THE UNIVERSITY OF VERMONT
COLLEGE OF ENGINEERING &
MATHEMATICAL SCIENCES

Representing Trees

CS 124 / Department of Computer Science

How do we represent a tree?

Choosing a data structure

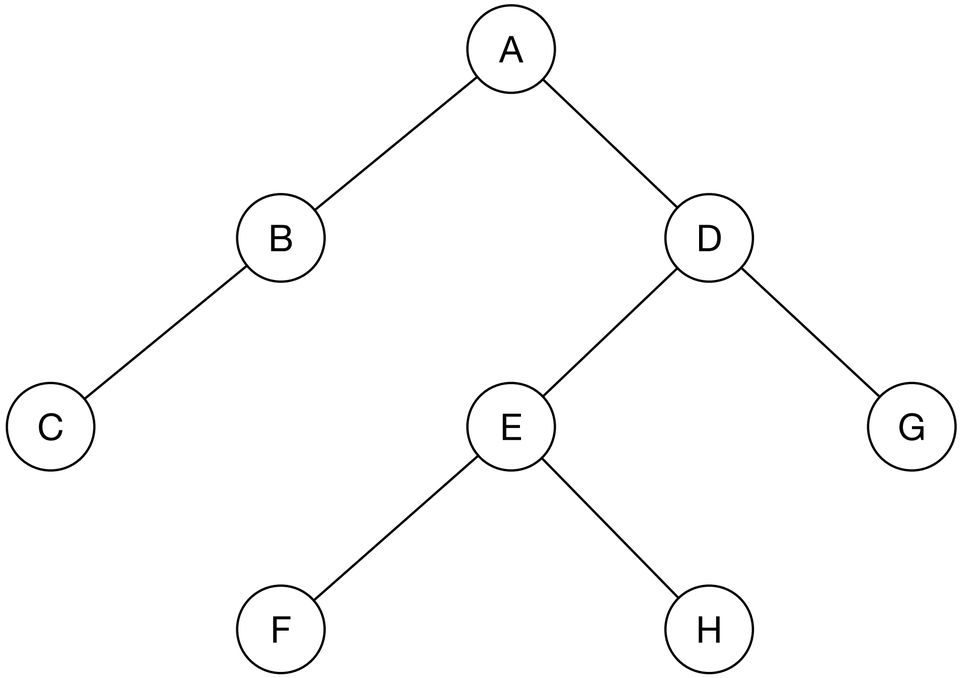
- Adjacency list
- Adjacency matrix
- Incidence matrix
- Simple L/R object-oriented approach

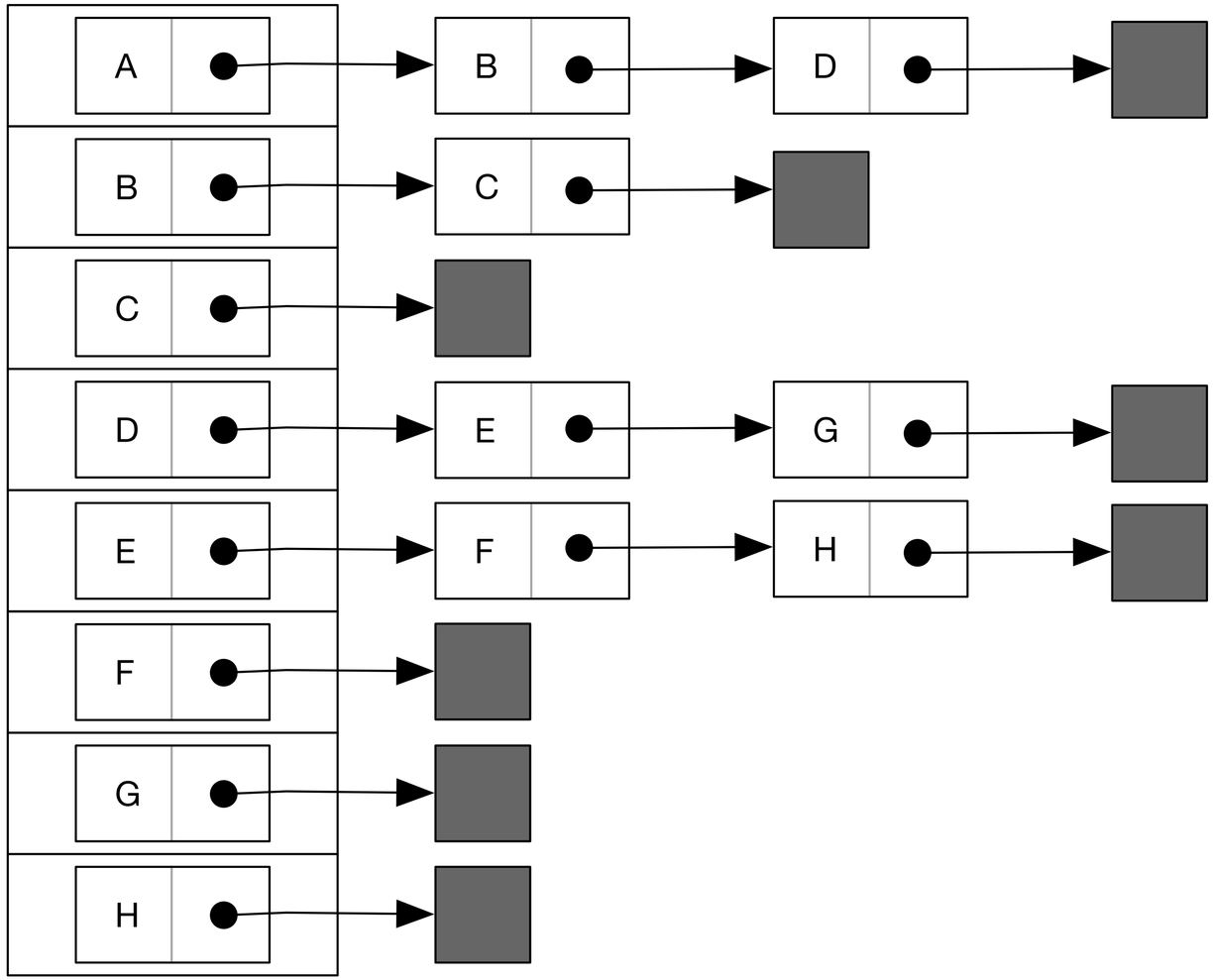
How do we represent a tree?

Choosing a data structure

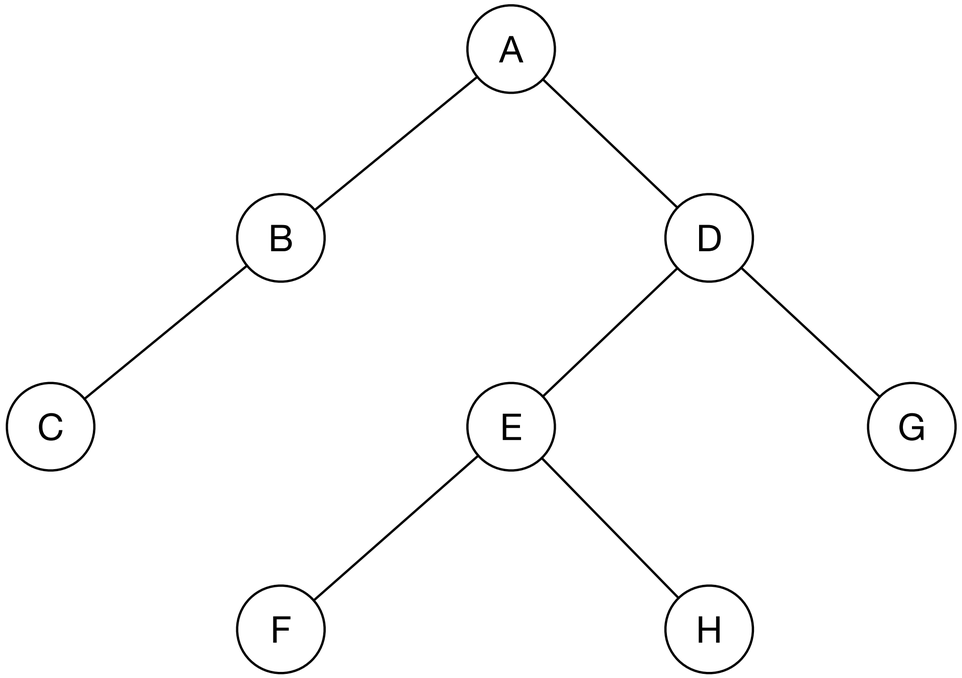
- Adjacency list
- Adjacency matrix
- Incidence matrix
- Simple L/R object-oriented approach

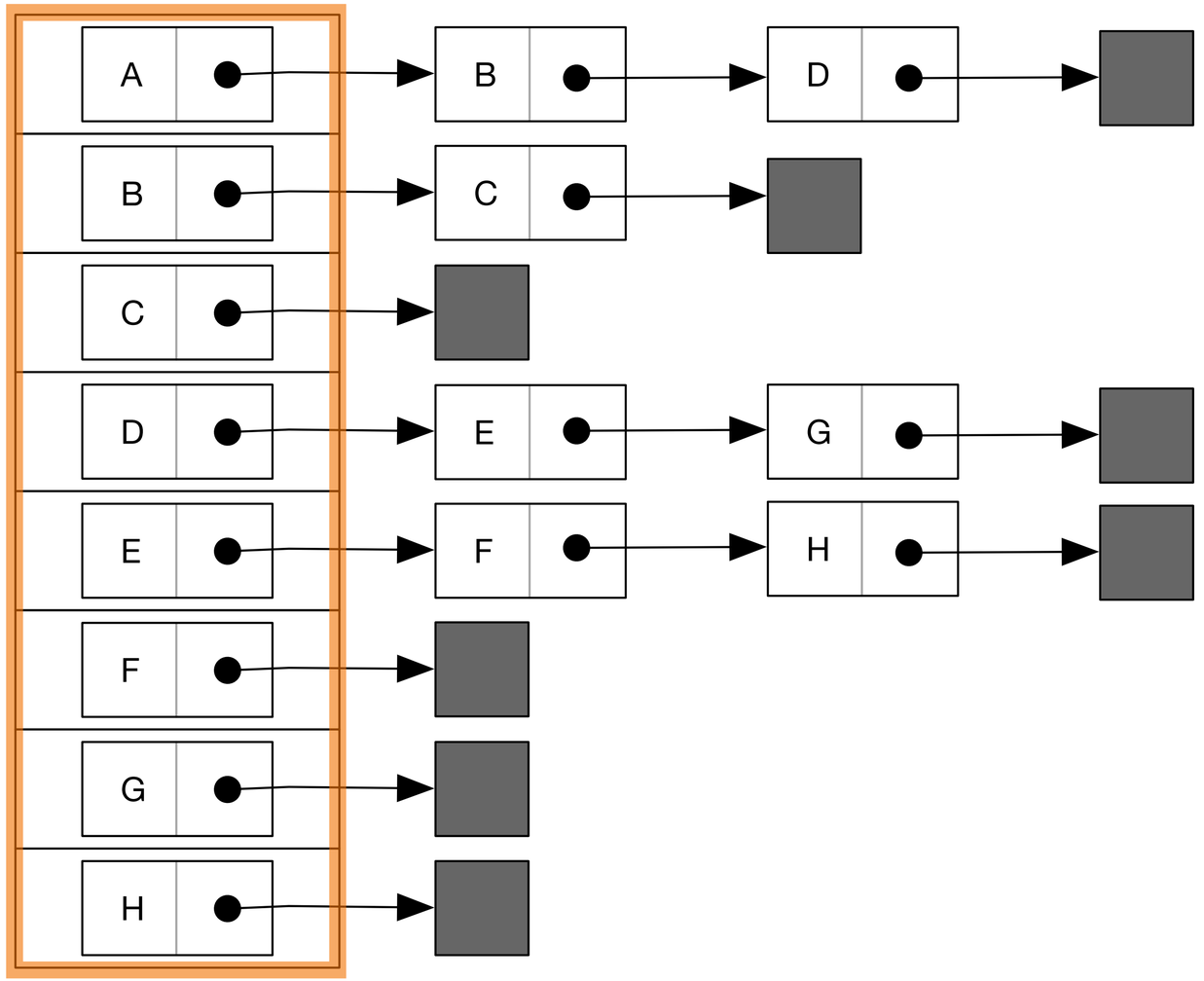
Adjacency list



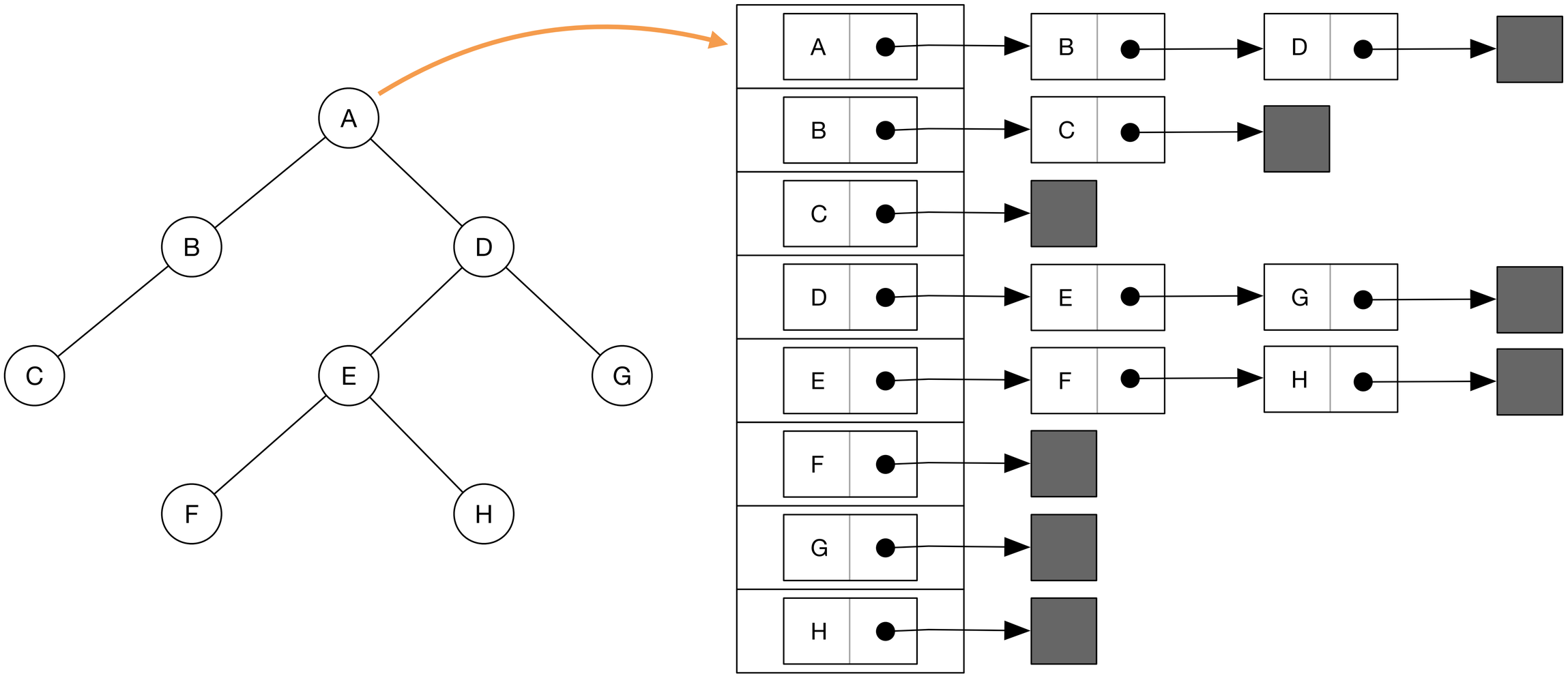


Adjacency list

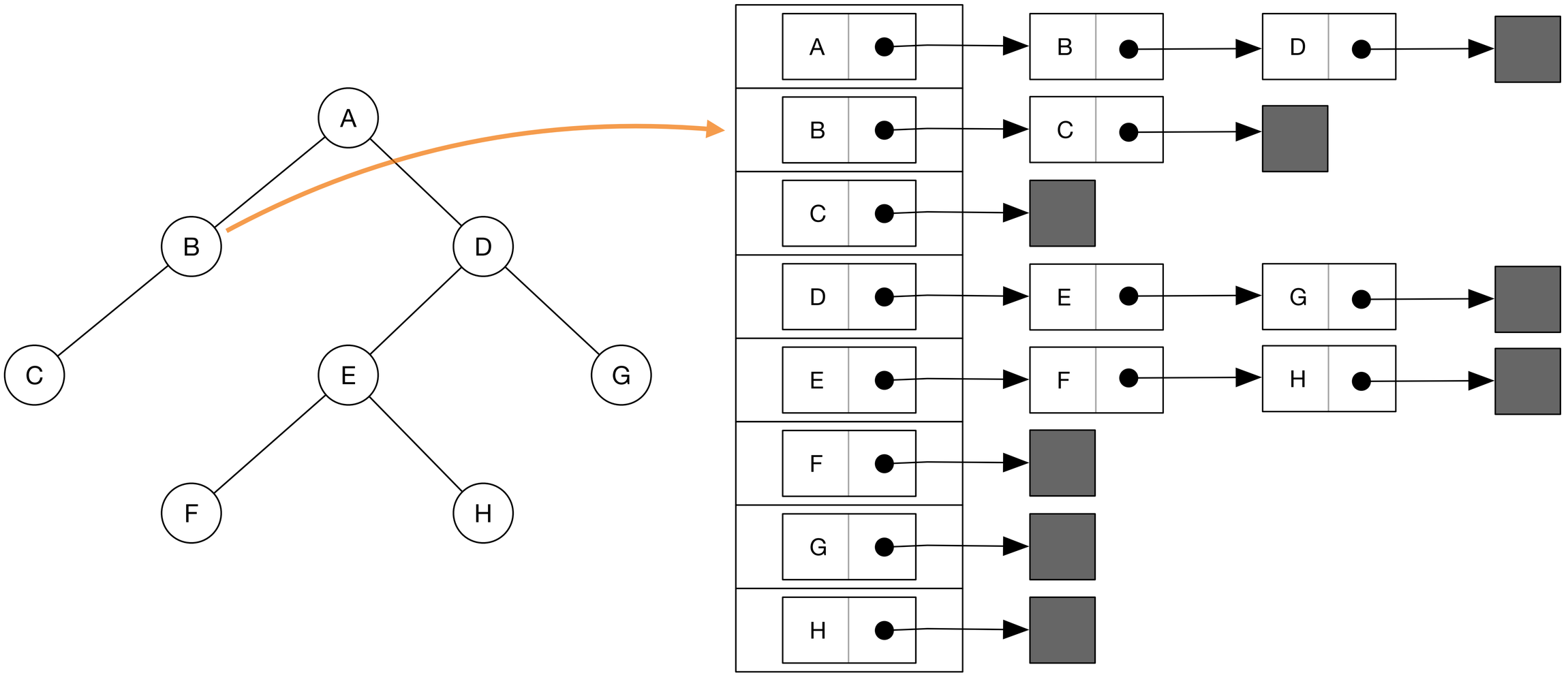




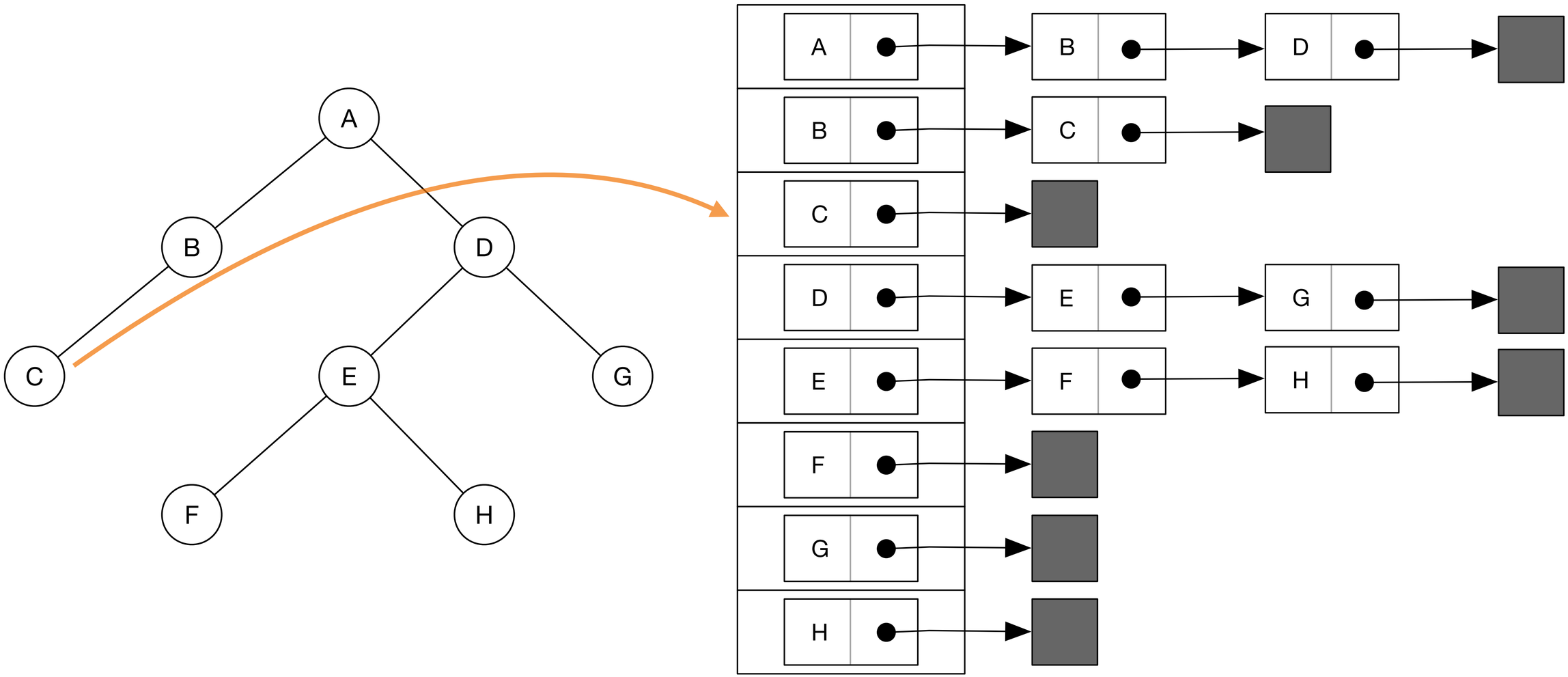
Adjacency list



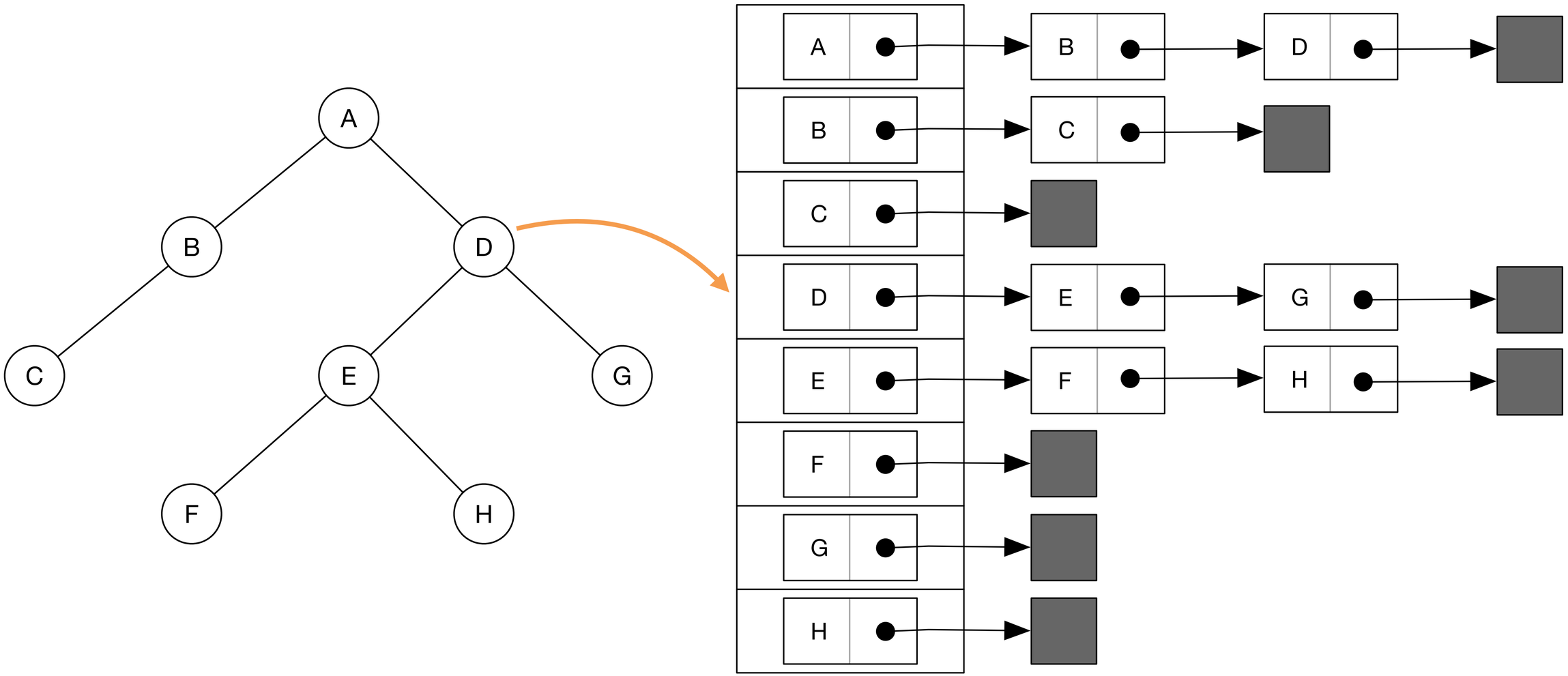
Adjacency list



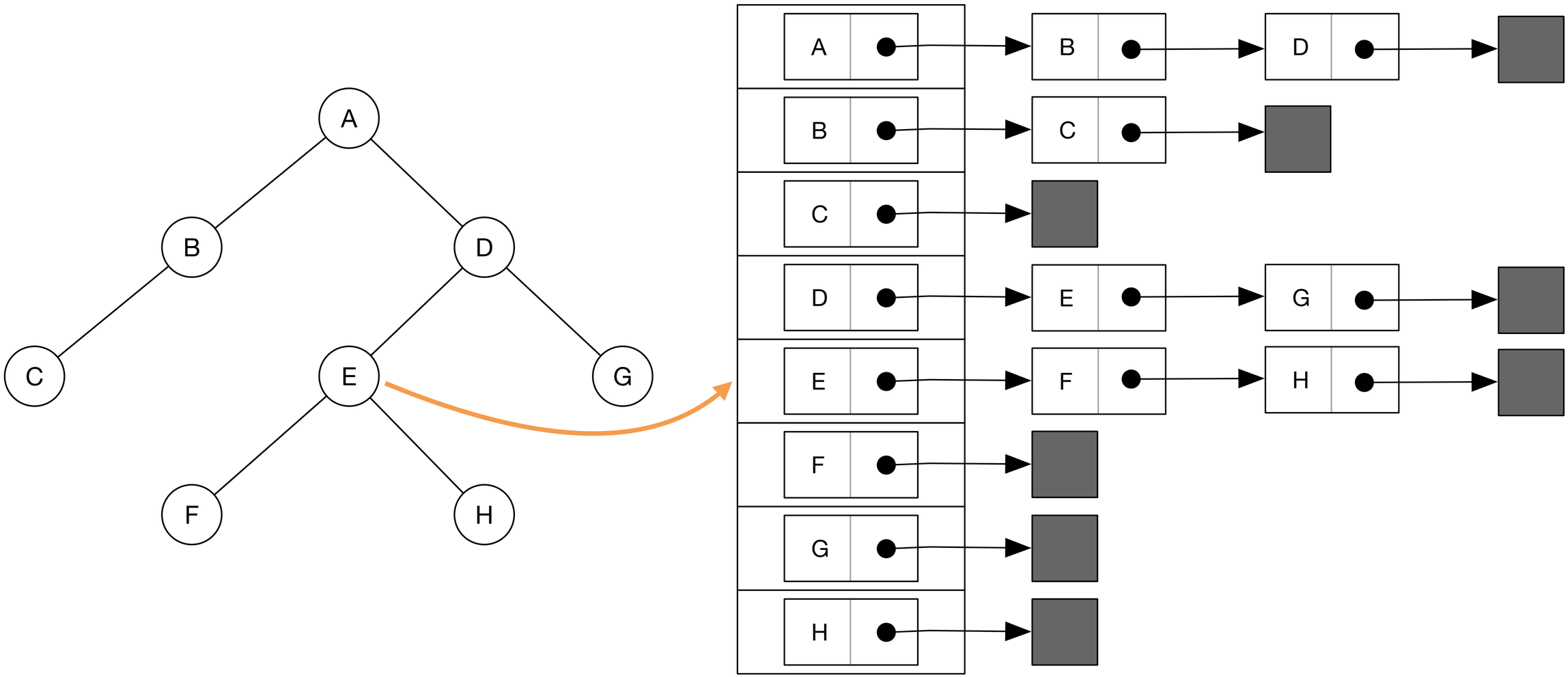
Adjacency list



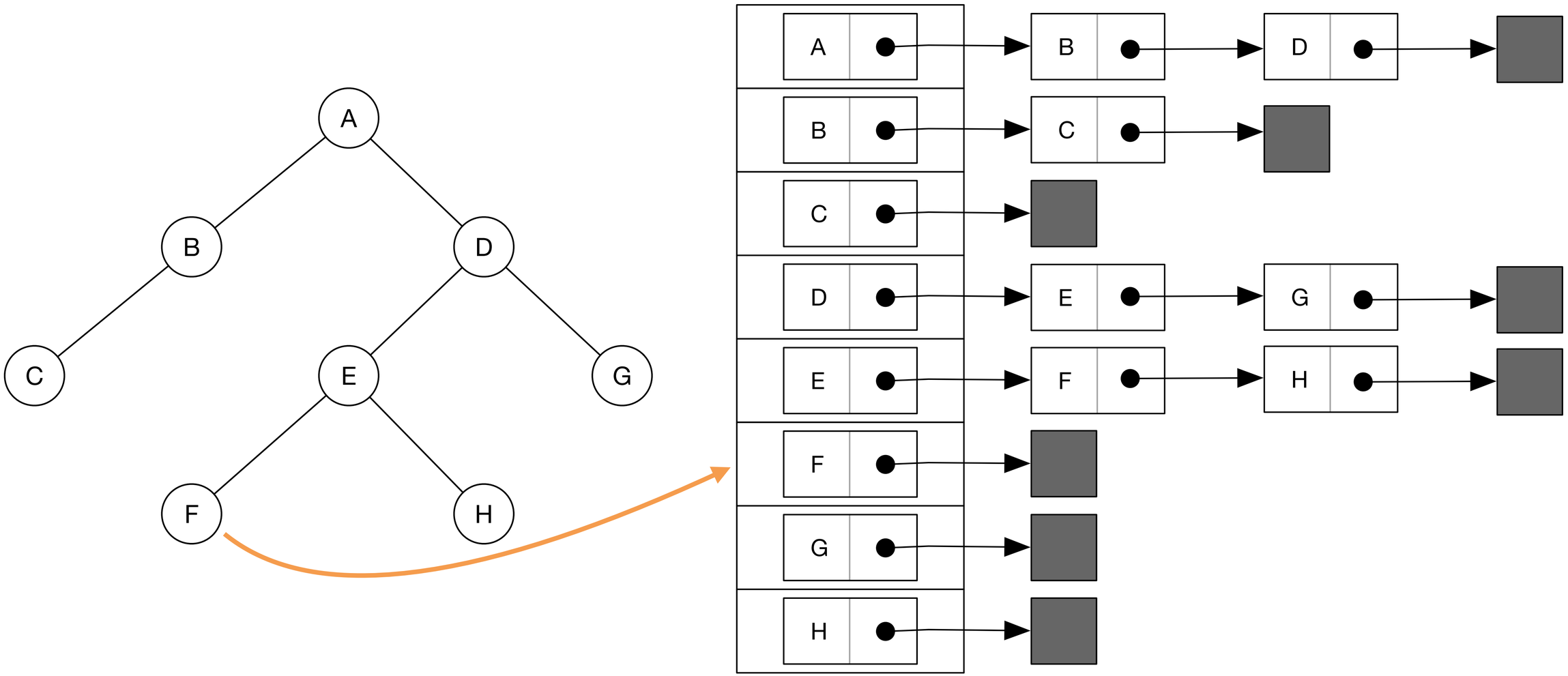
Adjacency list



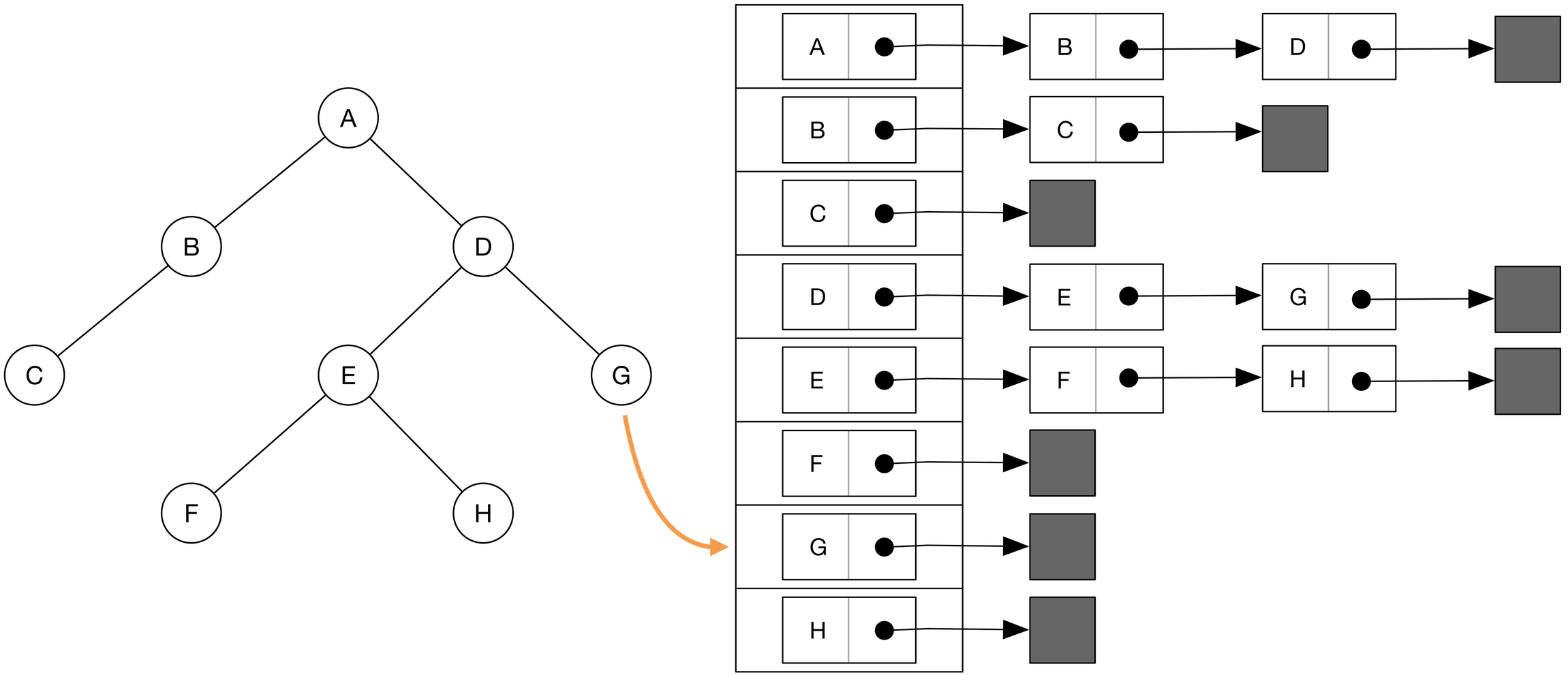
Adjacency list



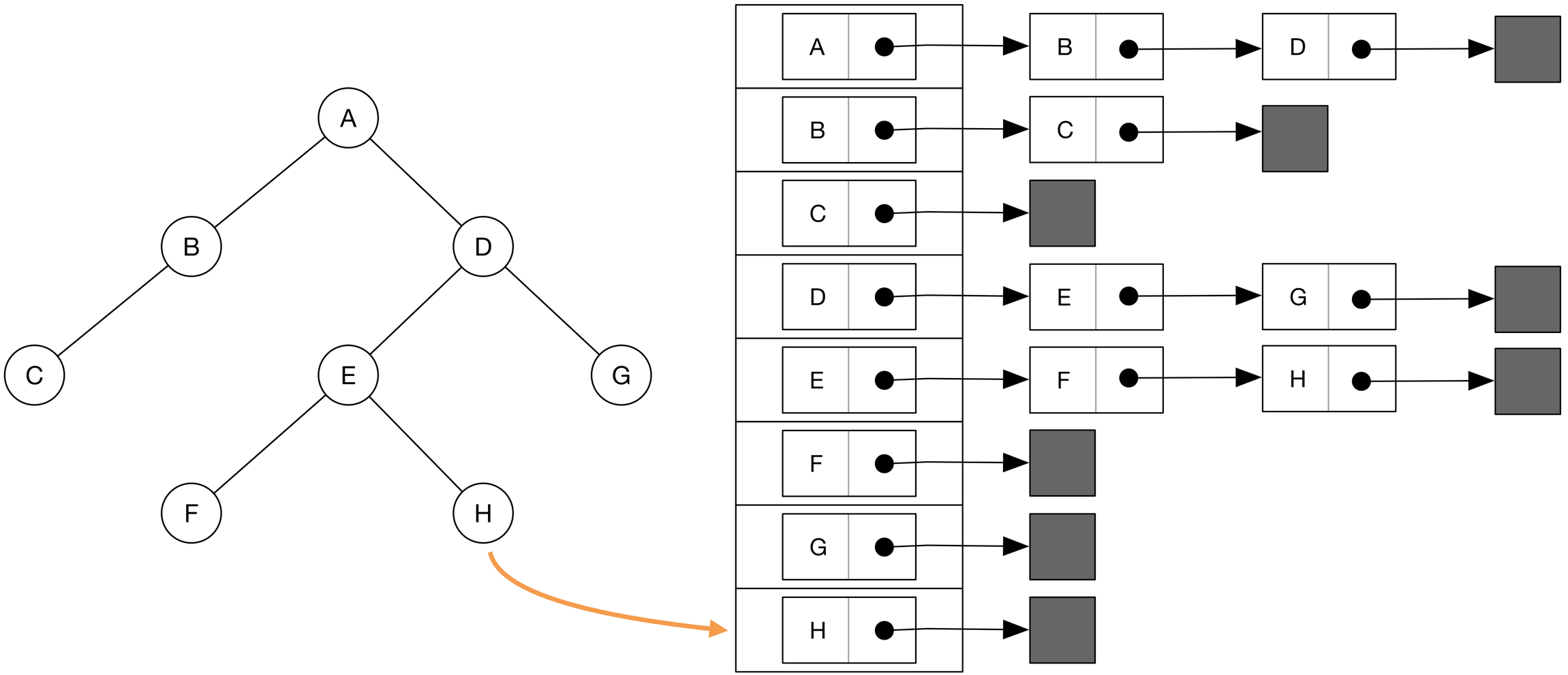
Adjacency list



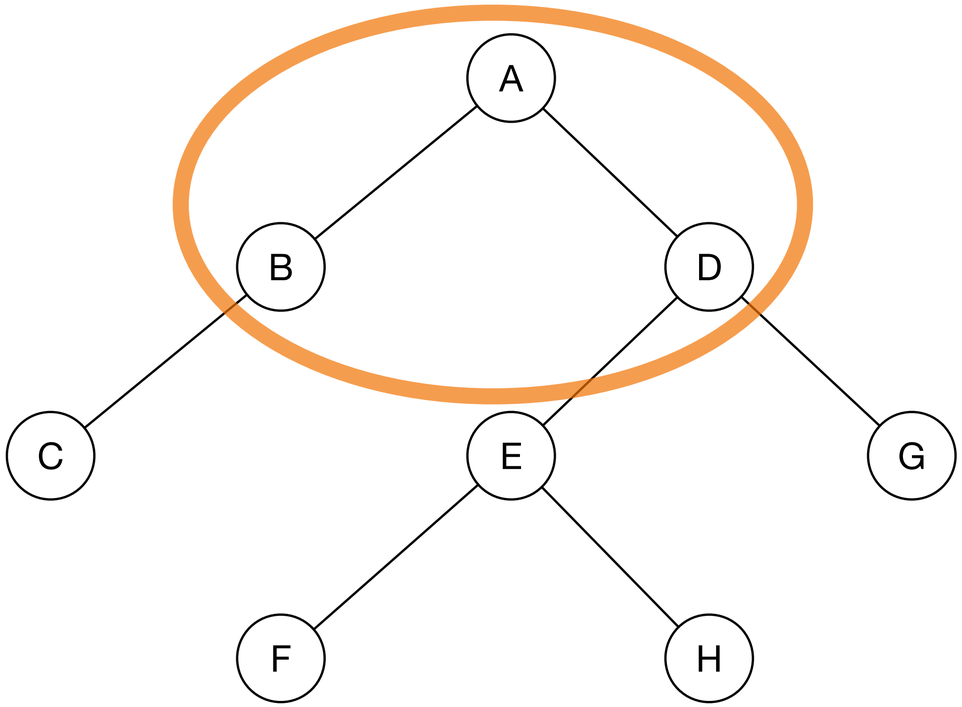
Adjacency list

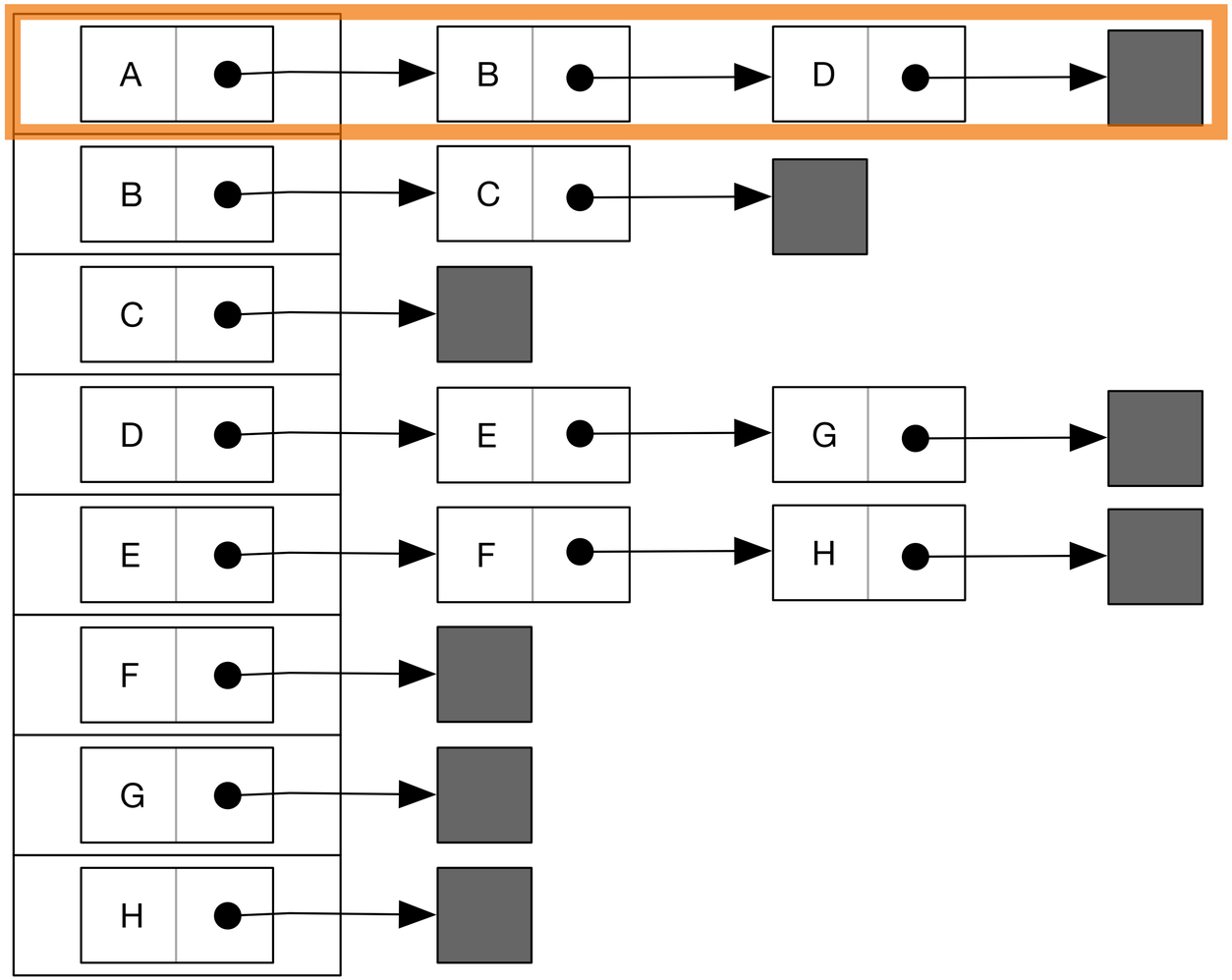


Adjacency list

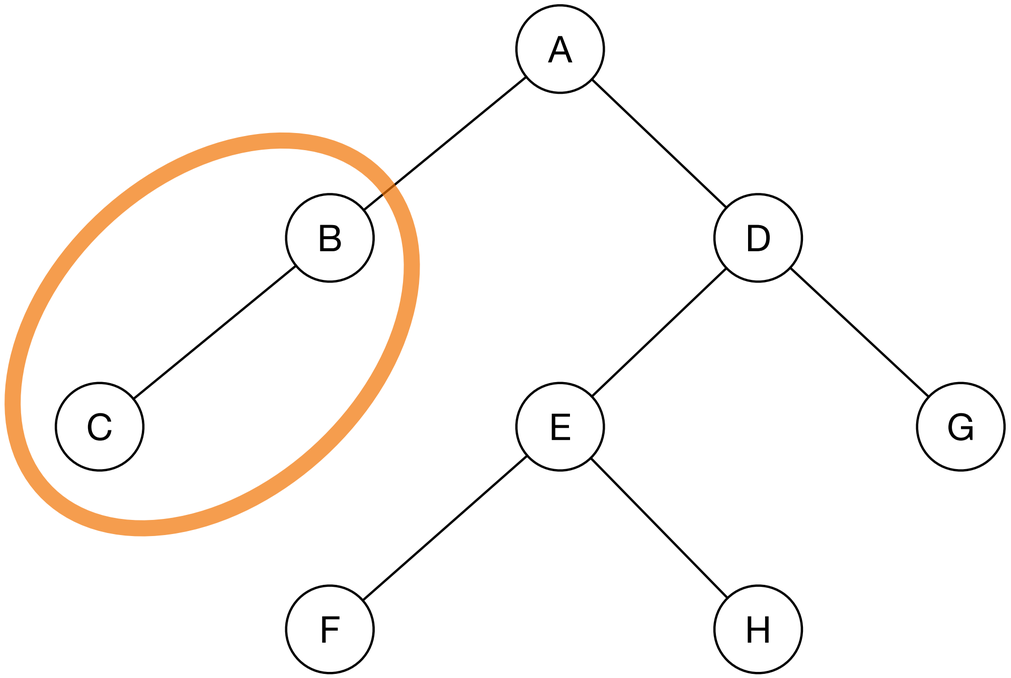


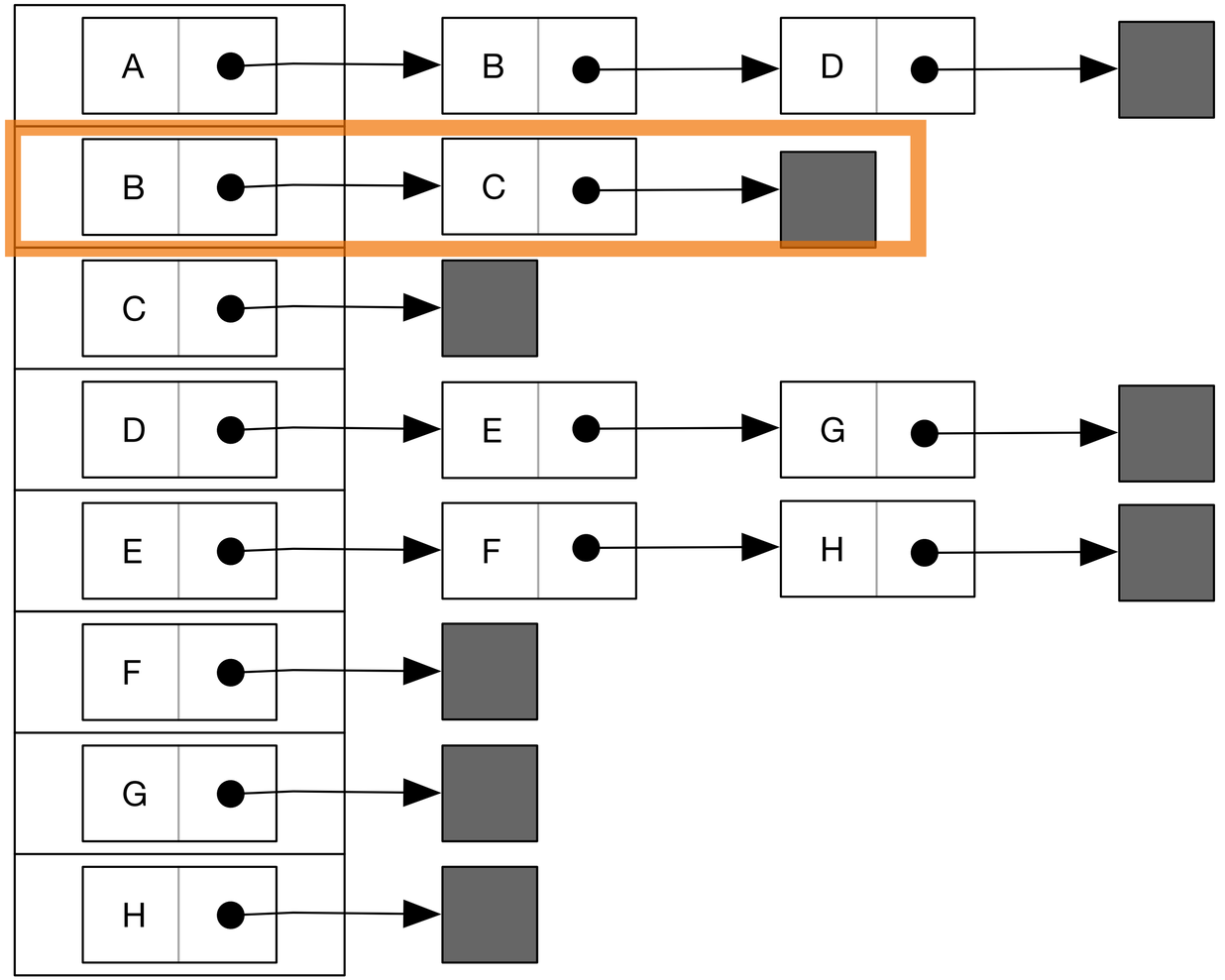
Adjacency list



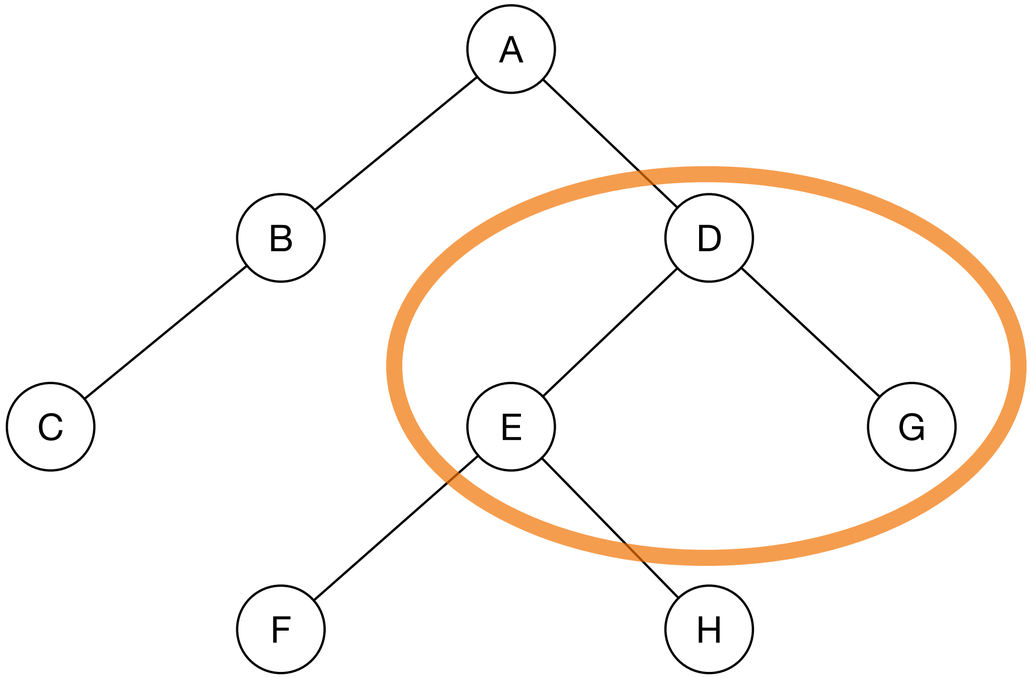


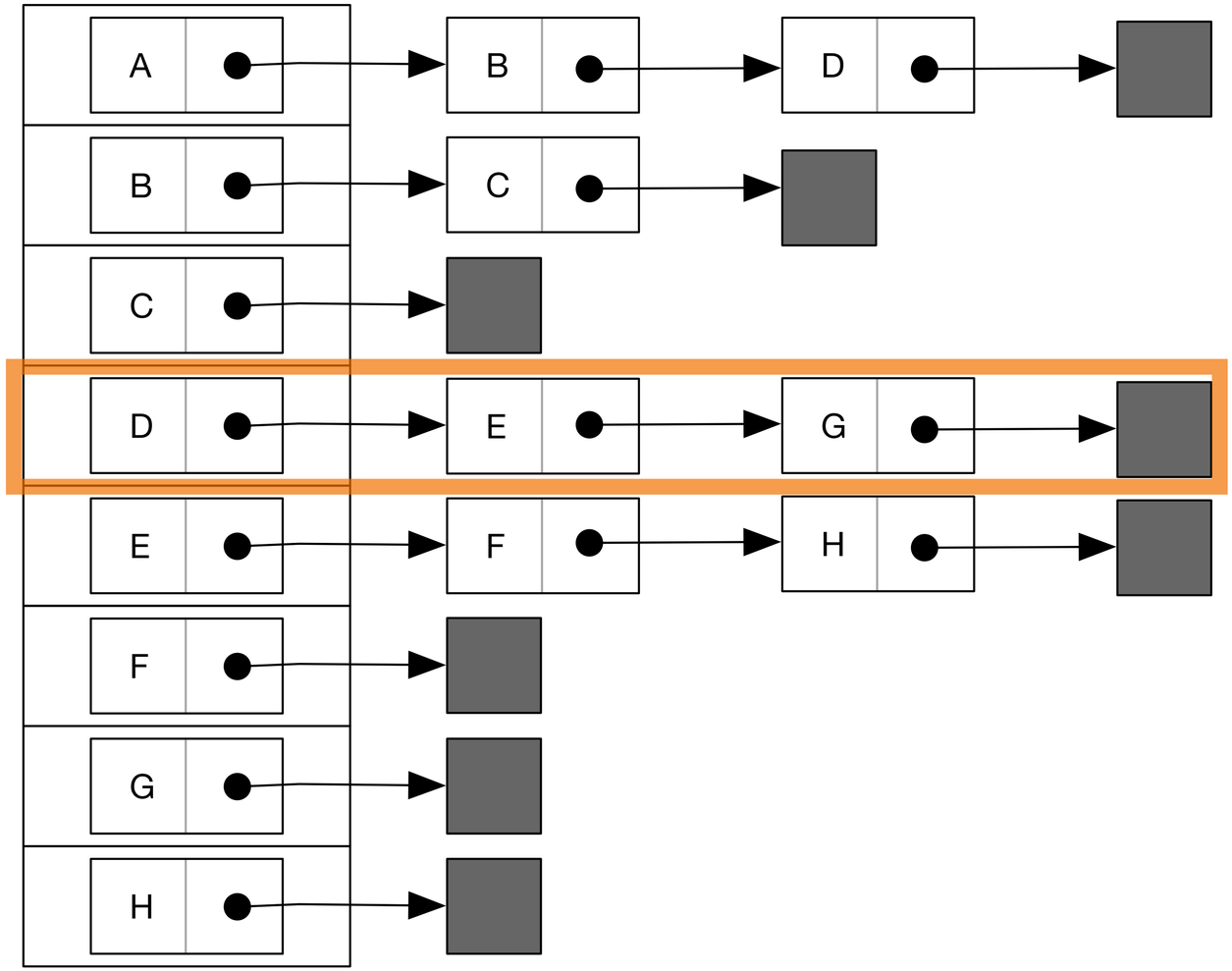
Adjacency list



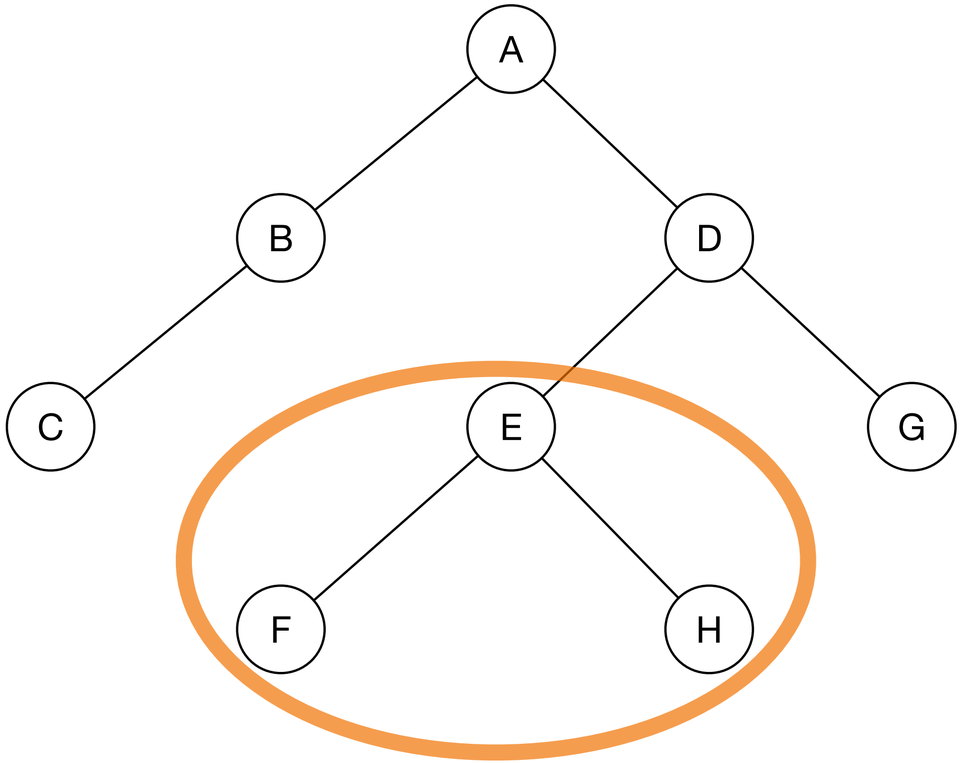


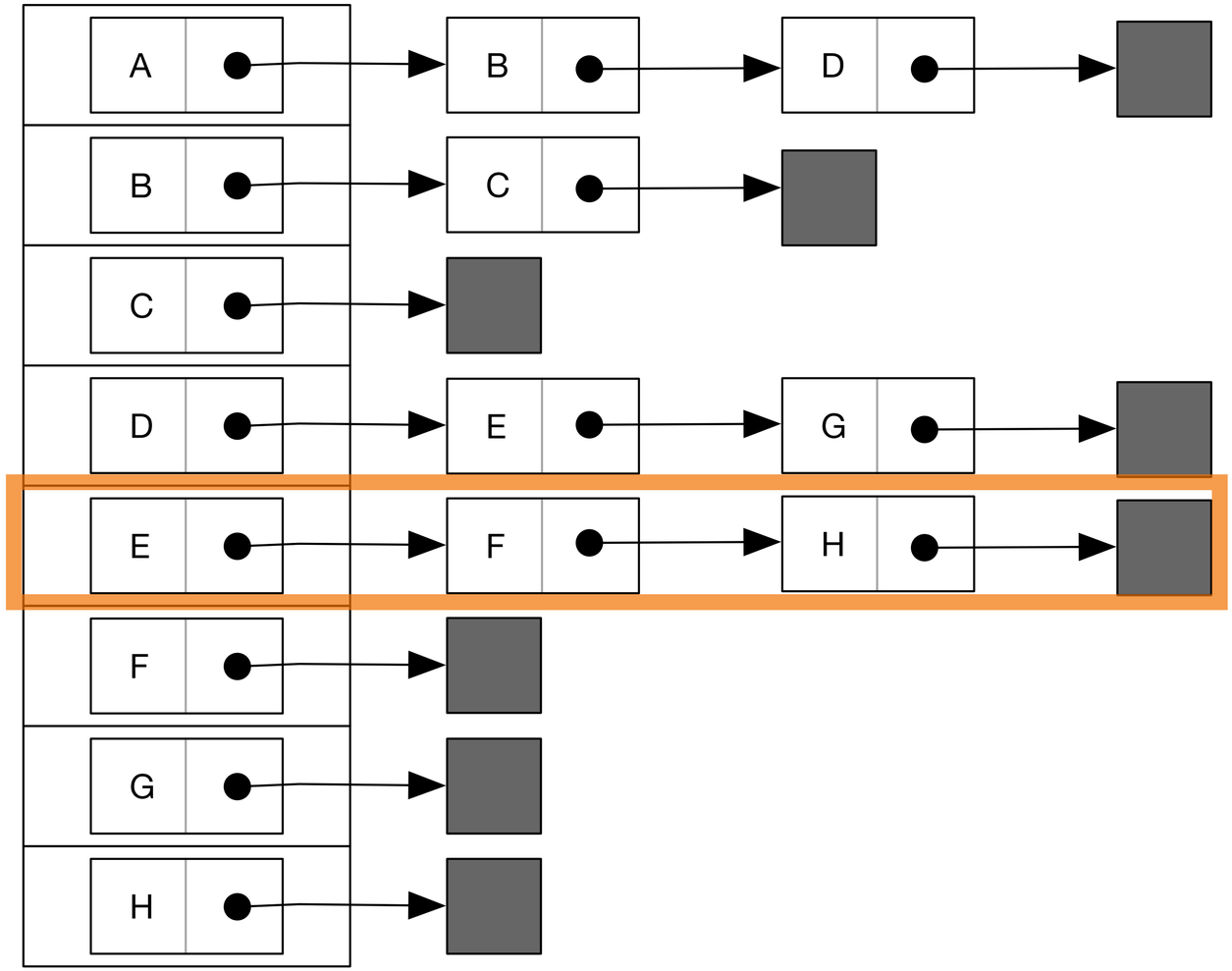
Adjacency list



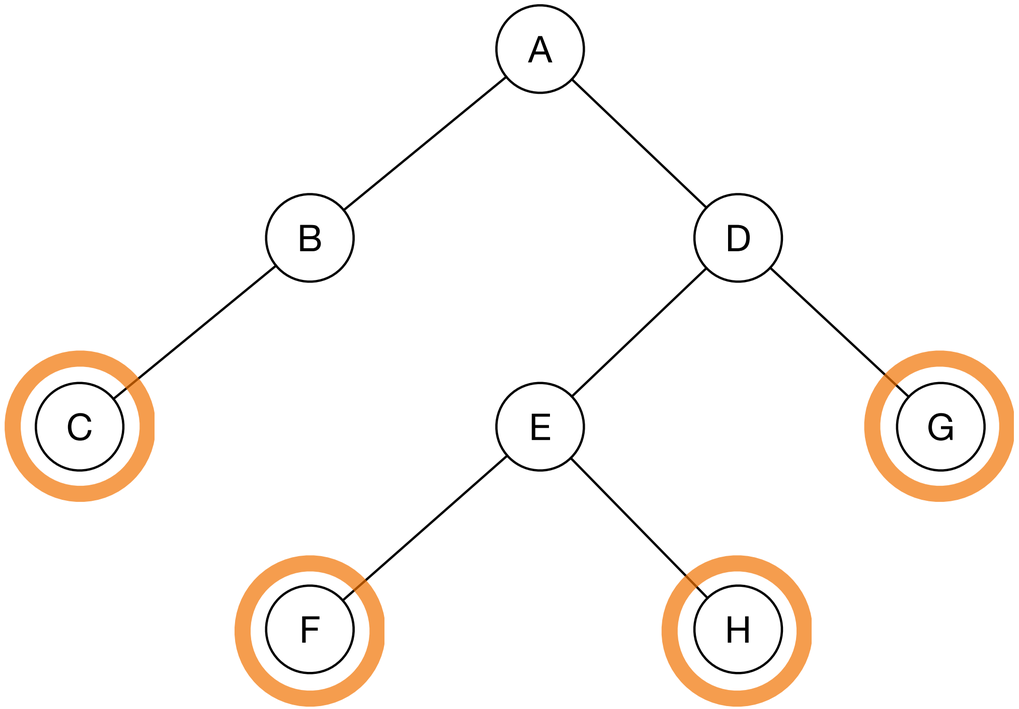


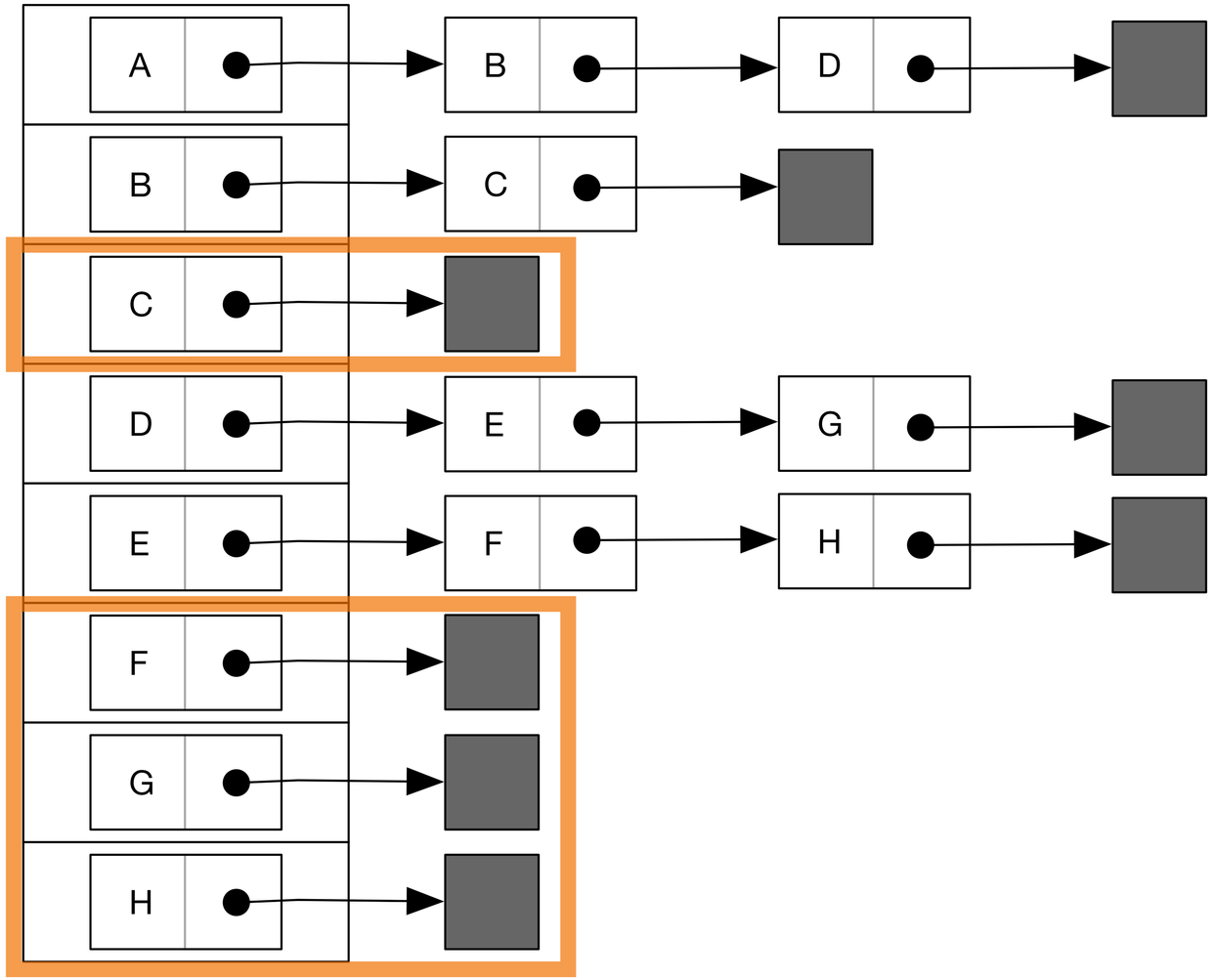
Adjacency list



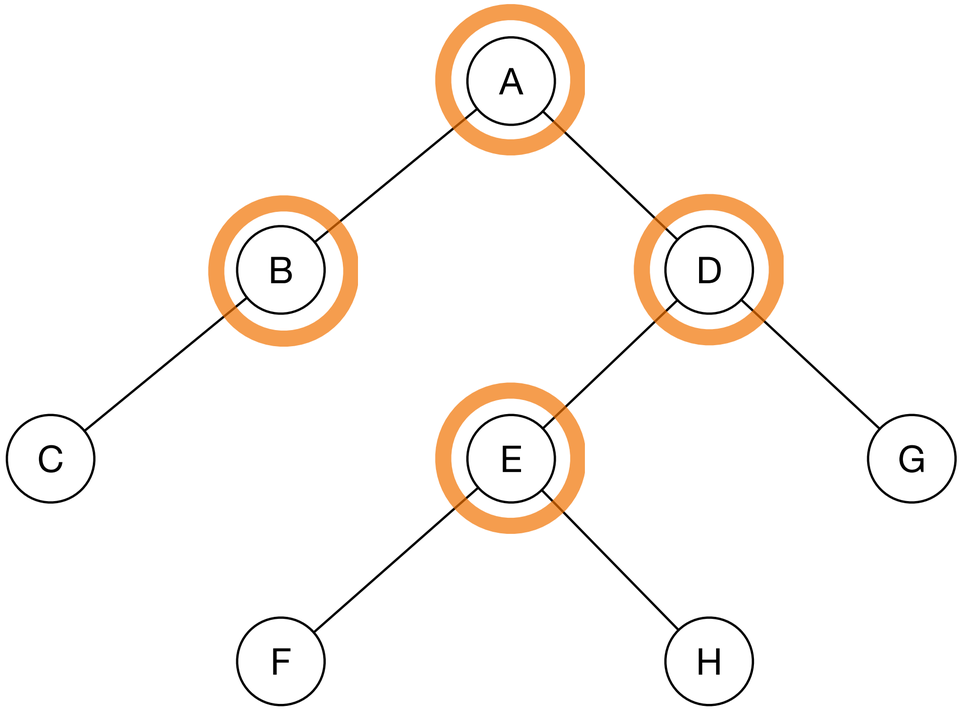


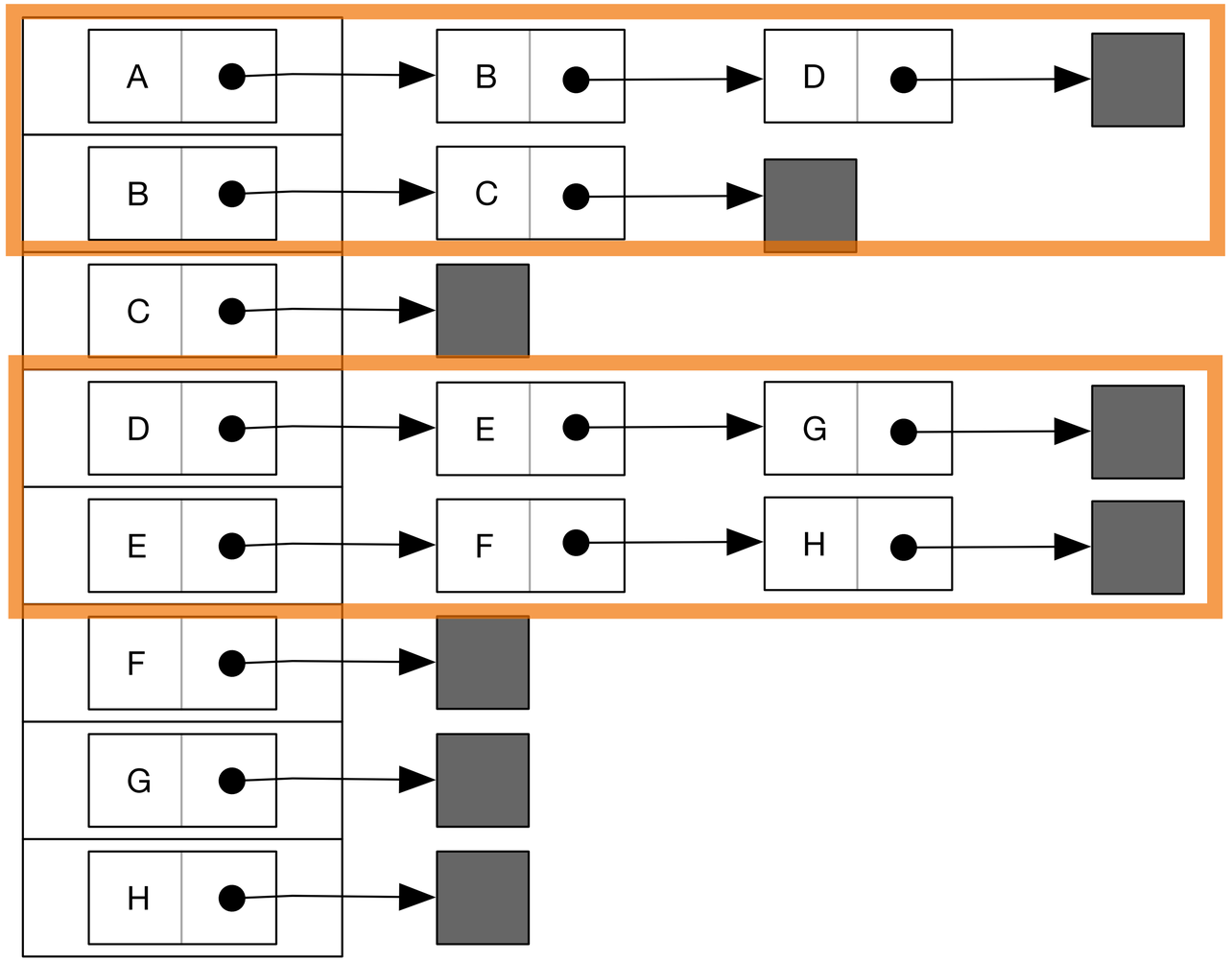
Adjacency list



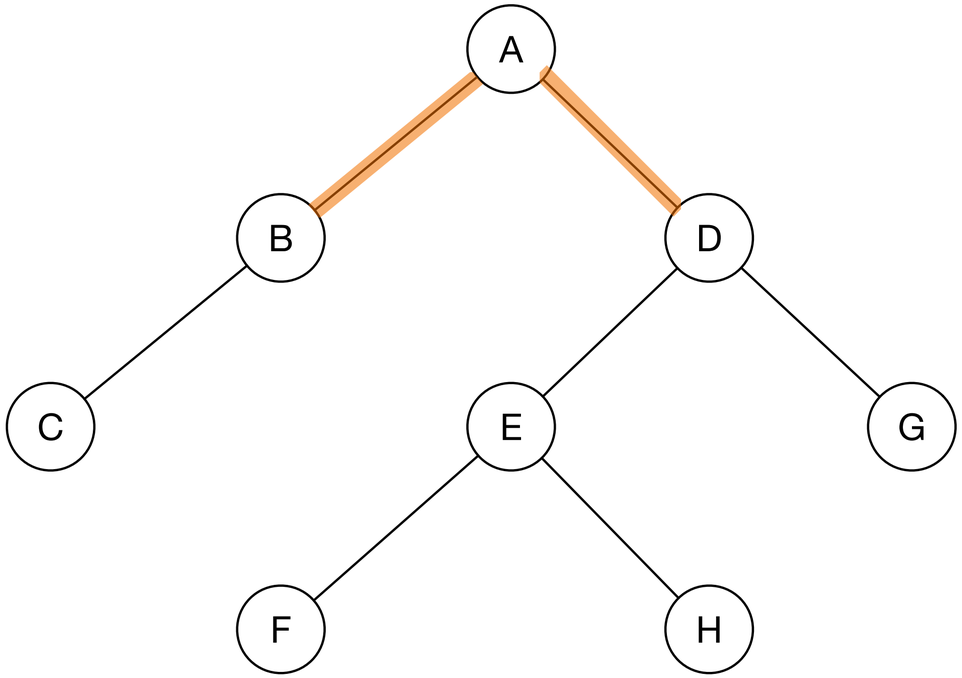


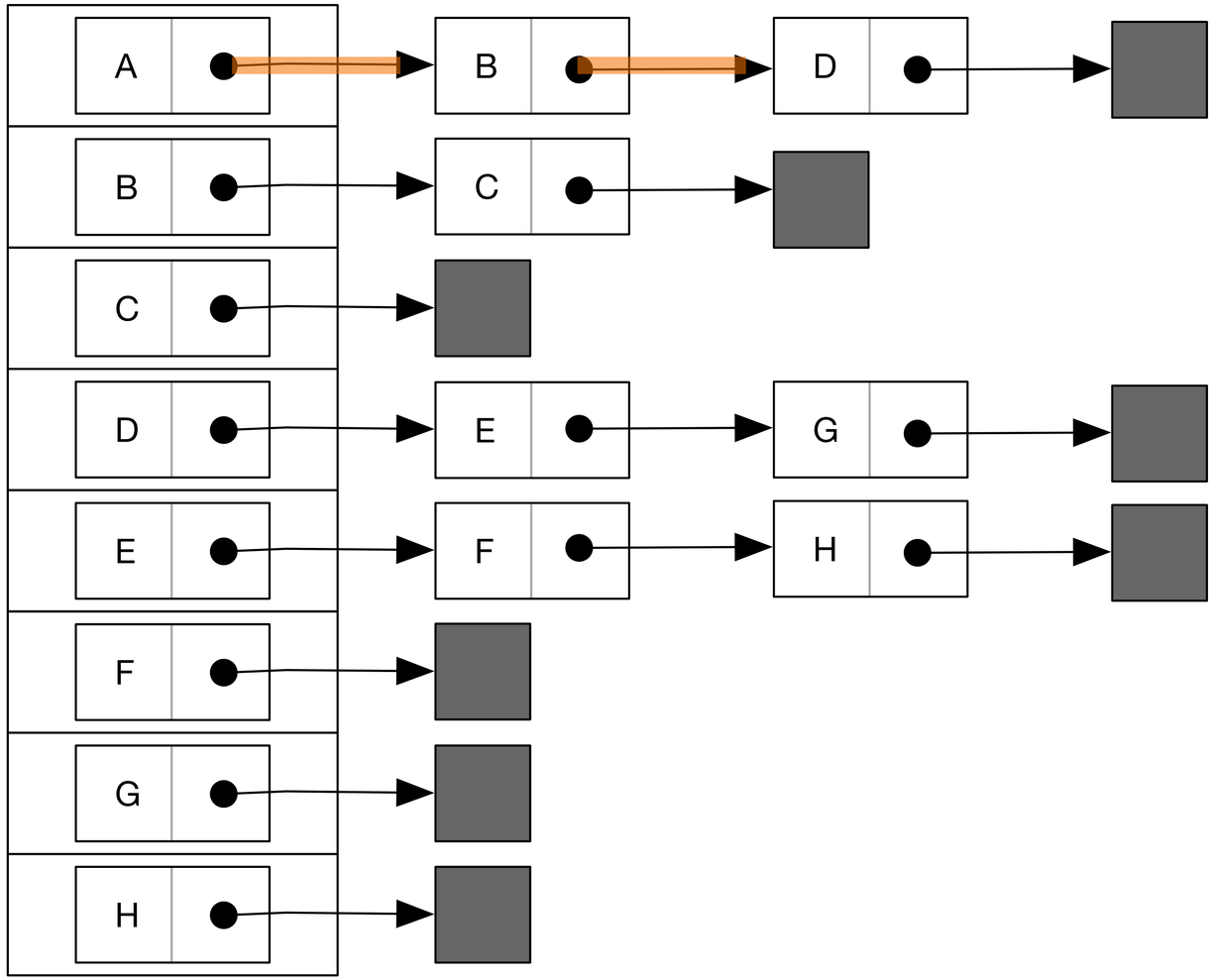
Adjacency list



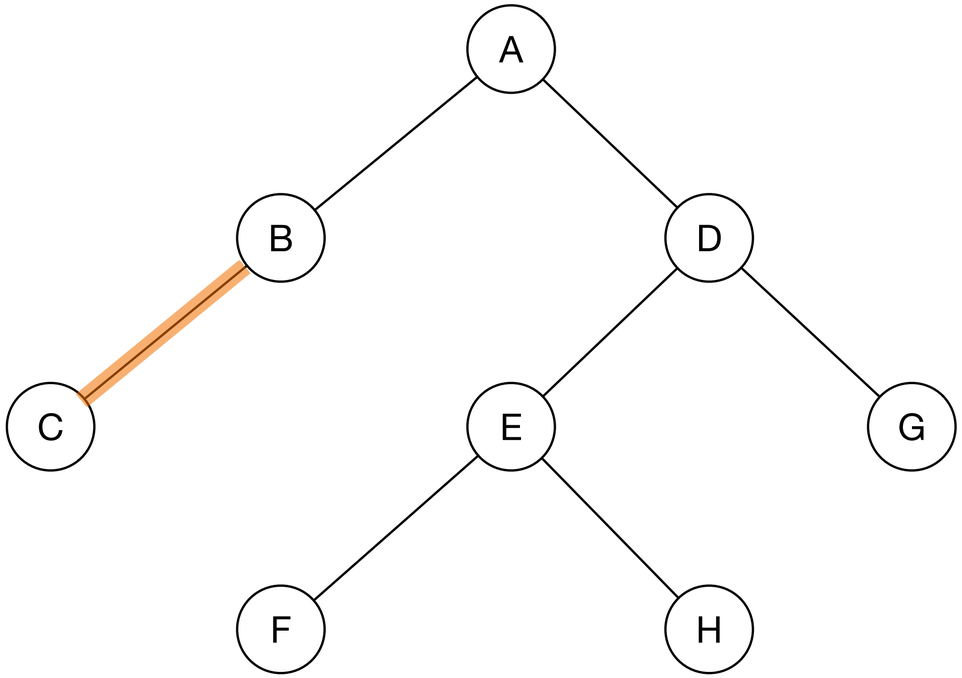


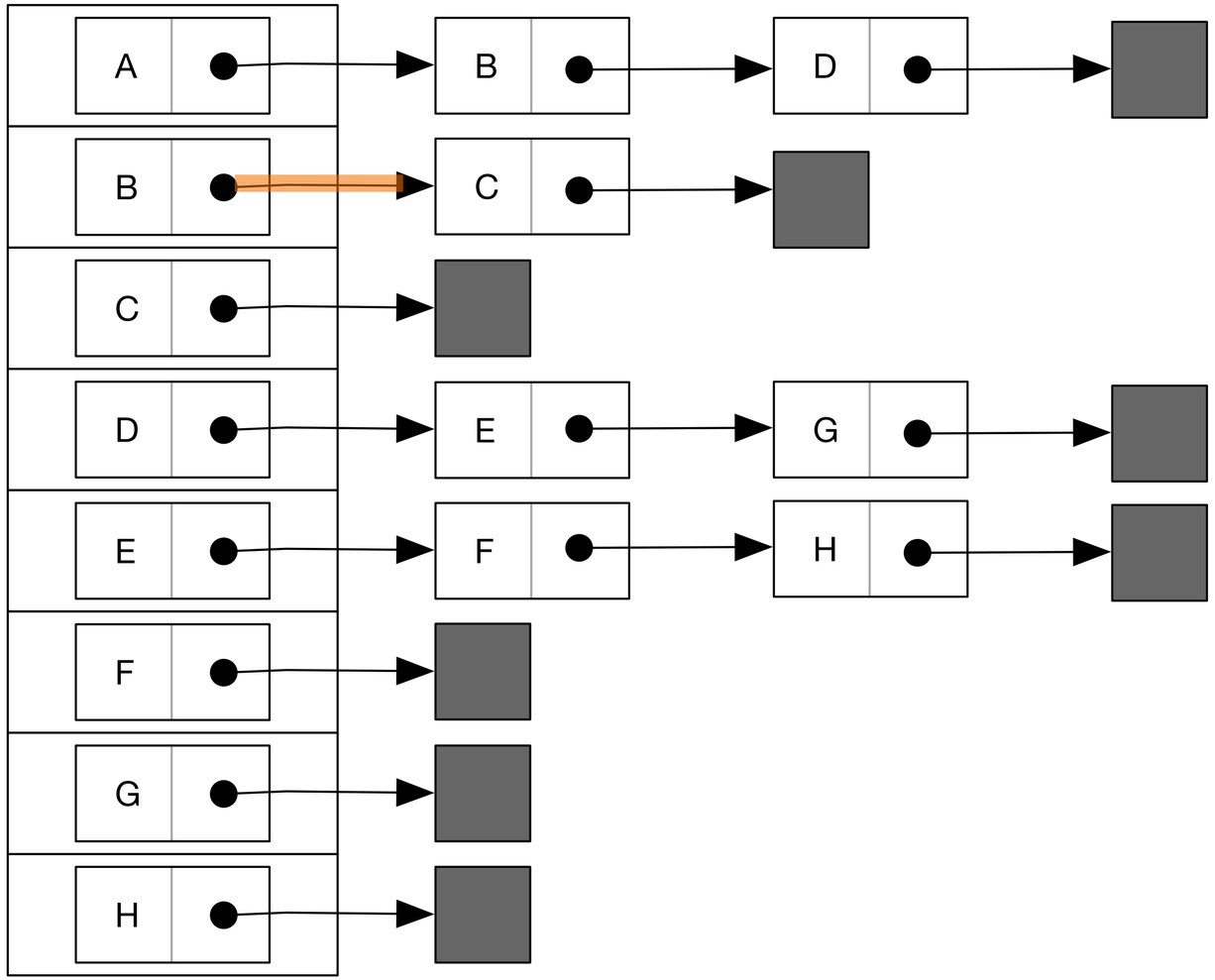
Adjacency list



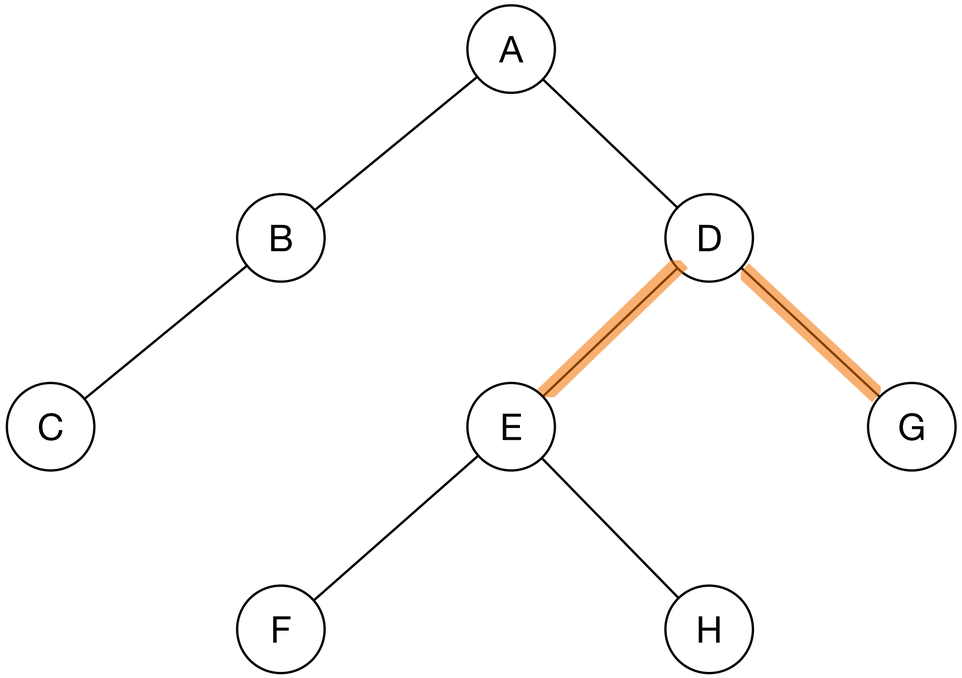


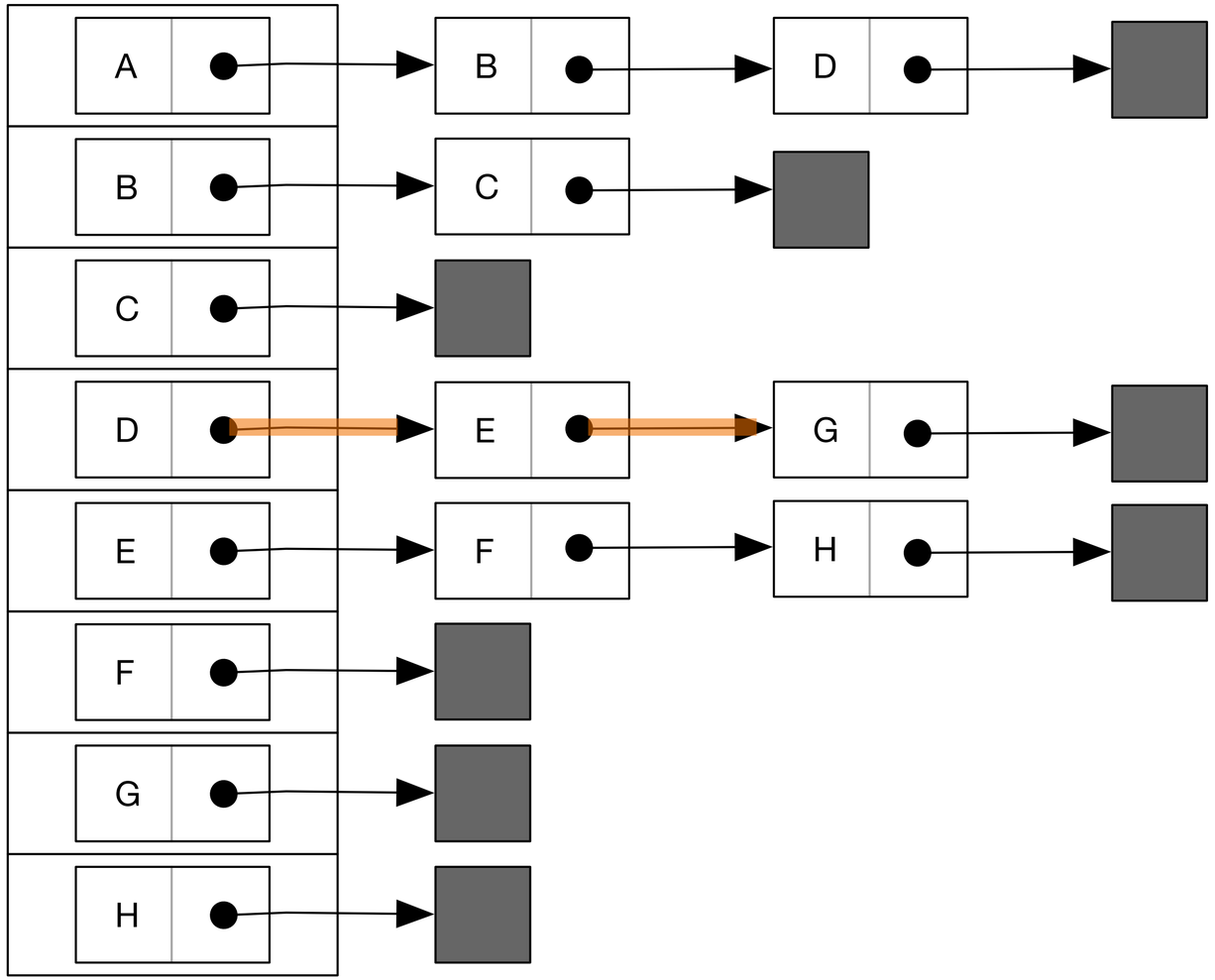
Adjacency list



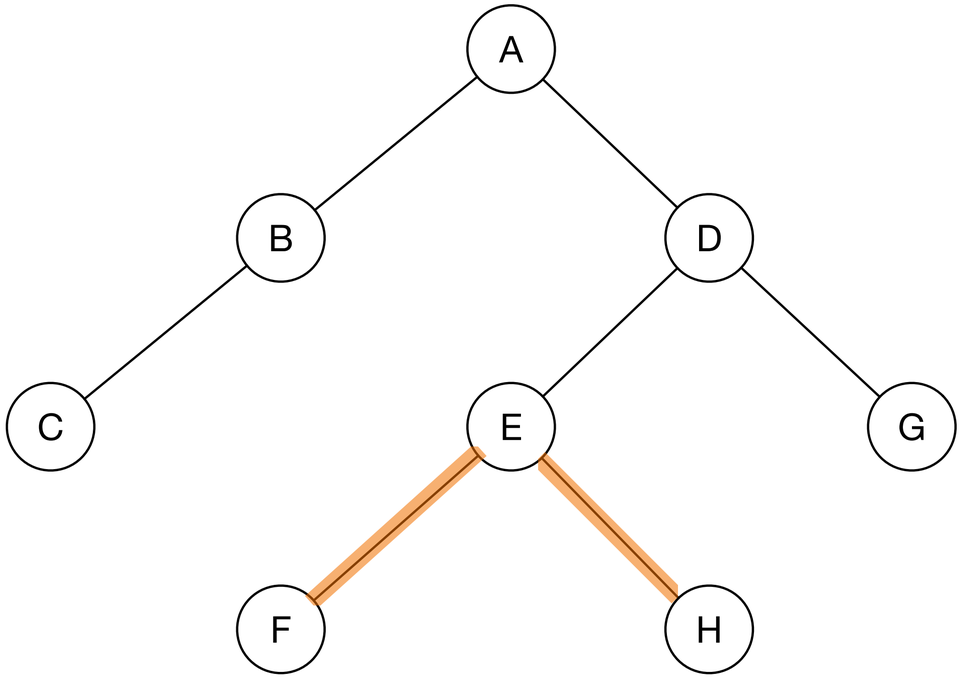


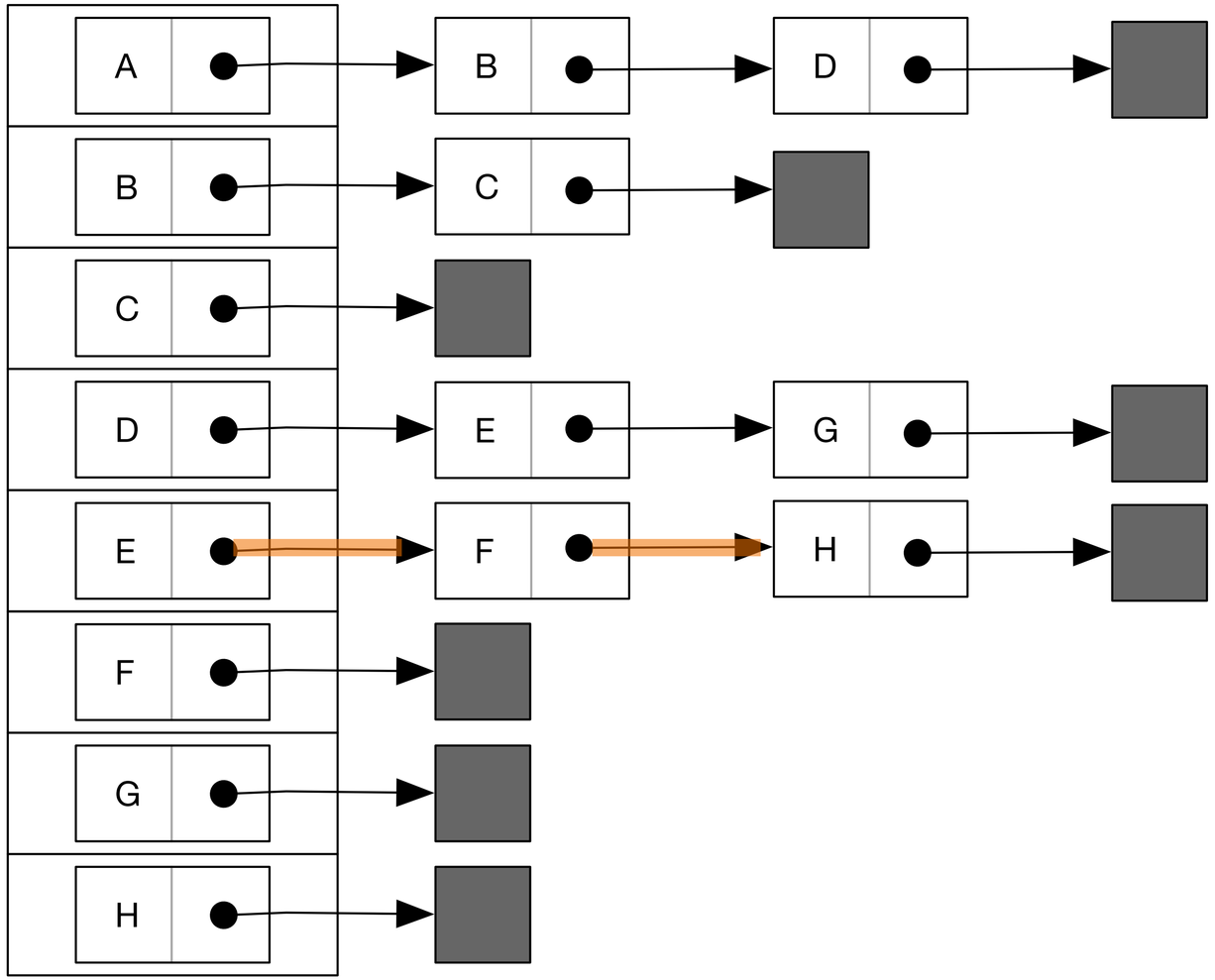
Adjacency list



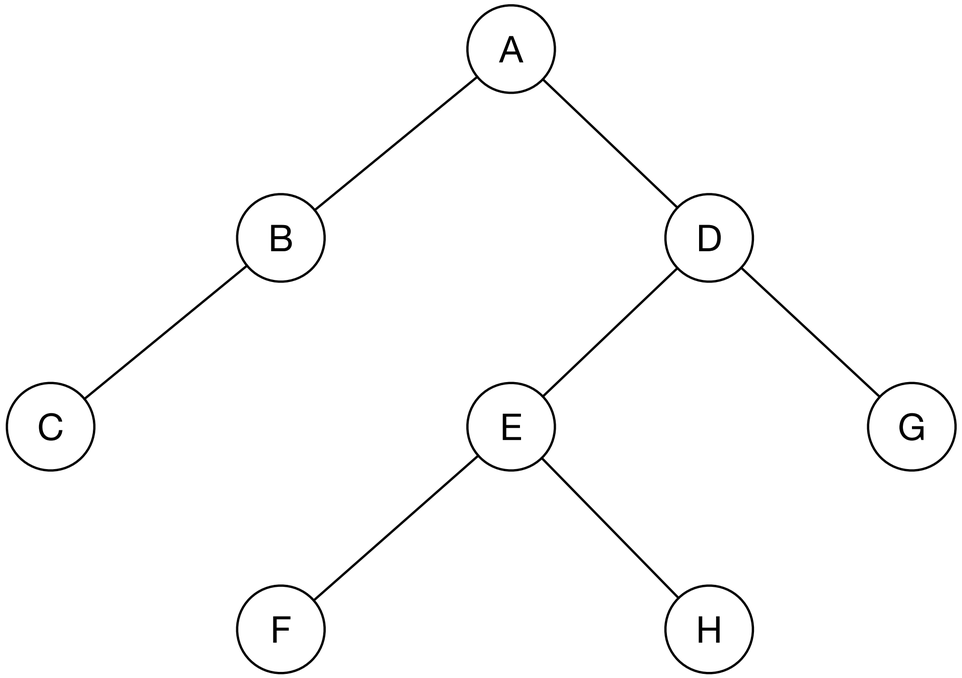


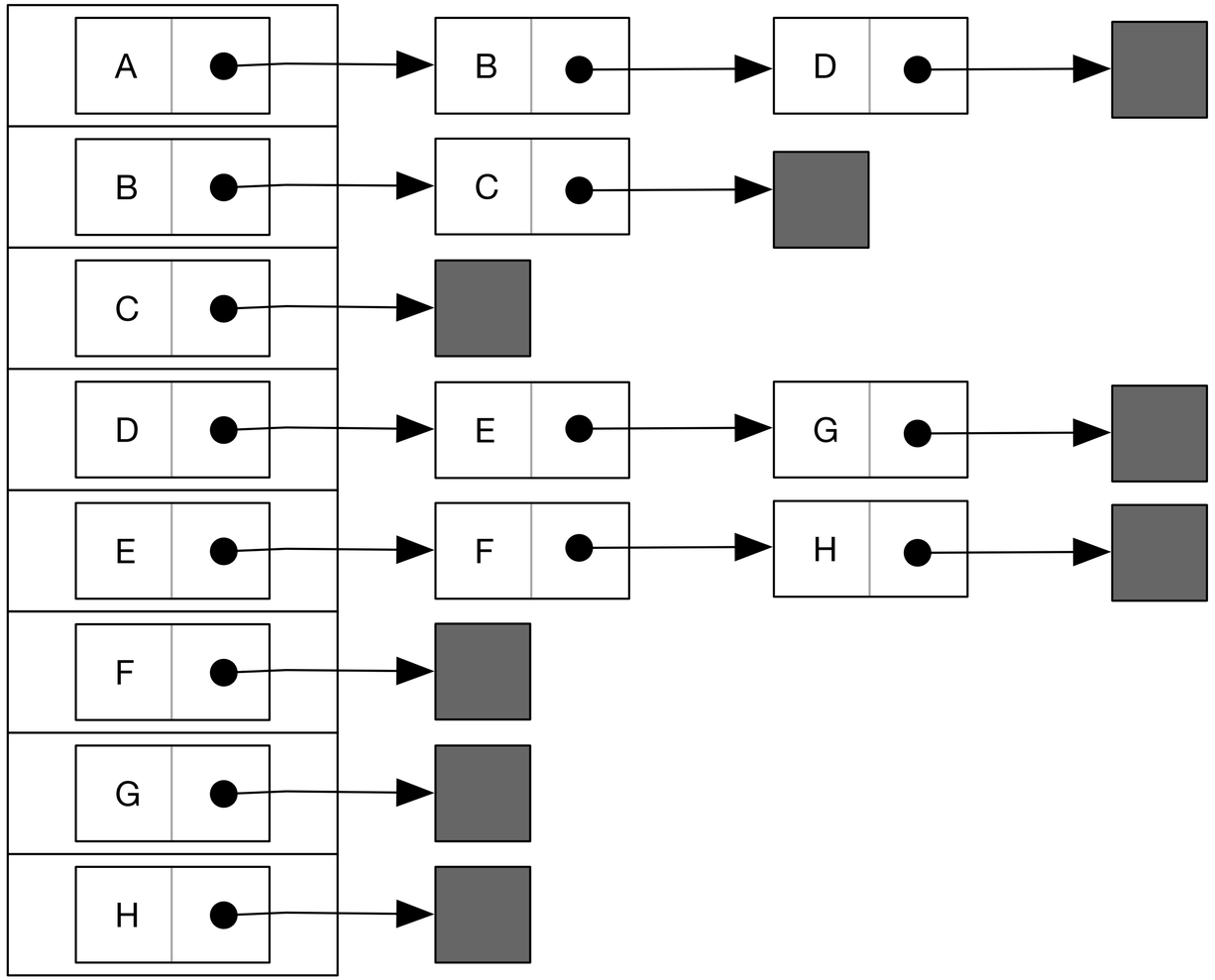
Adjacency list



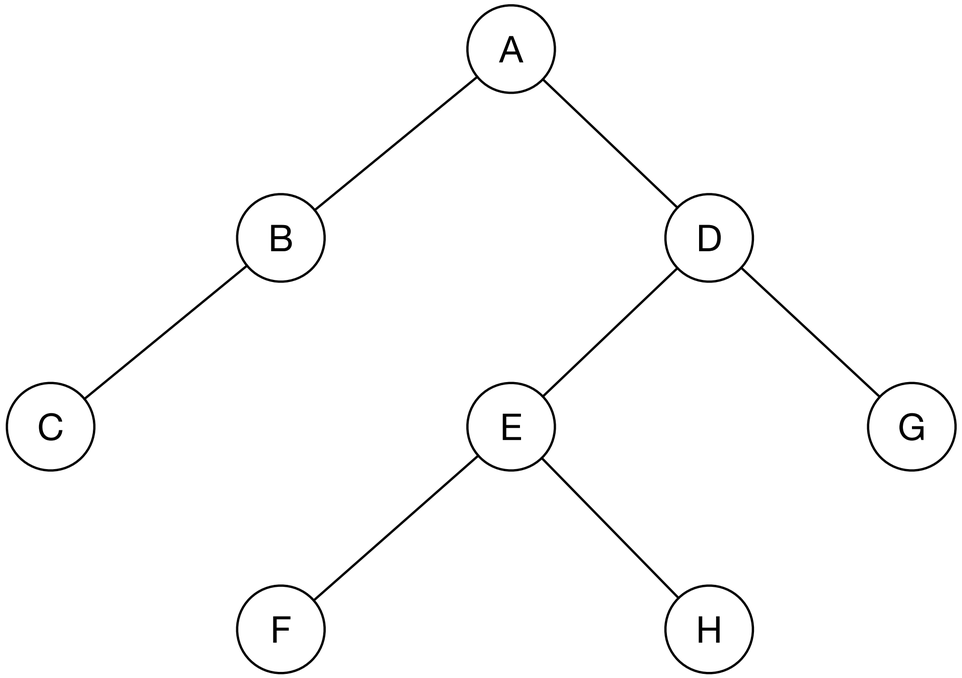


Adjacency list



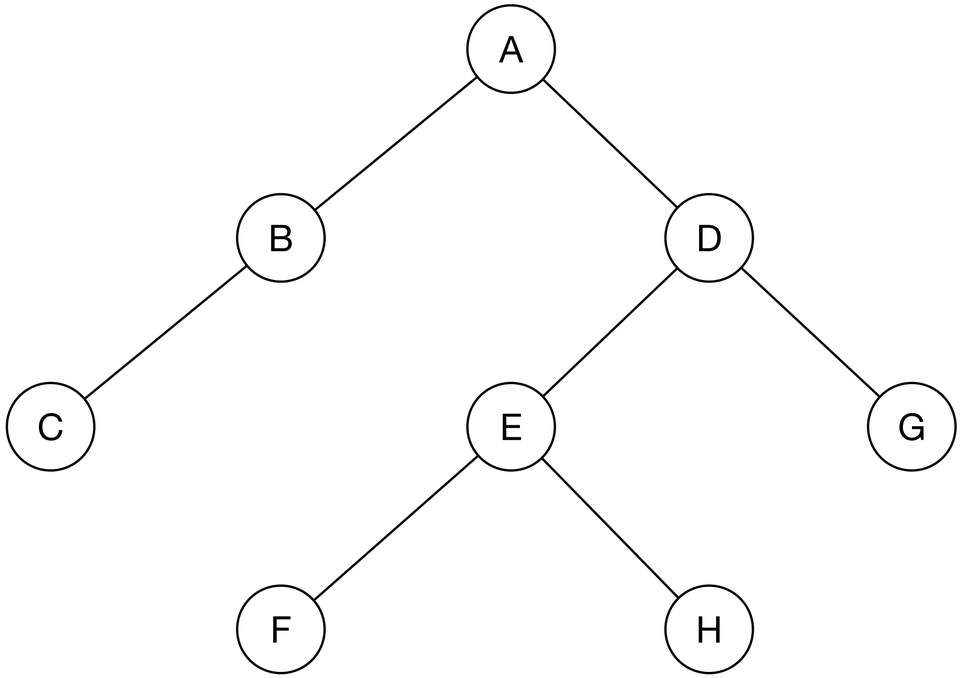


Adjacency matrix



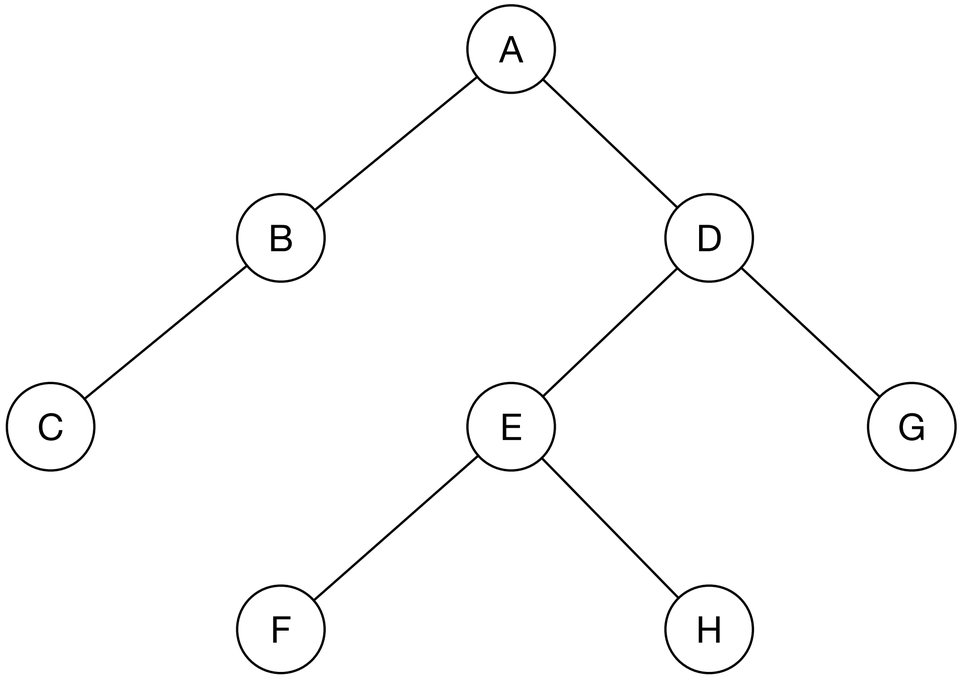
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix



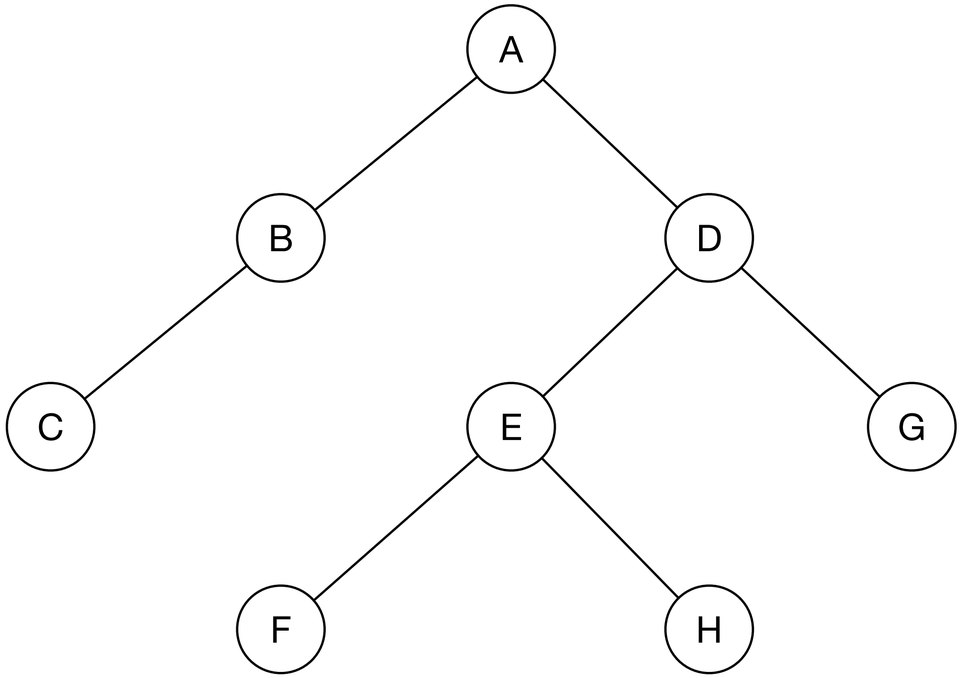
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix



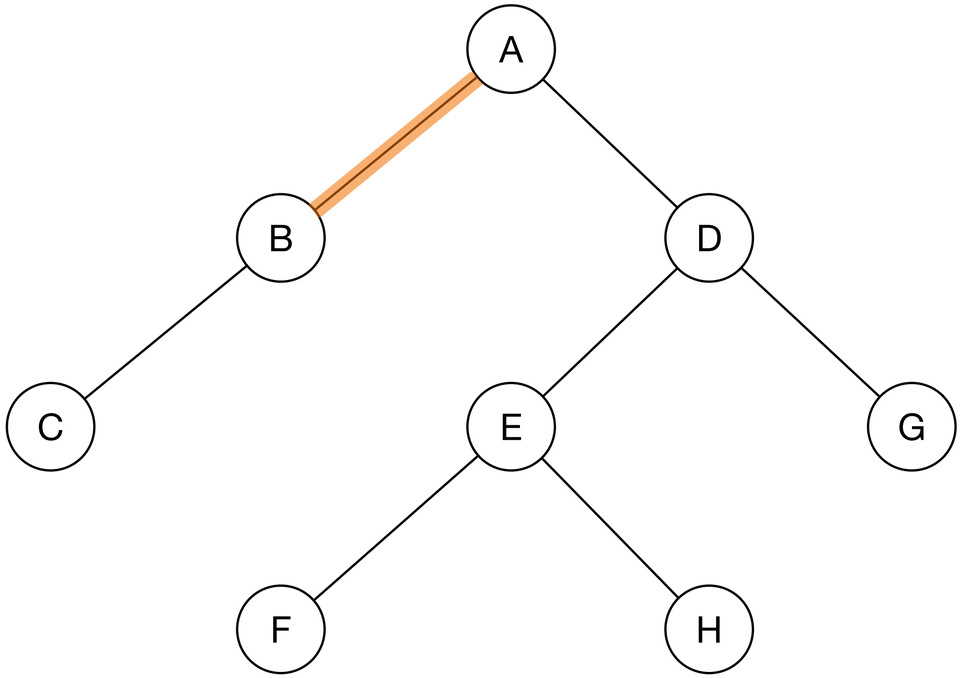
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix



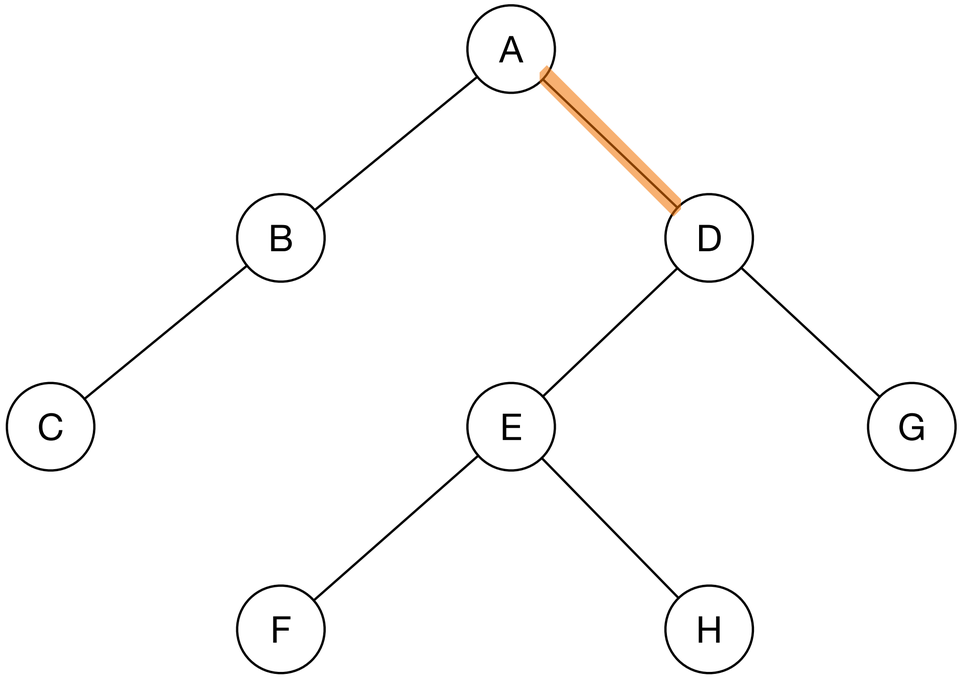
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix



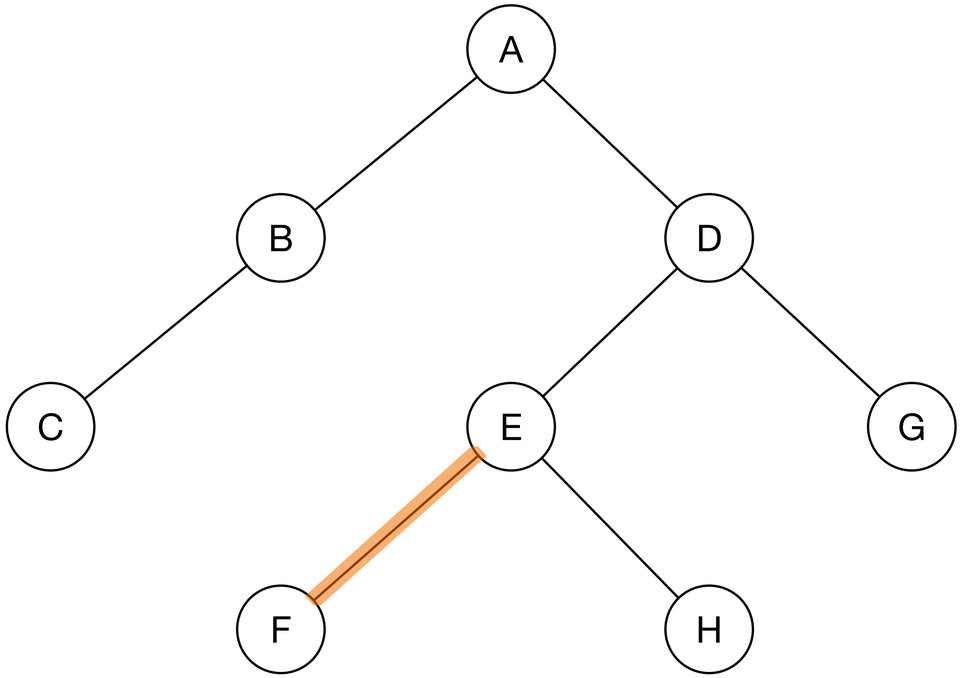
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix



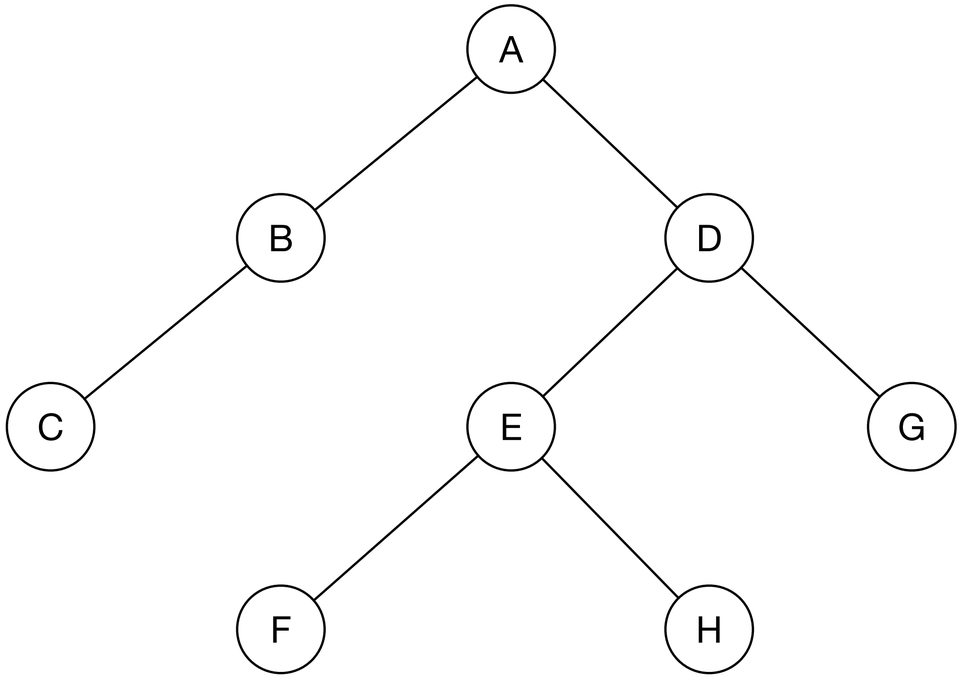
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix



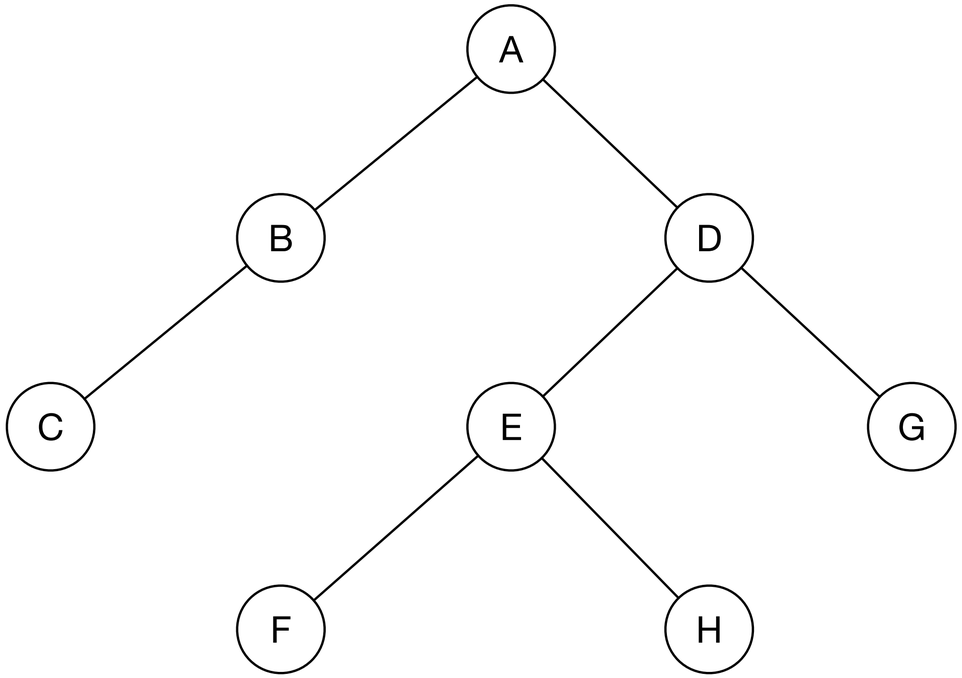
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix



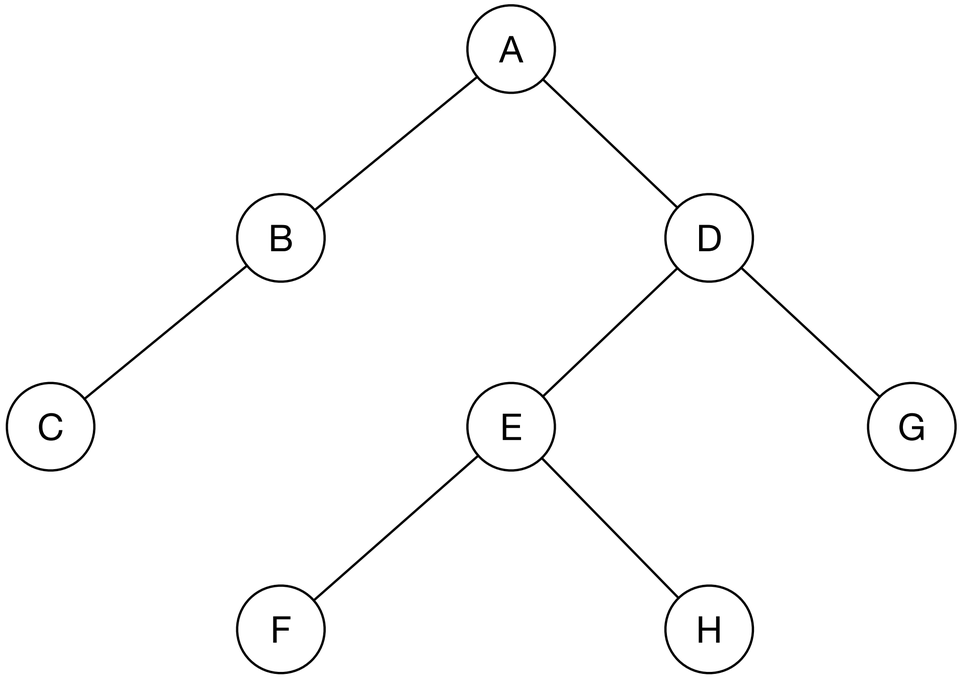
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix



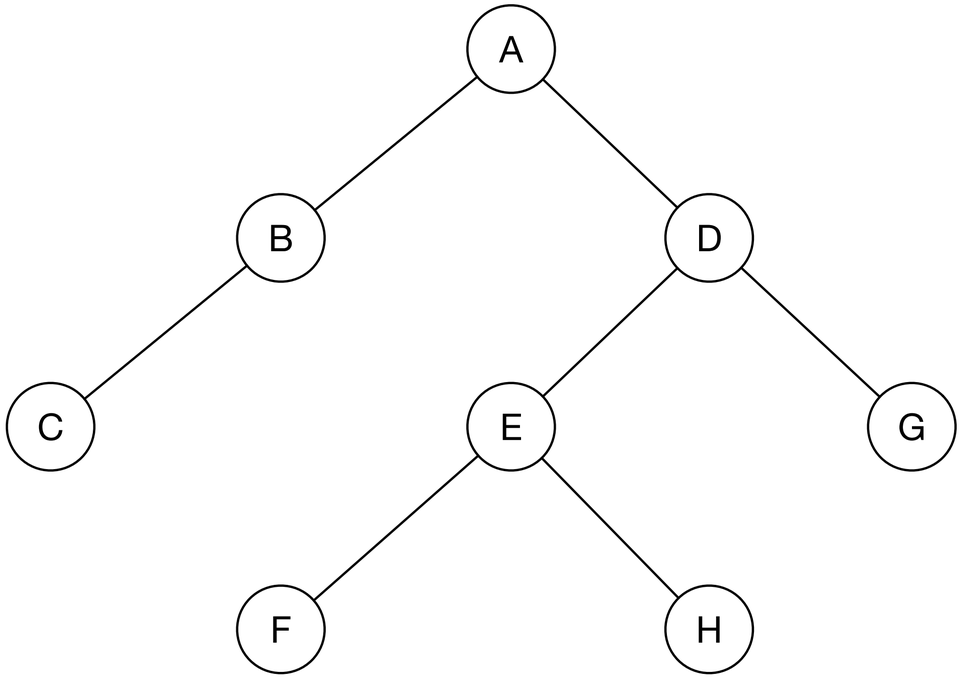
	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix



	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency matrix

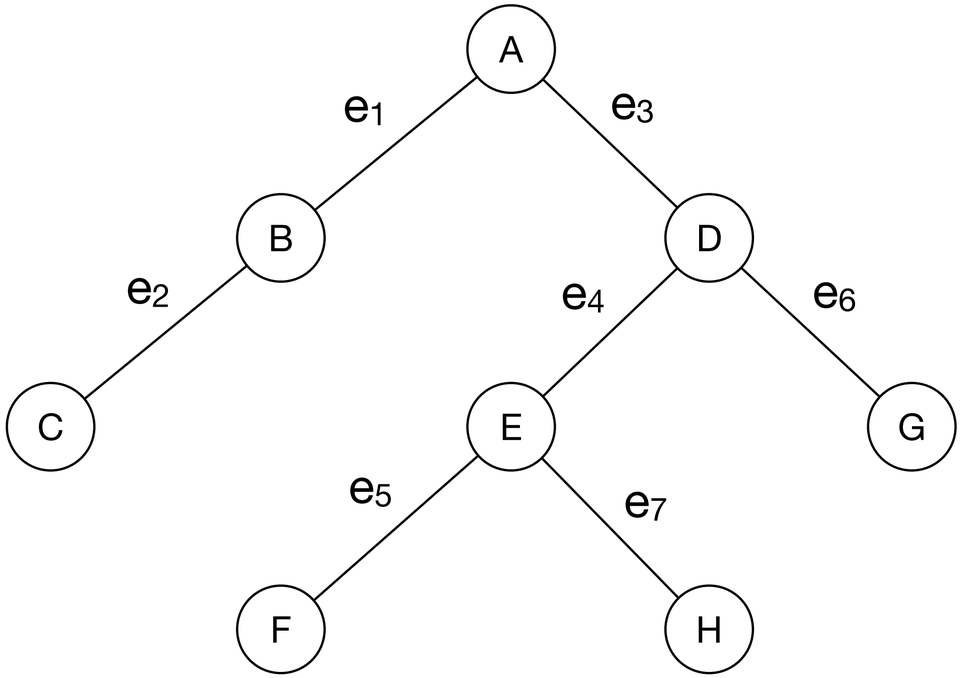


	A	B	C	D	E	F	G	H
A	0	1	0	1	0	0	0	0
B	1	0	1	0	0	0	0	0
C	0	1	0	0	0	0	0	0
D	1	0	0	0	1	0	1	0
E	0	0	0	1	0	1	0	1
F	0	0	0	0	1	0	0	0
G	0	0	0	1	0	0	0	0
H	0	0	0	0	1	0	0	0

Adjacency List vs Adjacency Matrix

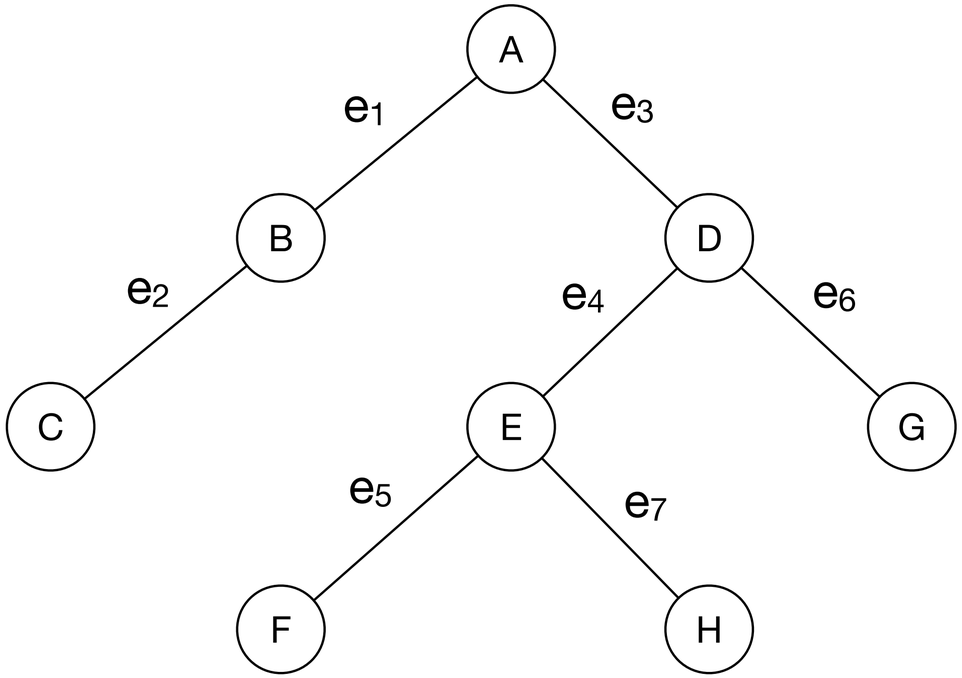
Adjacency list	Adjacency matrix
Good for sparse graphs	Good for dense graphs
Only encodes edges present	Encodes present *and* absent edges
Takes time to traverse list. If d is the degree of a node, this will be $O(d_{\max})$	Can check if two nodes are adjacent in constant time $O(1)$
Easier to find all adjacent nodes	Must scan entire row or column to find all adjacent nodes, $O(N)$
	Can do cool stuff with matrix calculations (spectral graph theory, an advanced topic)

Incidence matrix



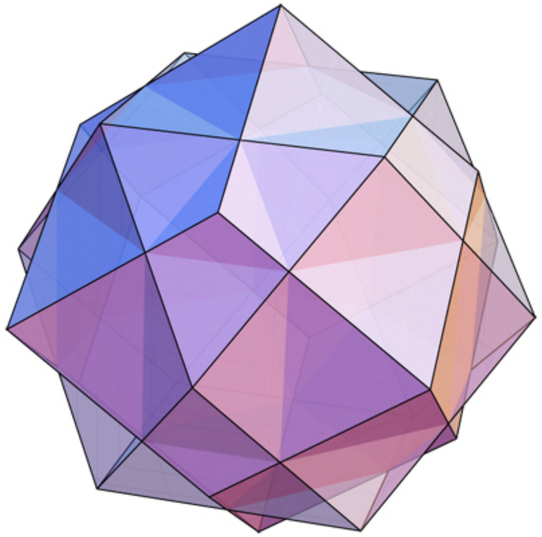
	e_1	e_2	e_3	e_4	e_5	e_6	e_7
A	1	0	1	0	0	0	0
B	1	1	0	0	0	0	0
C	0	1	0	0	0	0	0
D	0	0	1	1	0	1	0
E	0	0	0	1	1	0	1
F	0	0	0	0	1	0	0
G	0	0	0	0	0	1	0
H	0	0	0	0	1	0	1

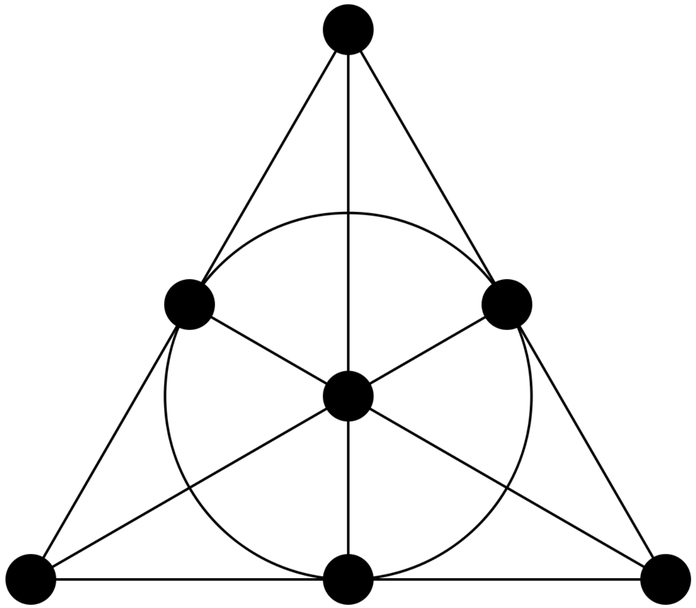
Incidence matrix

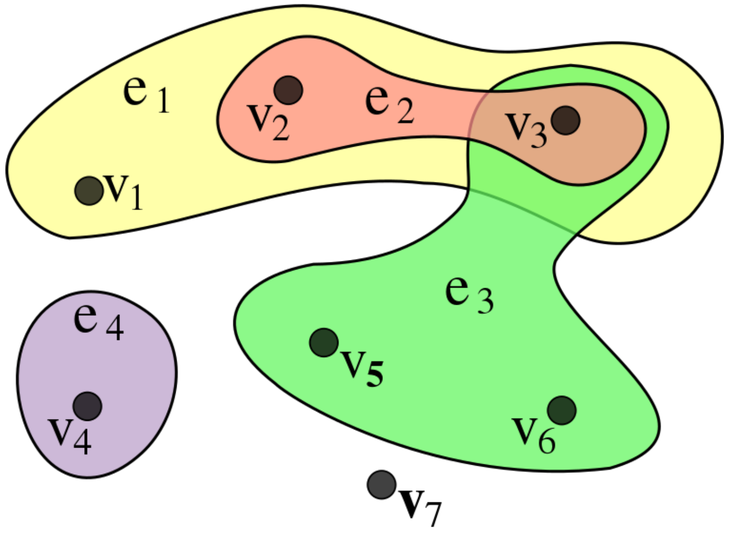


	e_1	e_2	e_3	e_4	e_5	e_6	e_7
A	1	0	1	0	0	0	0
B	1	1	0	0	0	0	0
C	0	1	0	0	0	0	0
D	0	0	1	1	0	1	0
E	0	0	0	1	1	0	1
F	0	0	0	0	1	0	0
G	0	0	0	0	0	1	0
H	0	0	0	0	1	0	1

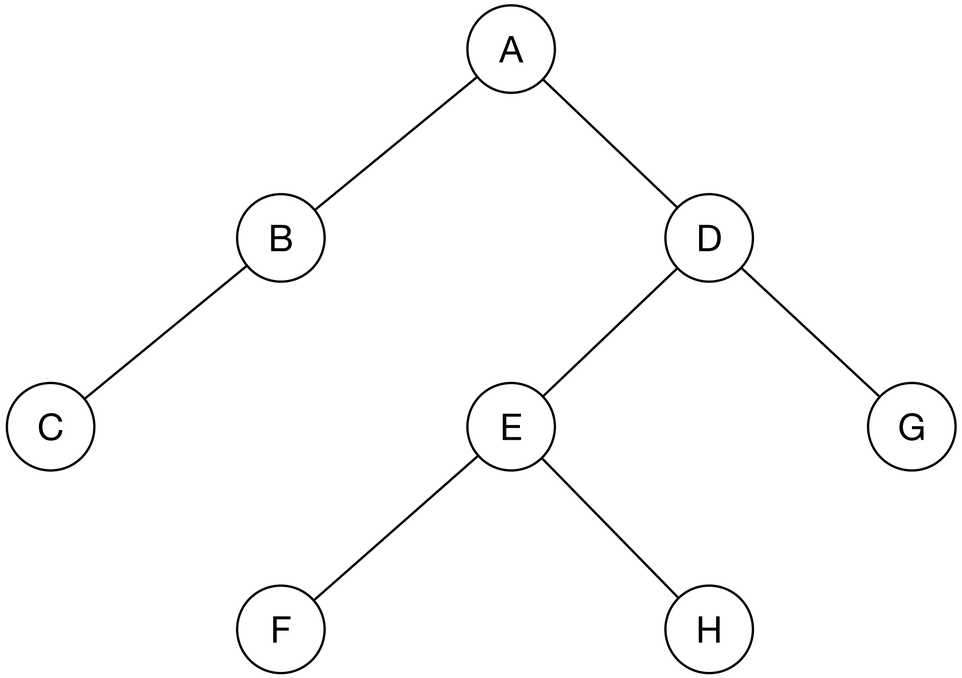
Incidence matrix





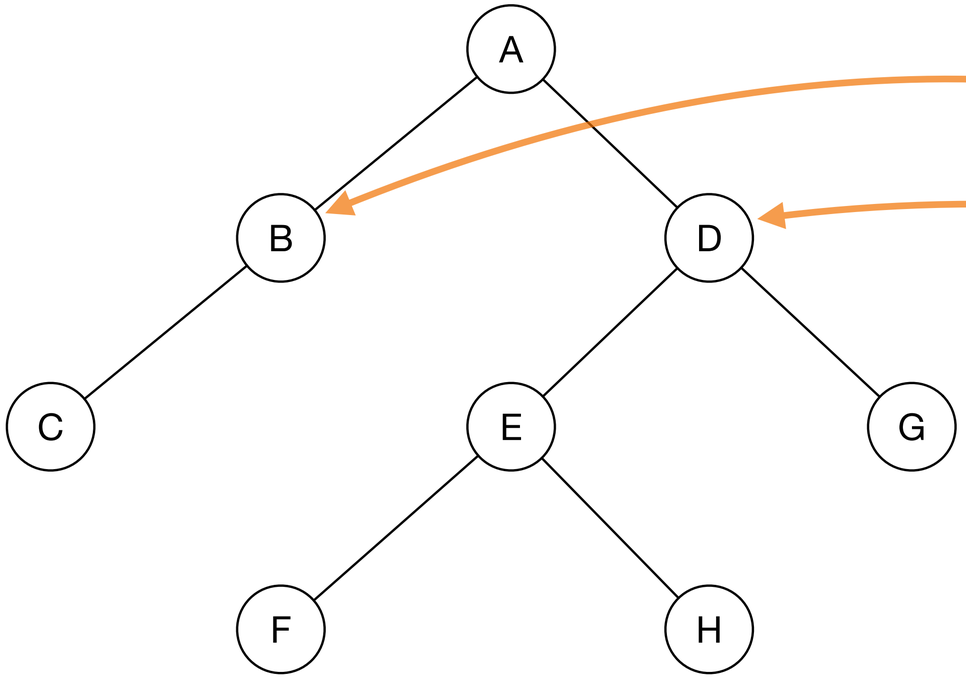


Simple L/R Object-oriented Approach




```
class Node {
private:
    Node *leftChild;
    Node *rightChild;
...
}
```

Simple L/R Object-oriented Approach

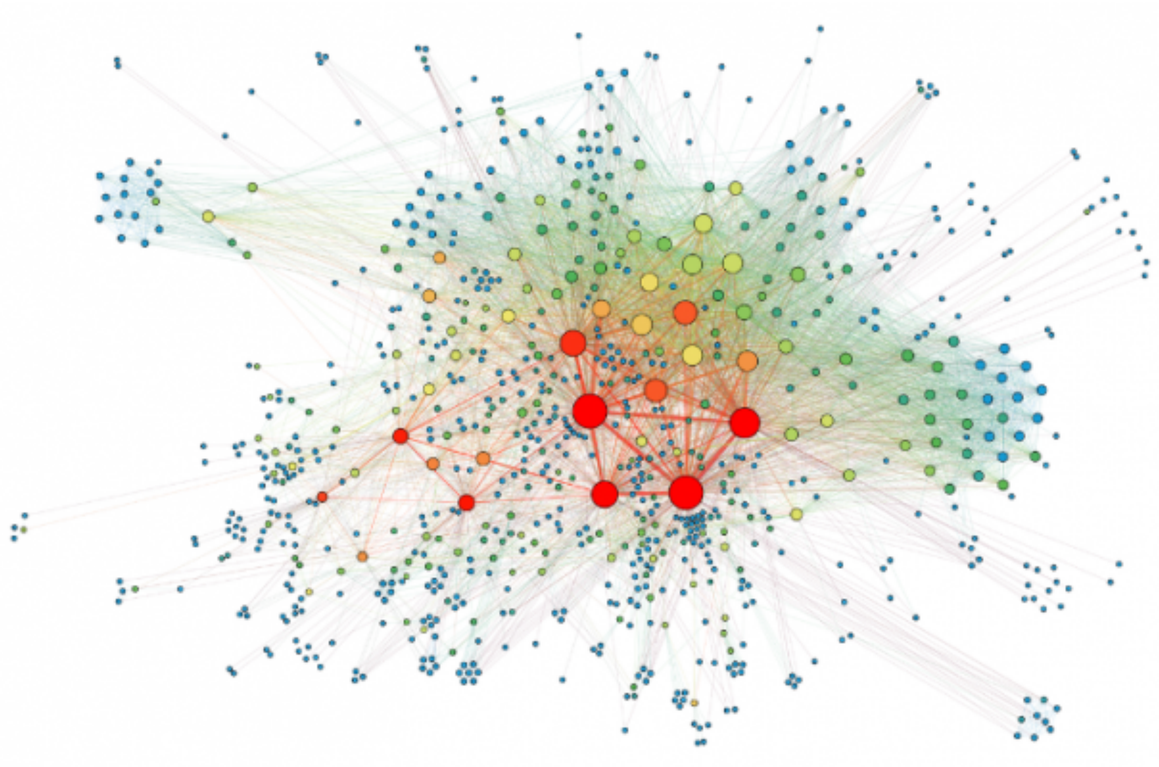


A

```
class Node {
private:
    Node *leftChild;
    Node *rightChild;

...
}
```

Simple L/R Object-oriented Approach



```
class Node {
private
    Node *leftChild;
    Node *rightChild;
...
}
```

How do we represent a tree?

Choosing a data structure

- Adjacency list - sparse objects
- Adjacency matrix - dense objects
- Simple L/R object-oriented approach - binary trees