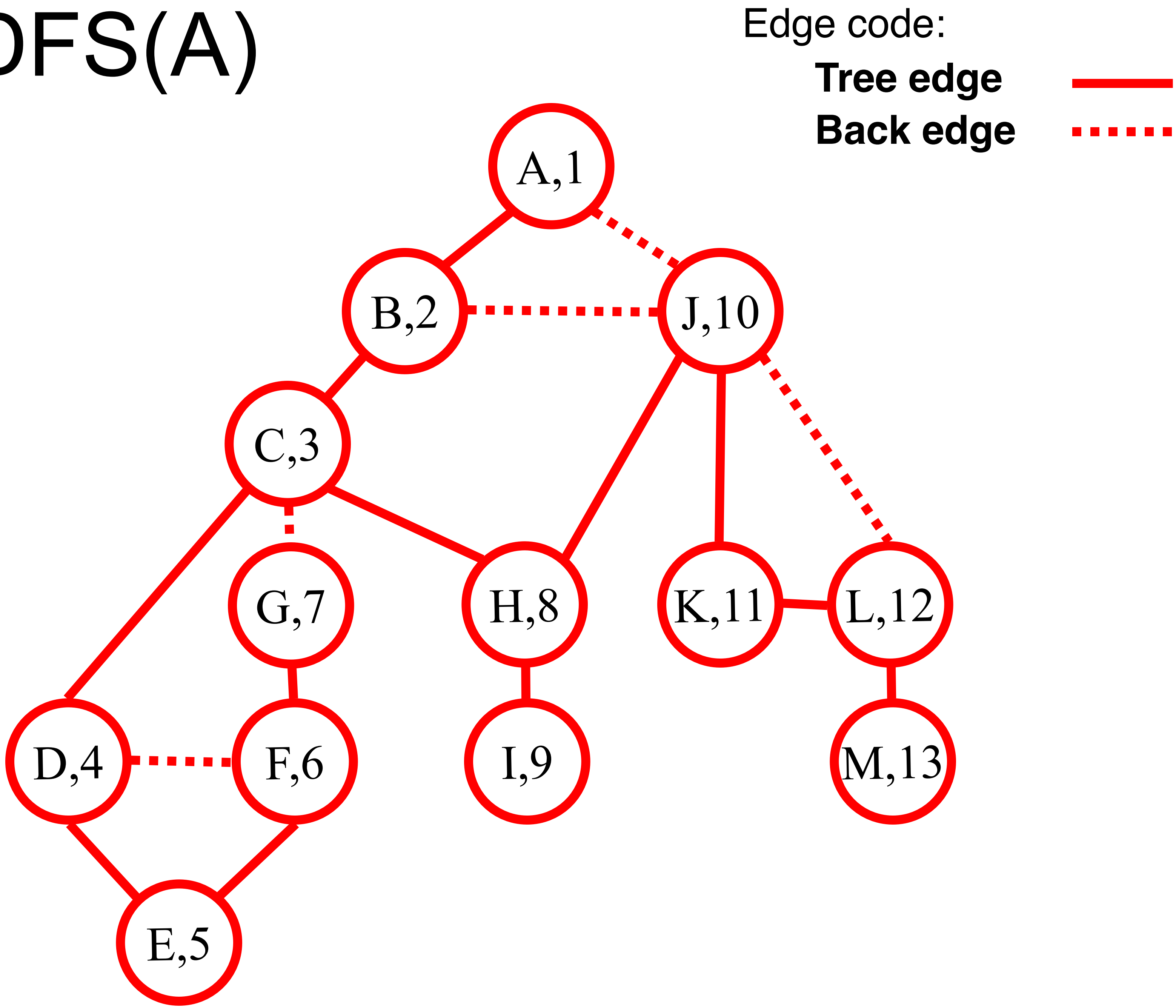
undiscovered
discovered
fully-explored



Call Stack: (Edge list)
TA-DA!!

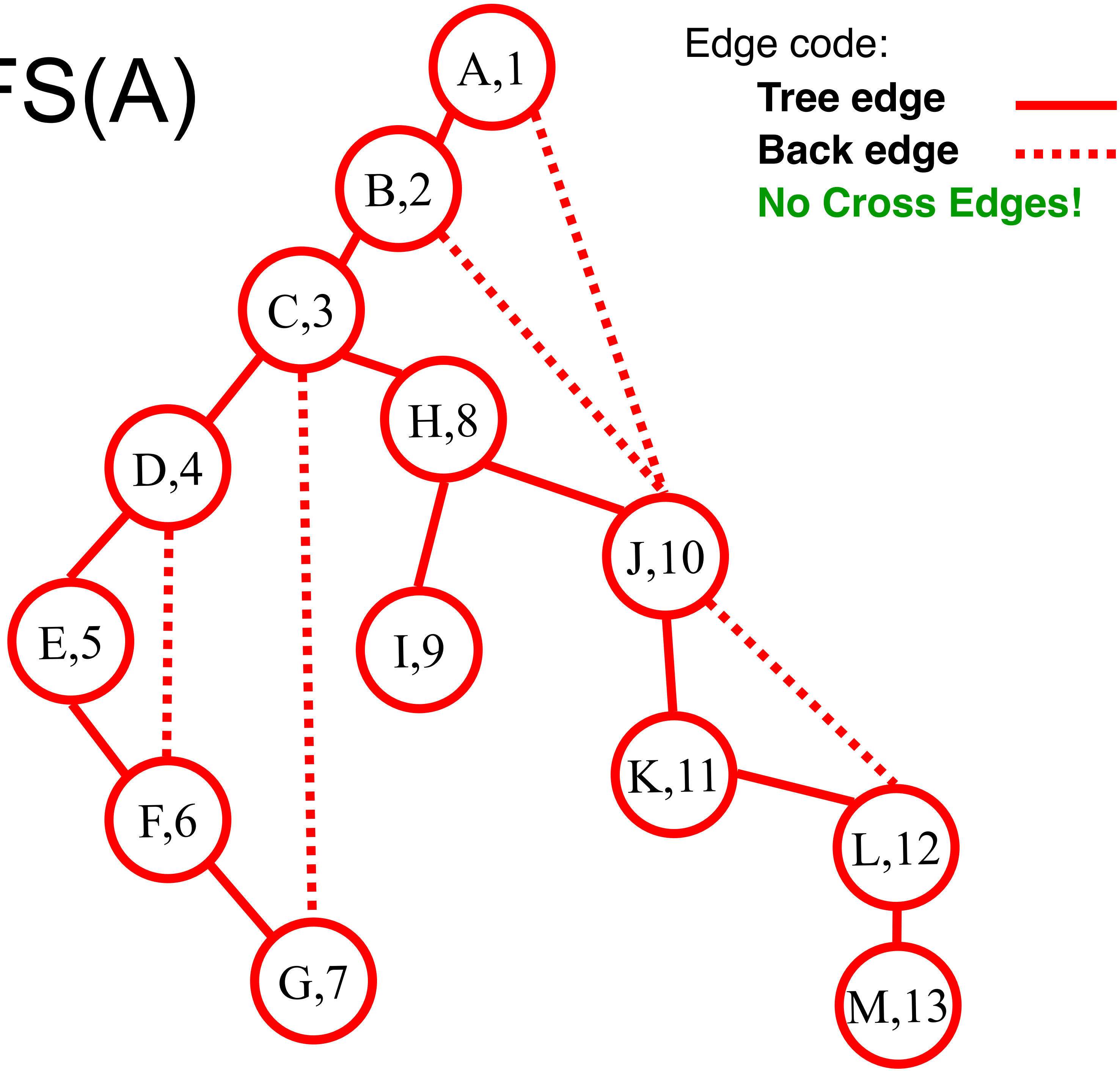
st[] = {}

DFS(A)



Edge code:
Tree edge
Back edge
A,1
B,2
J,10
C,3
G,7
H,8
K,11
L,12
D,4
F,6
I,9
M,13
E,5

DFS(A)


A,1
B,2
C,3
D,4
E,5
F,6
G,7
H,8
I,9
J,10
K,11
L,12
M,13
Edge code:
Tree edge
Back edge
No Cross Edges!