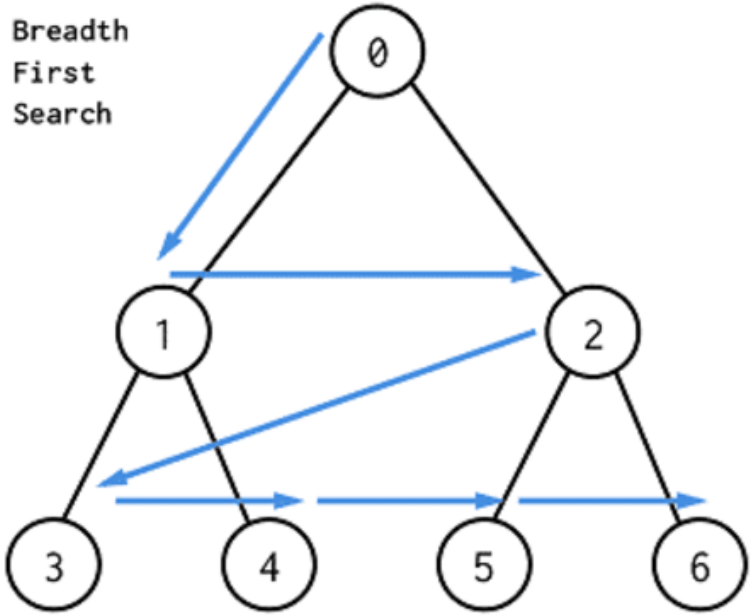
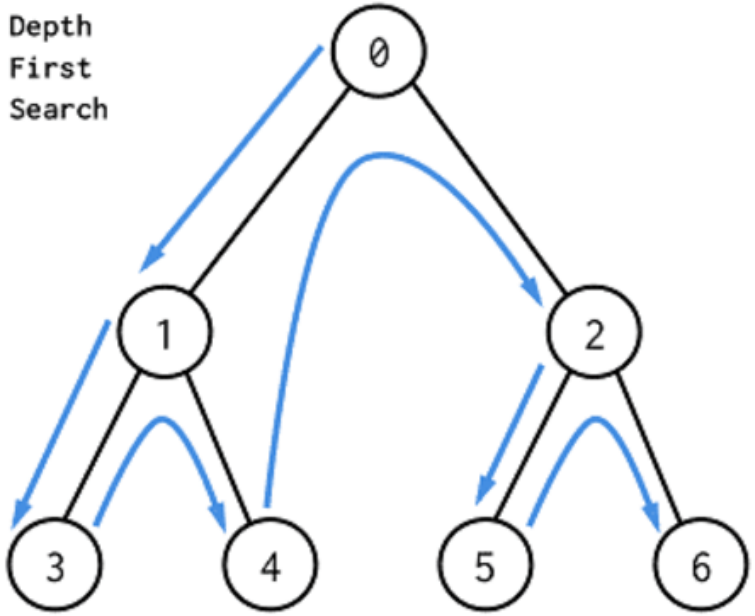
Depth-First Search

Overview

› Depth-first and breadth-first (next section) searches are two classic and diametrically opposed ways to systematically process the vertices of a graph





Depth-first strategy

- Let V be an initially empty set of visited vertices
- Process and place the starting vertex s in V
- The next (unvisited) vertex to process and place in V is the furthest (deepest) vertex from s that is adjacent to a vertex along the current path

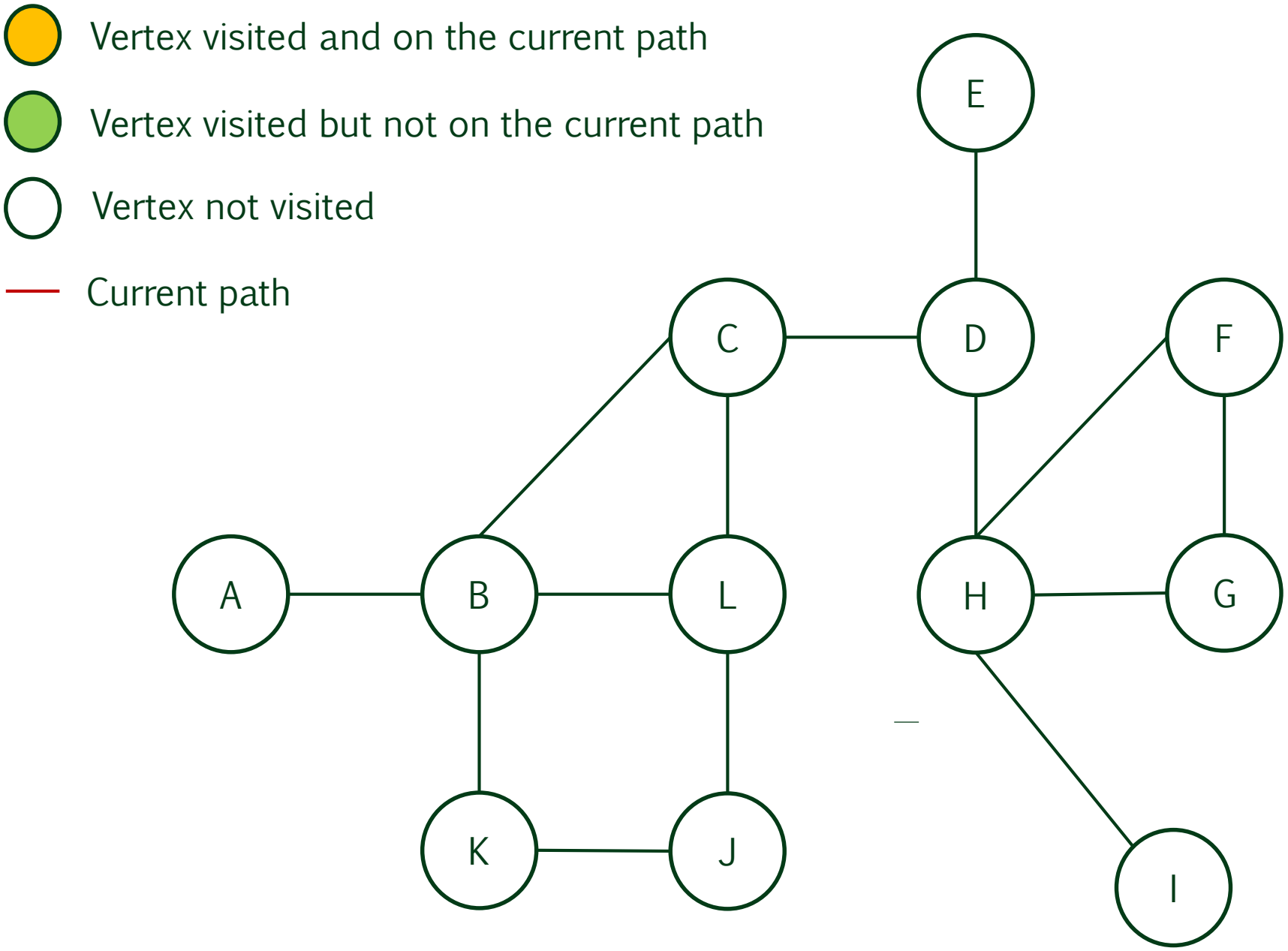
- Underlying data structure: Stack (recursion)

Alternate descriptions

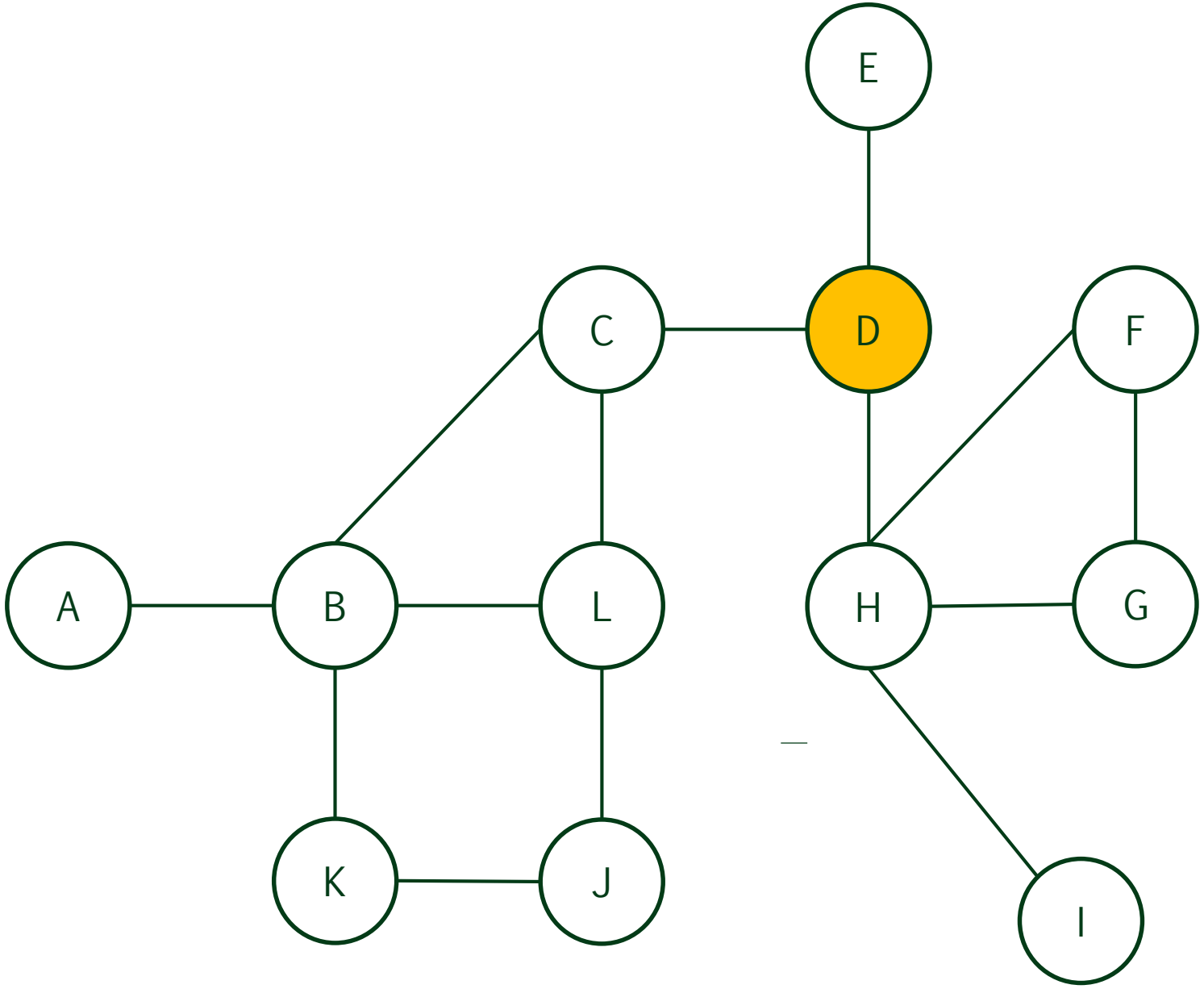
- “Explore the graph one path at time.”
- “Explore the graph along a path from the starting vertex for as long as possible before backing up to explore another path.”

Basic algorithm

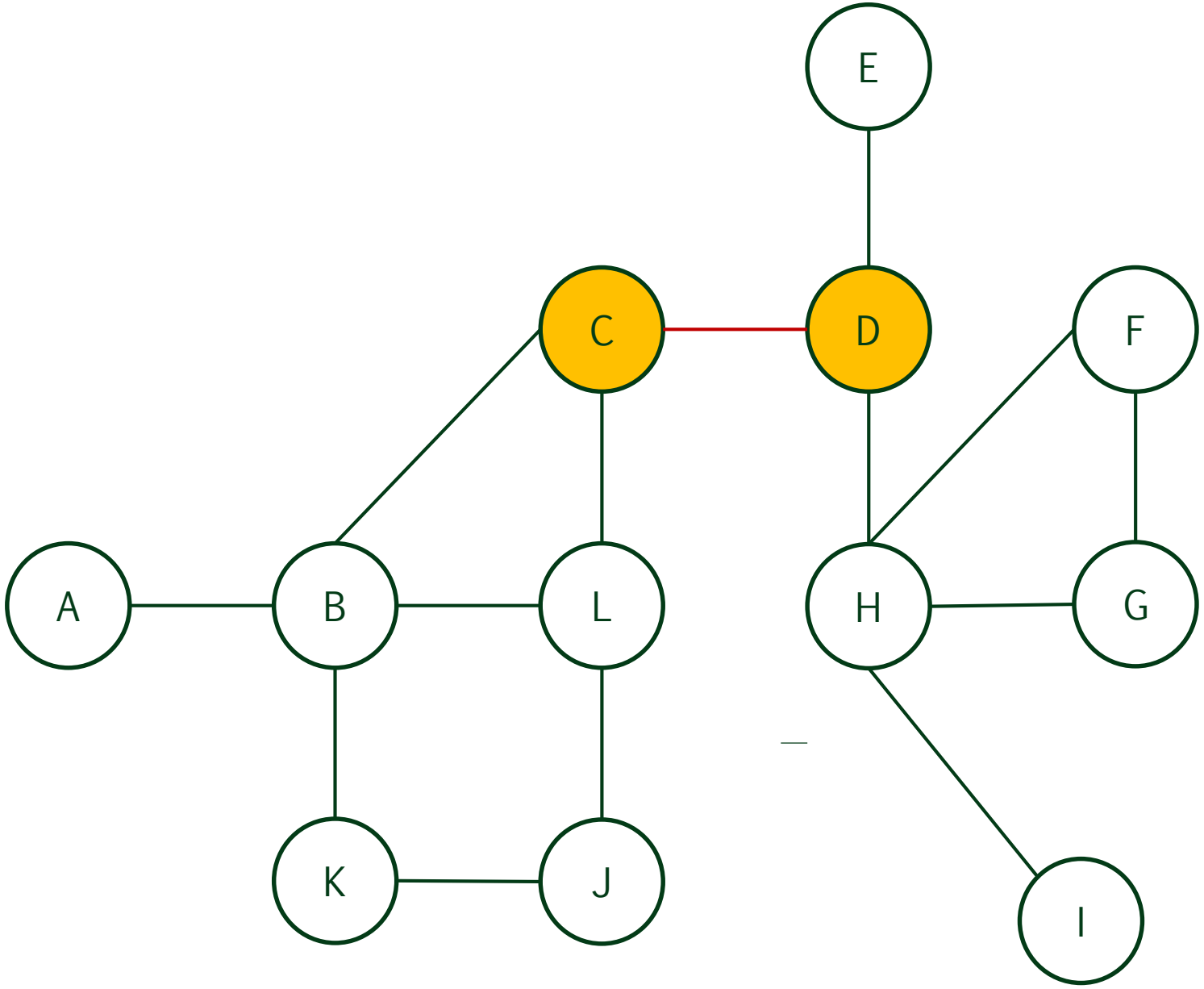
```
Depthfirstsearch (vertex v)
    mark v as visited
    process v
    for each unvisited adjacent vertex w to v do
        Depthfirstsearch(w)
```


Vertex visited and on the current path
Vertex visited but not on the current path
Vertex not visited
Current path
E
C
D
F
A
B
L
H
G
K
J
I

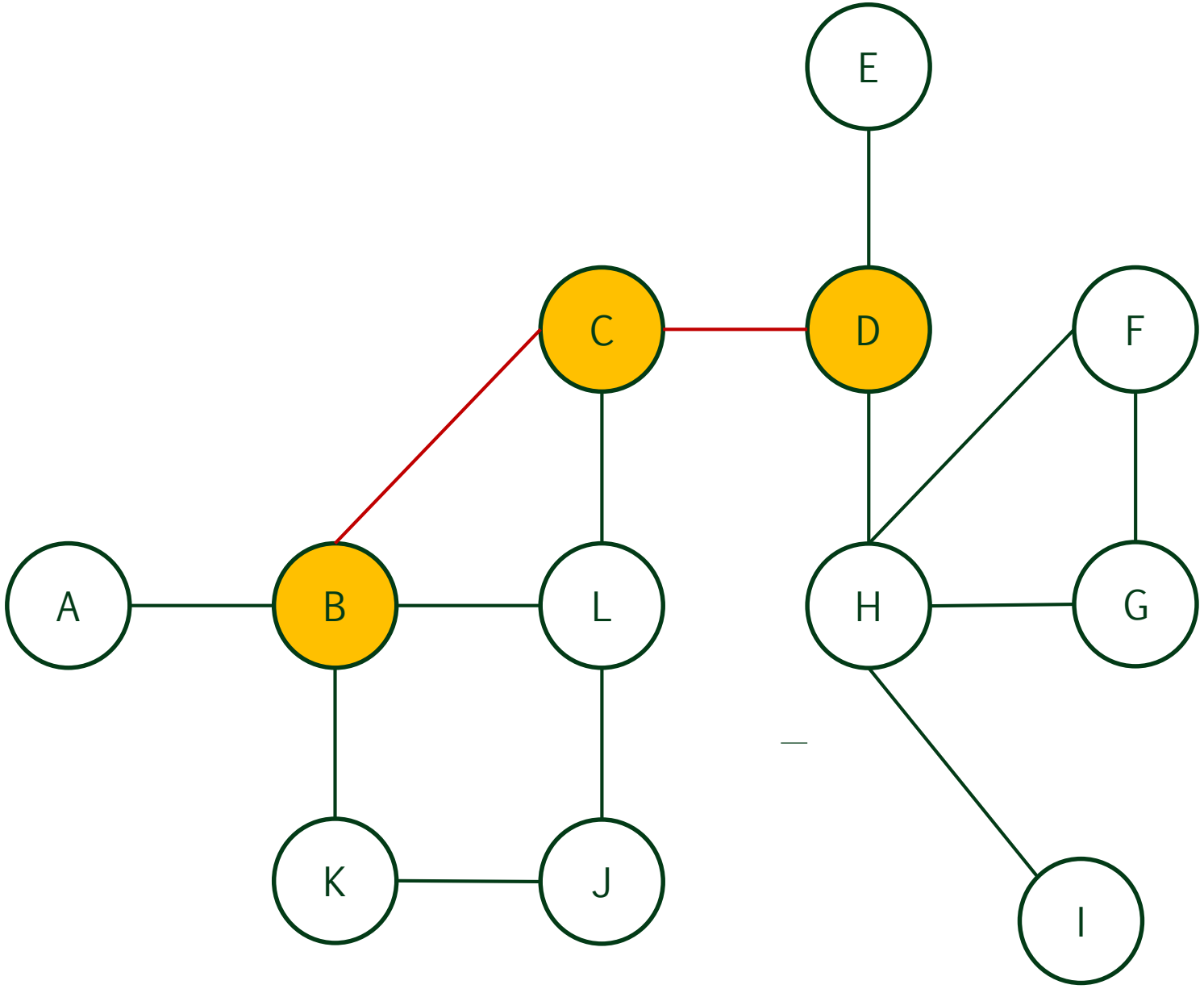
D



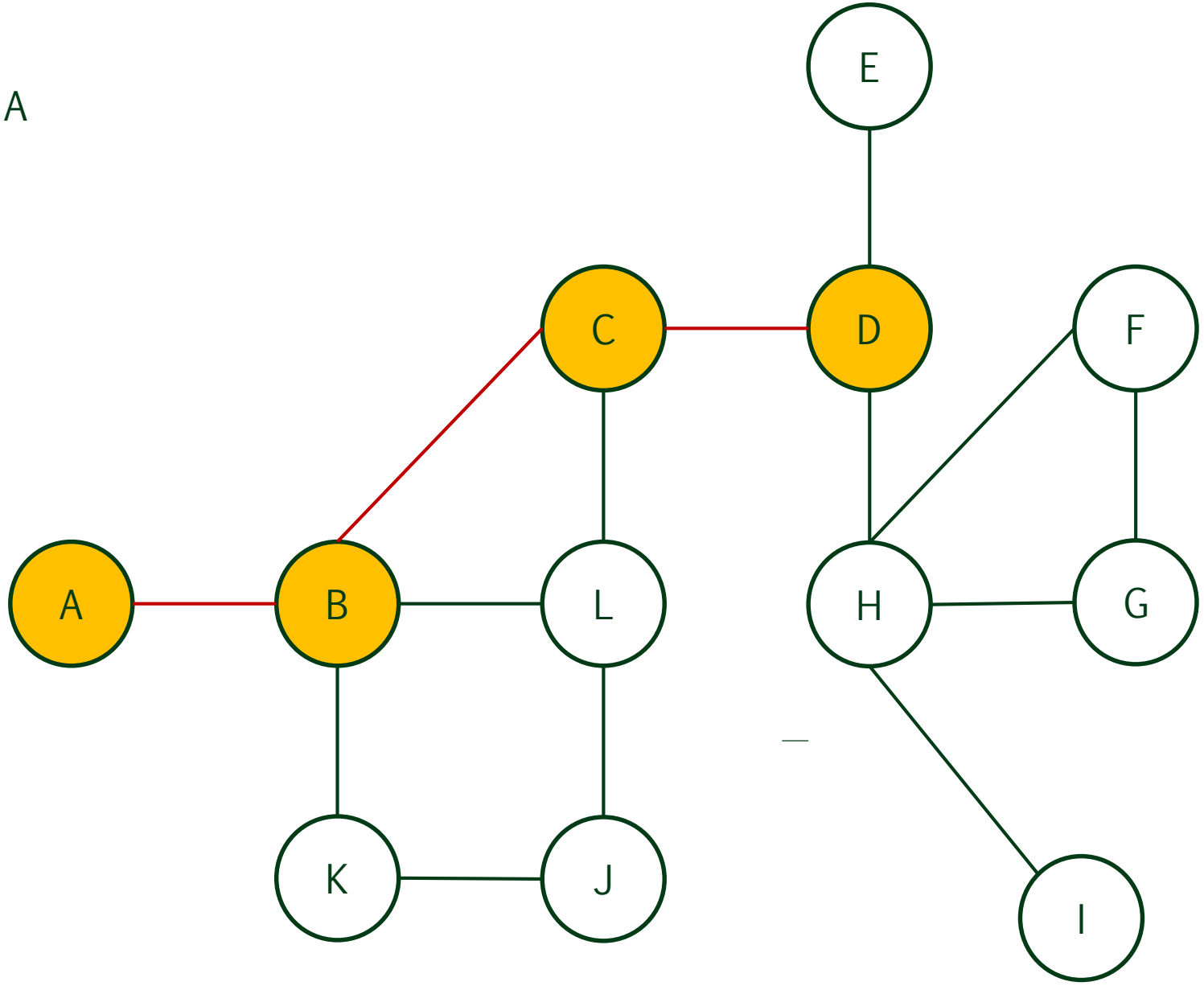
D C



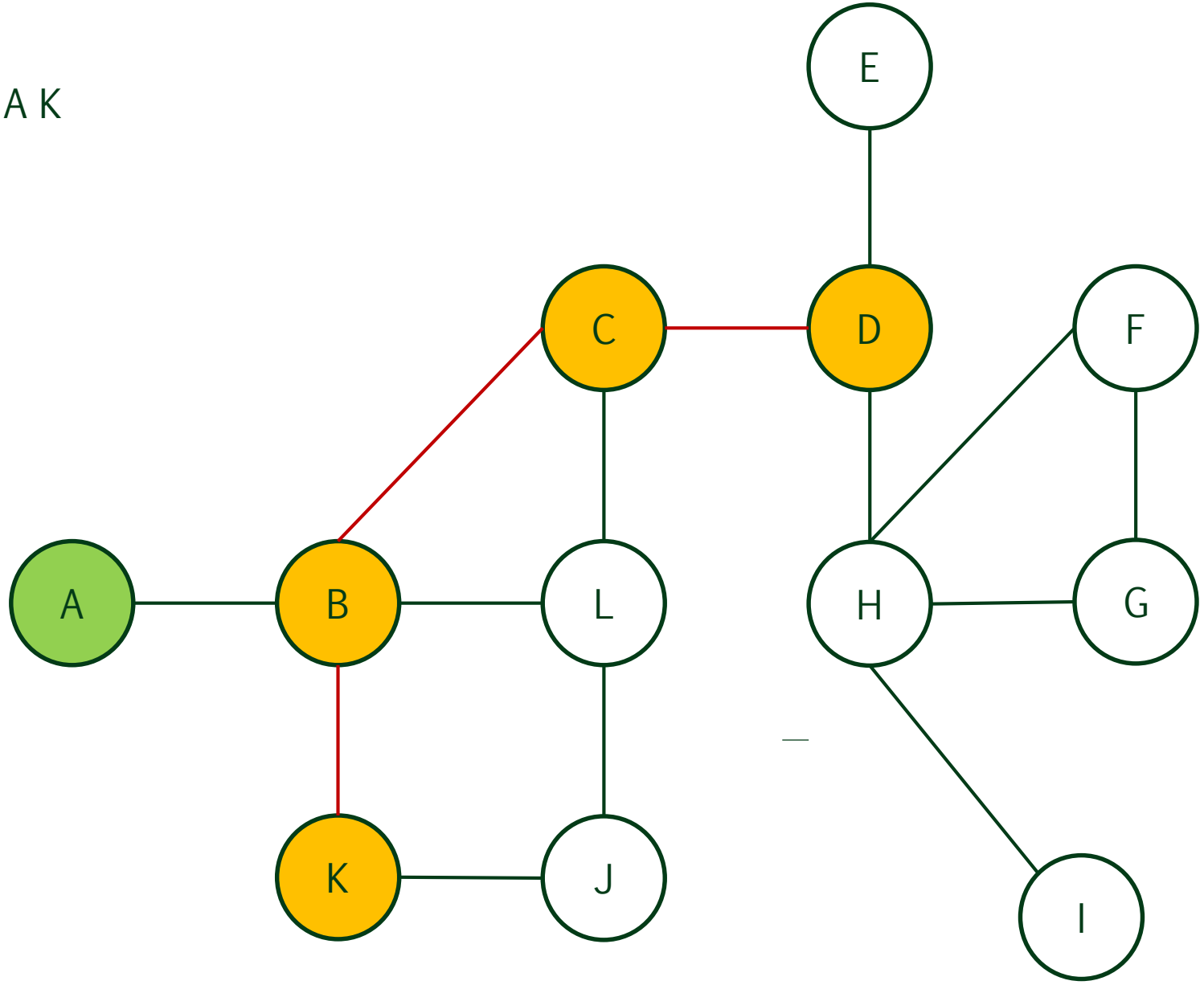
D C B



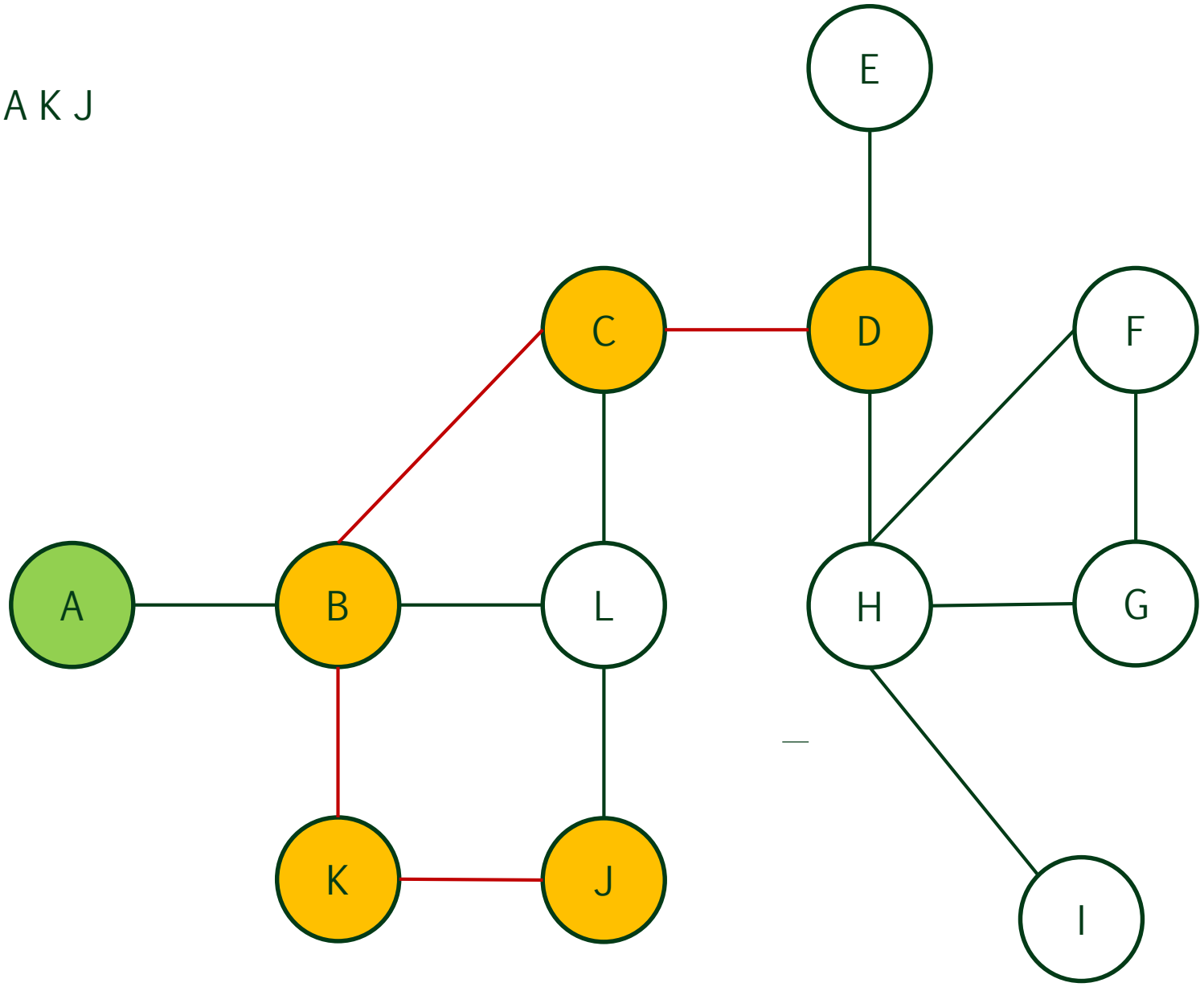
D C B A



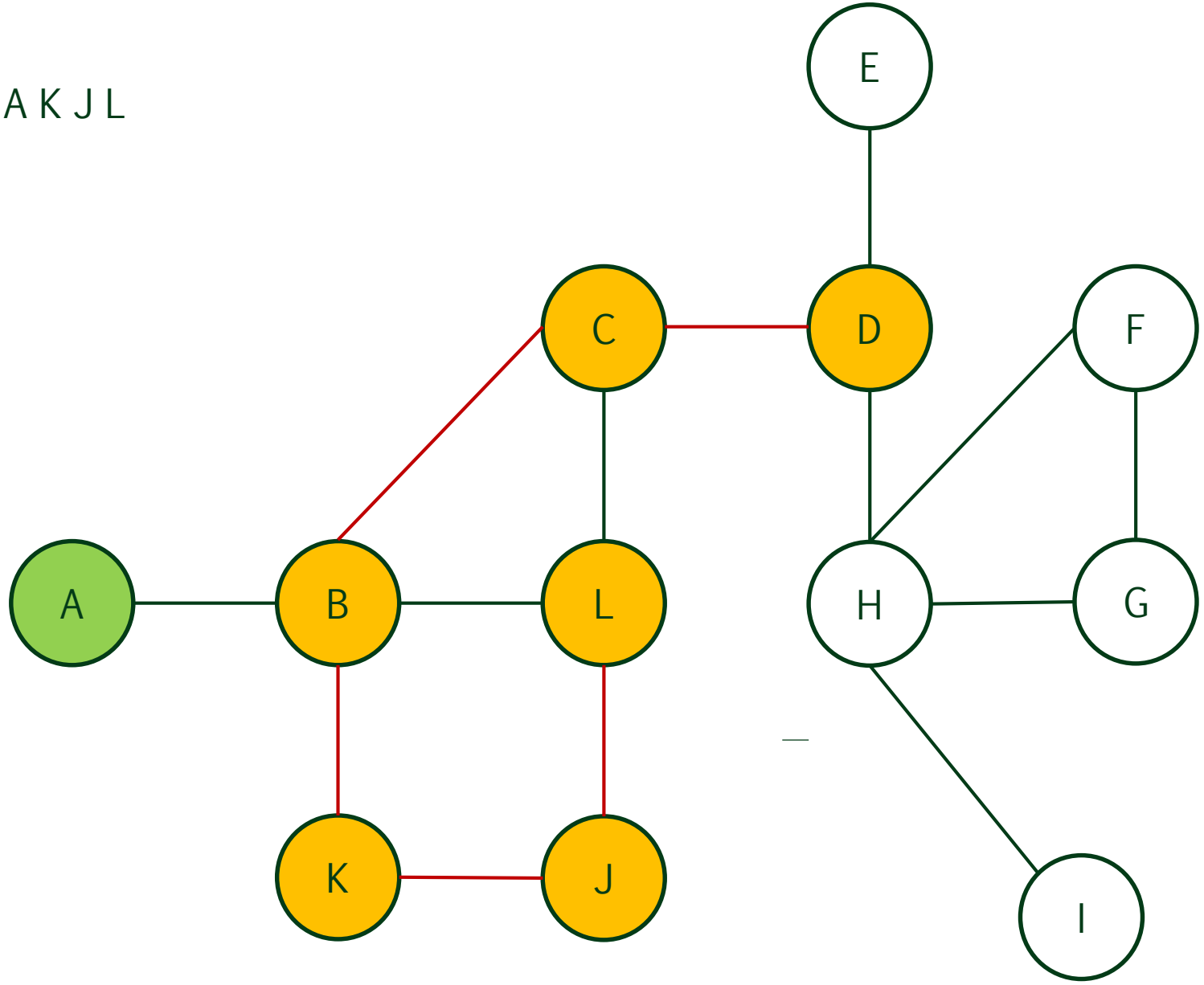
D C B A K



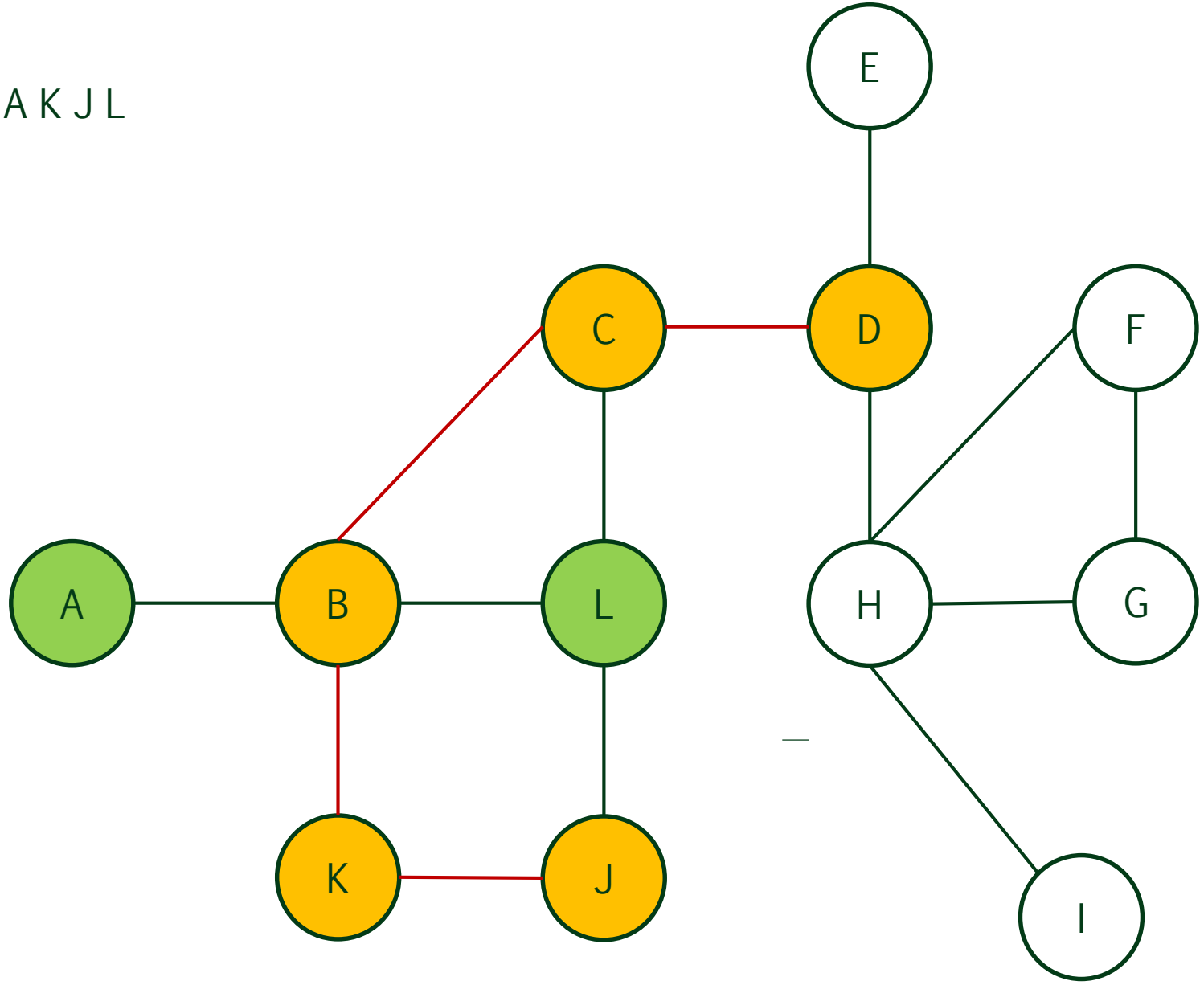
D C B A K J



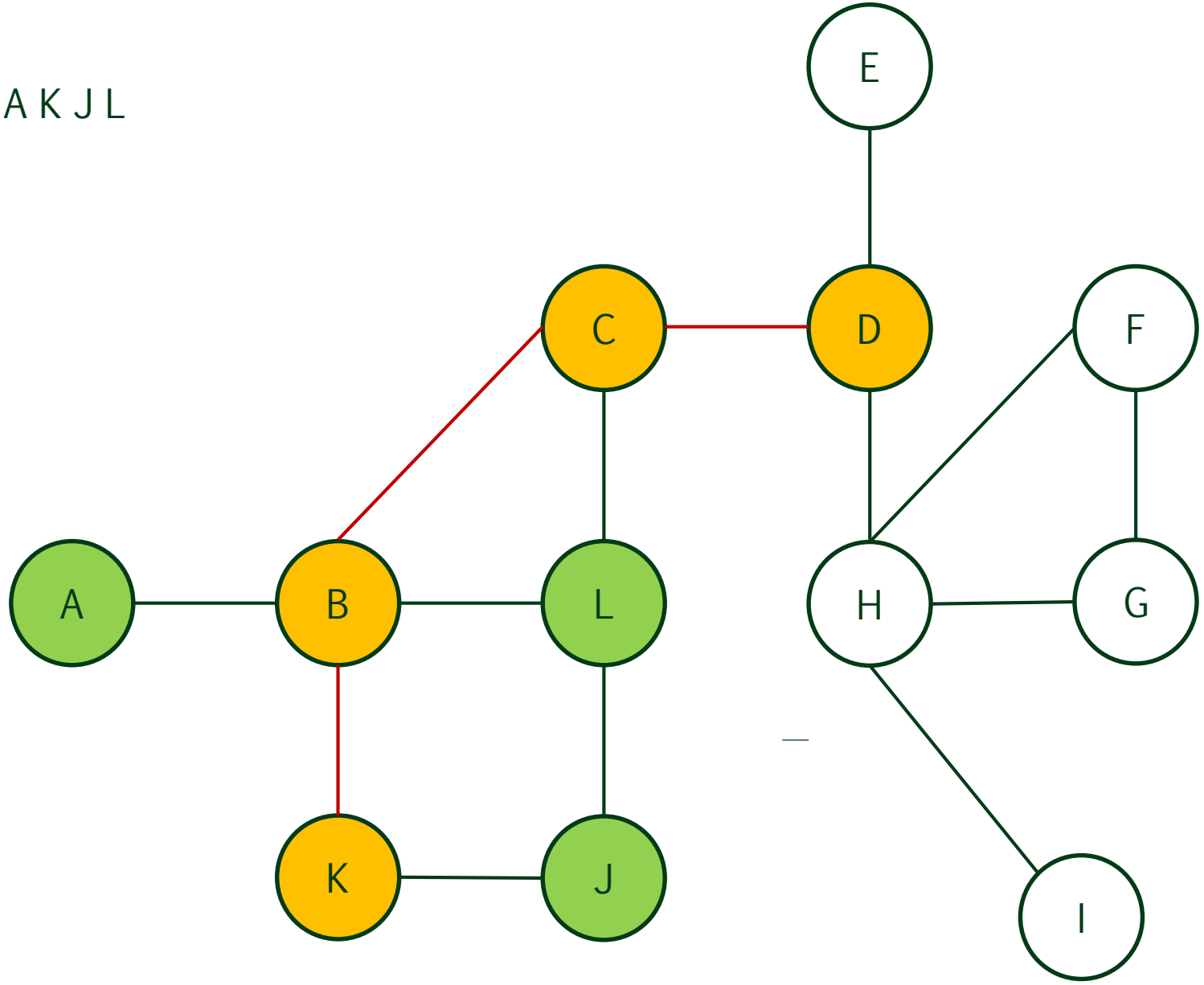
D C B A K J L



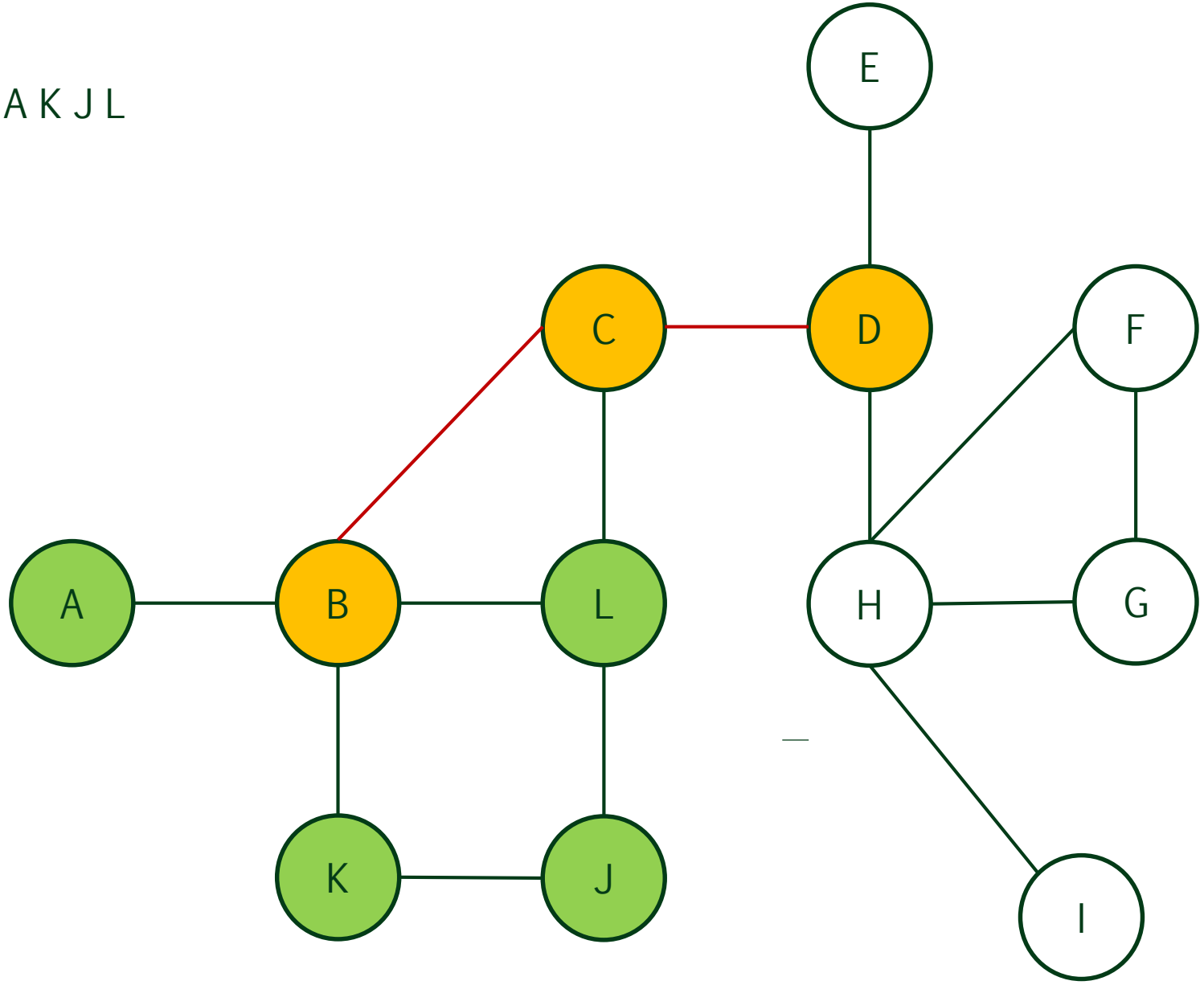
D C B A K J L



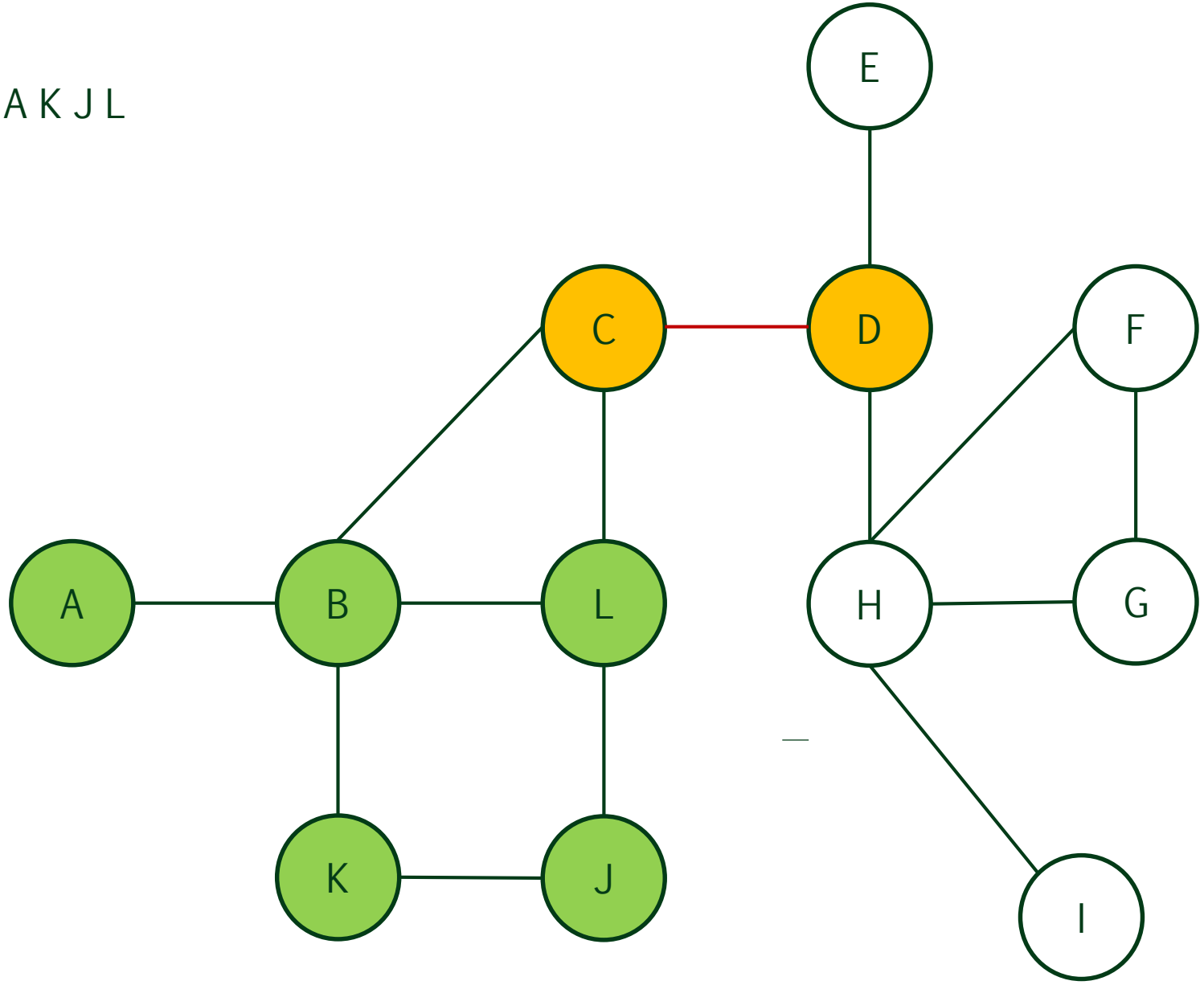
D C B A K J L



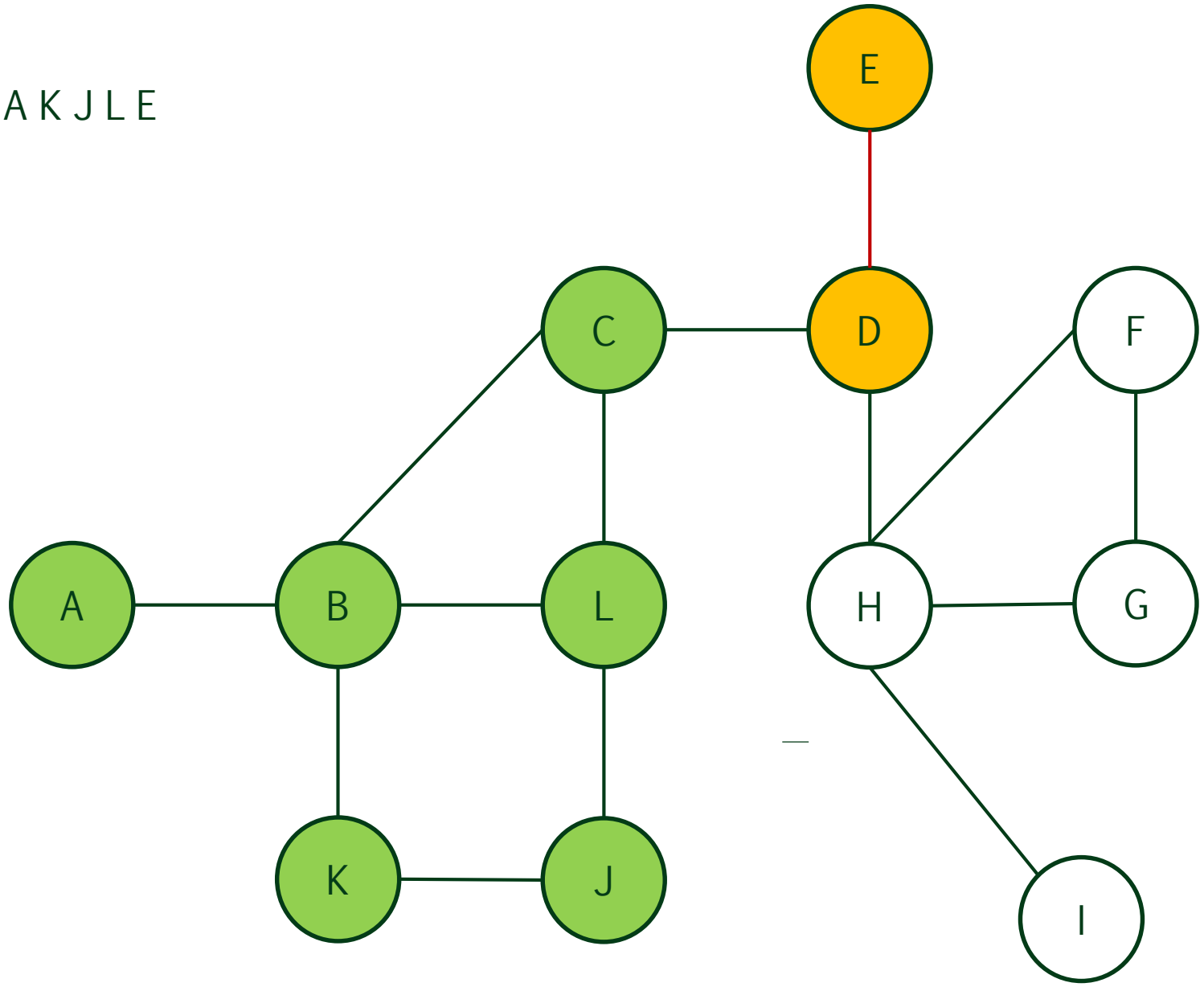
D C B A K J L



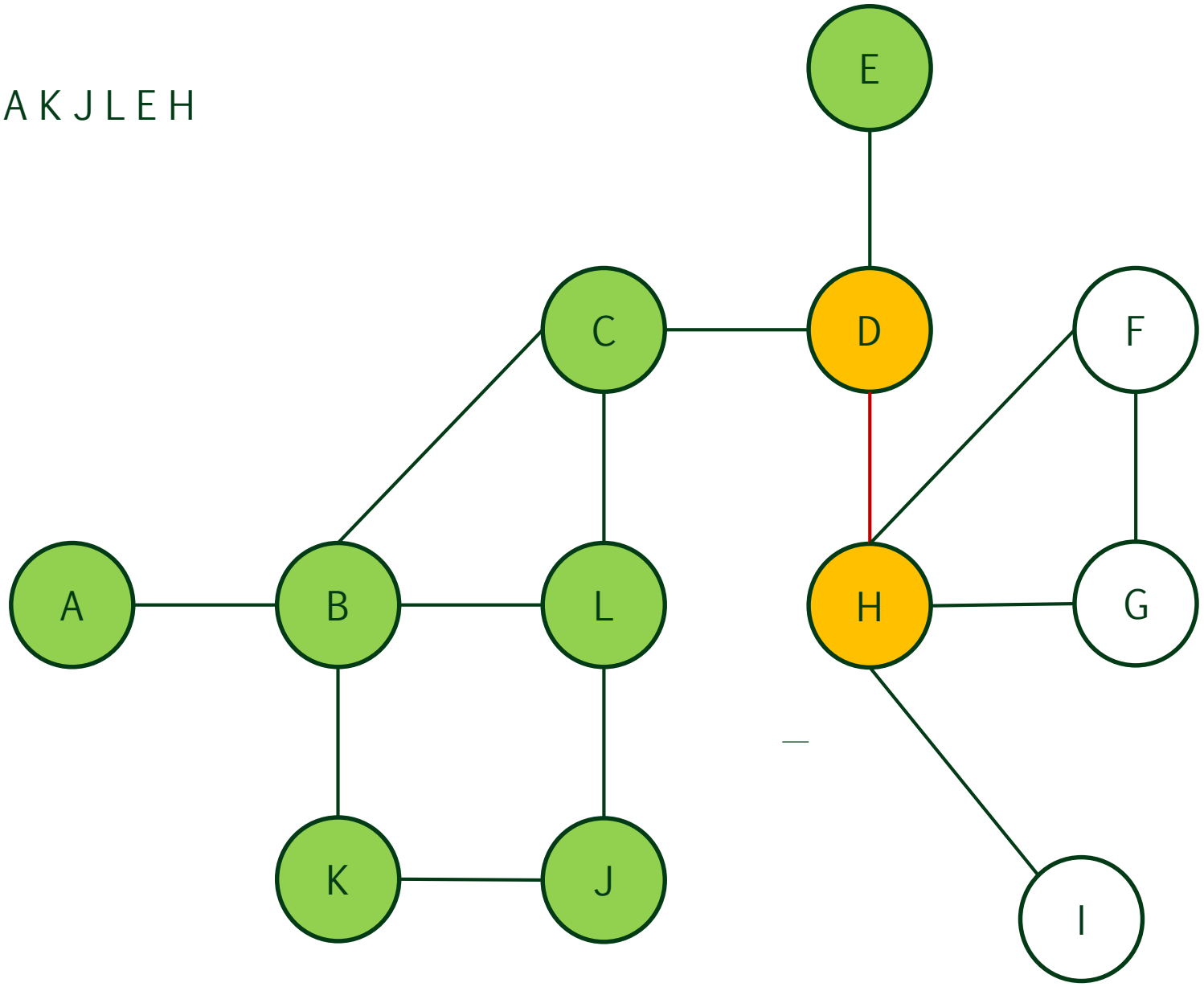
D C B A K J L



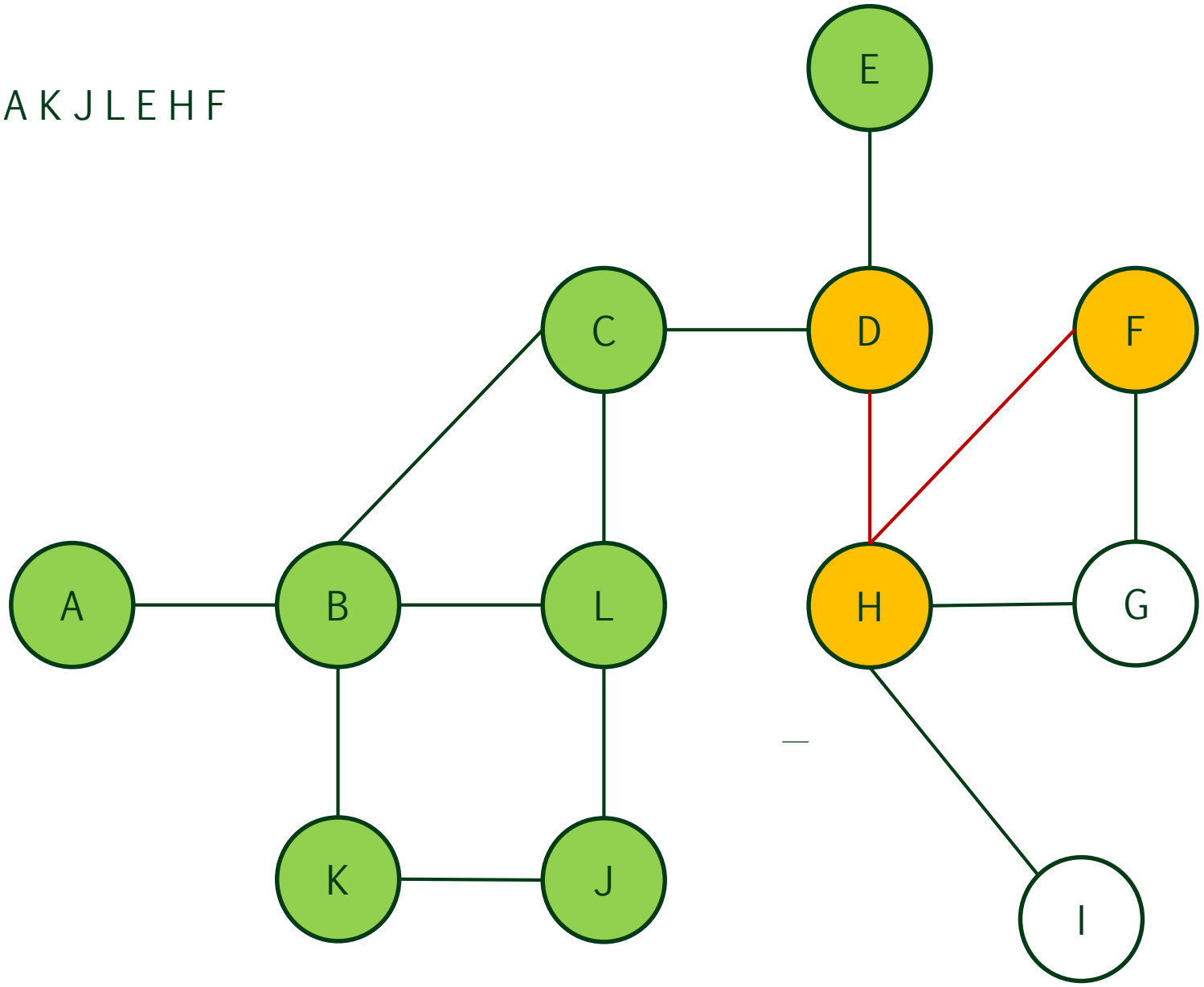
D C B A K J L E



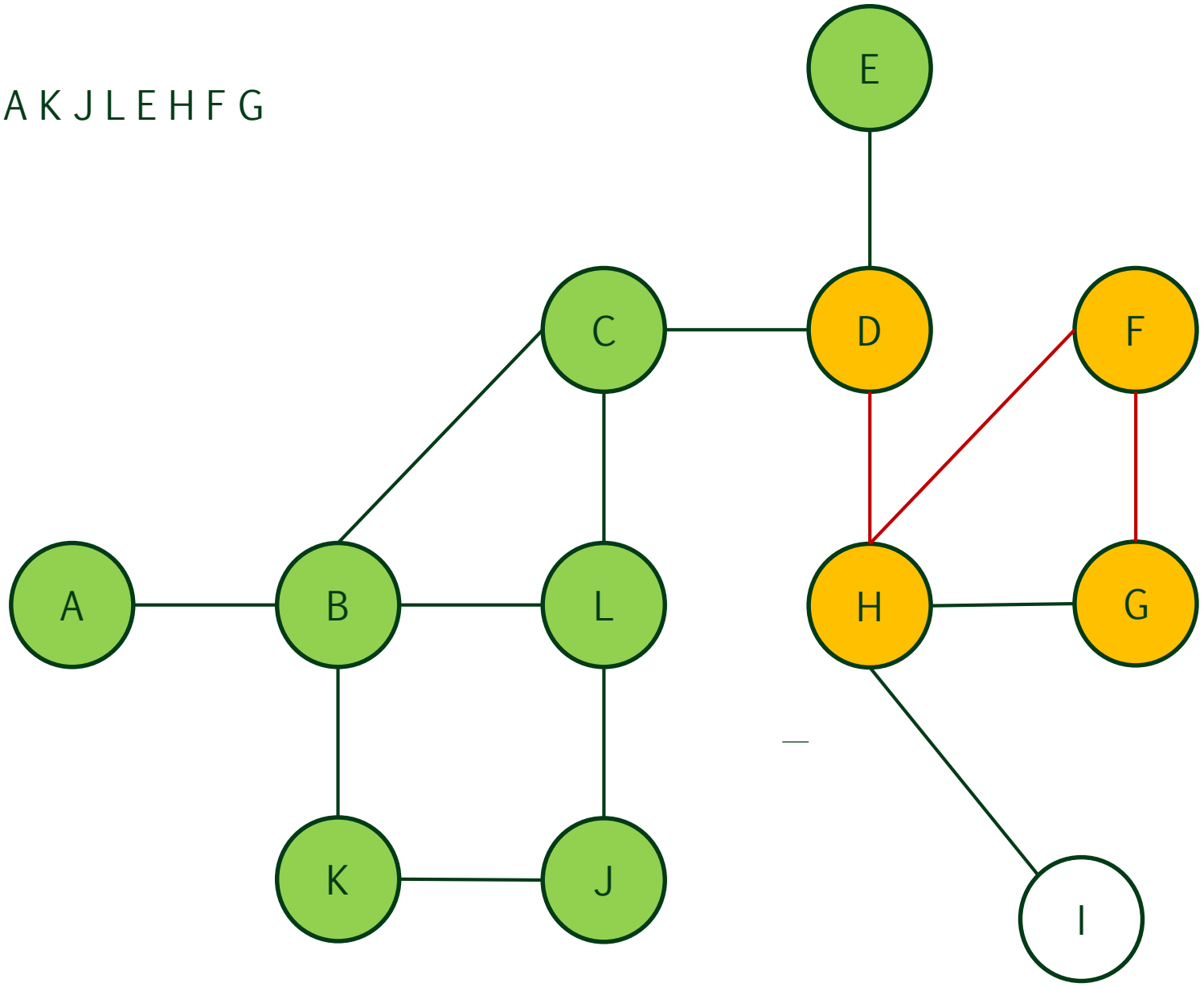
D C B A K J L E H



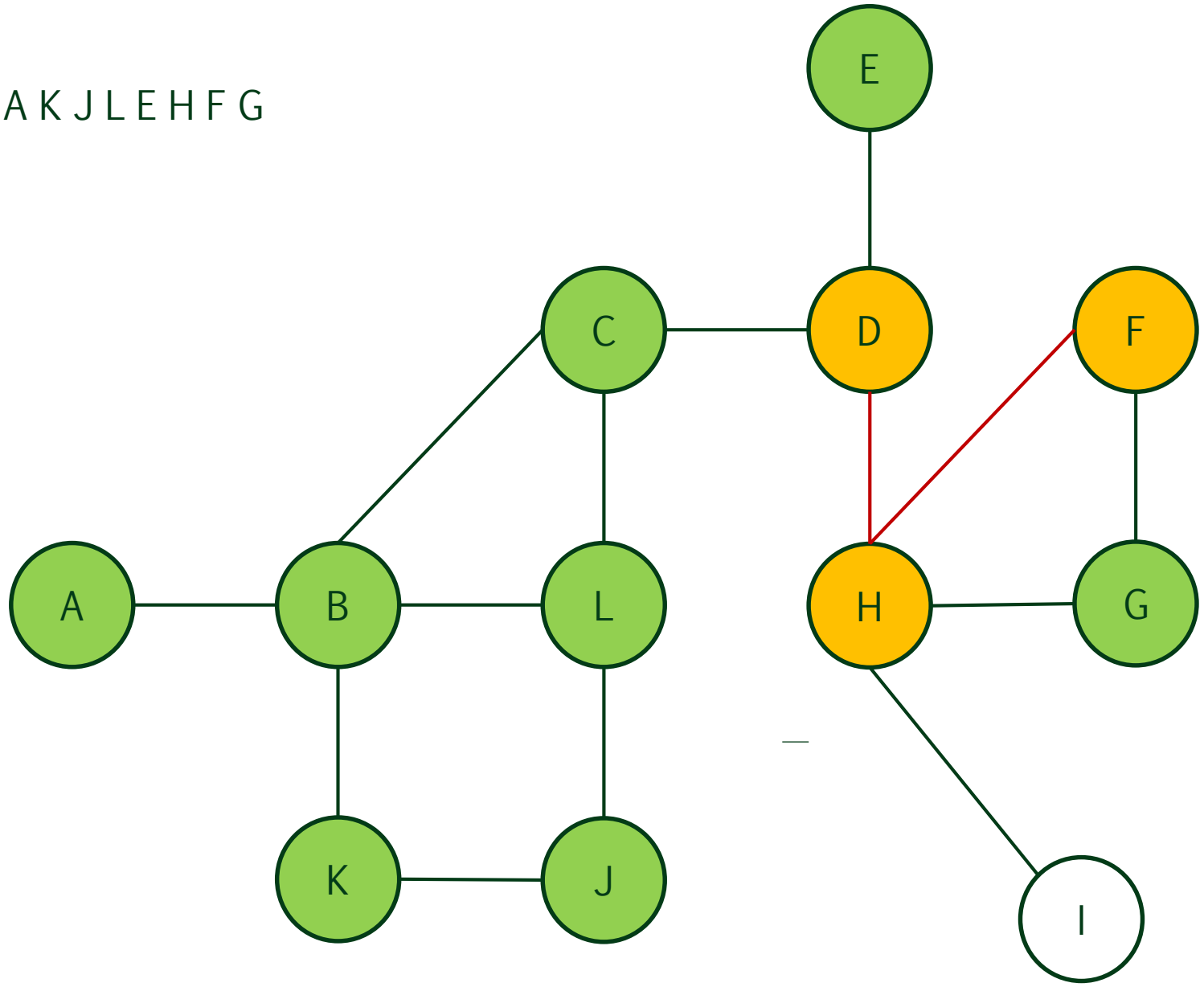
D C B A K J L E H F



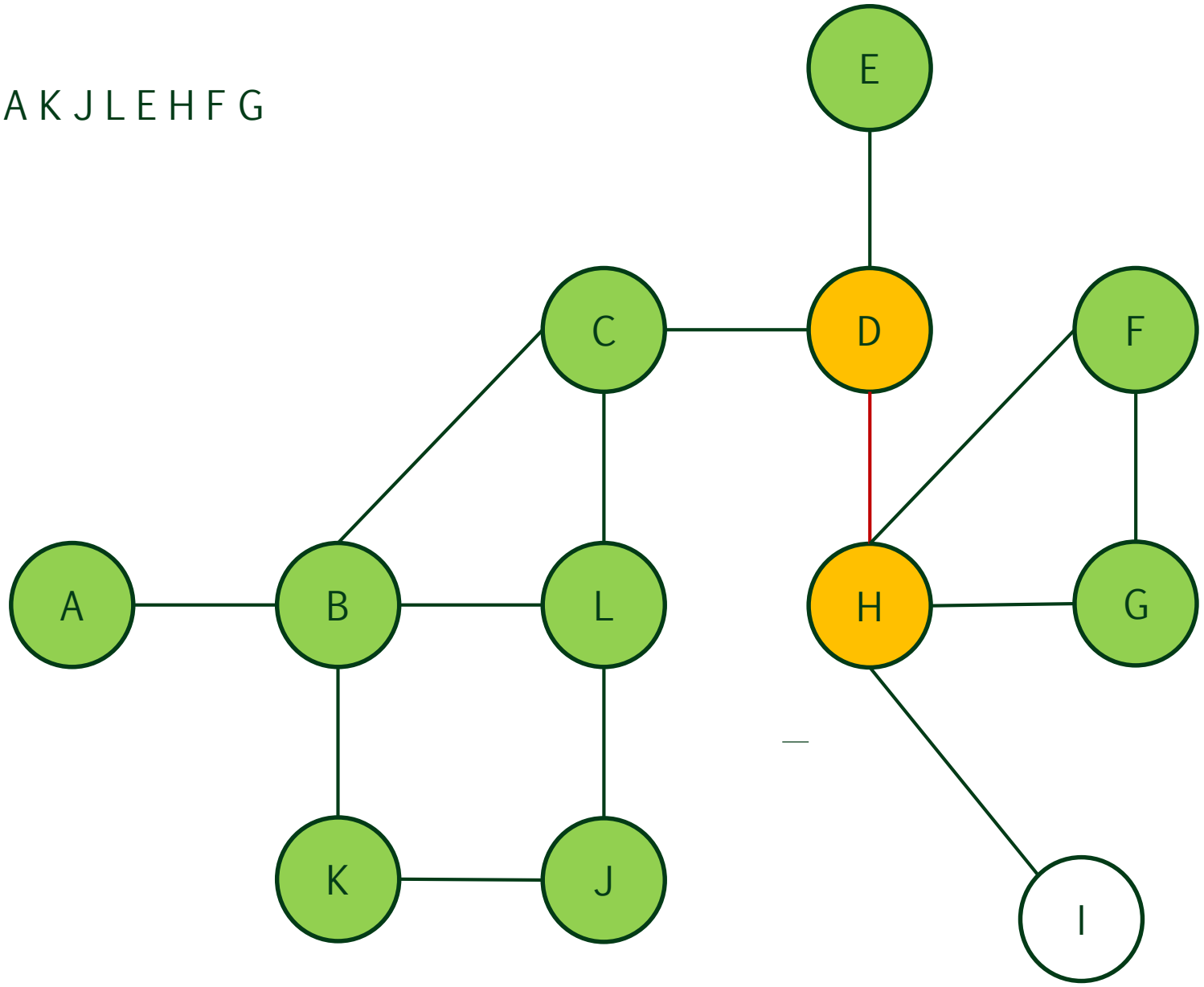
D C B A K J L E H F G



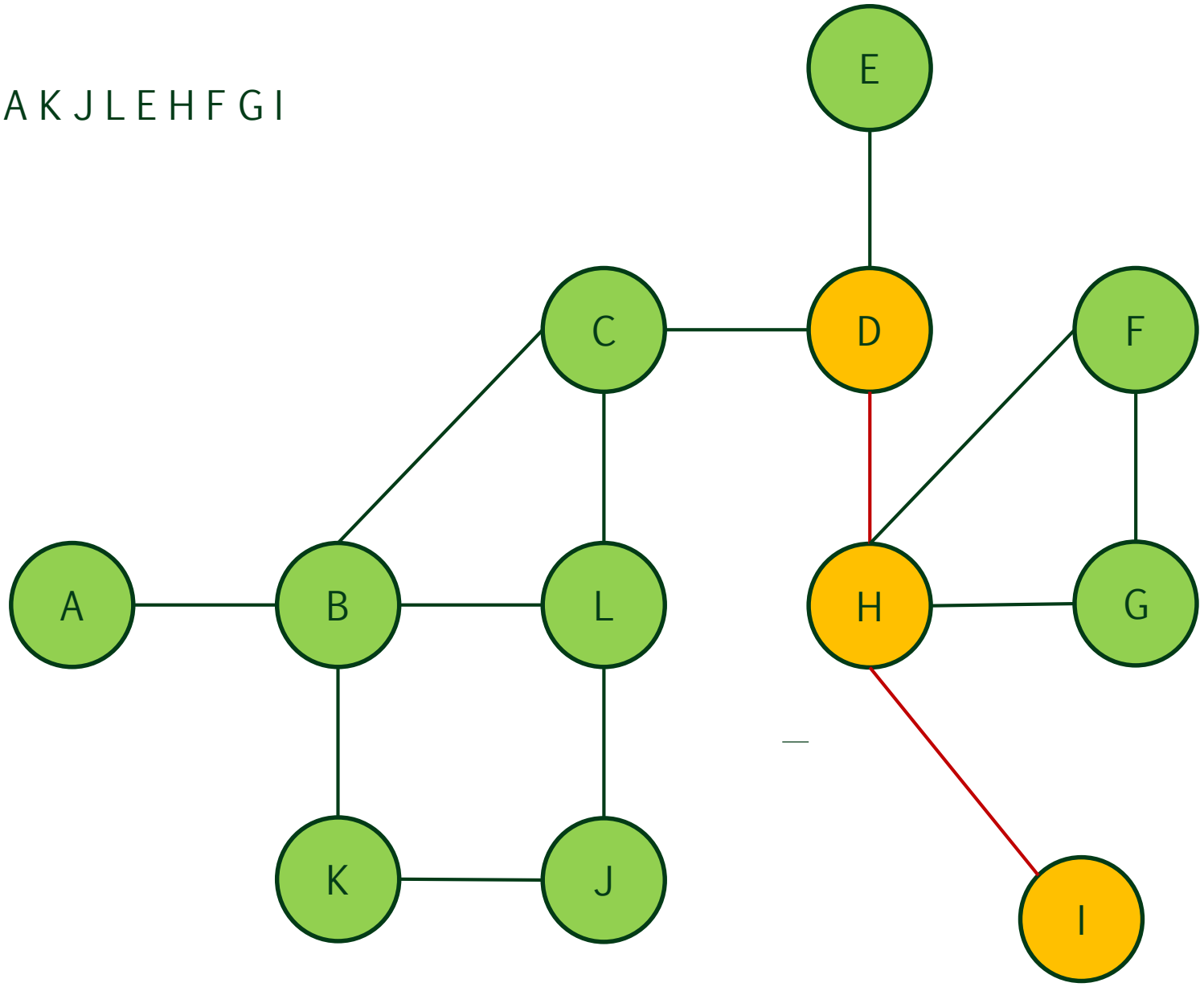
D C B A K J L E H F G



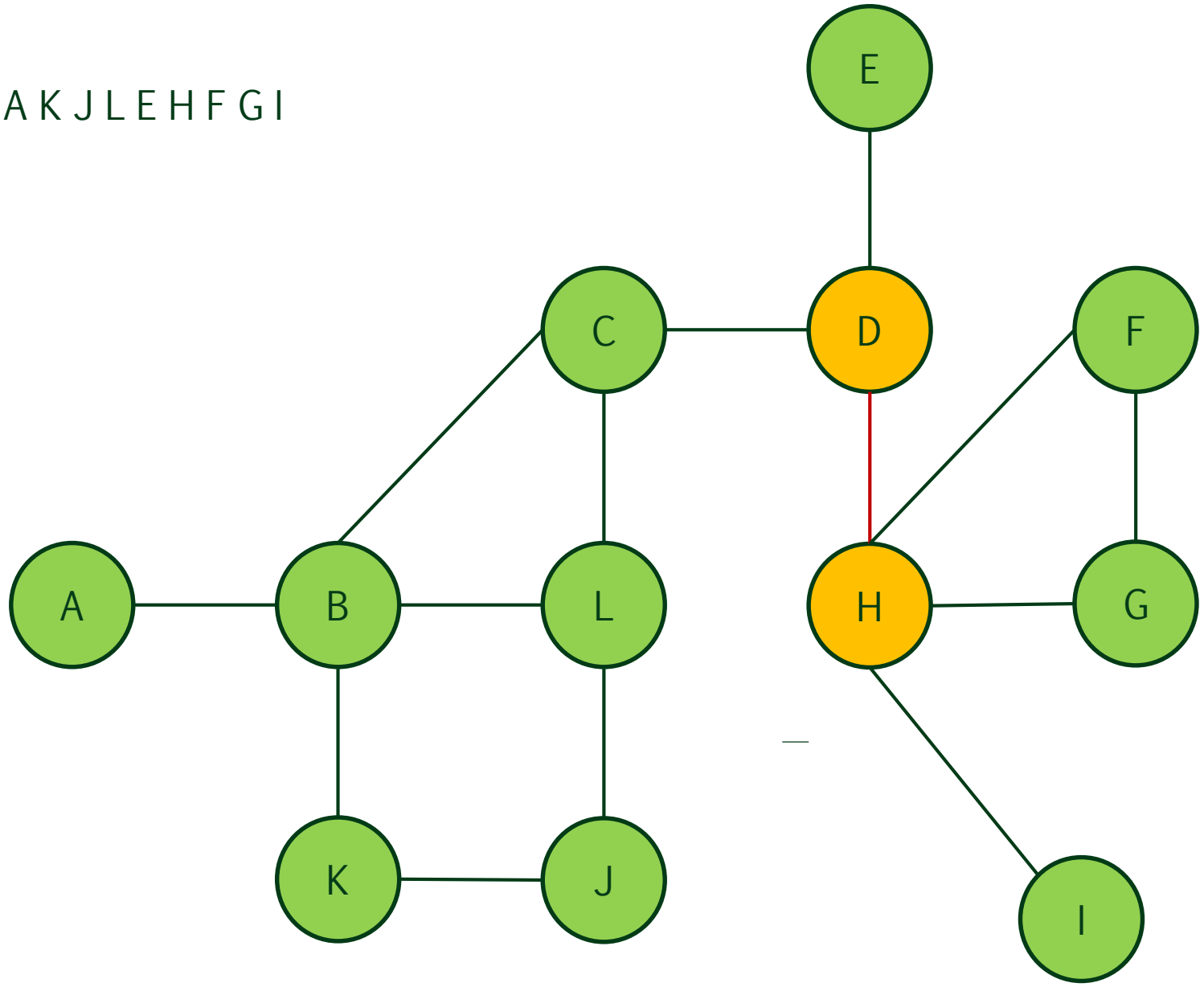
D C B A K J L E H F G



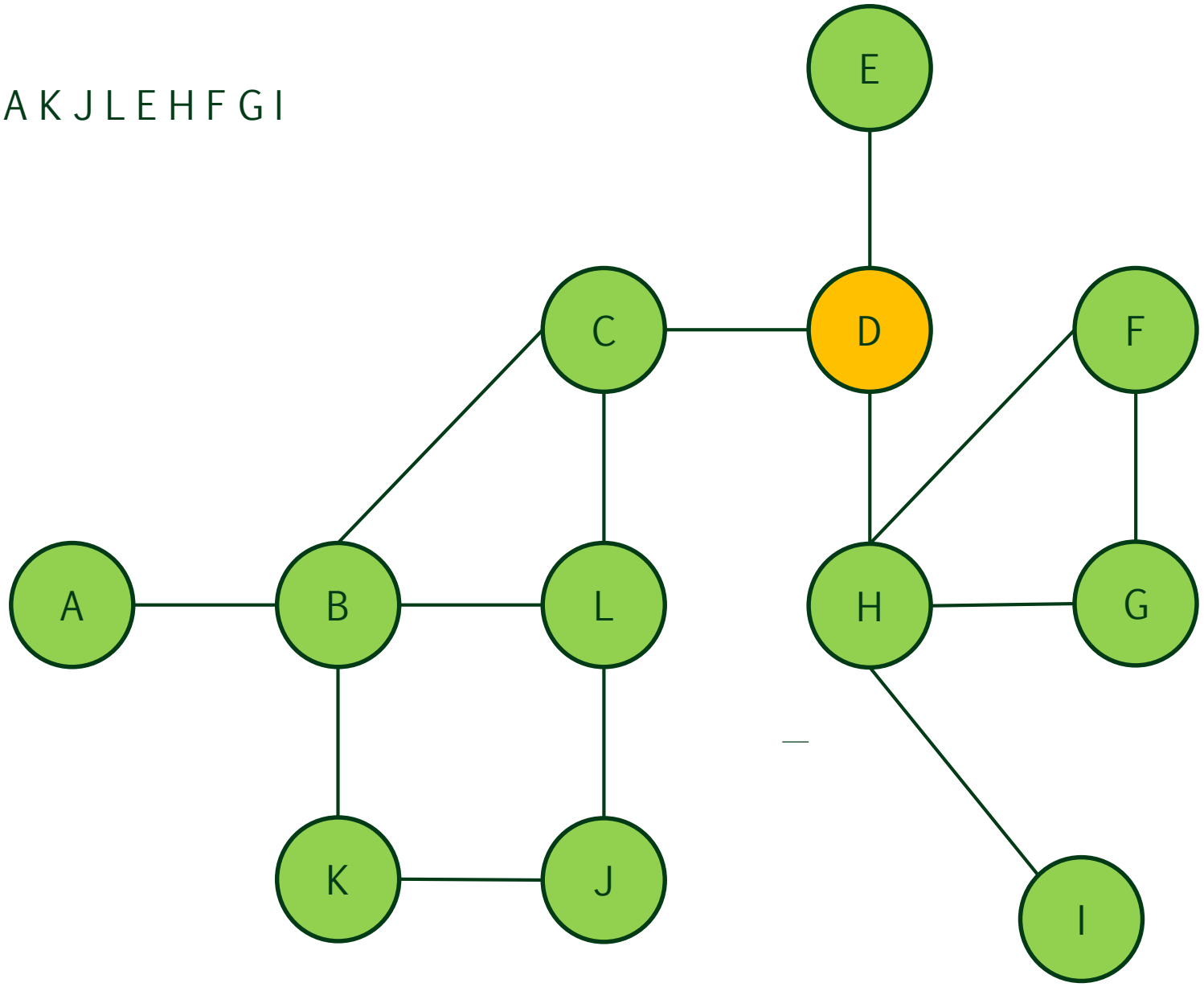
D C B A K J L E H F G I



D C B A K J L E H F G I

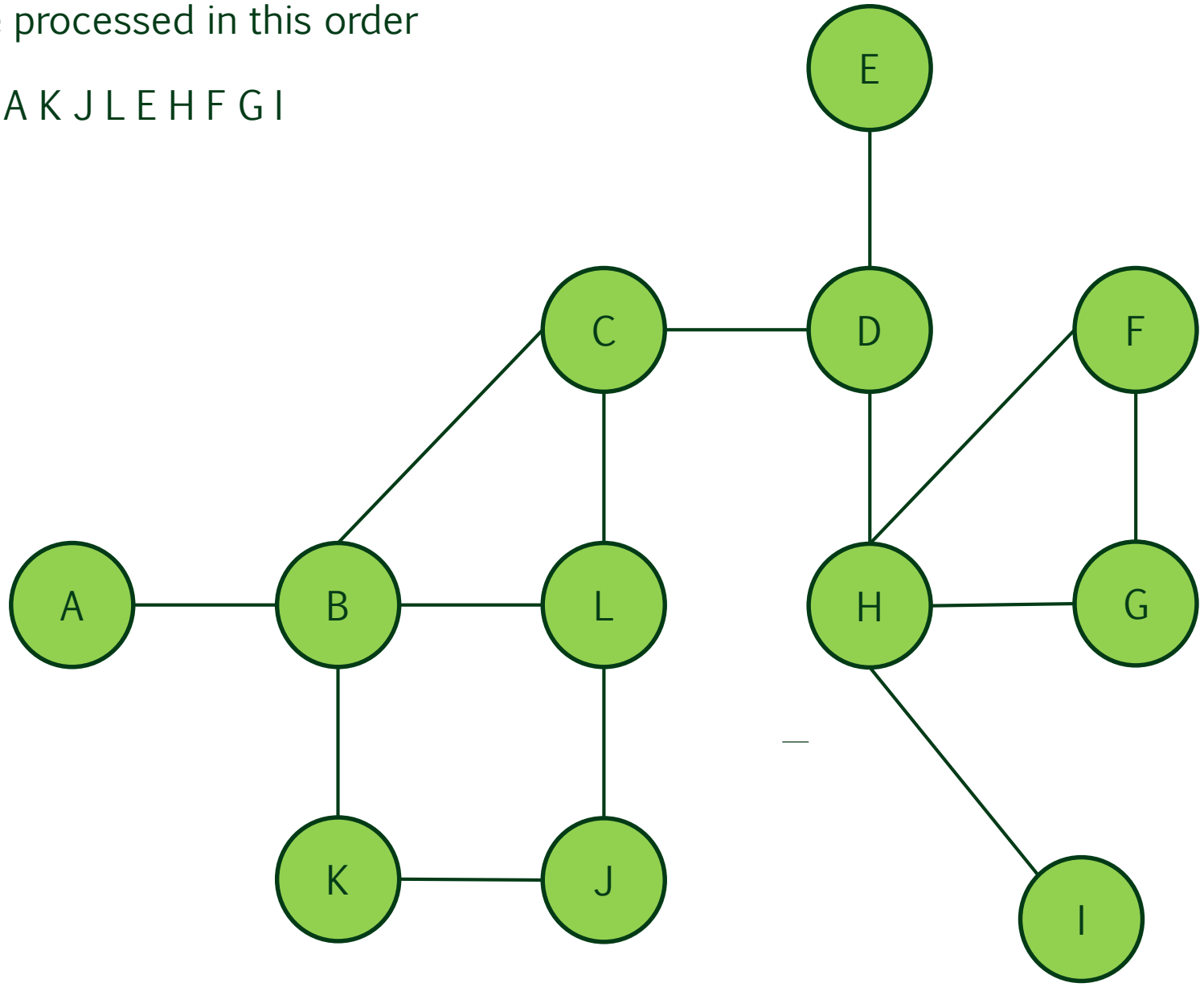


D C B A K J L E H F G I



Vertices are processed in this order

D C B A K J L E H F G I



Breadth-First Search

Breadth-first strategy

- Let V be an initially empty set of visited vertices
- Let Q be an initially empty queue of vertices
- Place the starting vertex s in both V and Q
- The next vertex v to process is retrieved and removed from the front of Q. Once v is processed, each of its unvisited adjacent vertices is placed in V and Q

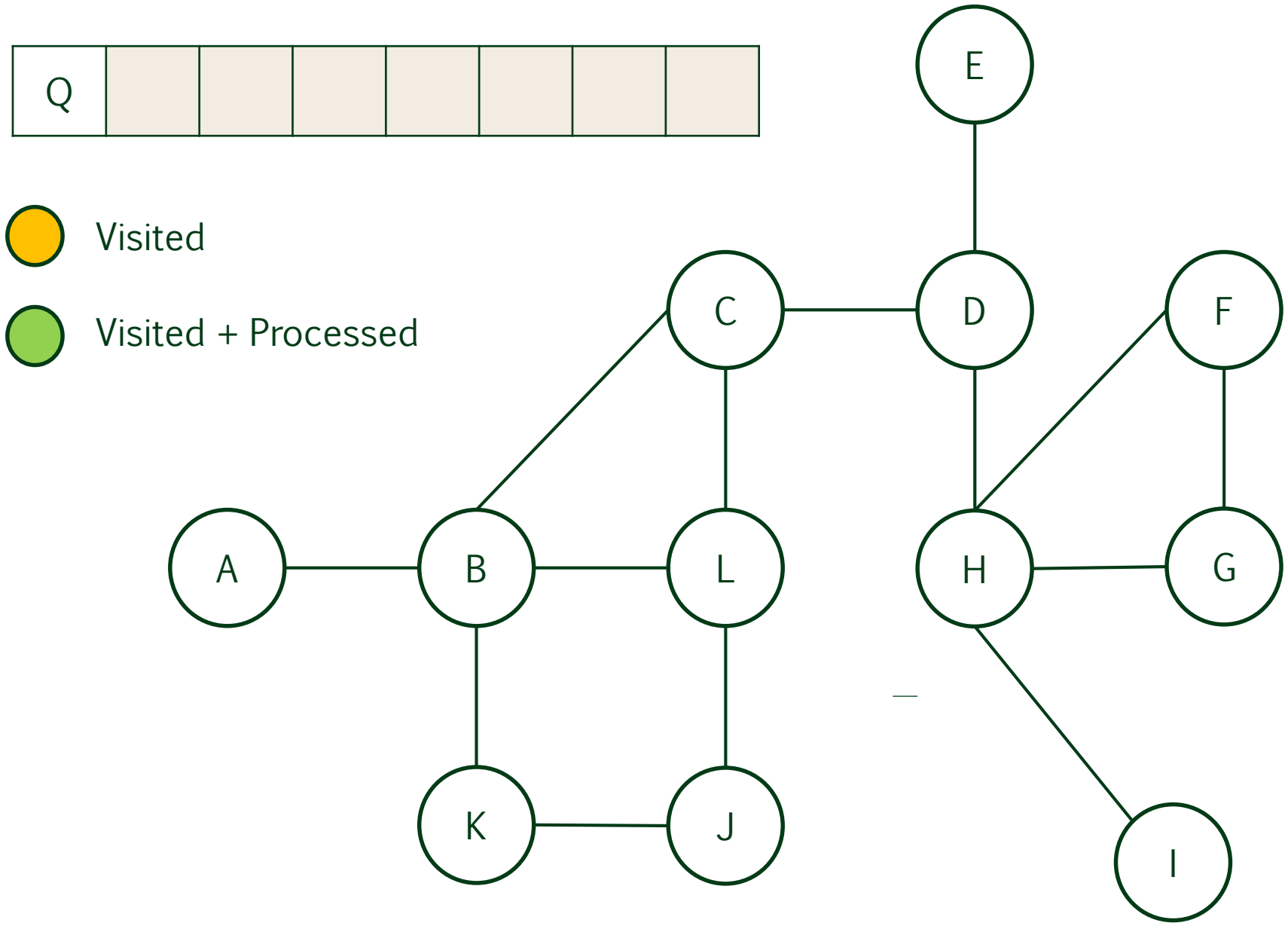
- Underlying data structure: Queue

Alternate descriptions

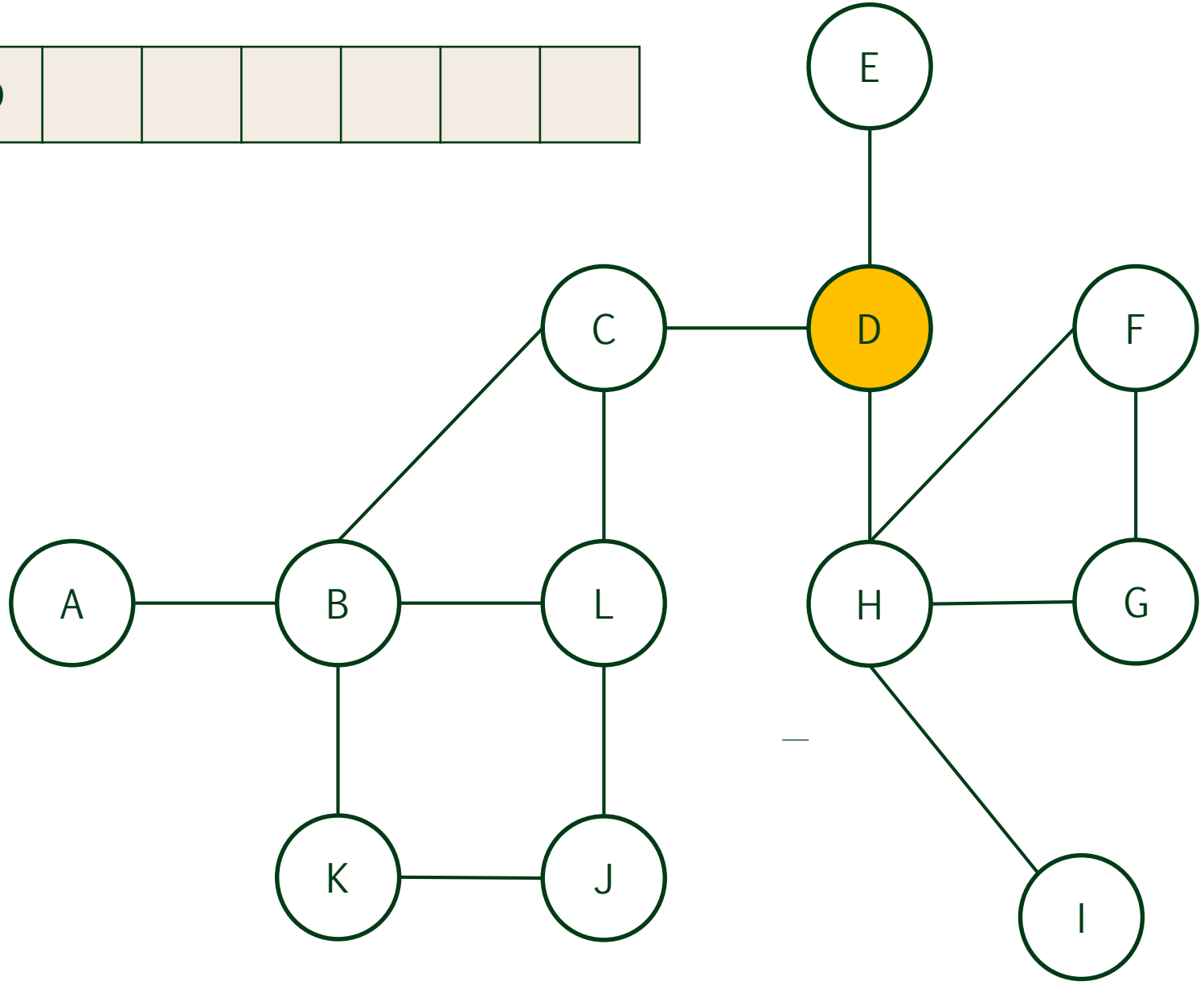
- “Explore vertices of a graph level-by-level from the starting vertex.”

Basic algorithm

```
Breadthfirstsearch (vertex v)
    mark v as visited
    Q.Enqueue(v)
    while !Q.Empty() do
        v = Q.Dequeue();
        process v
        for each unvisited adjacent vertex w to v do
            mark w as visited
            Q.Enqueue(w)
```

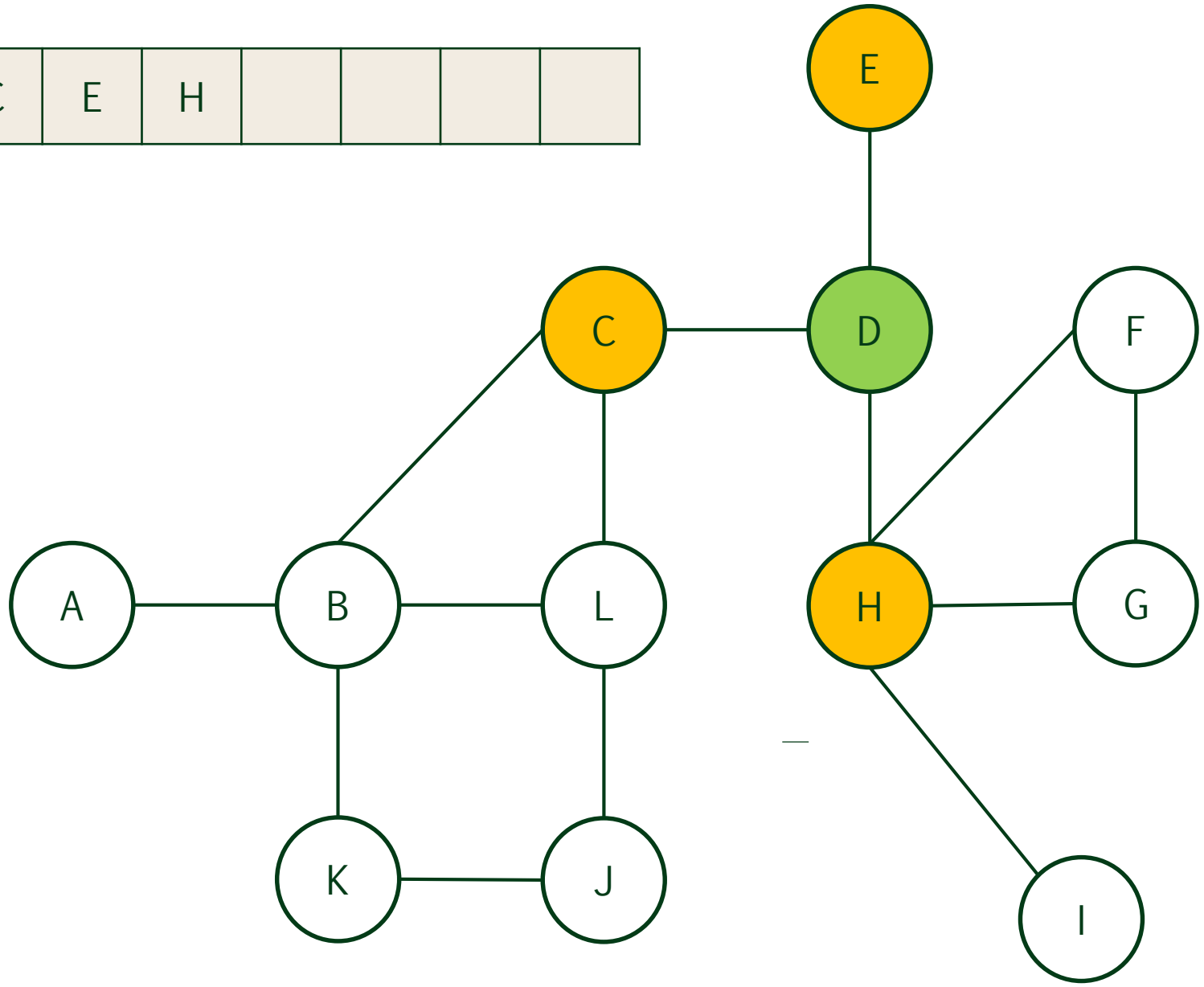

Q
Visited
Visited + Processed
E
C
D
F
A
B
L
H
G
K
J
I

Q	D						



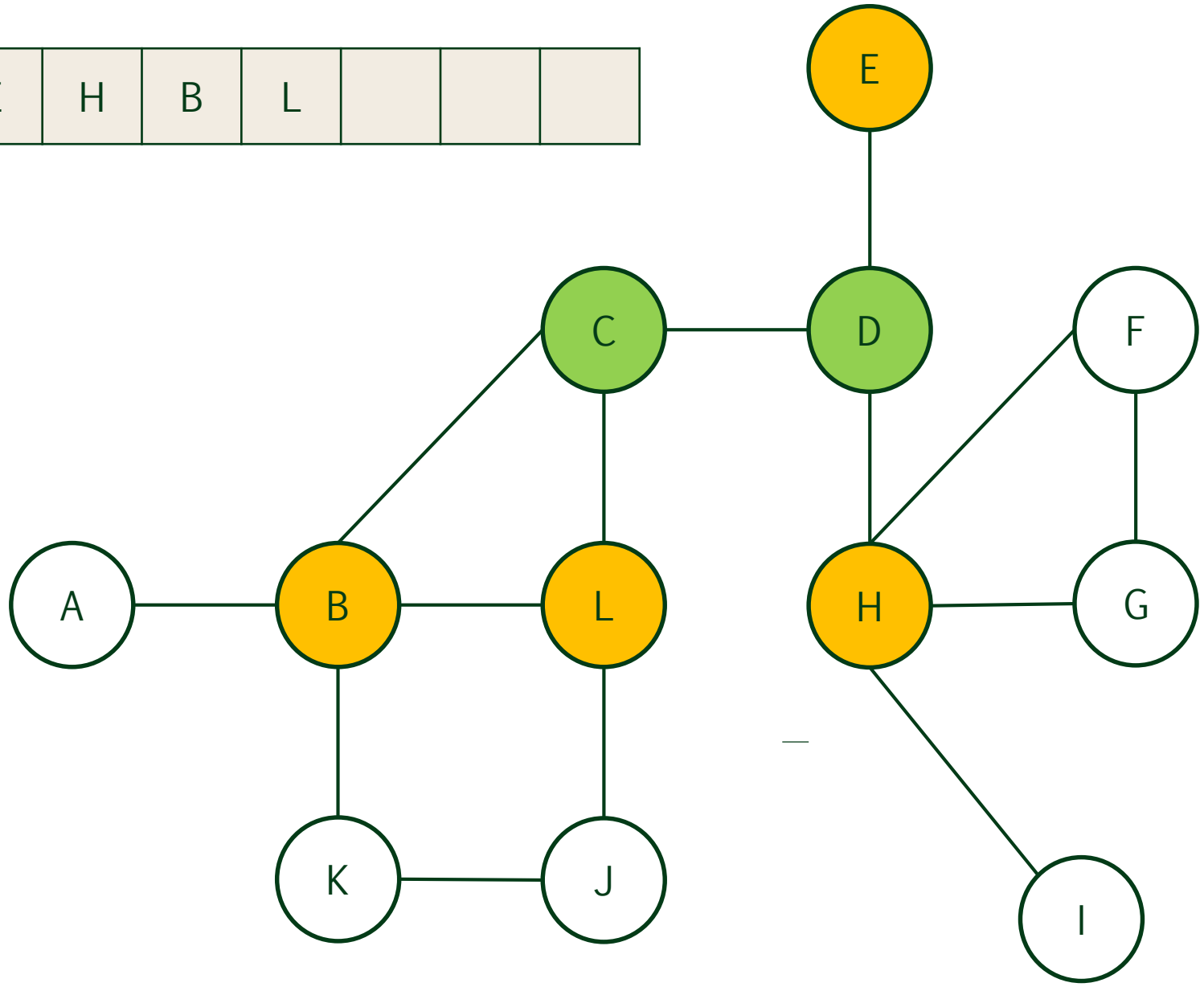
Q	C	E	H				

D



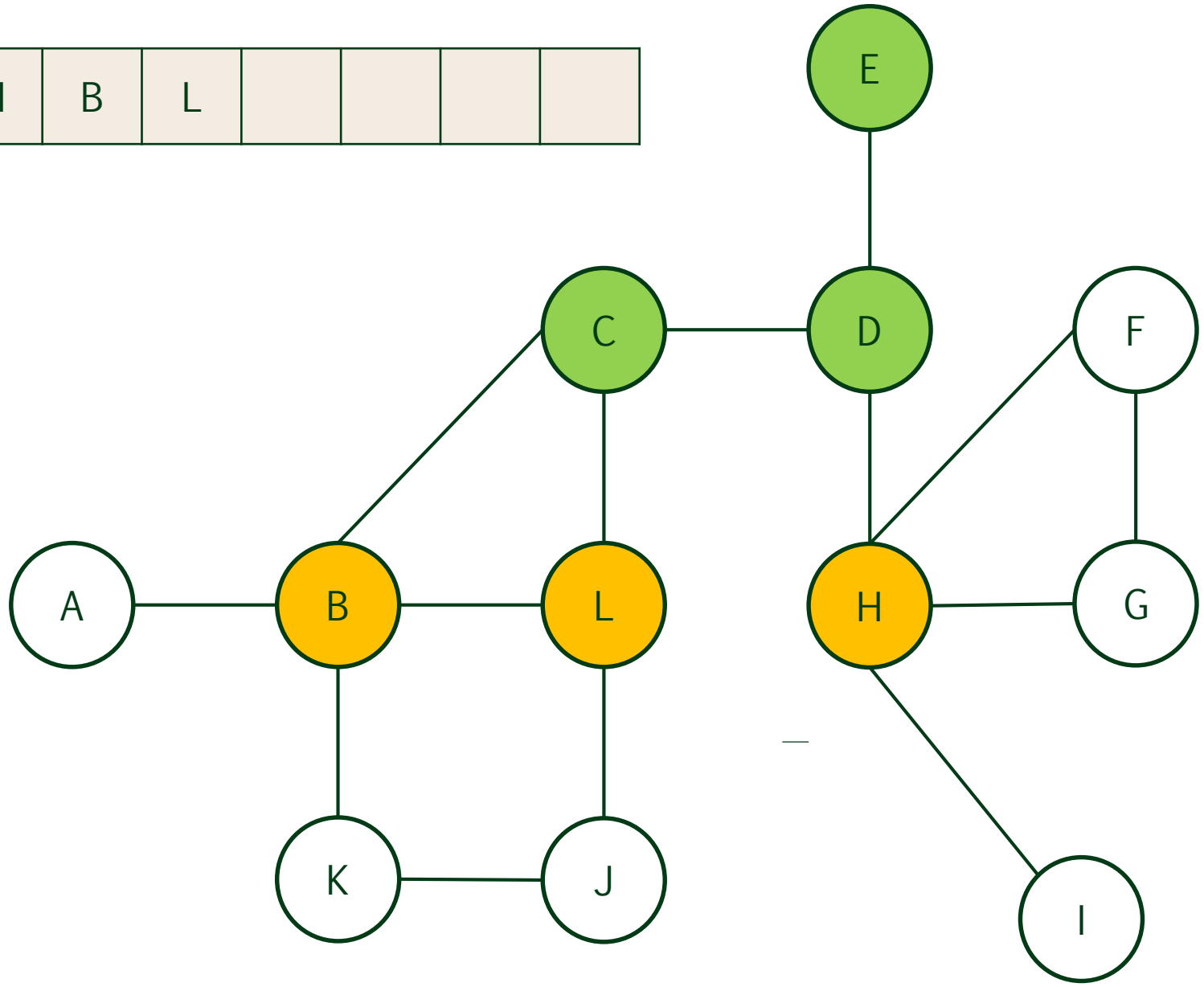
Q	E	H	B	L			

D C



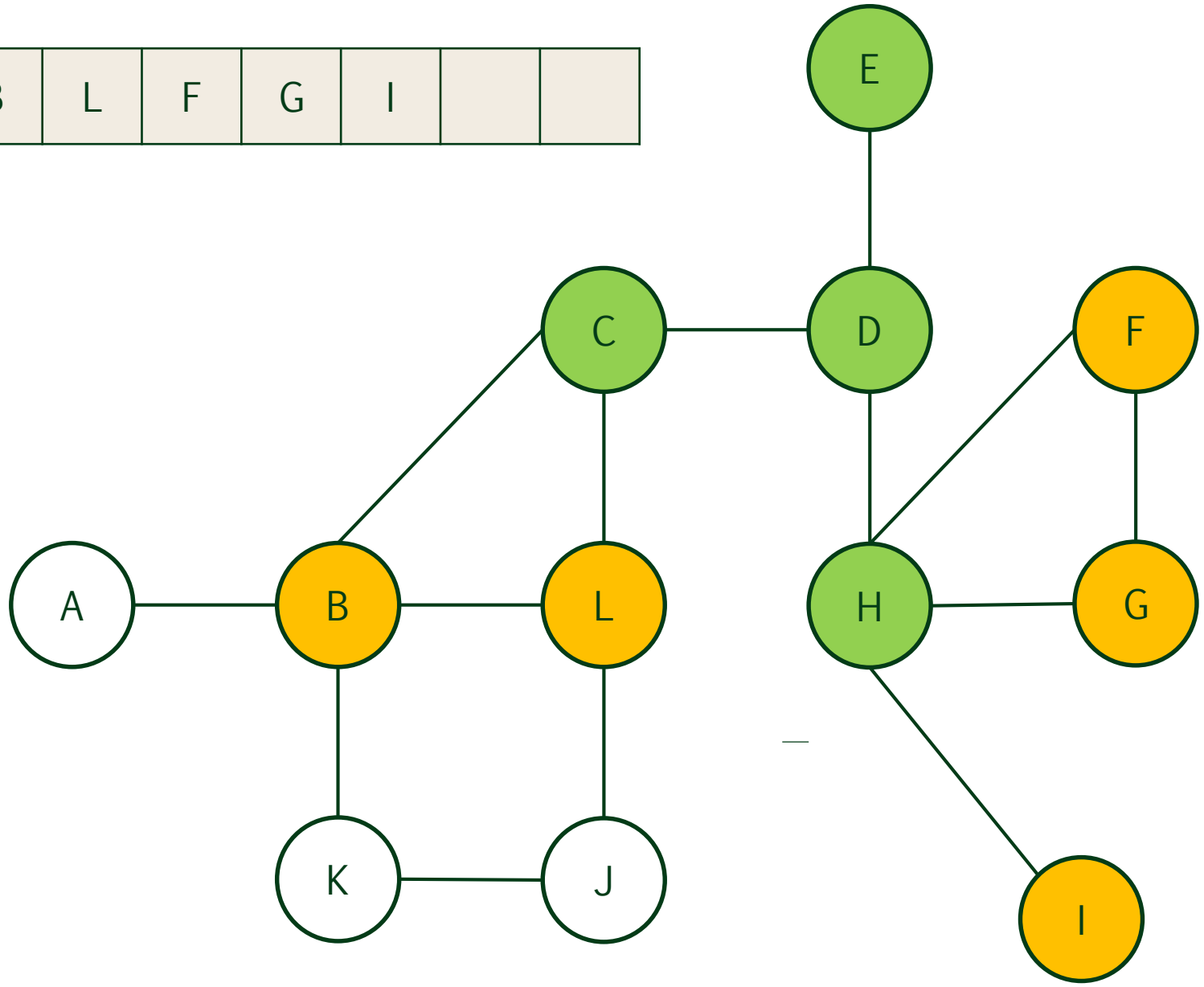
Q	H	B	L				

D C E



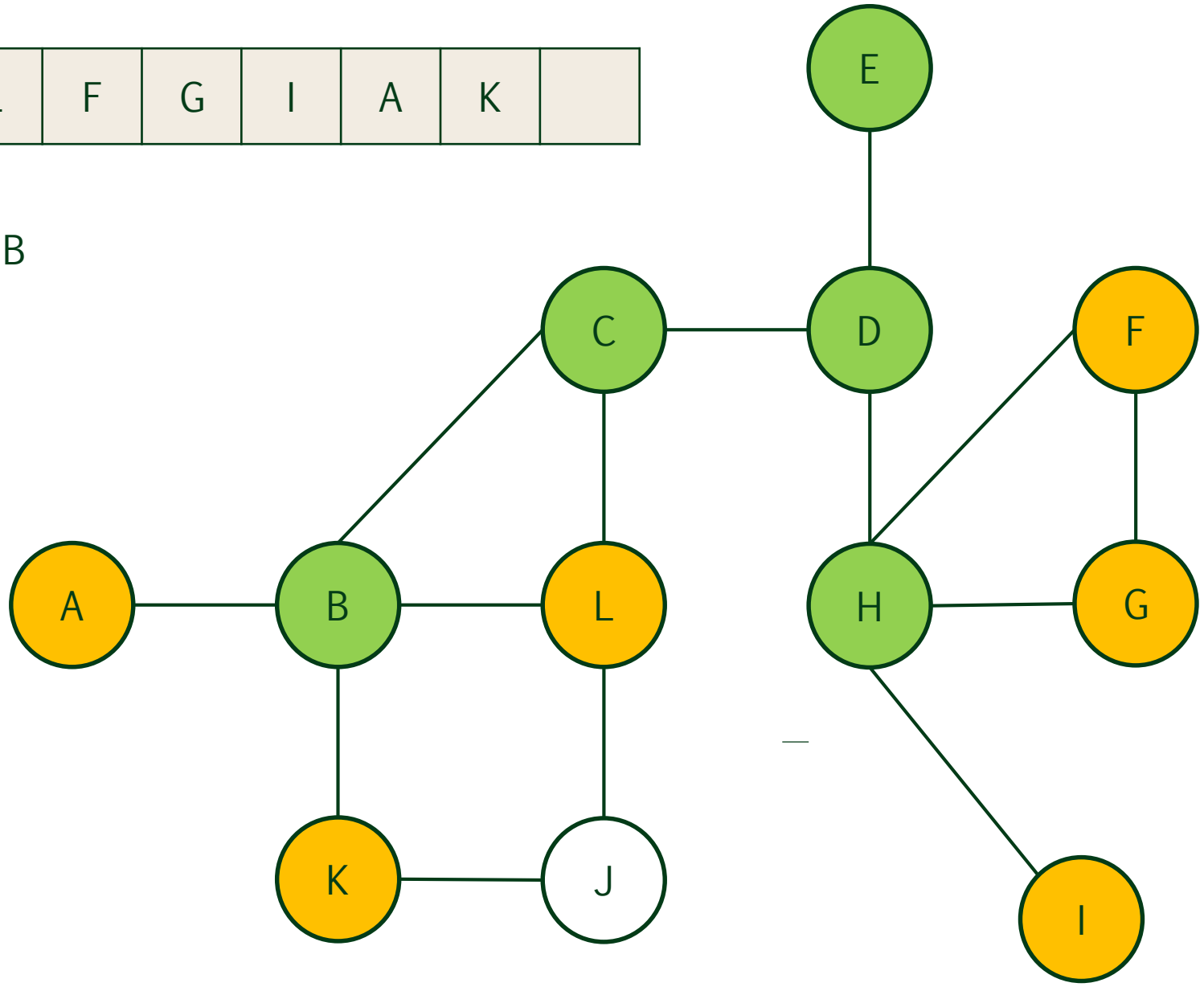
Q	B	L	F	G	I		

D C E H



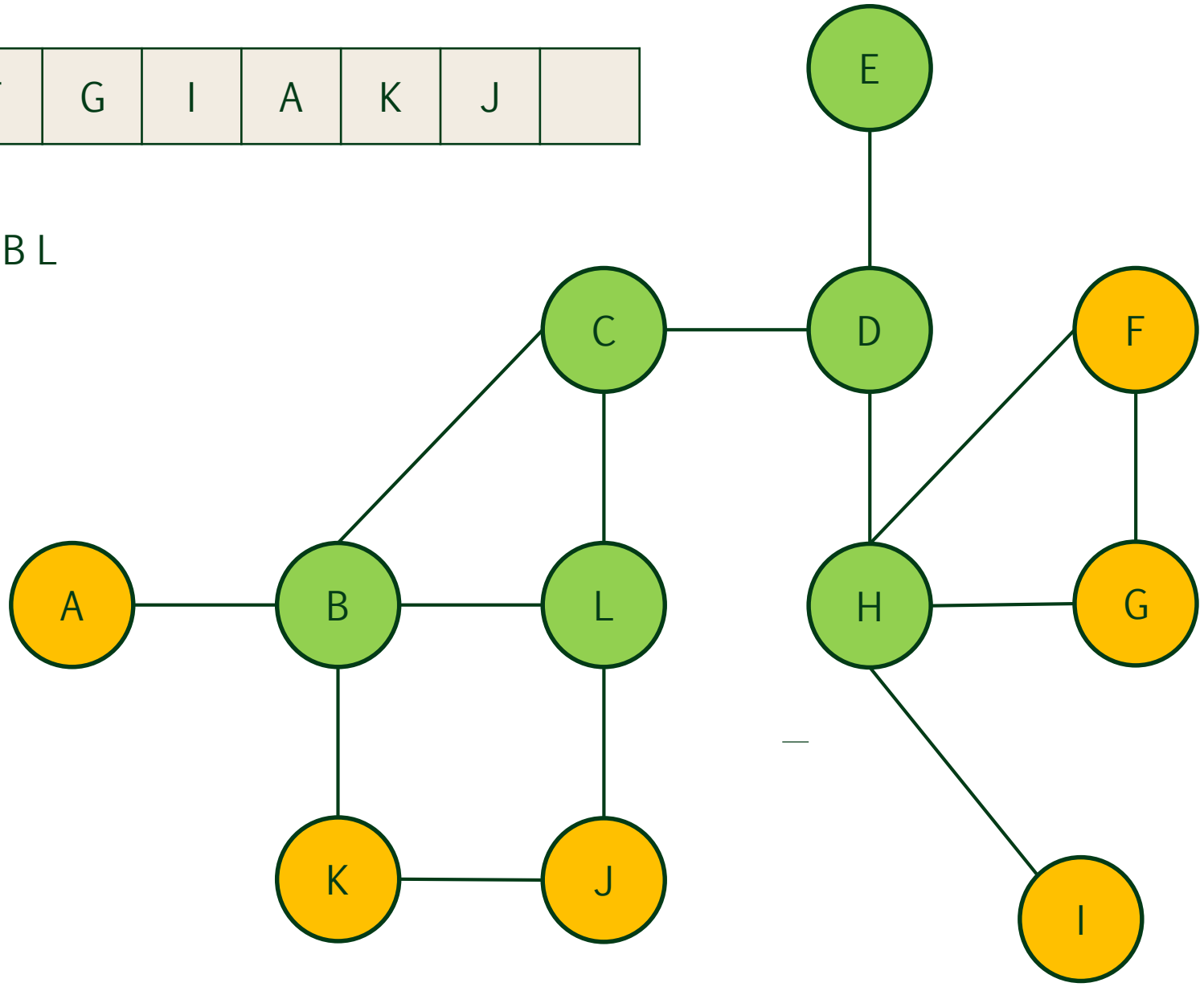
Q	L	F	G	I	A	K	

D C E H B



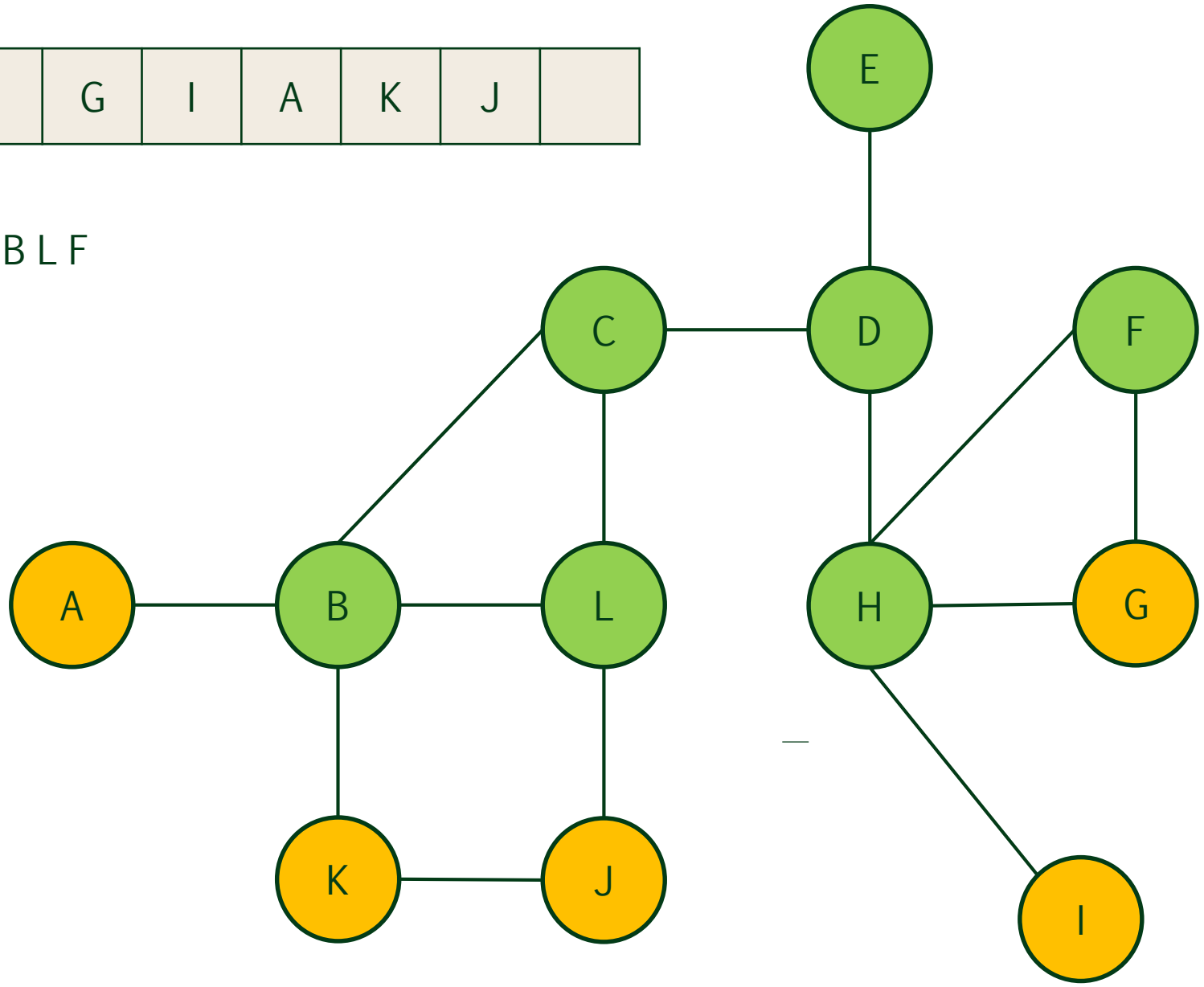
Q	F	G	I	A	K	J	

D C E H B L



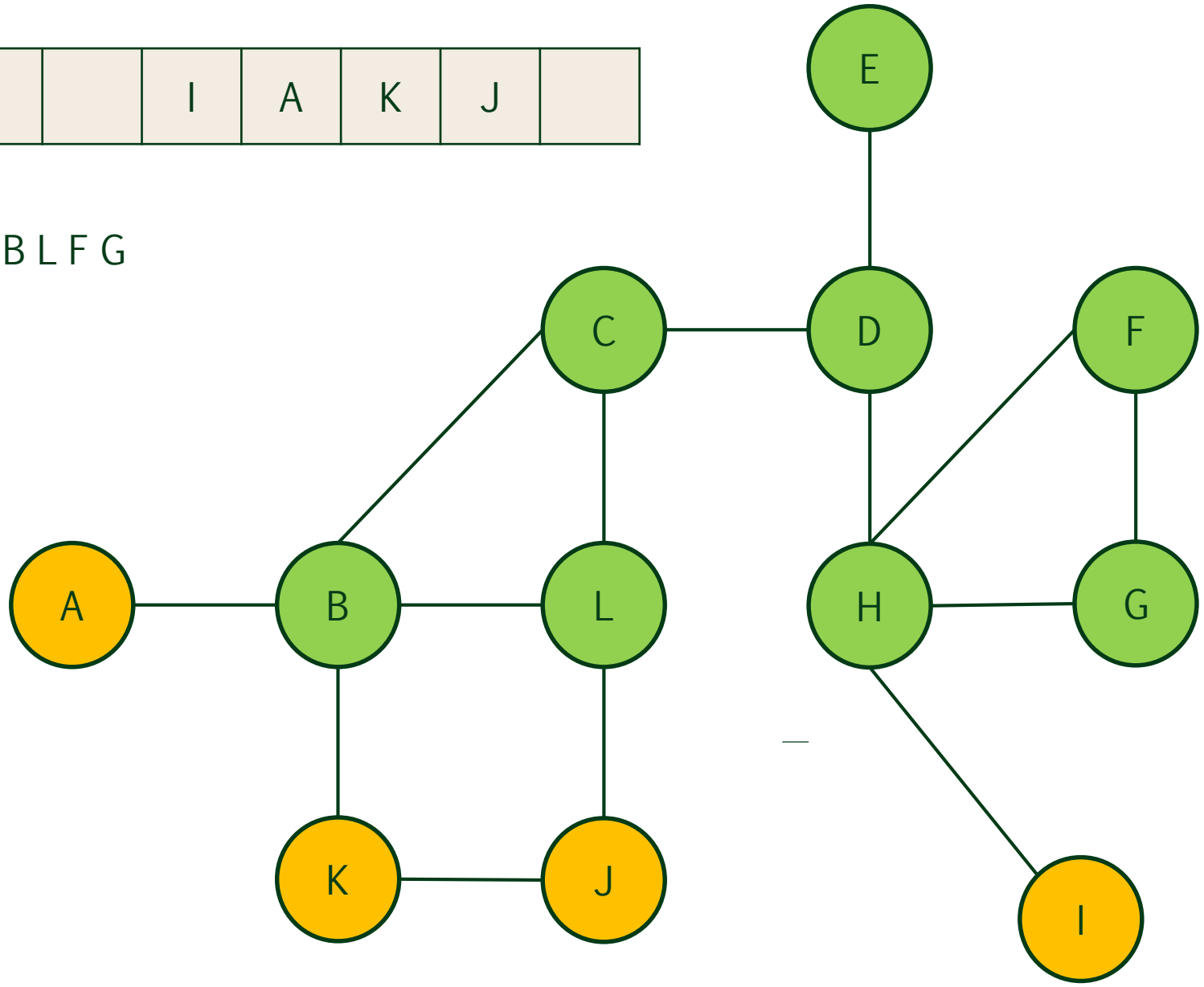
Q		G	I	A	K	J	

D C E H B L F



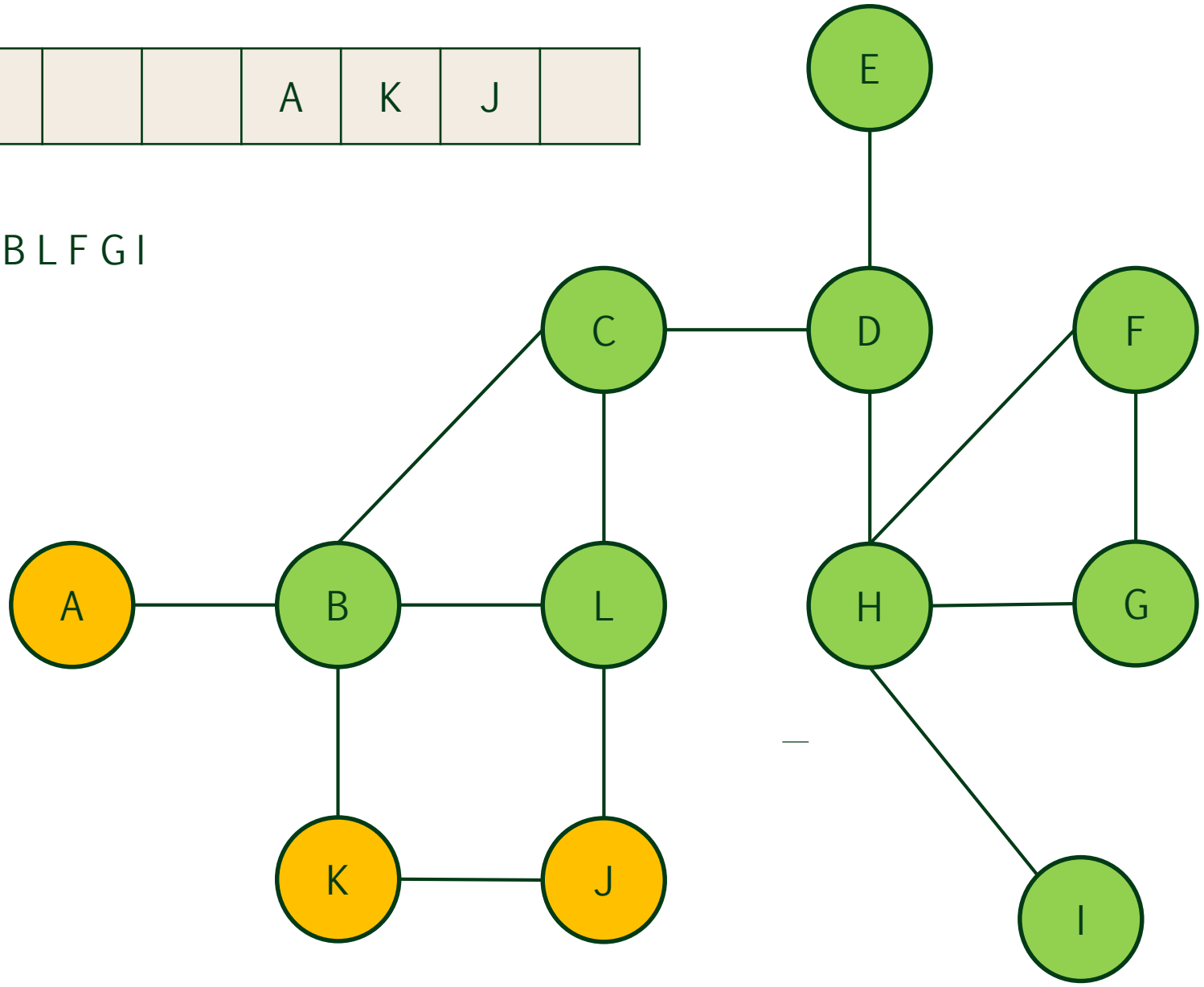
Q			I	A	K	J	

D C E H B L F G



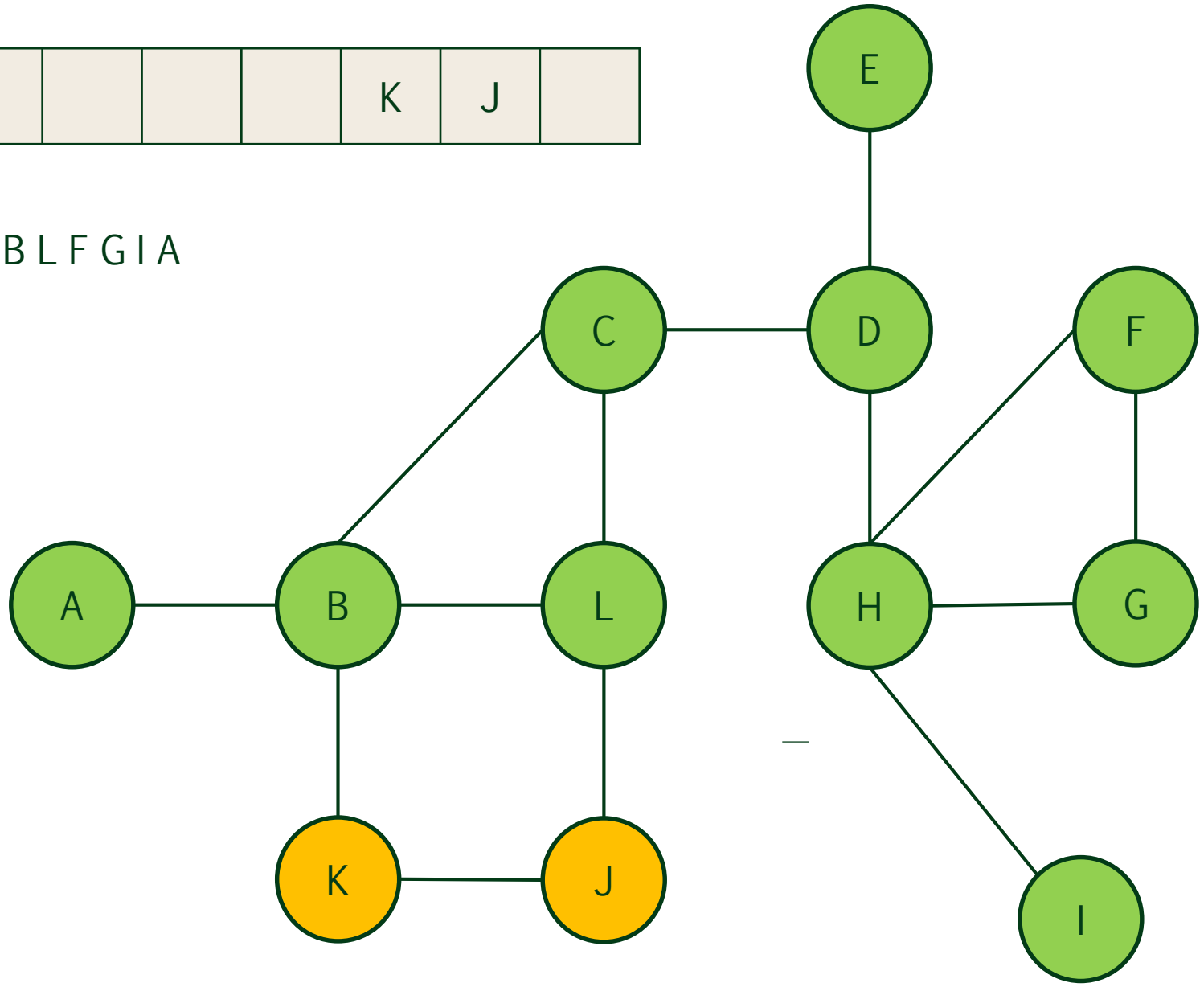
Q				A	K	J	

D C E H B L F G I



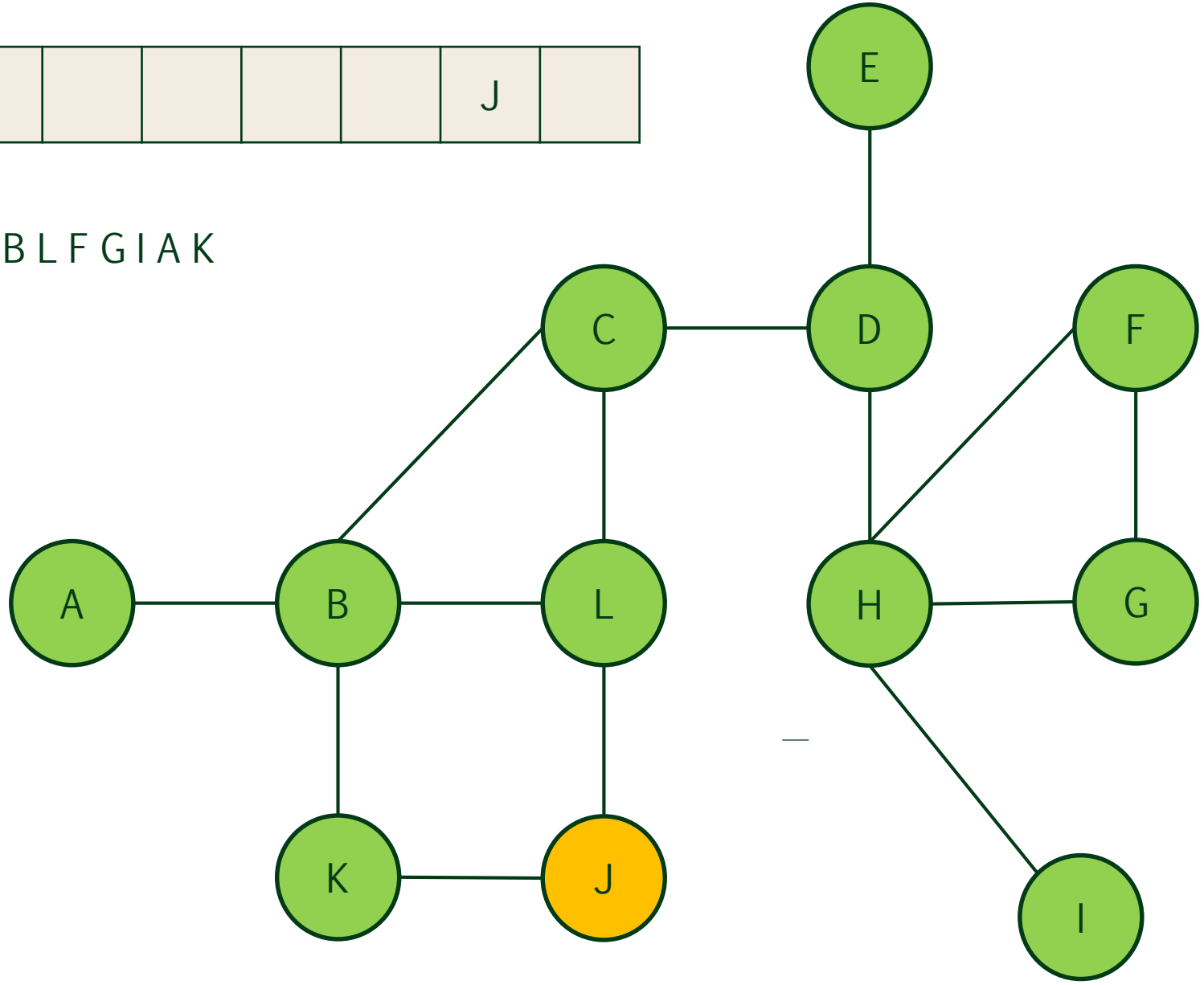
Q					K	J	

D C E H B L F G I A



Q						J	

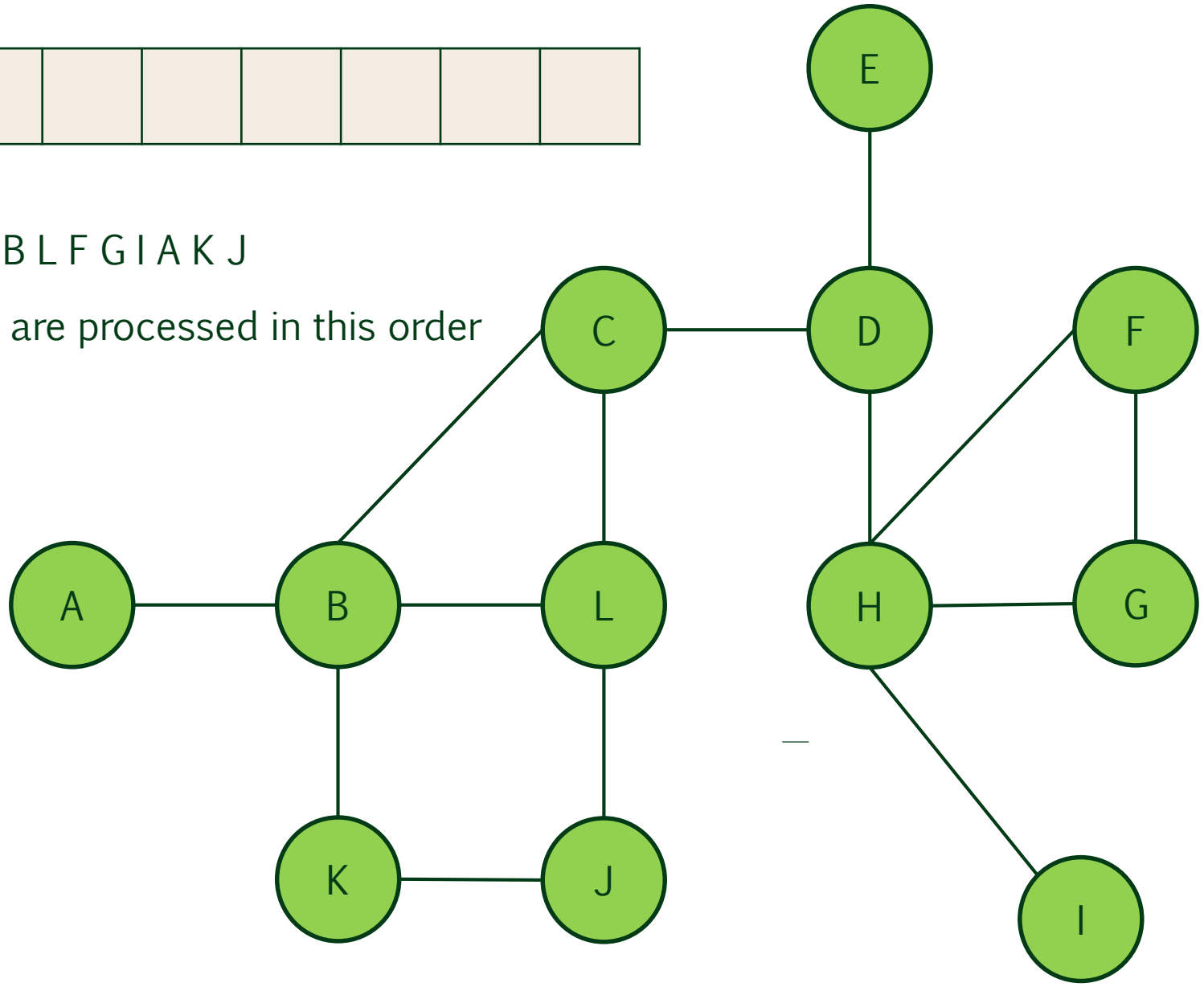
D C E H B L F G I A K



Q							

D C E H B L F G I A K J

Vertices are processed in this order



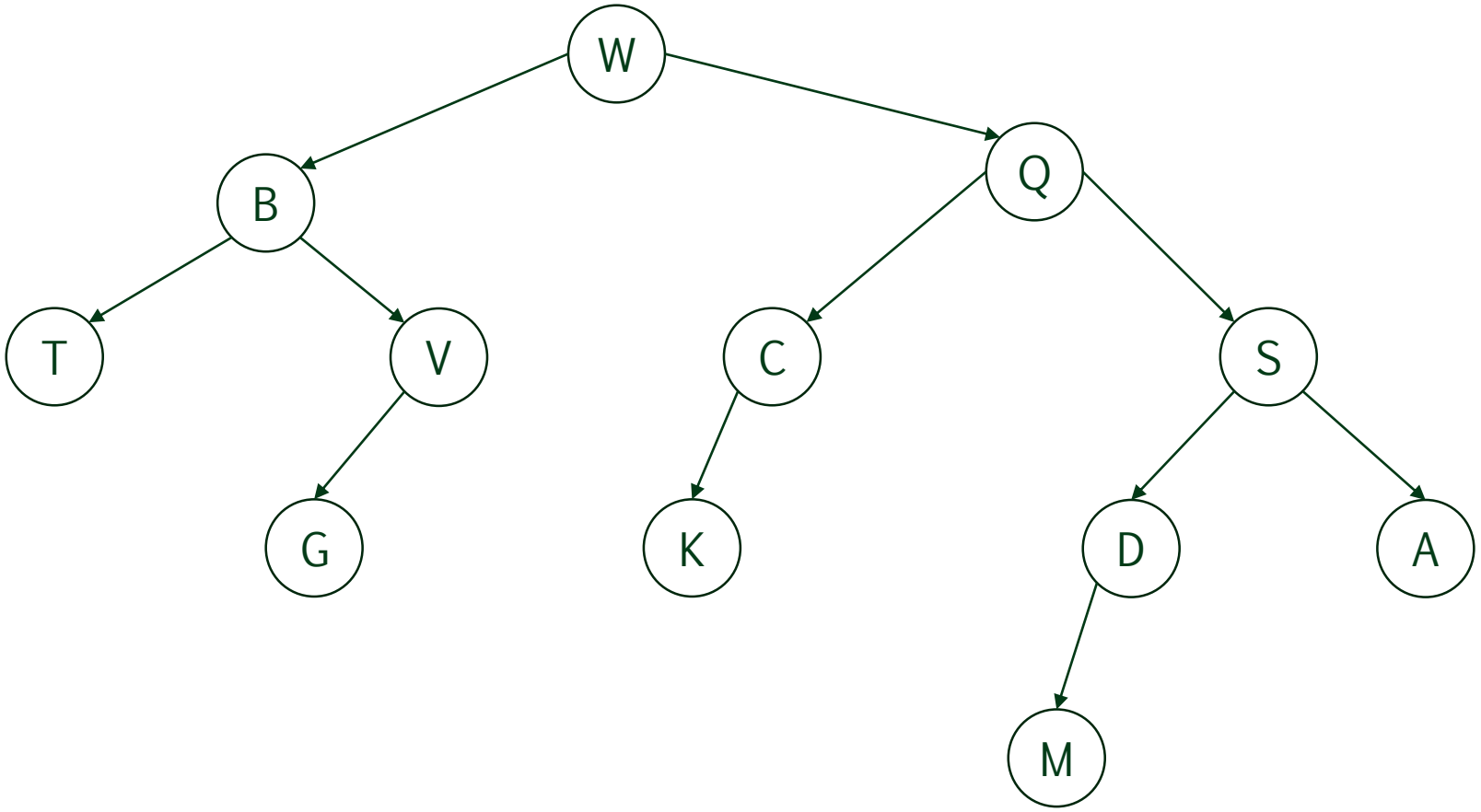
Note that all vertices at distance of i from the starting vertex are processed before all vertices at distance i+1

Vertex	D	C	E	H	B	L	F	G	I	A	K	J
Distance	0	1			2					3		

Implication

Once a vertex is processed, the shortest path to that vertex has been found!

Consider the following binary tree T



Exercise

- Perform a depth-first and breadth-first search on T starting at vertex W
- Output in each case, the vertices as they are processed
- If the search begins at vertex Q, how does the depth-first or breadth-first search proceed to explore the entire tree T?