Union Find!

(Disjoint Set)

William Fiset

Outline

- **Discussion & Examples**
  - What is Union Find?
  - Magnets example
  - When and where is a Union Find used?
  - Kruskal’s minimum spanning tree algorithm
  - Complexity analysis
- **Implementation Details**
  - Find & Union operations
  - Path compression
- **Code Implementation**

Discussion and Examples

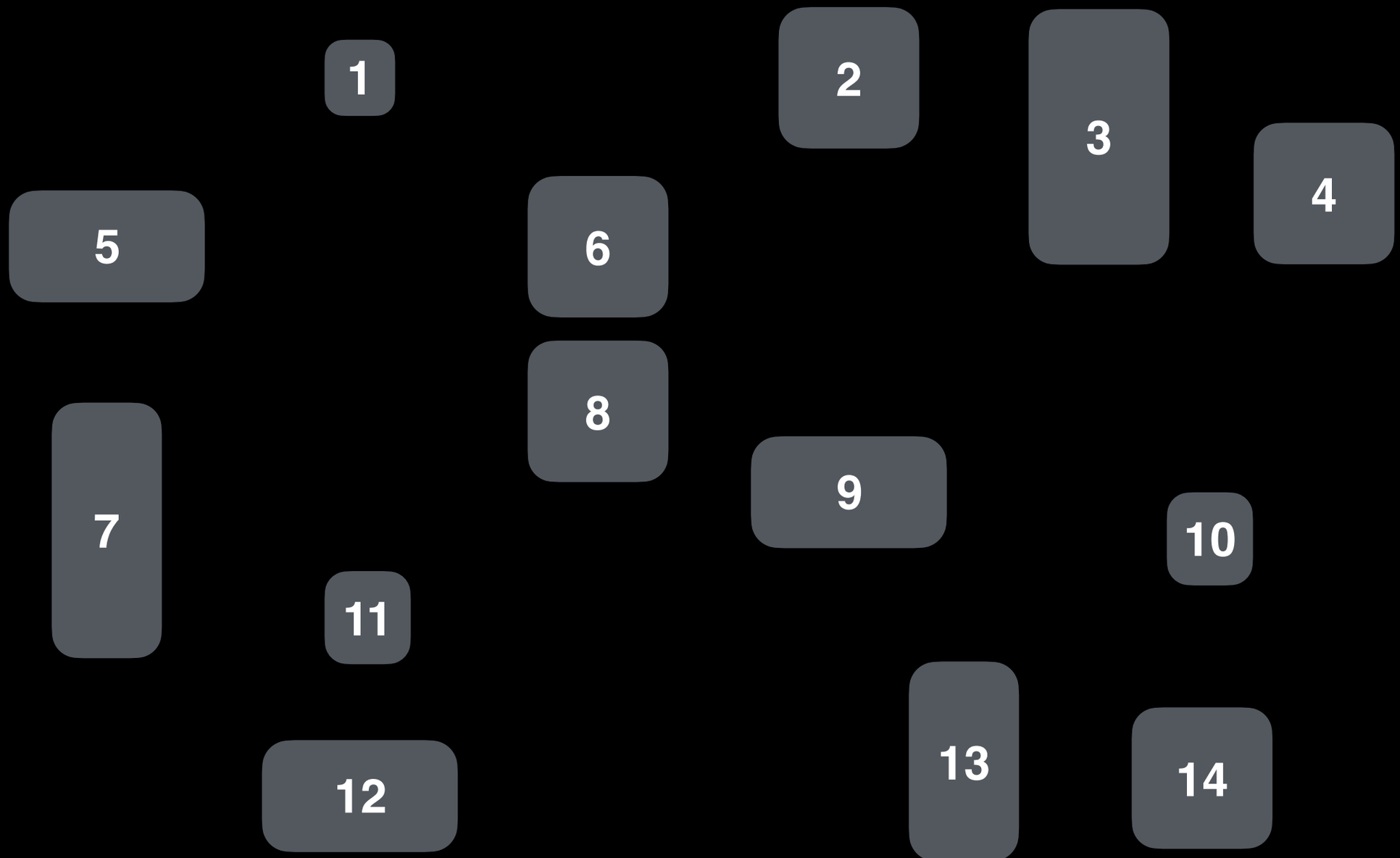
What is Union Find?

Union Find is a data structure that keeps track of elements which are split into one or more disjoint sets. Its has two primary operations: ***find*** and ***union***.

Union Find Magnets Example

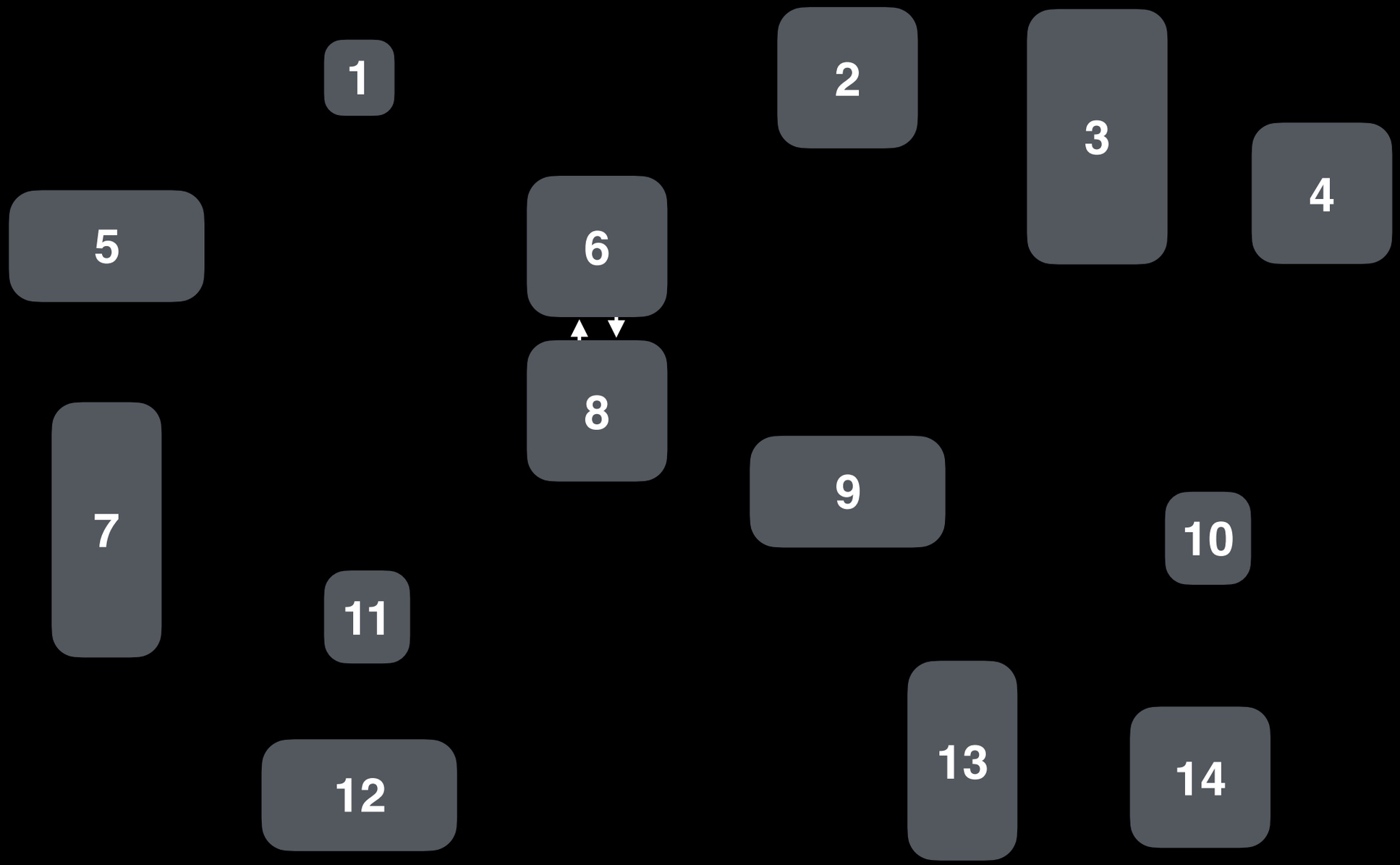
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



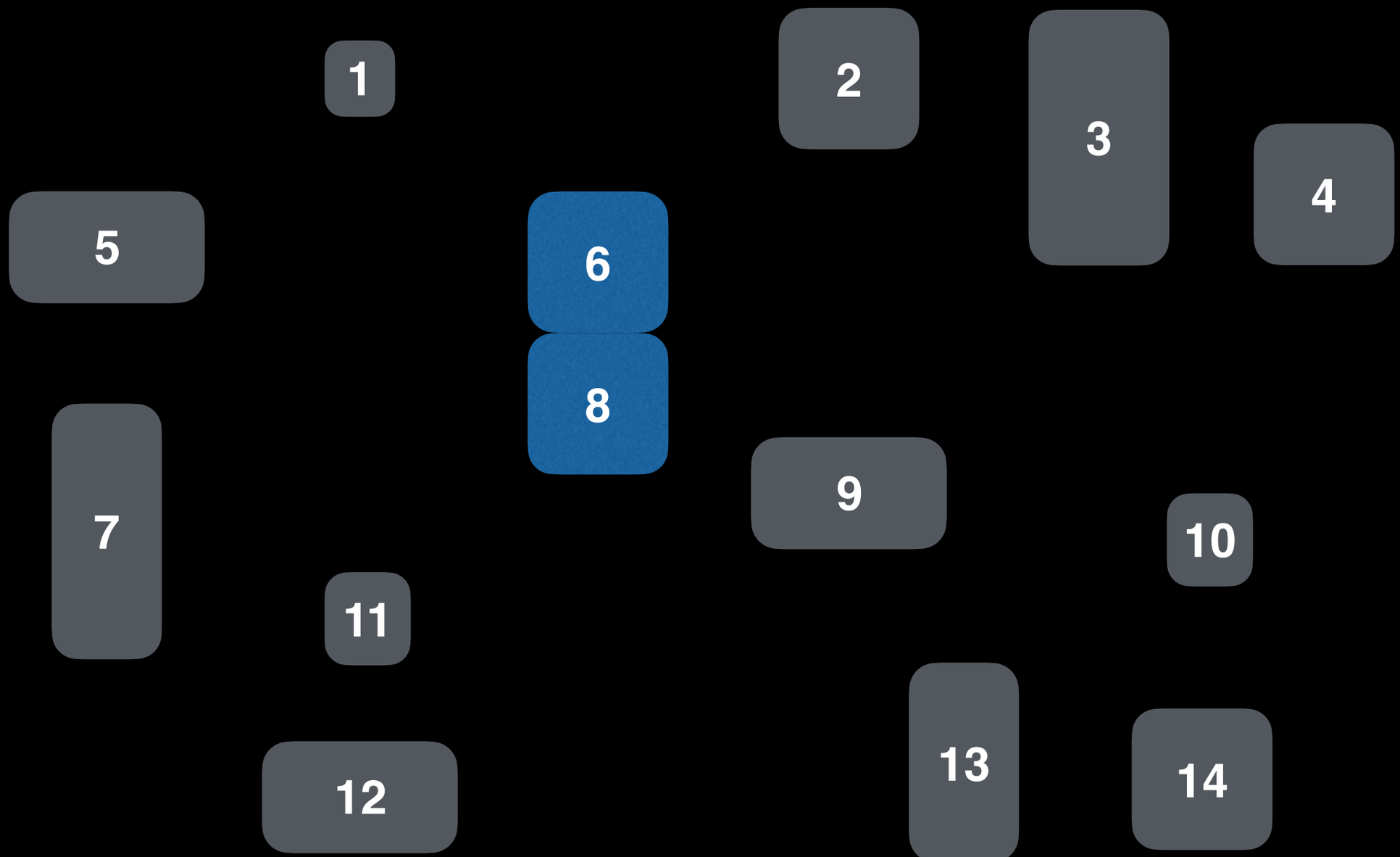
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



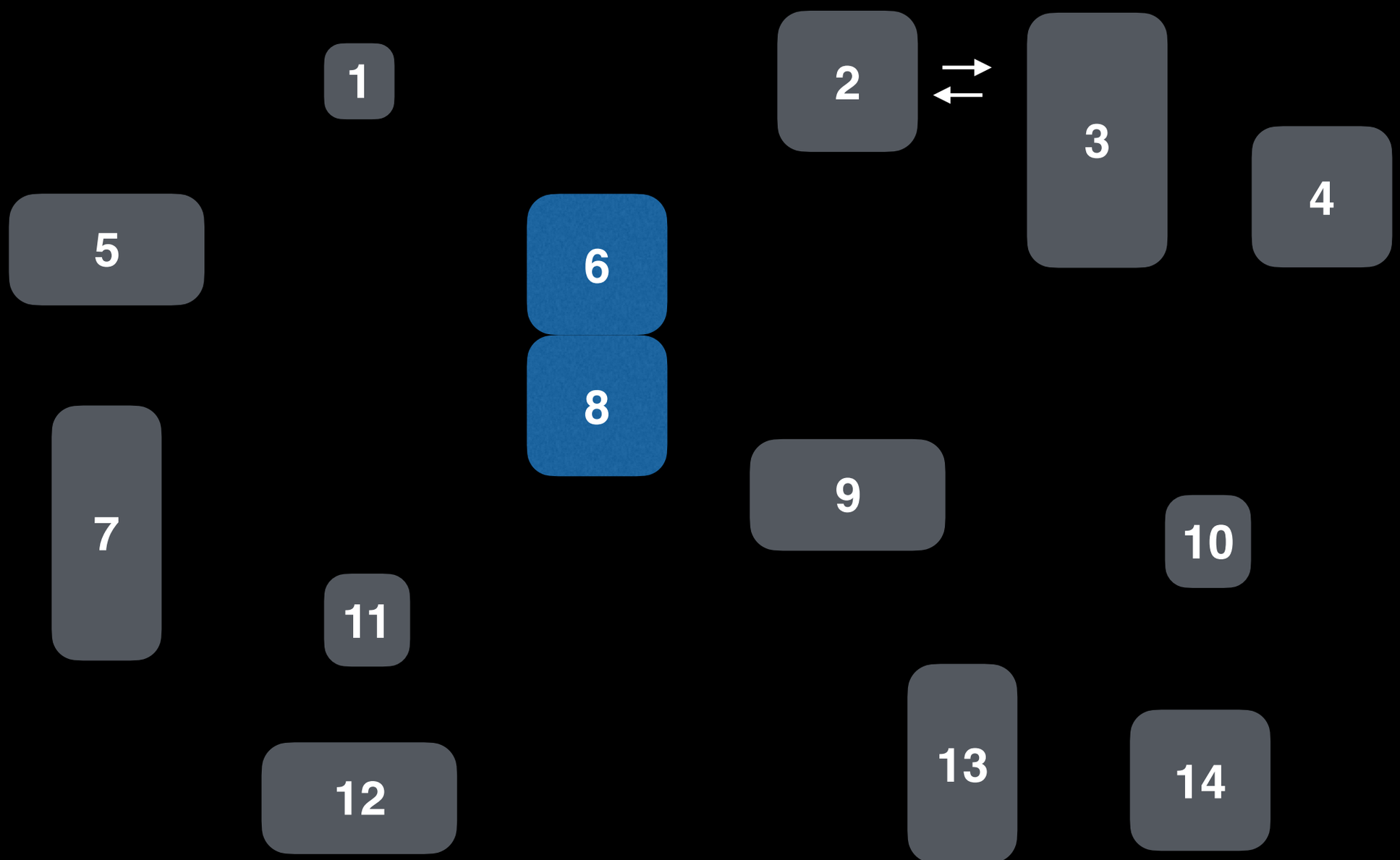
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



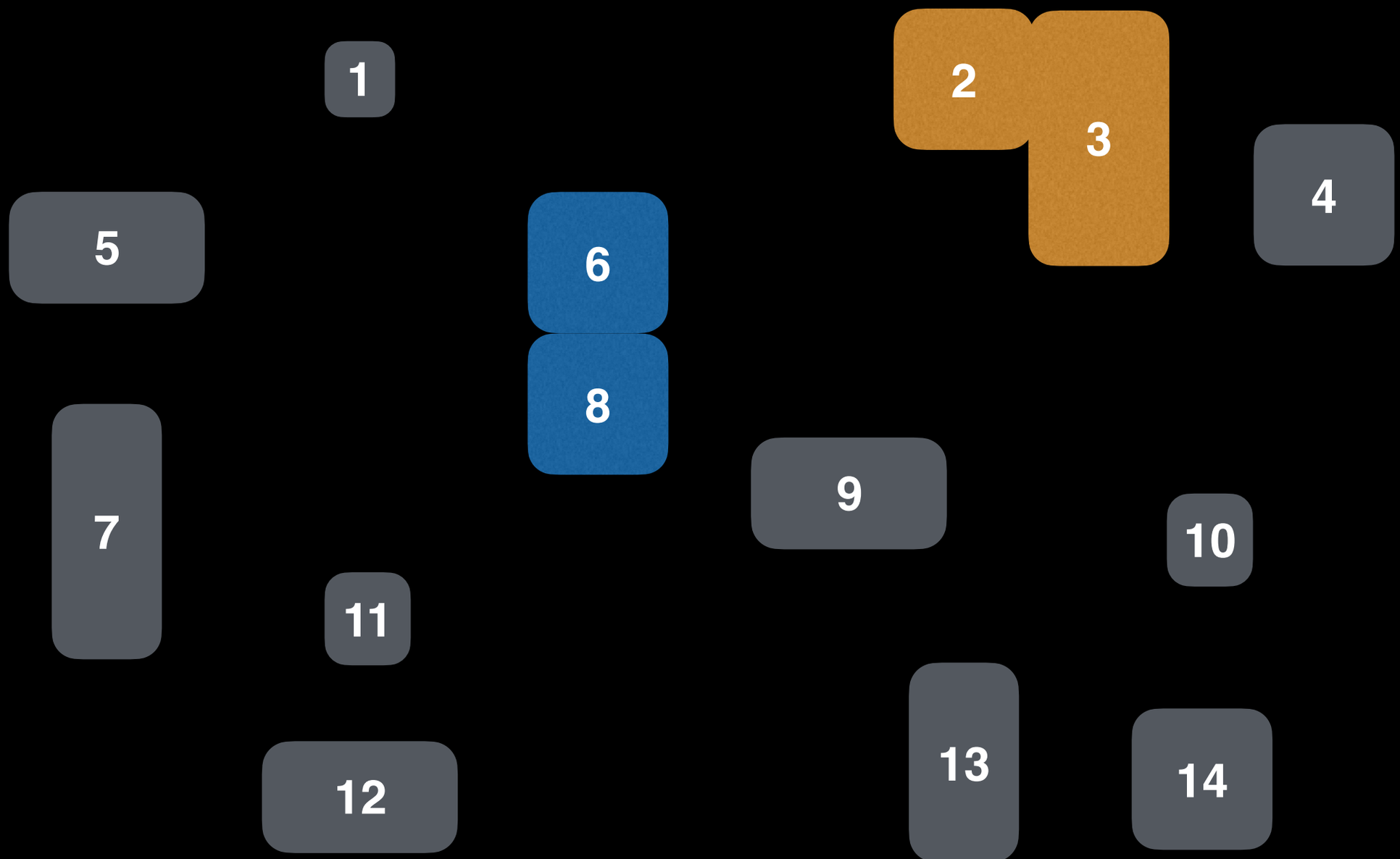
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



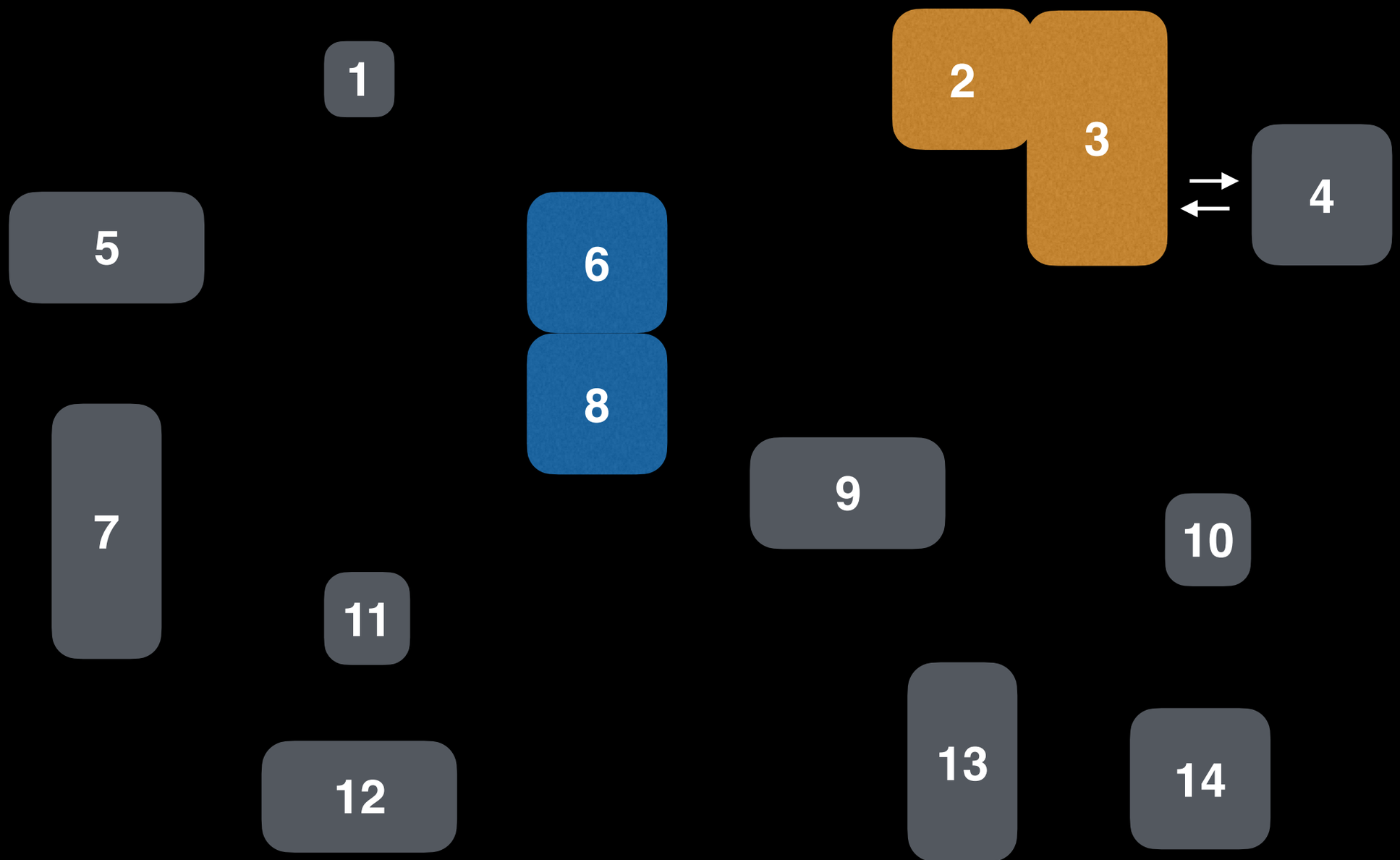
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



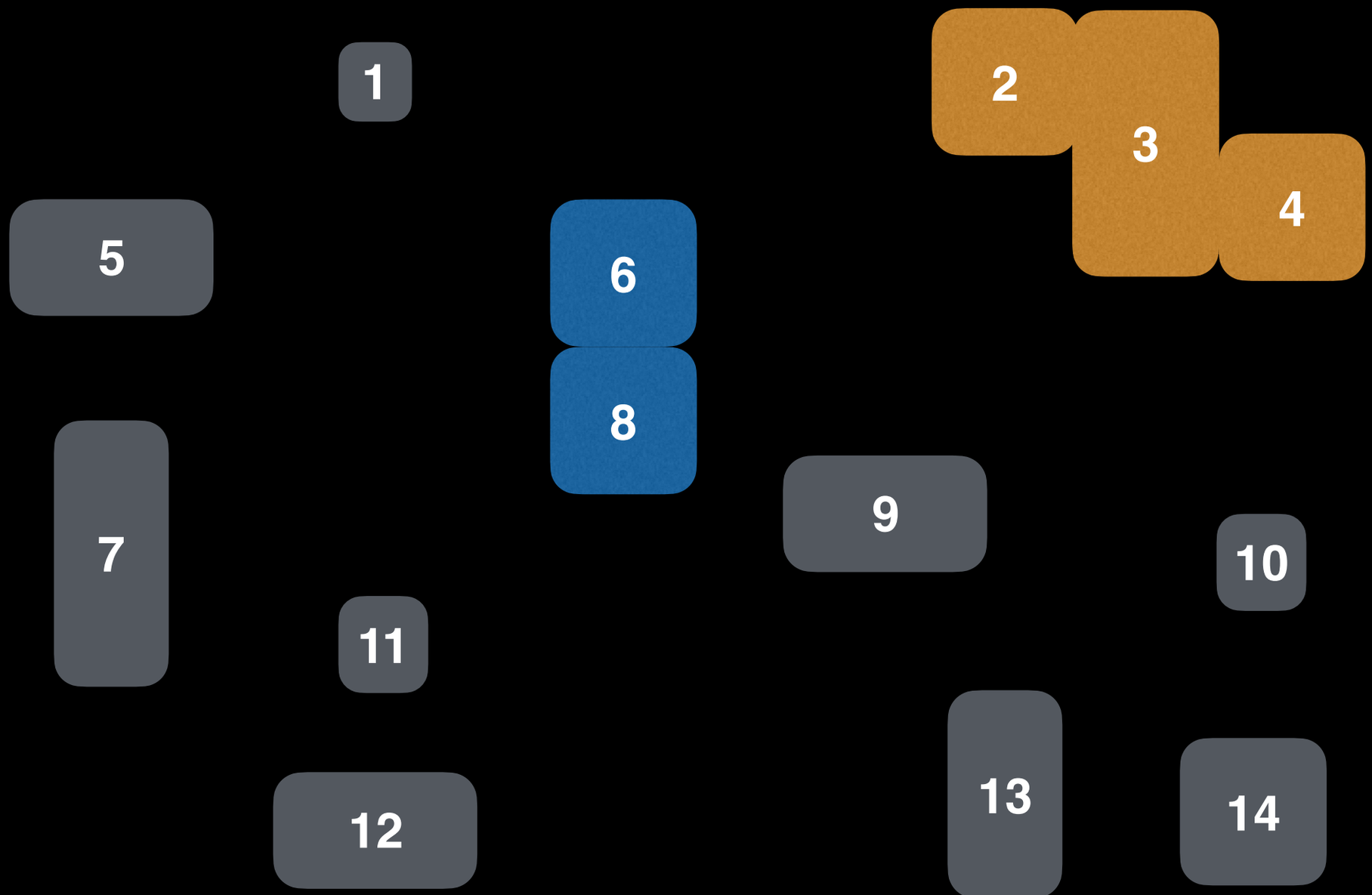
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



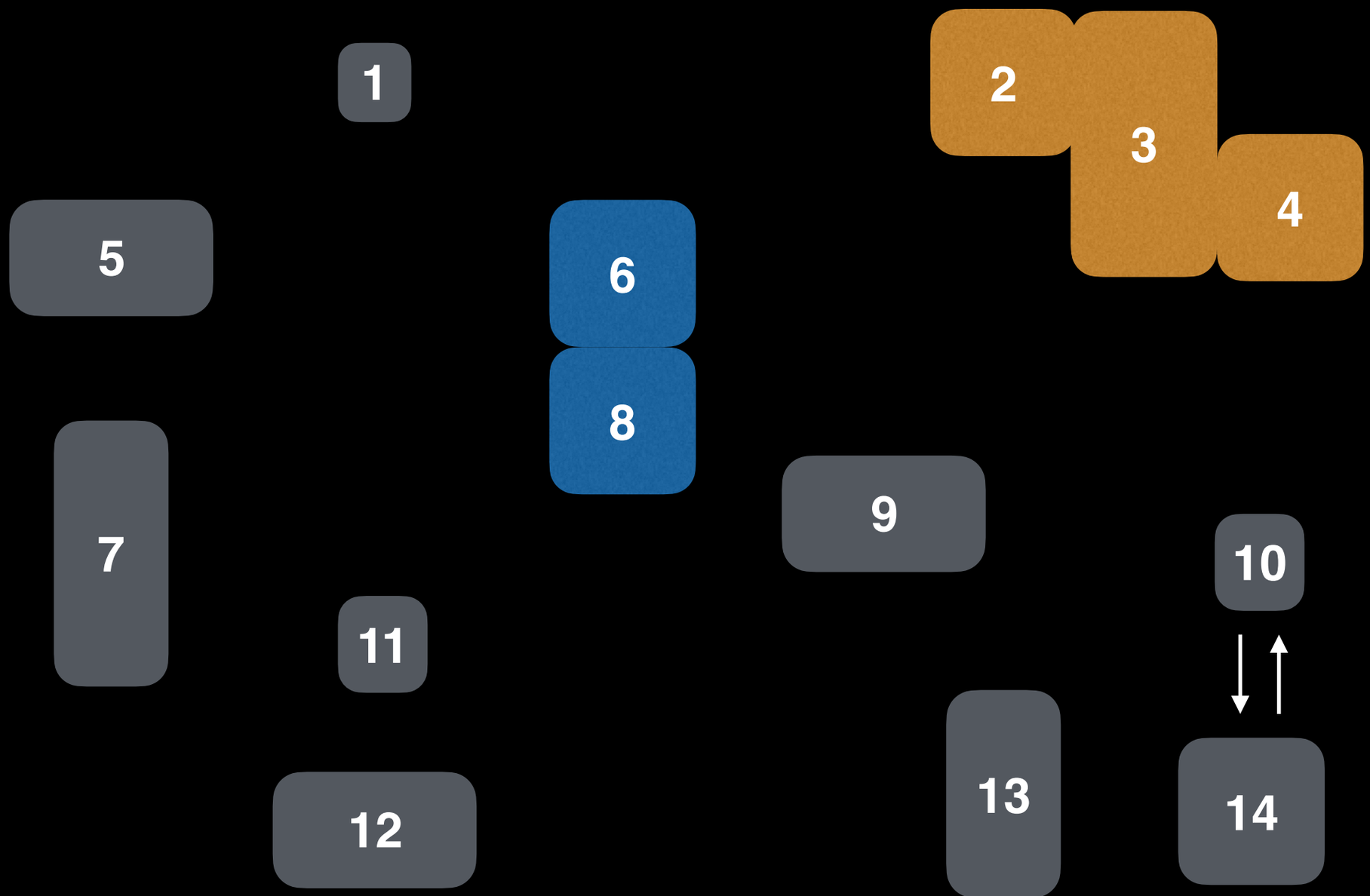
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



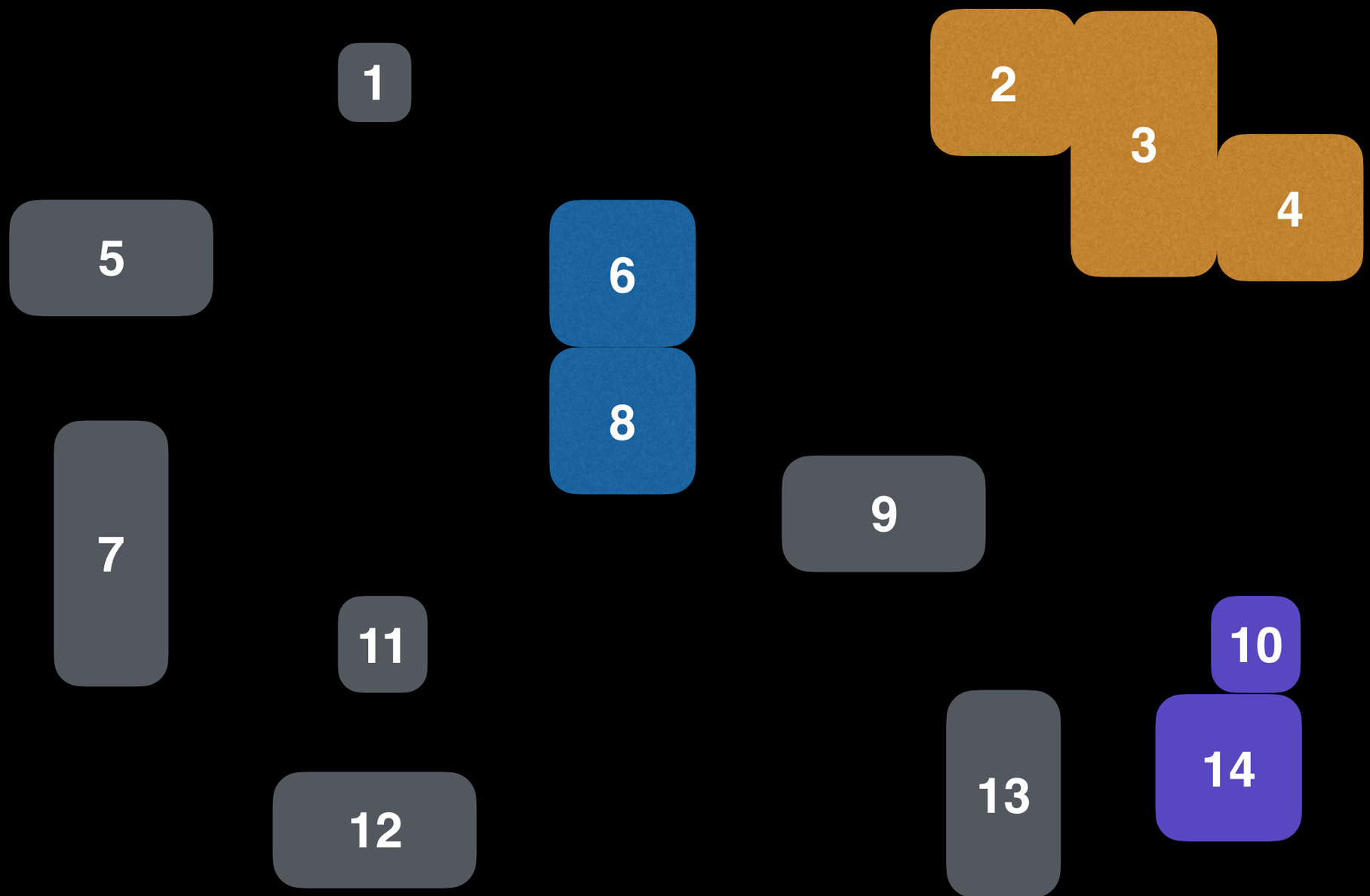
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



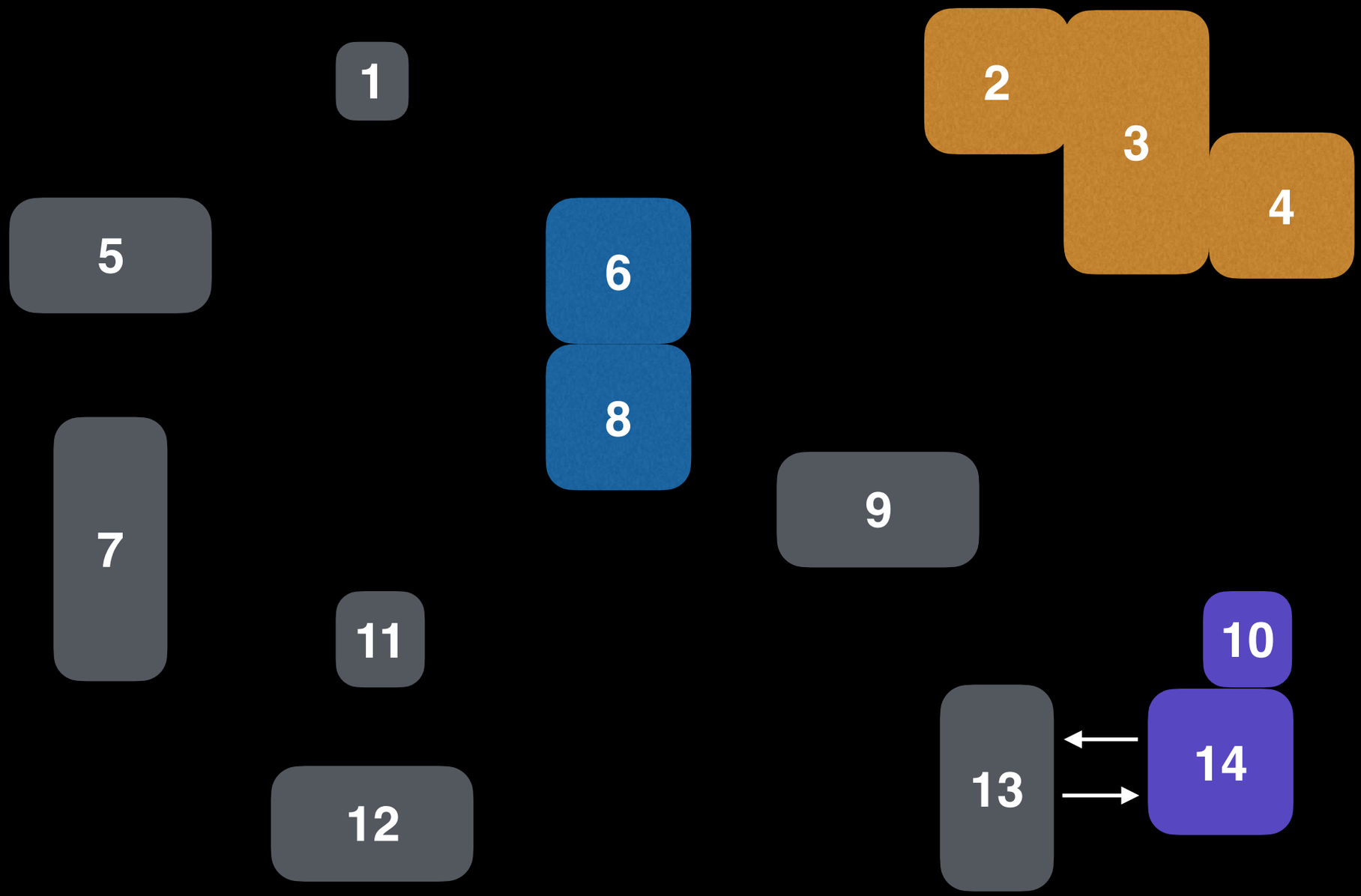
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



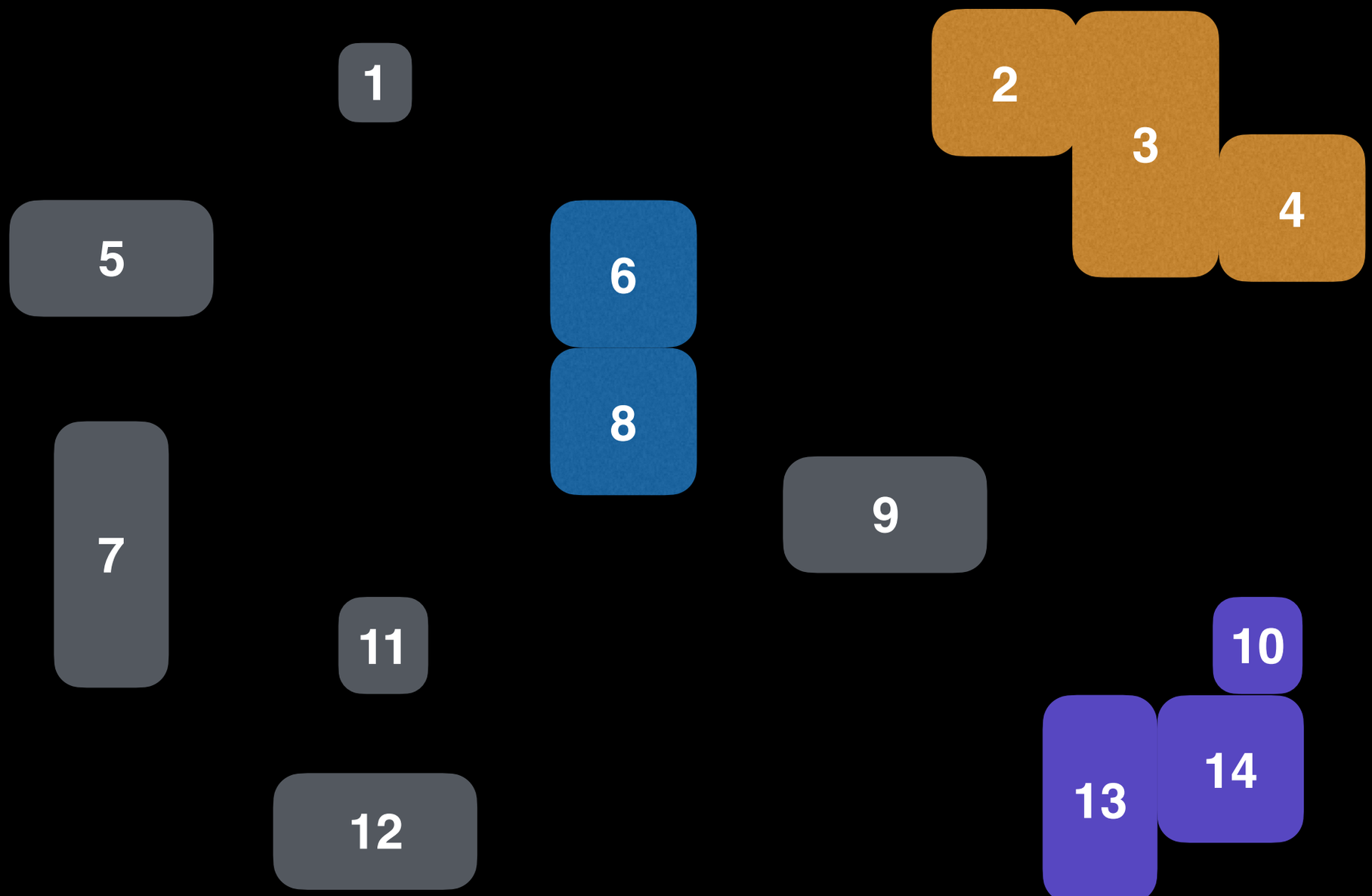
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



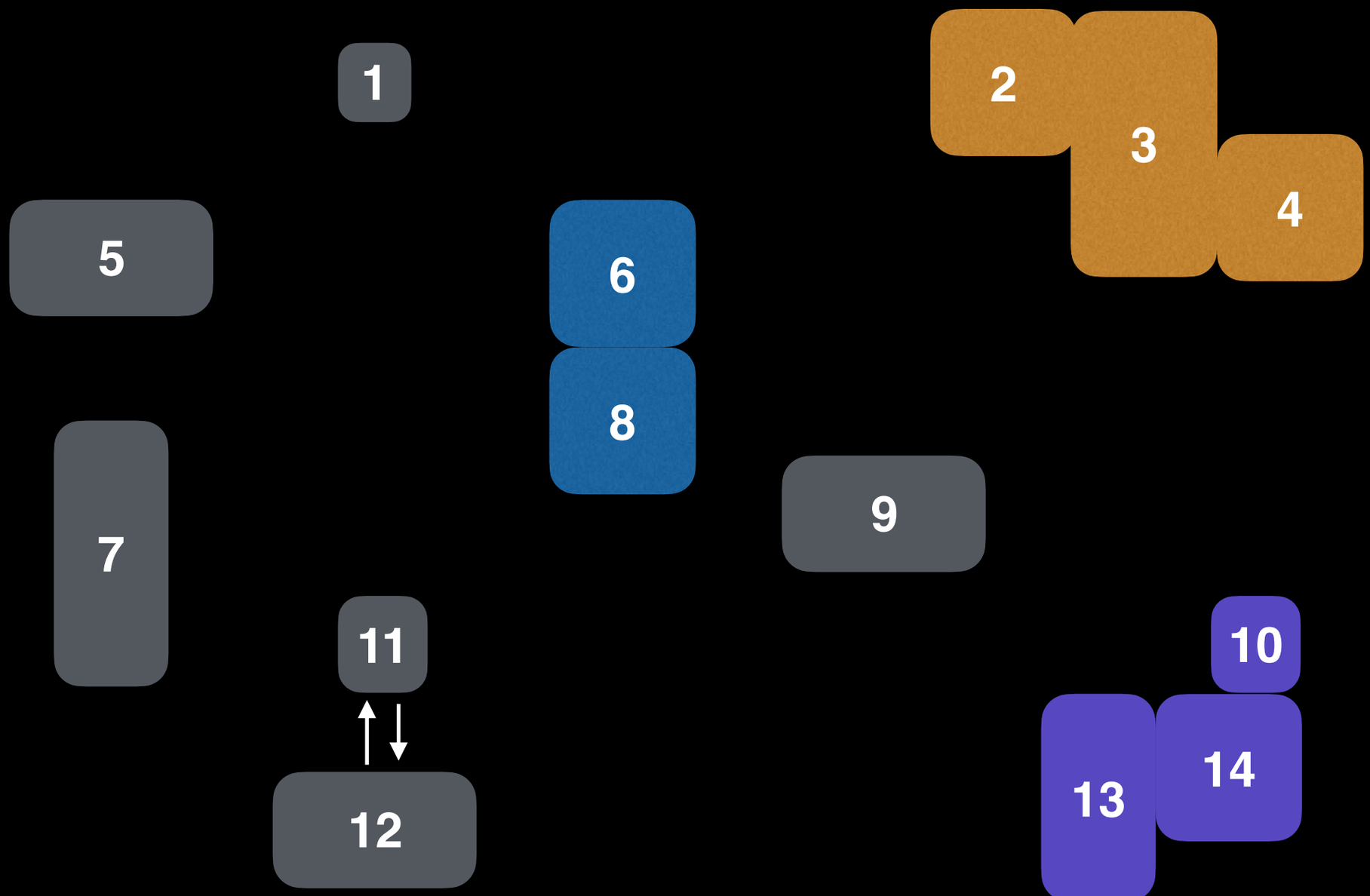
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



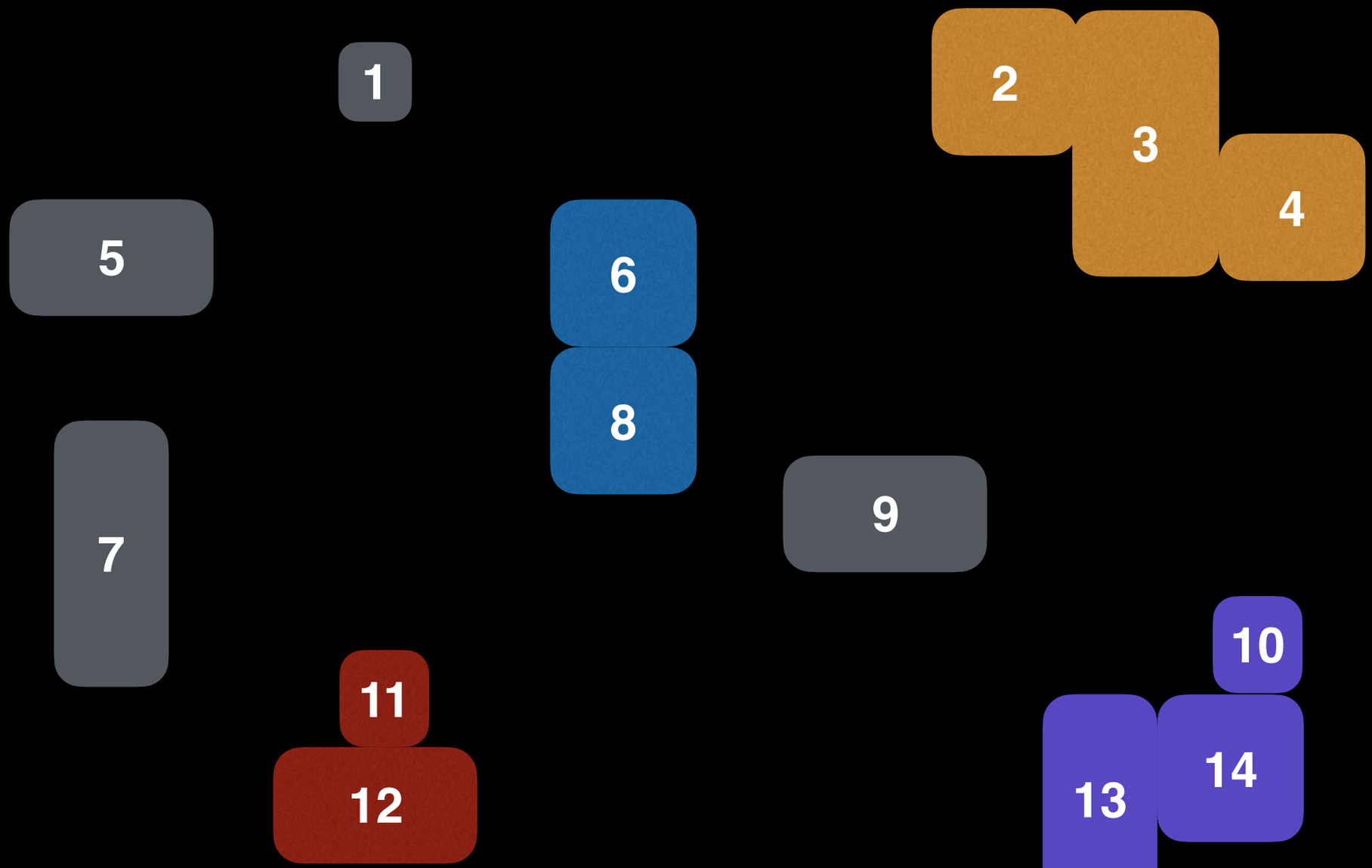
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



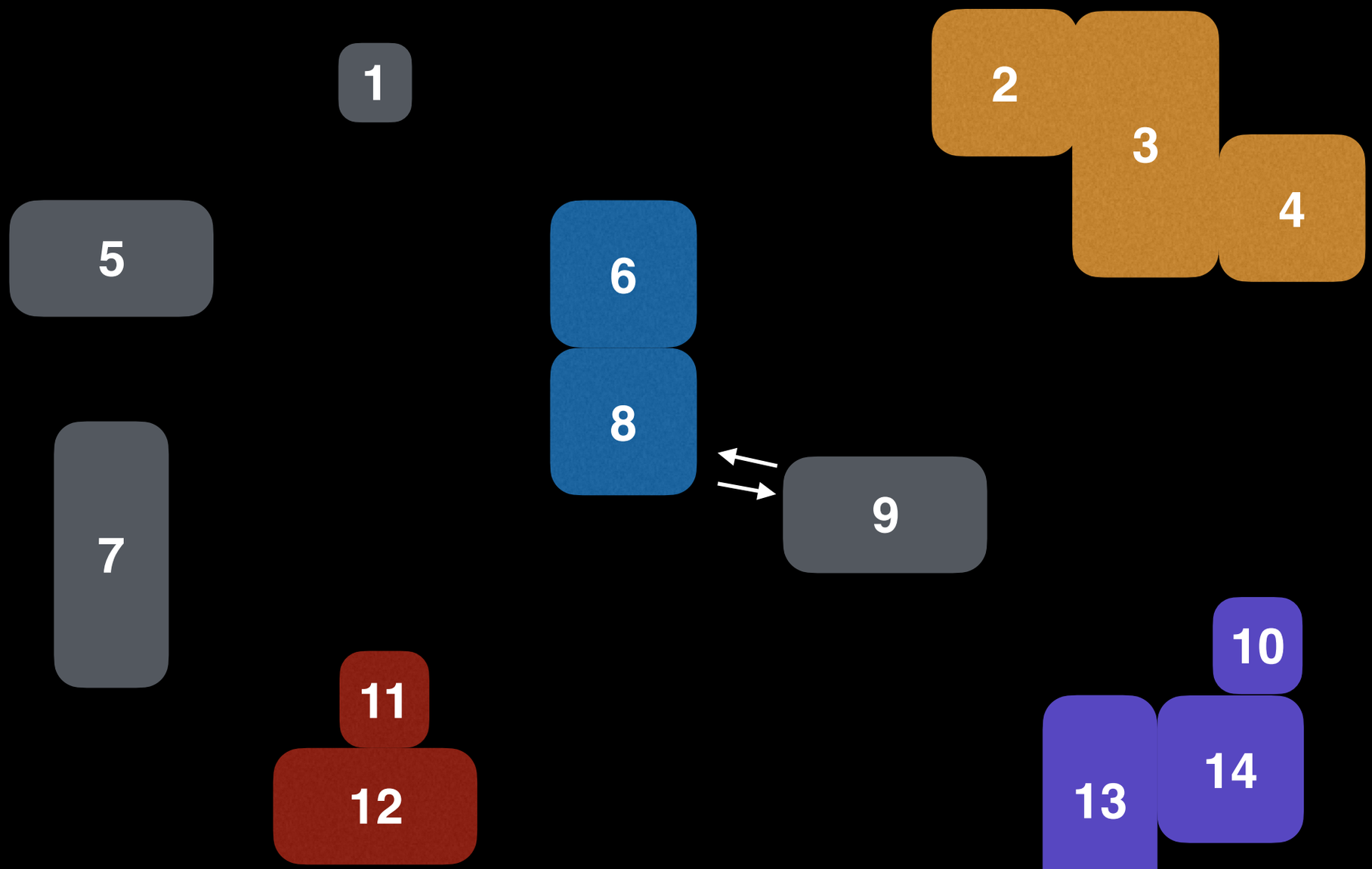
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



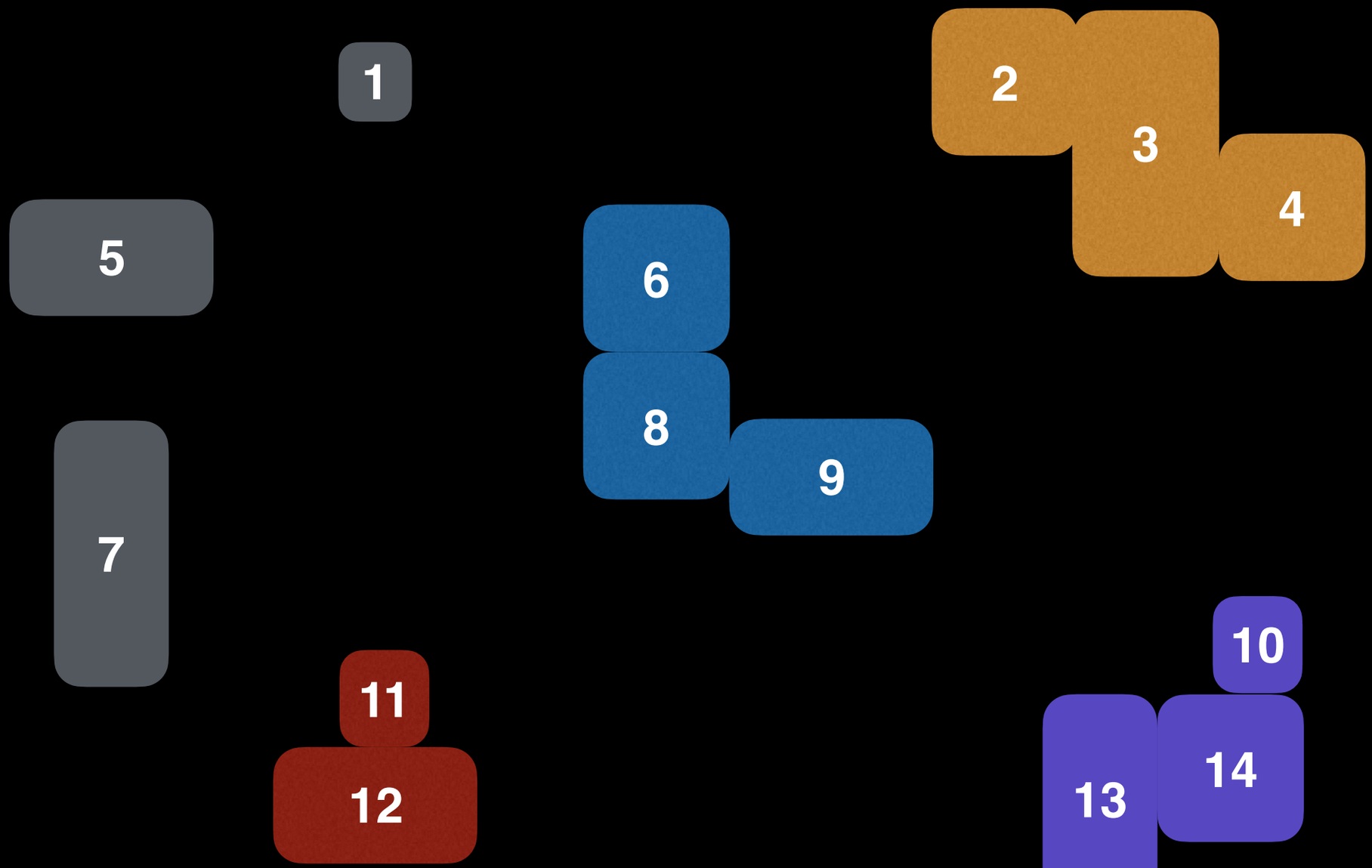
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



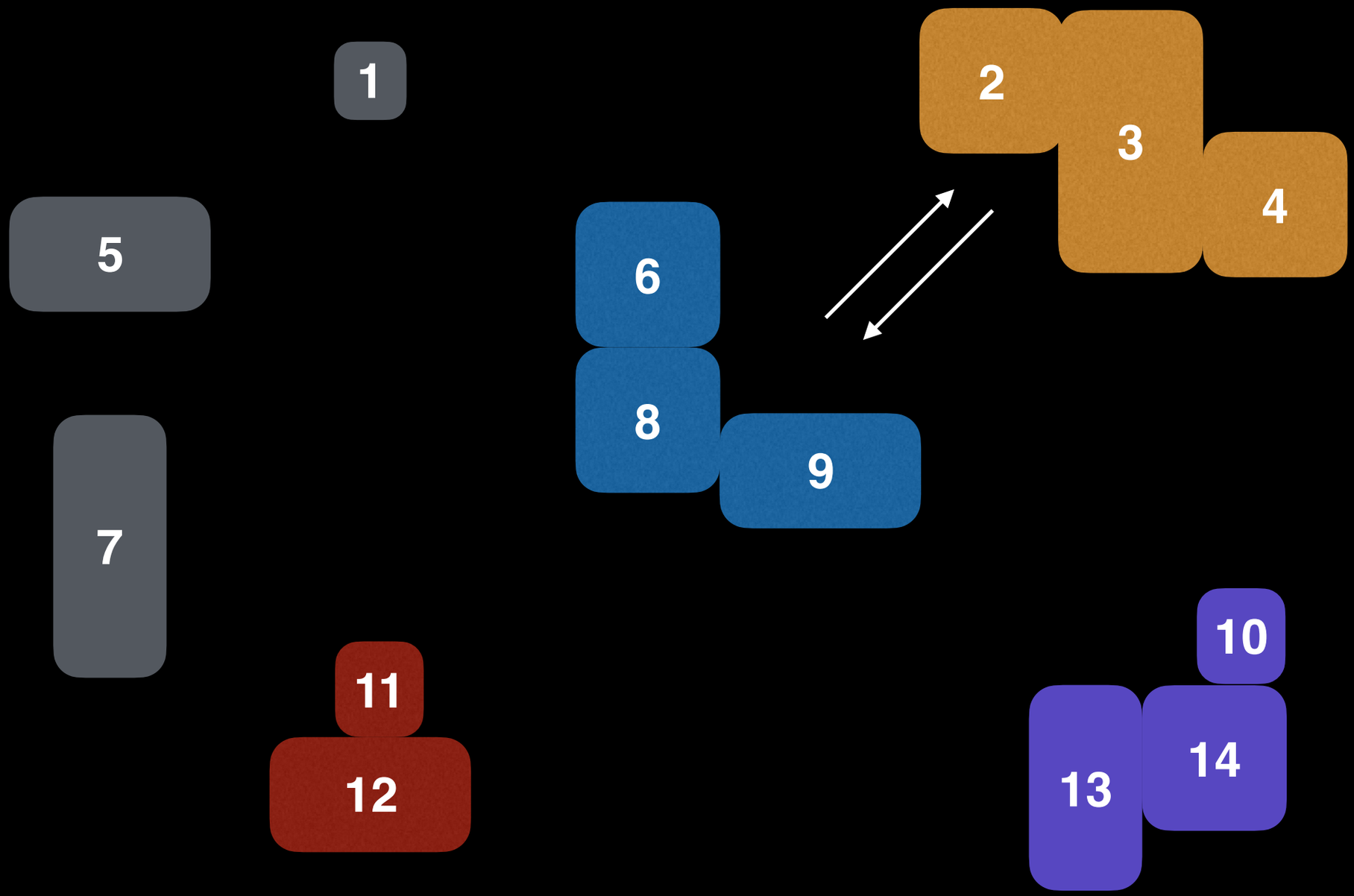
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



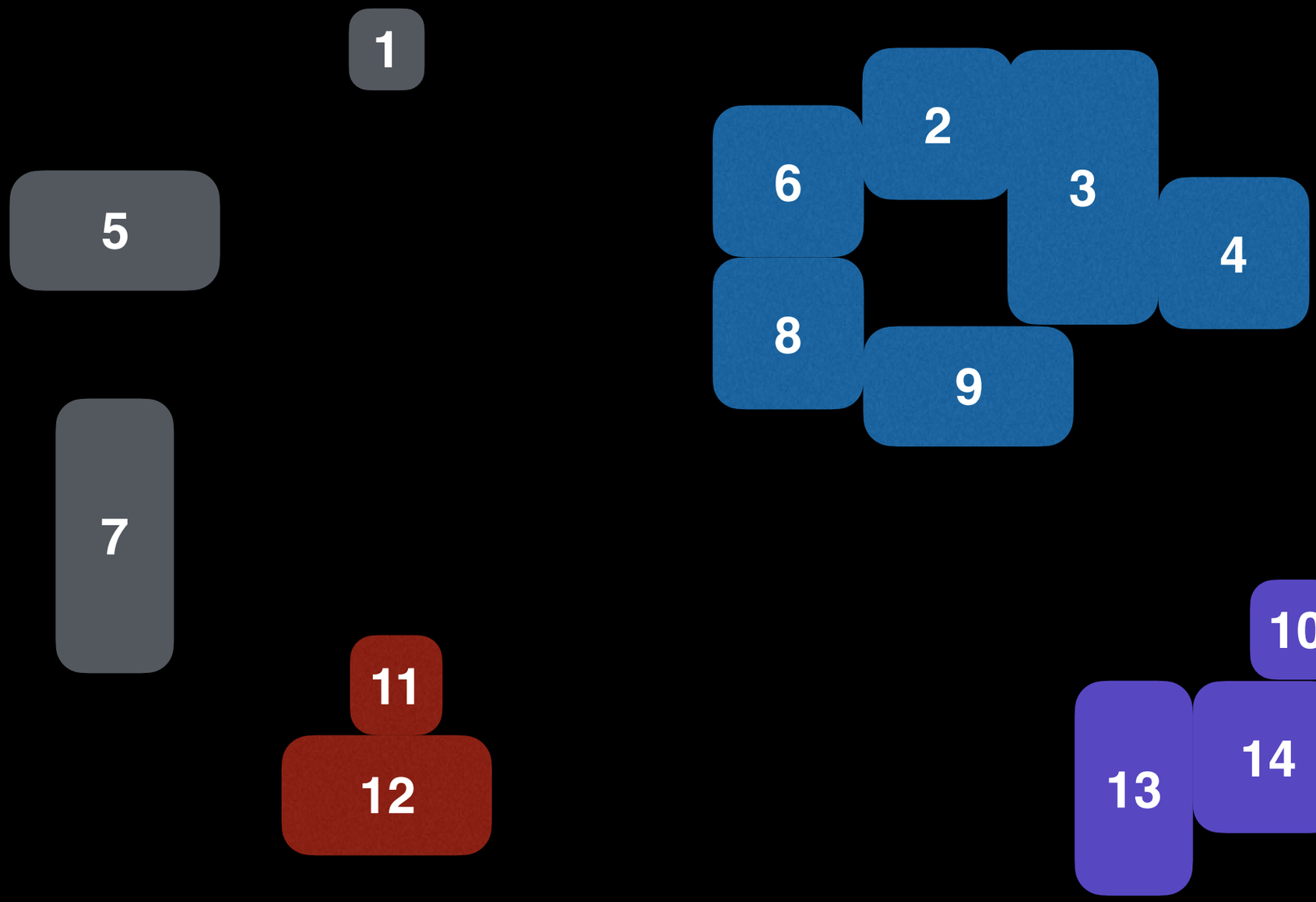
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



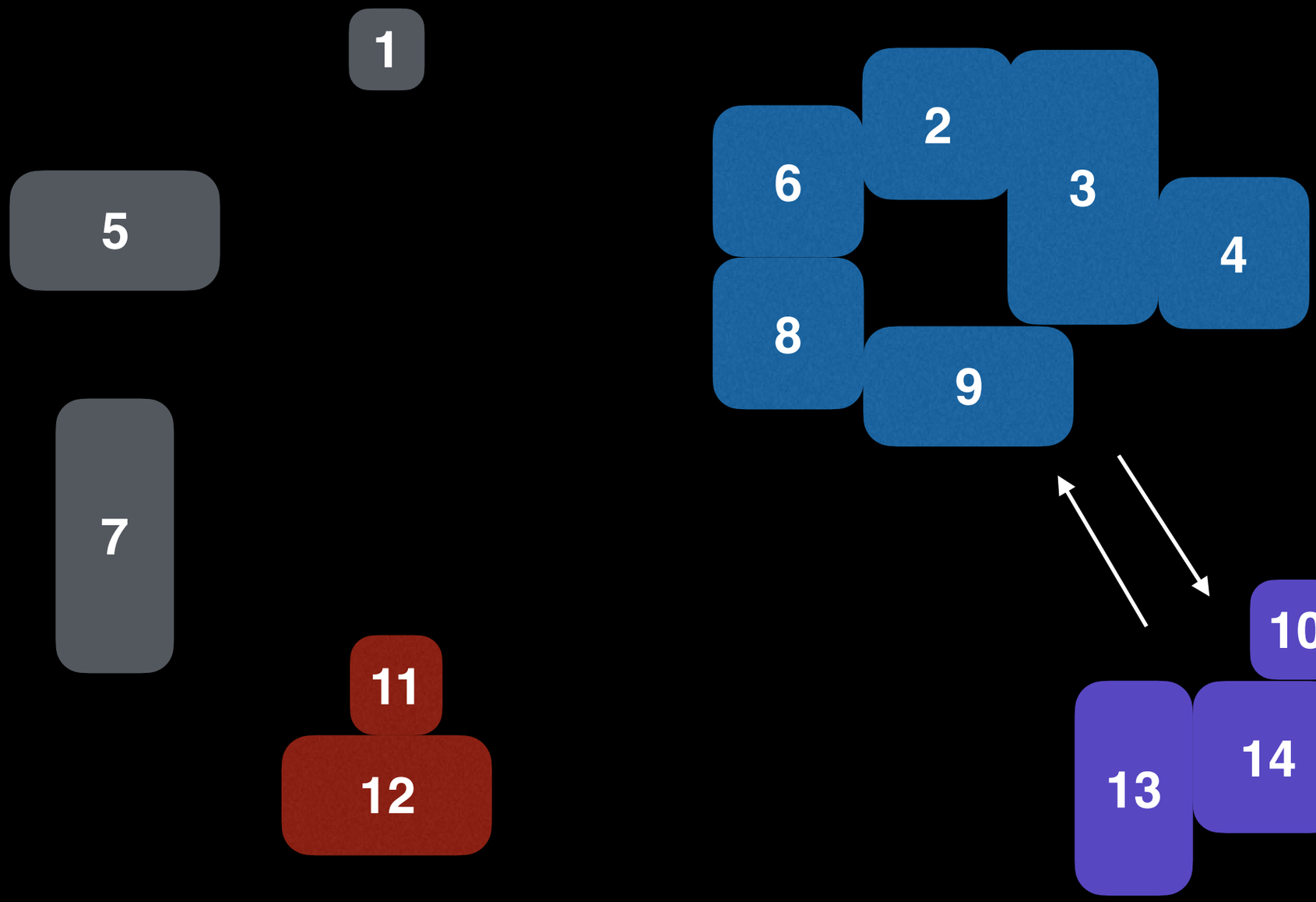
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



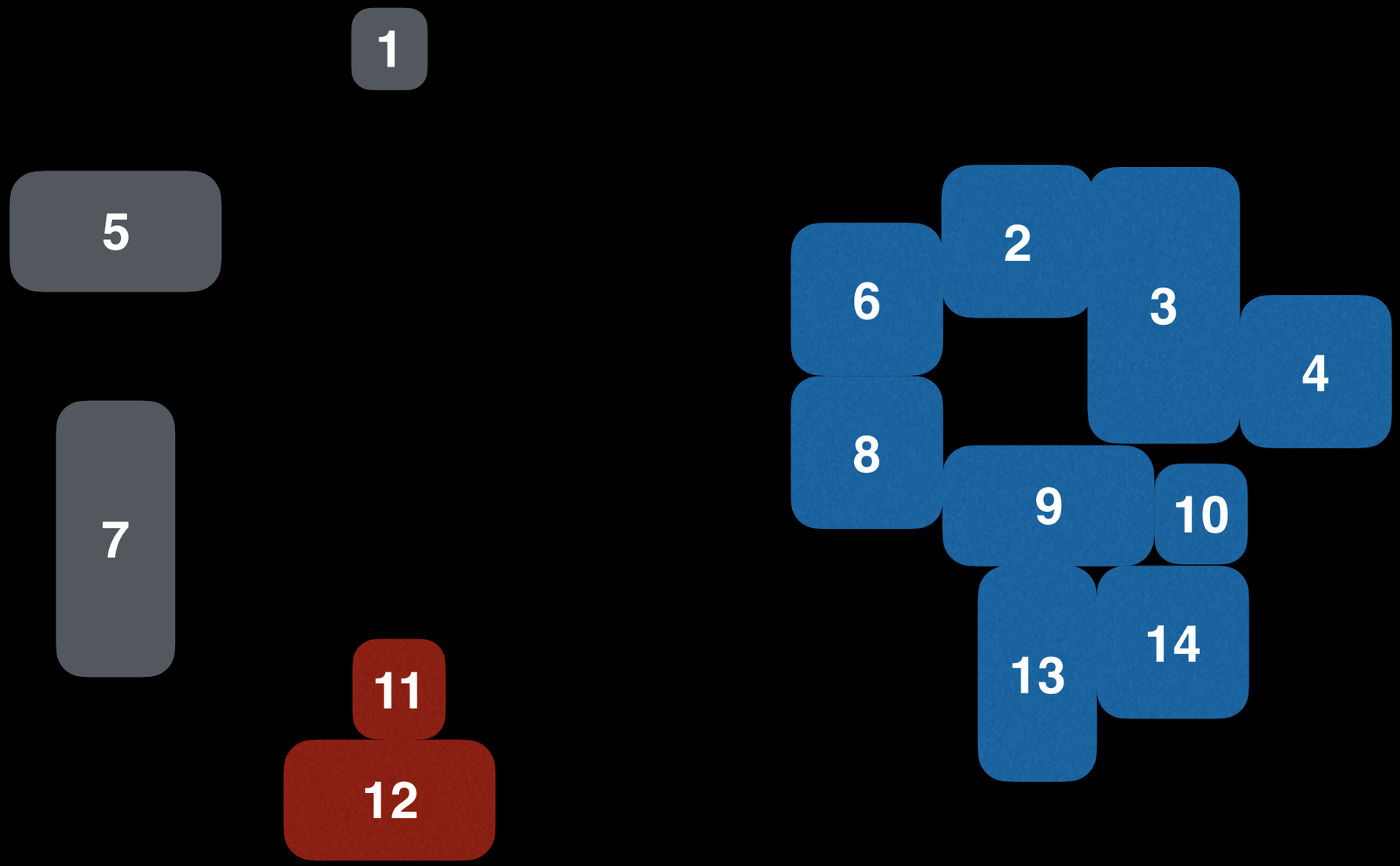
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



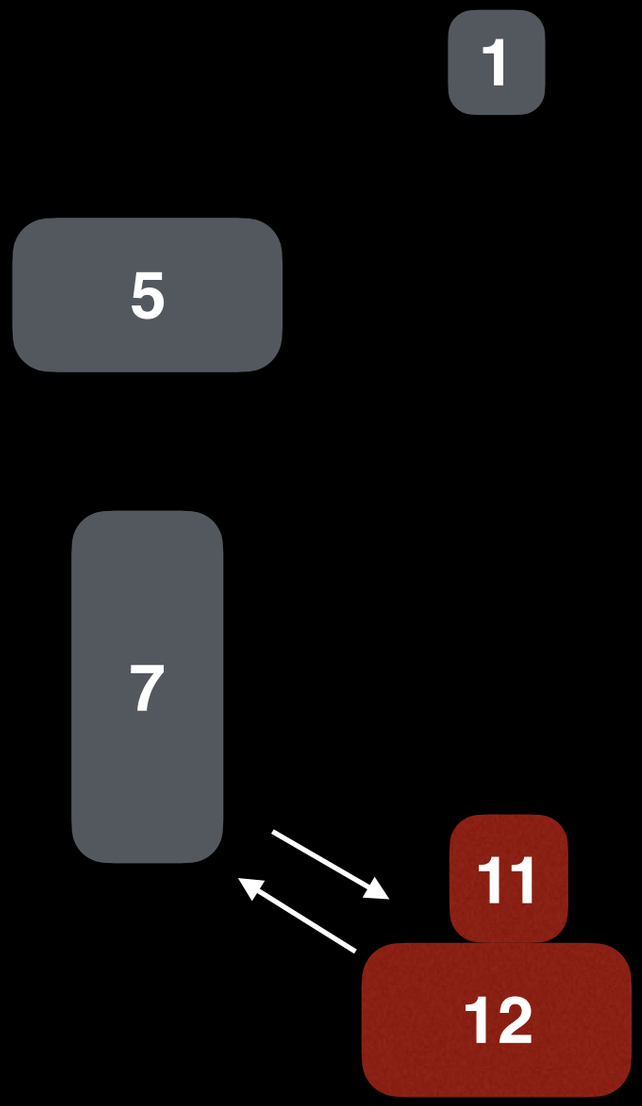
Union Find Magnets Example

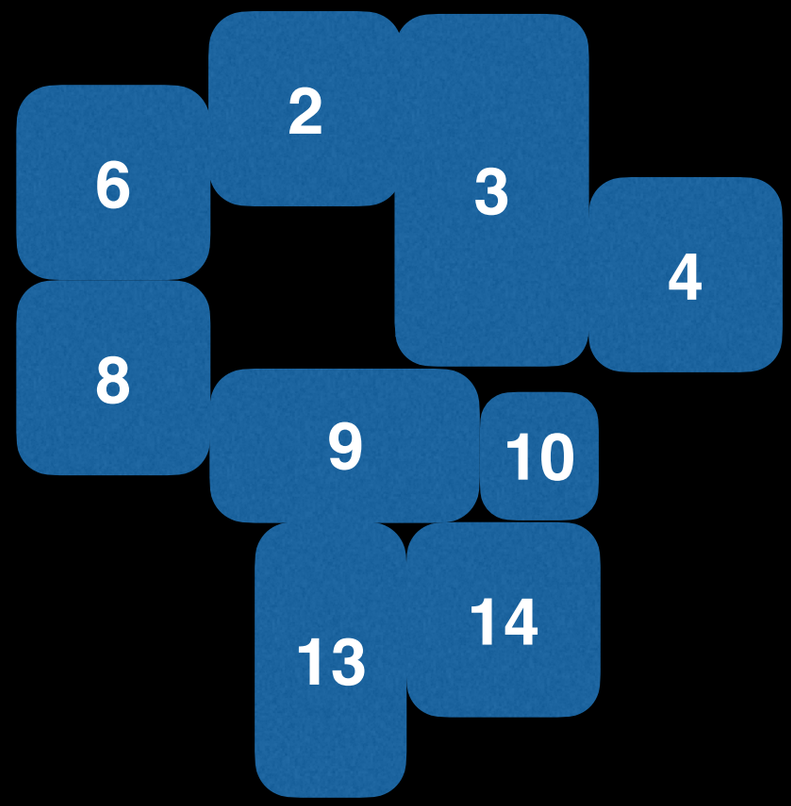
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



Union Find Magnets Example

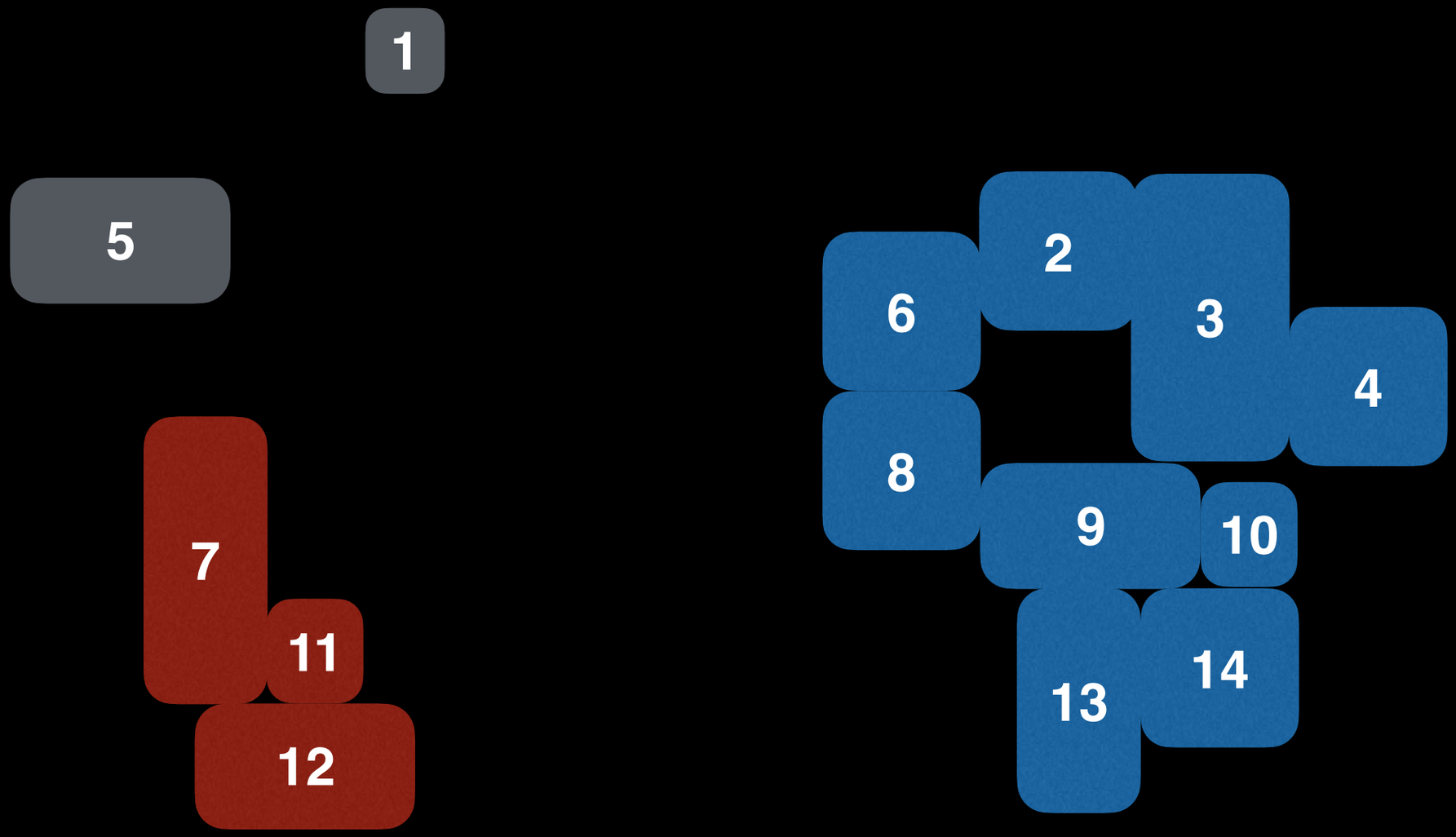
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14





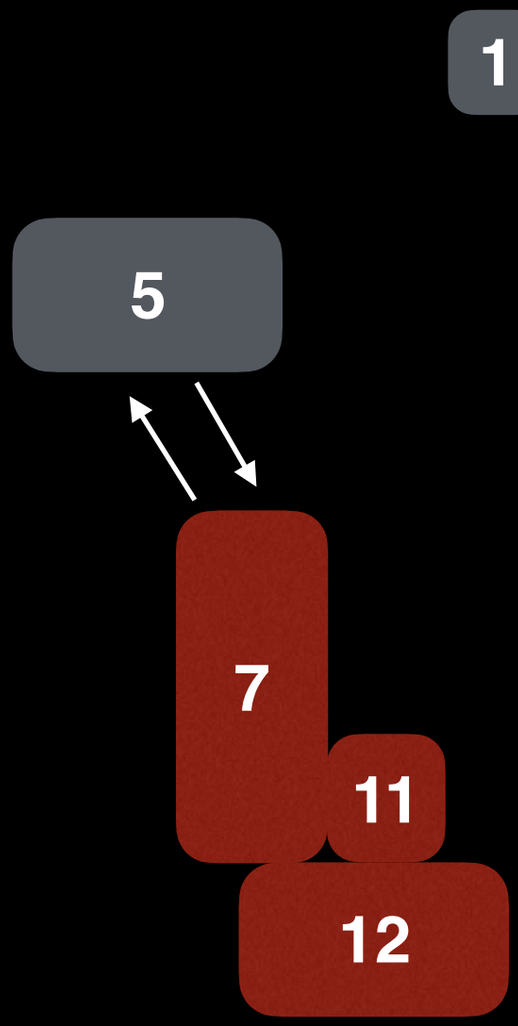
Union Find Magnets Example

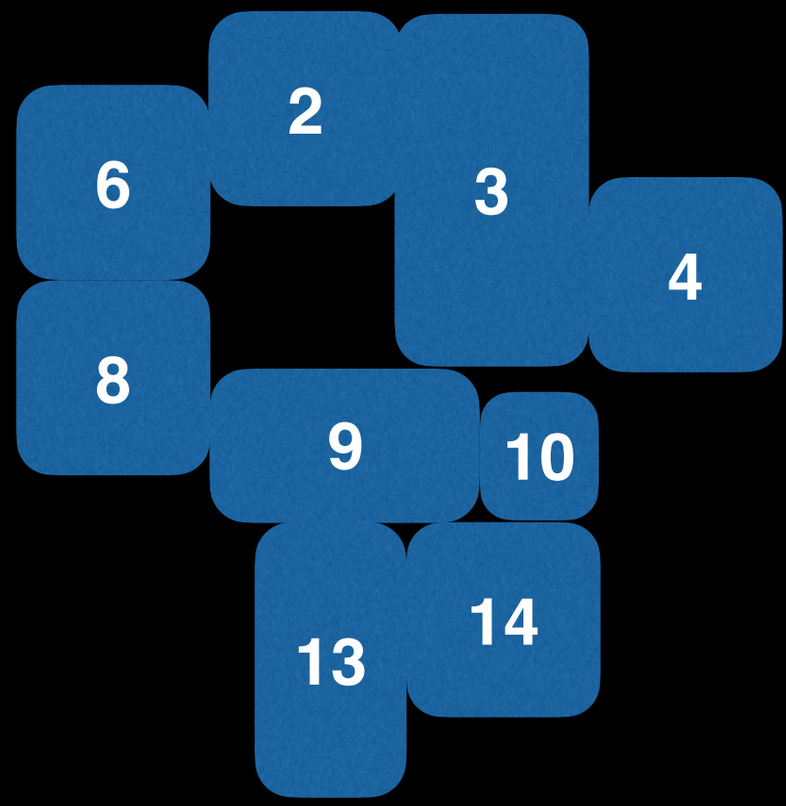
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



Union Find Magnets Example

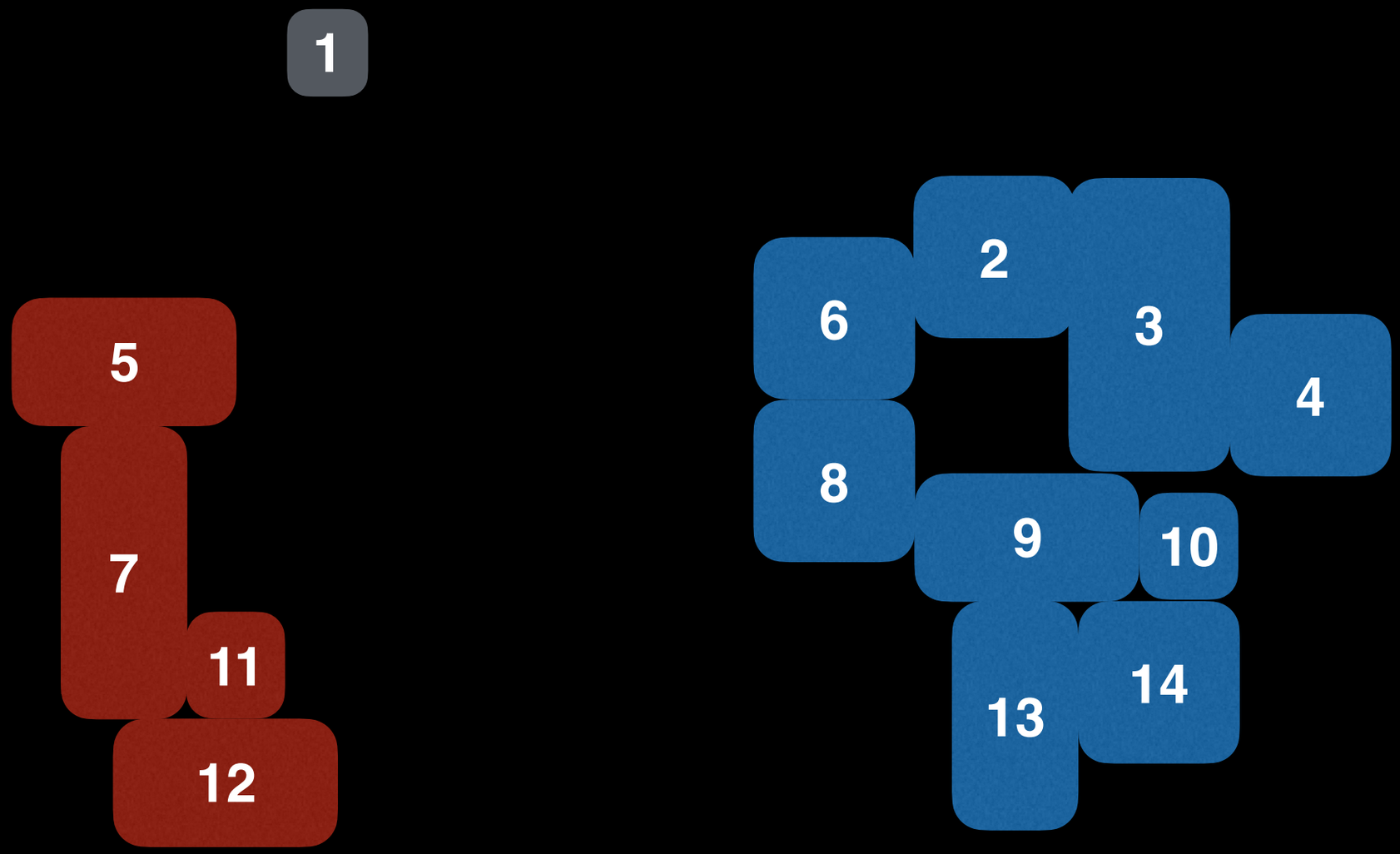
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14





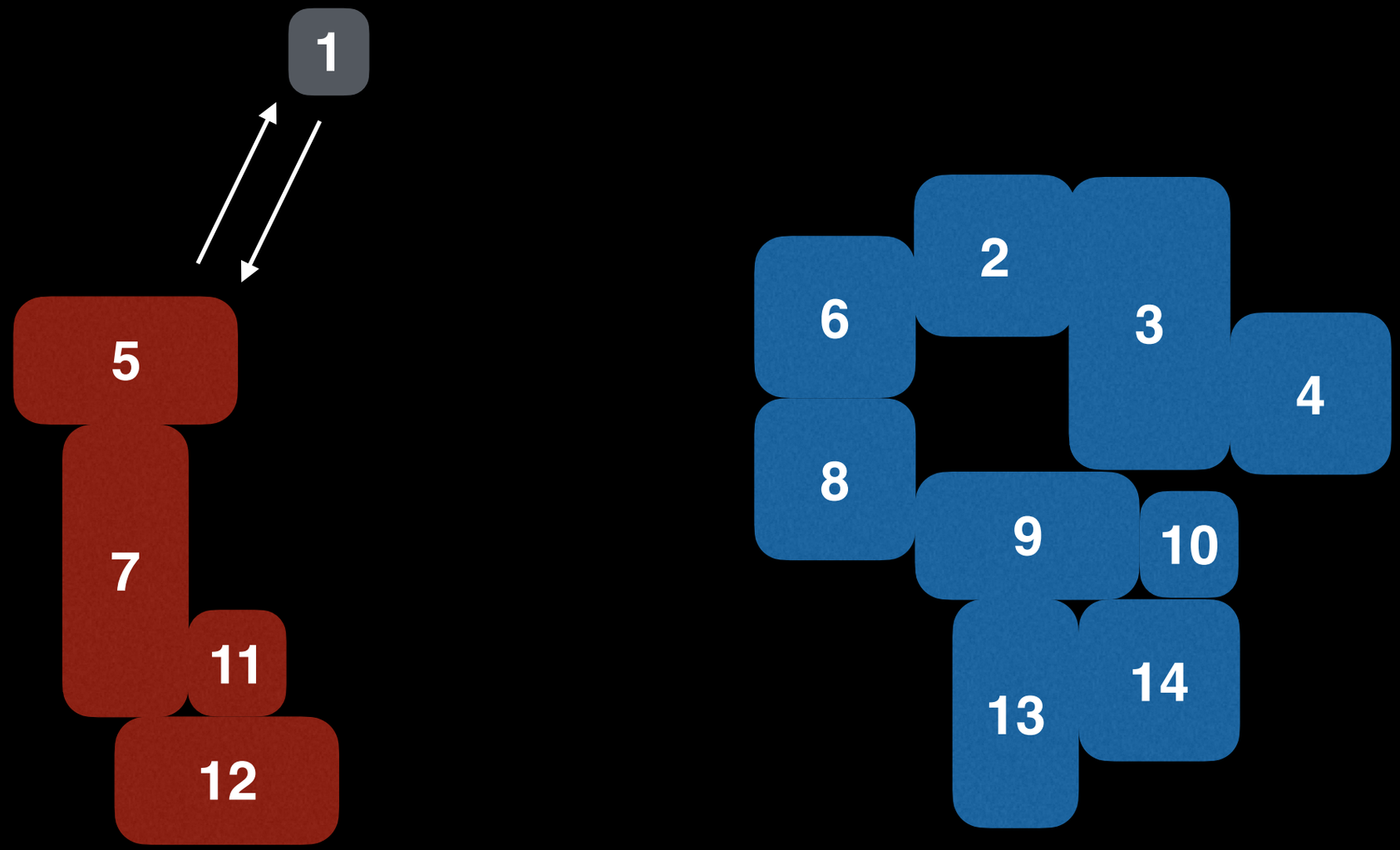
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



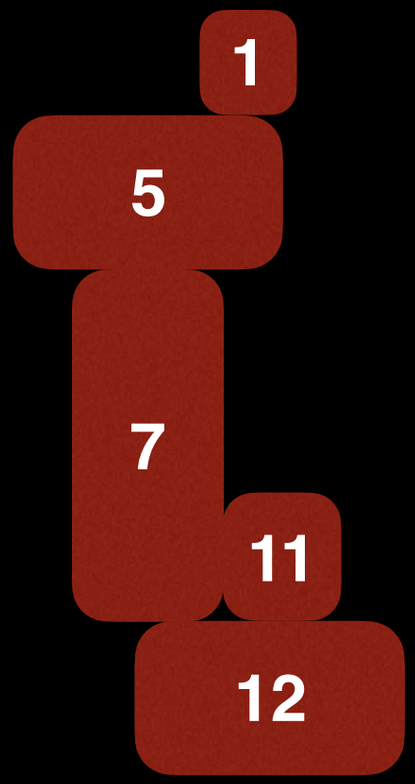
Union Find Magnets Example

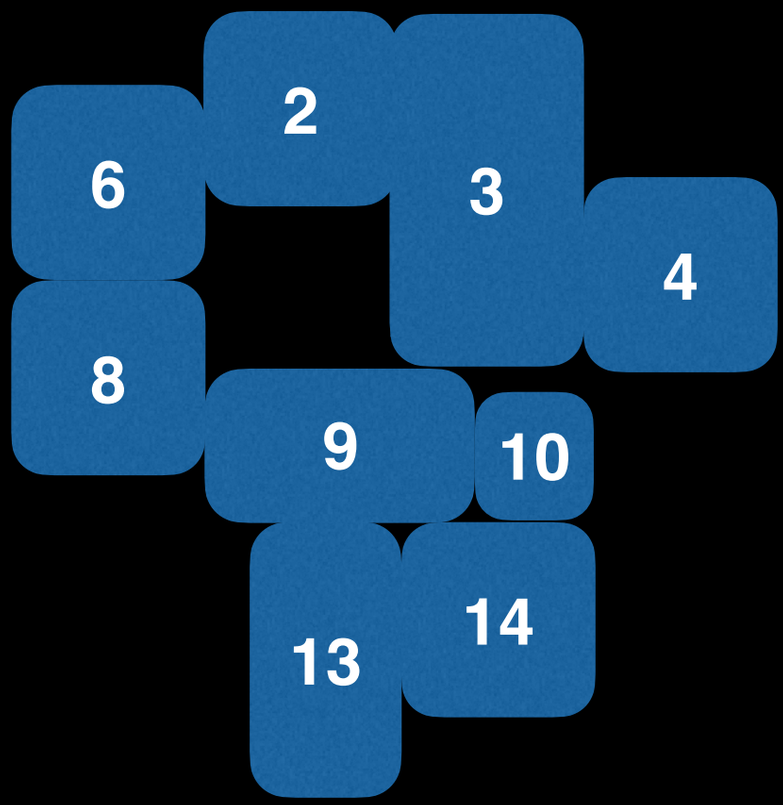
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



Union Find Magnets Example

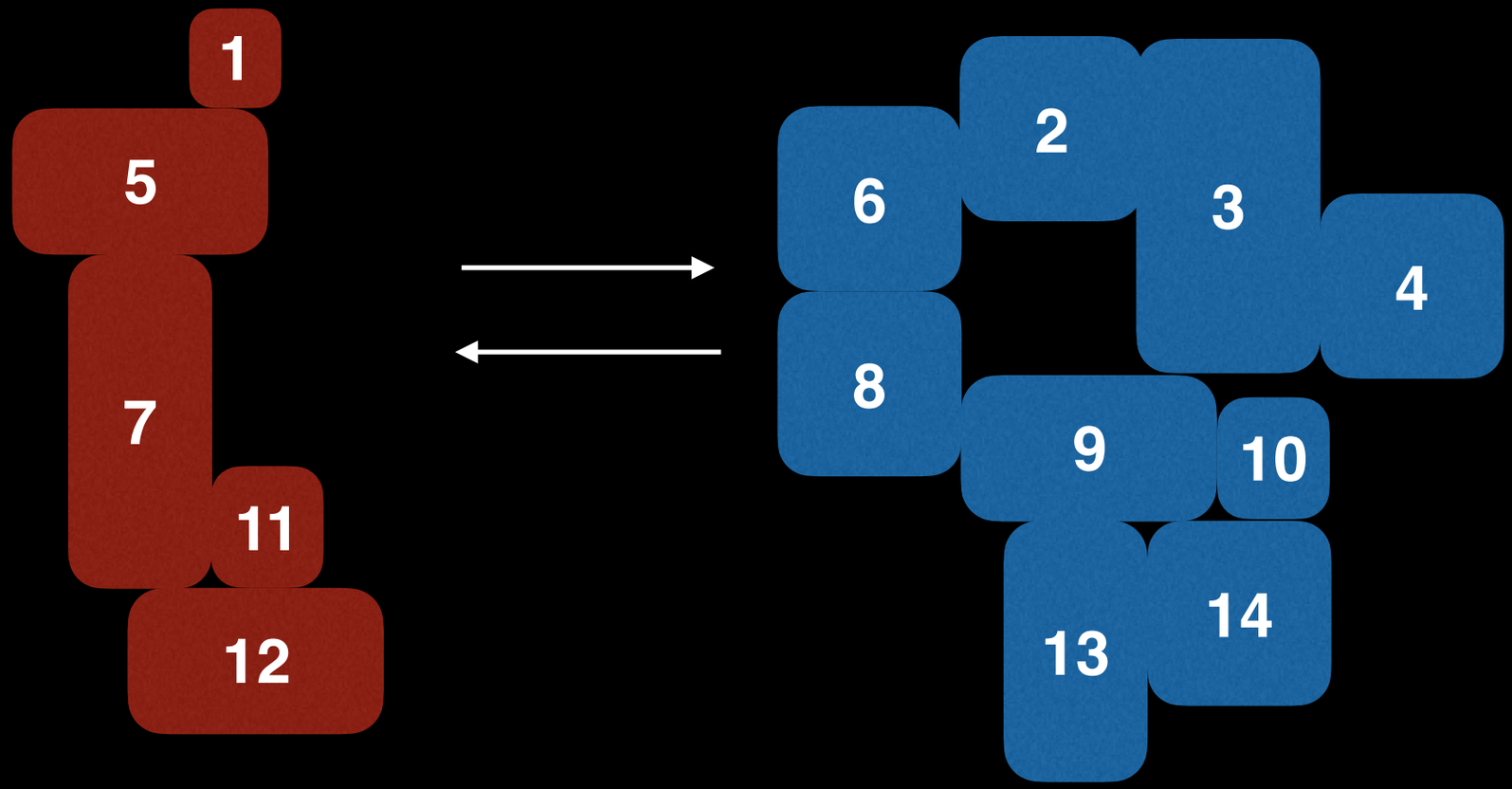
Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14





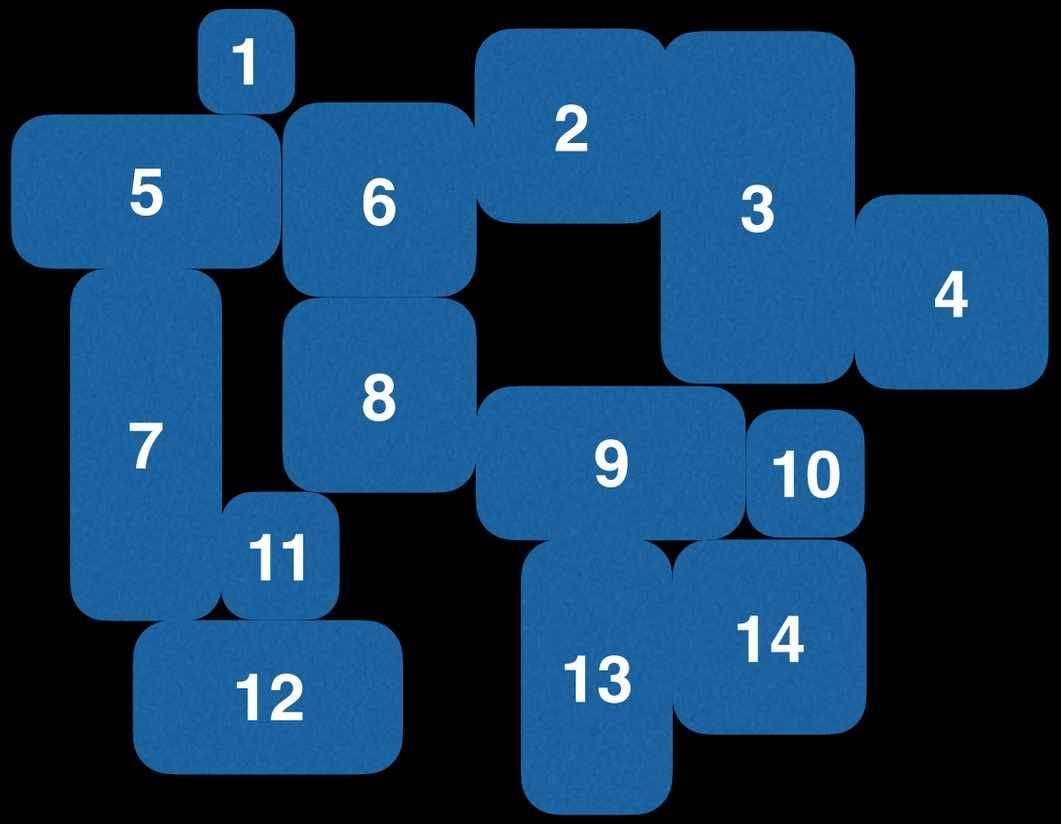
Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



Union Find Magnets Example

Magnet 1
Magnet 2
Magnet 3
Magnet 4
Magnet 5
Magnet 6
Magnet 7
Magnet 8
Magnet 9
Magnet 10
Magnet 11
Magnet 12
Magnet 13
Magnet 14



When and where is a Union Find used?

Kruskal's minimum spanning tree algorithm

Grid percolation

Network connectivity

Lowest common ancestor in trees

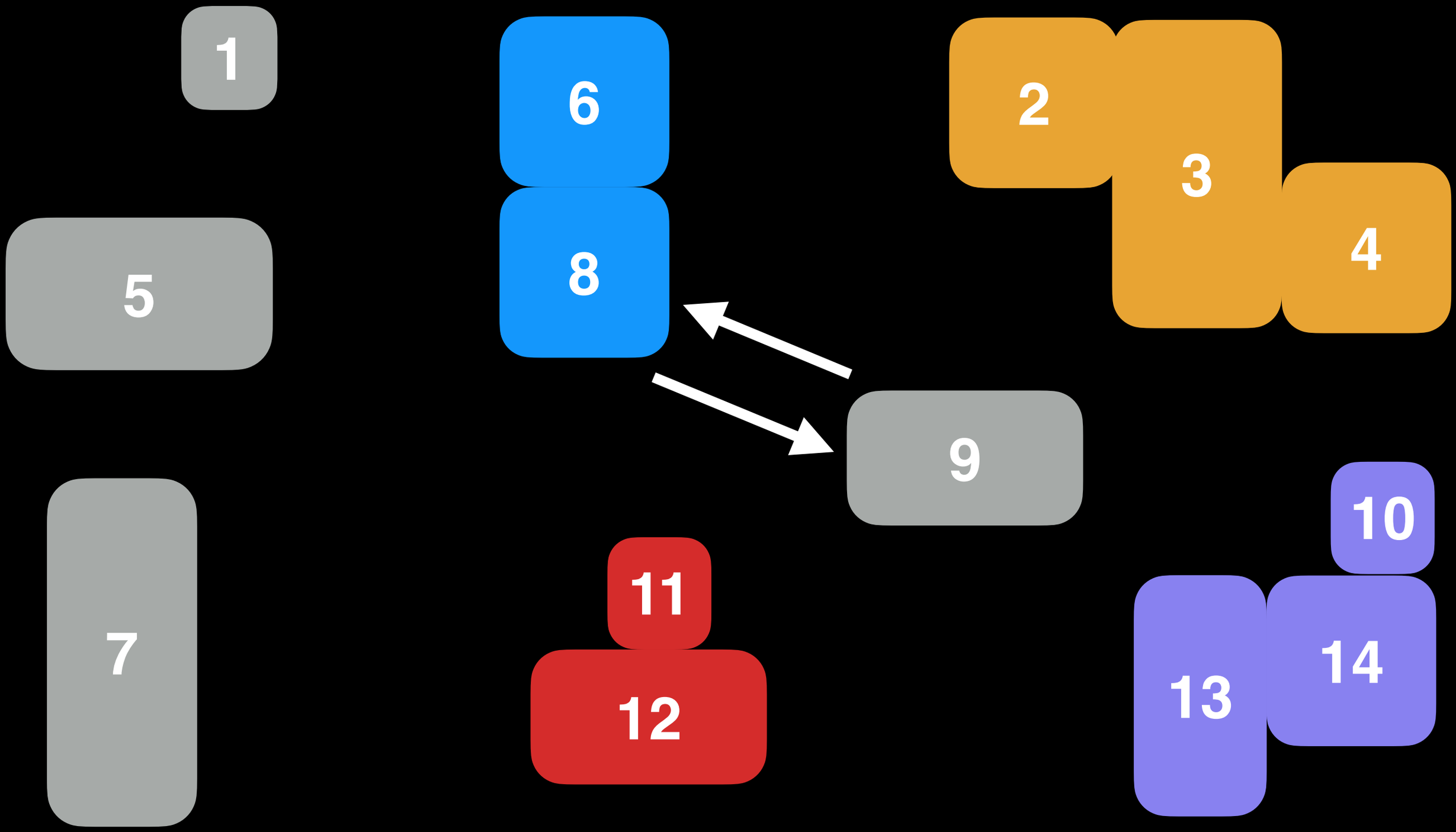
Image processing

Complexity

Construction	O(n)
Union	O(α(n))
Find	O(α(n))
Get component size	O(α(n))
Check if connected	O(α(n))
Count components	O(1)

Where α(n) is the inverse Ackerman function

What is a Union Find?



Union Find

Kruskal's Algorithm

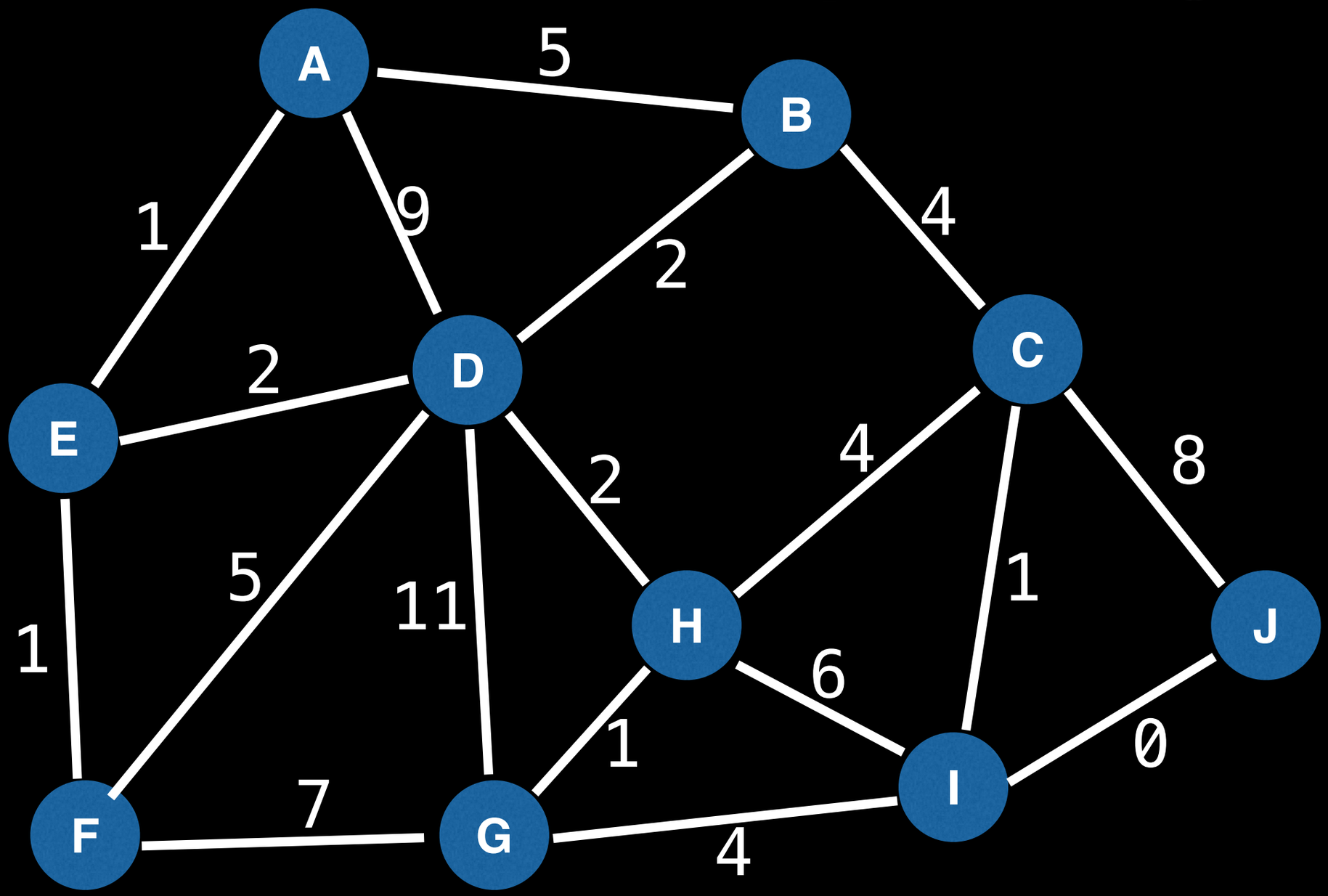
William Fiset

Union Find application: Kruskal's Minimum Spanning Tree

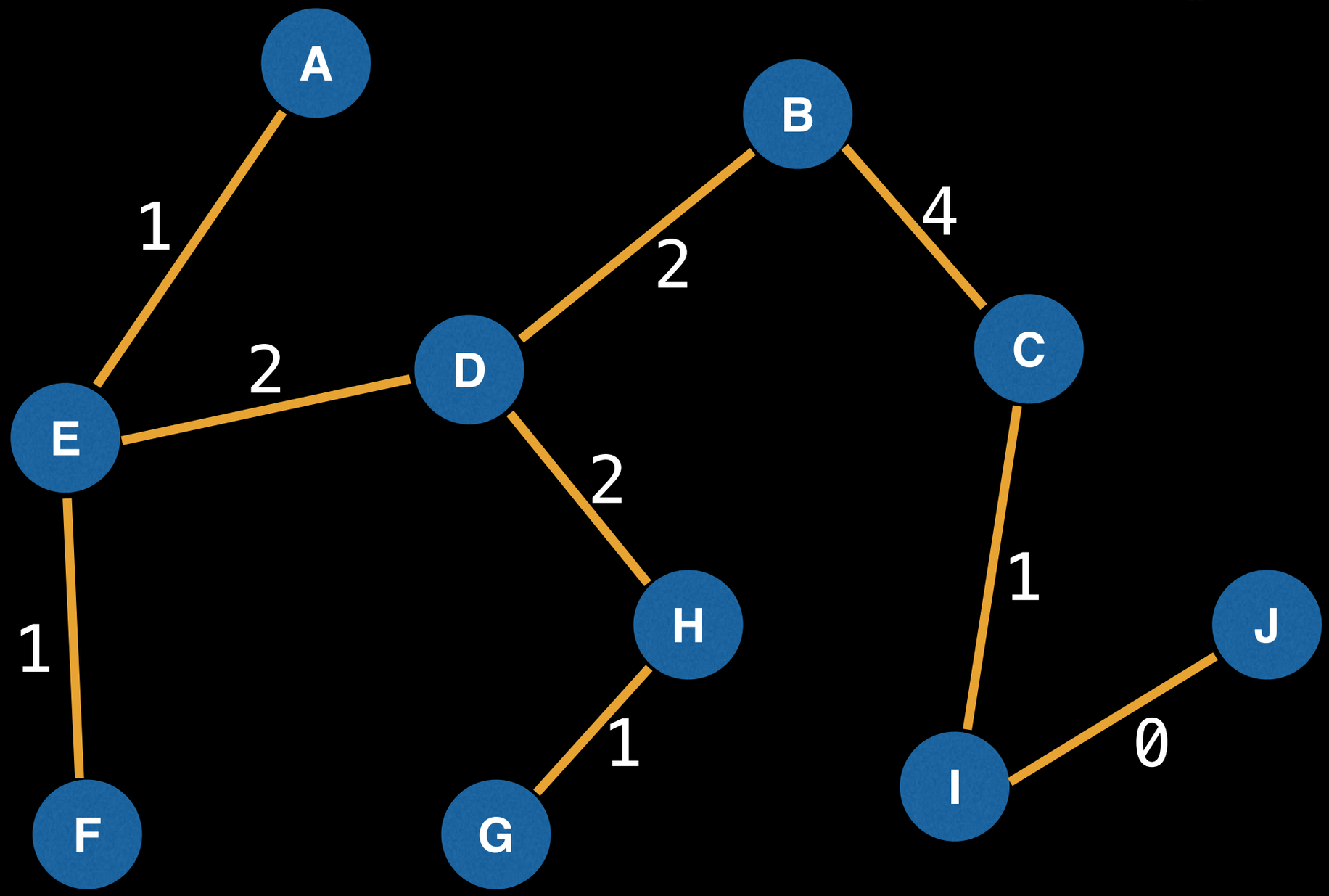
Given a graph G = (V,E) we want to find a **Minimum Spanning Tree** in the graph (it may not be unique).

A minimum spanning tree is a subset of the edges which connect[s] all vertices in the graph with the minimal total edge cost and which contains no cycles

Union Find application: Kruskal's Minimum Spanning Tree



Union Find application: Kruskal's Minimum Spanning Tree



Minimum spanning tree with weight 14

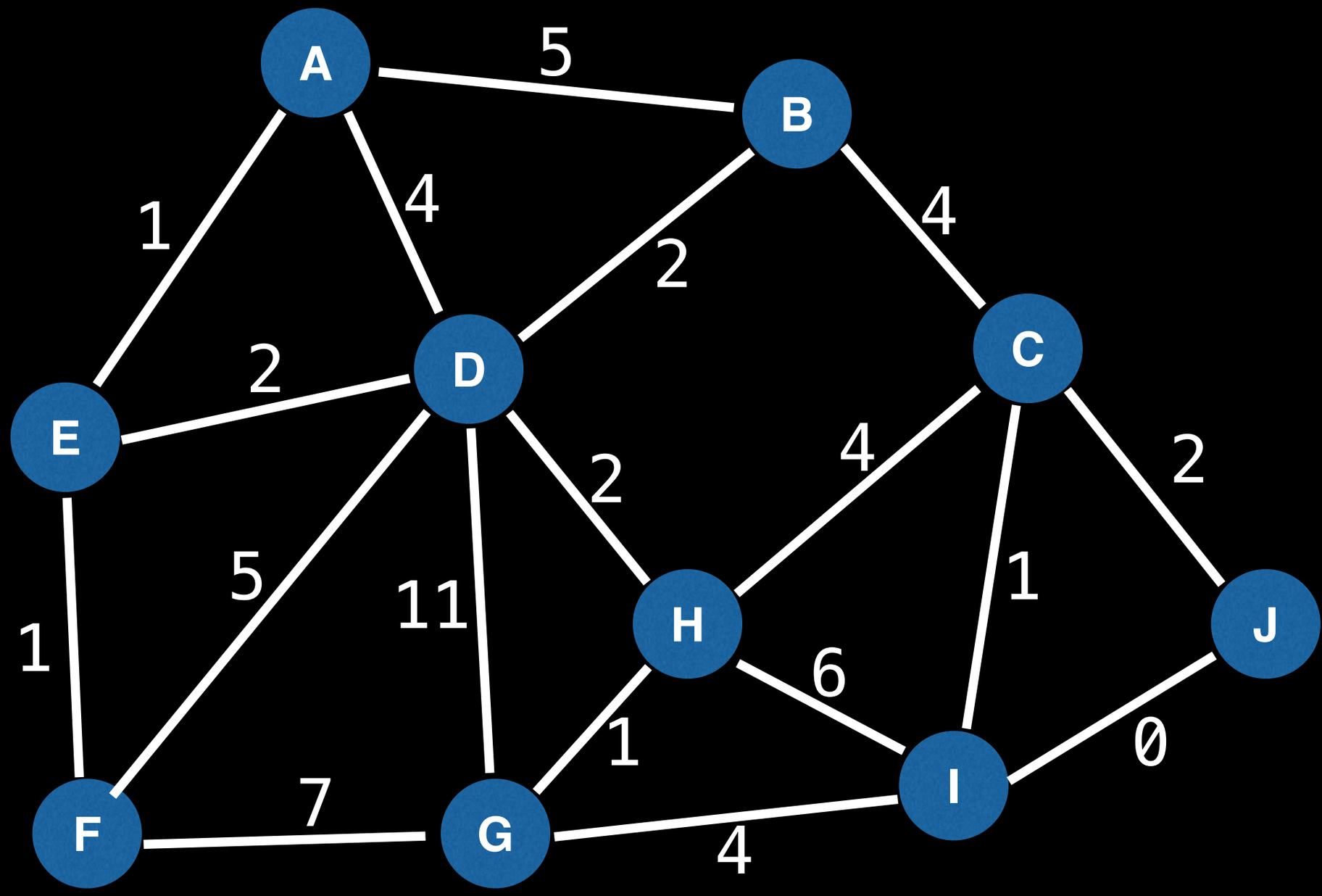
Union Find application: Kruskal’s Minimum Spanning Tree

1) Sort edges by ascending edge weight.

2) Walk through the sorted edges and look at the two nodes the edge belongs to, if the nodes are already unified we don’t include this edge, otherwise we include it and unify the nodes.

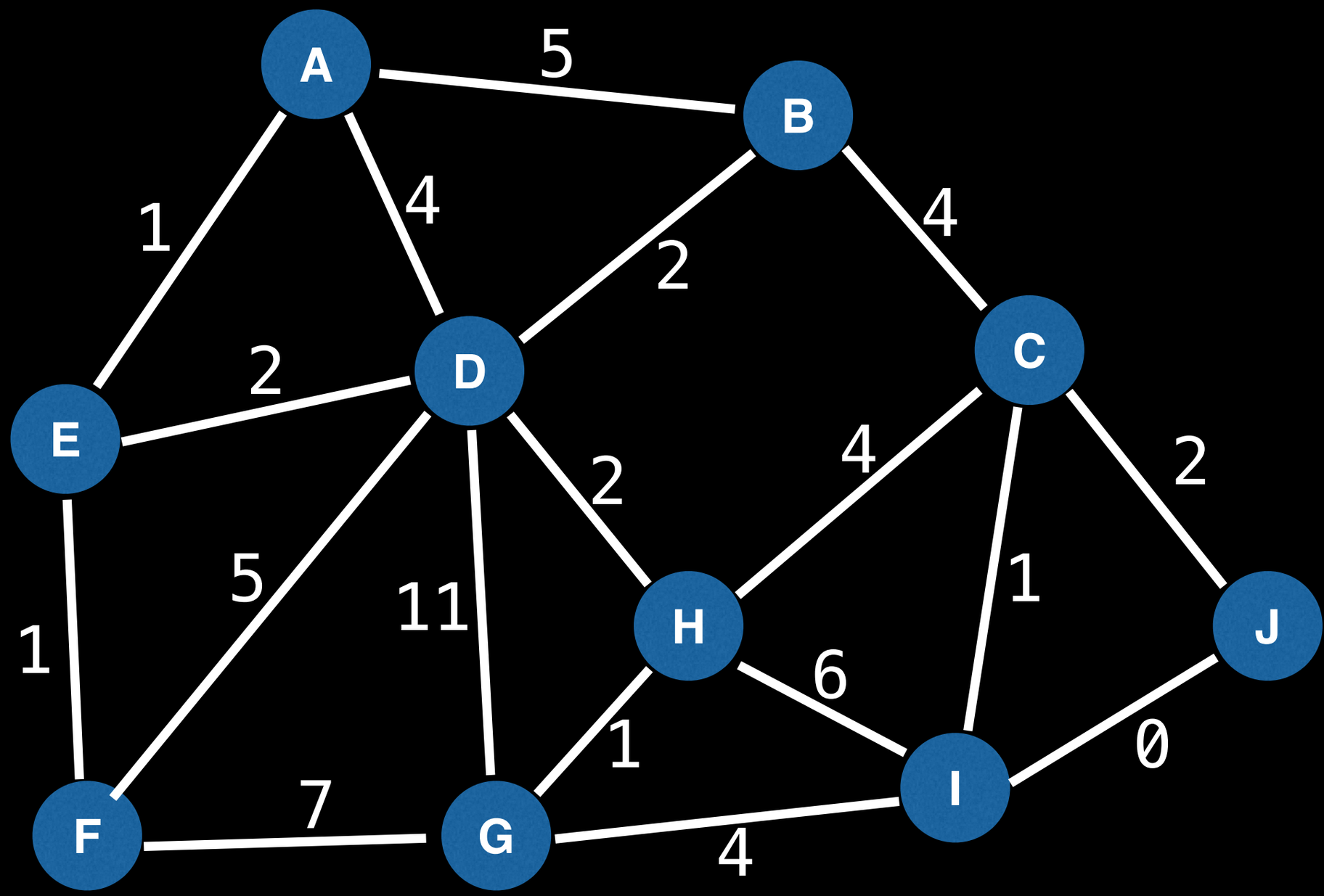
3) The algorithm terminates when every edge has been processed or all the vertices have been unified.

Union Find application: Kruskal's Minimum Spanning Tree



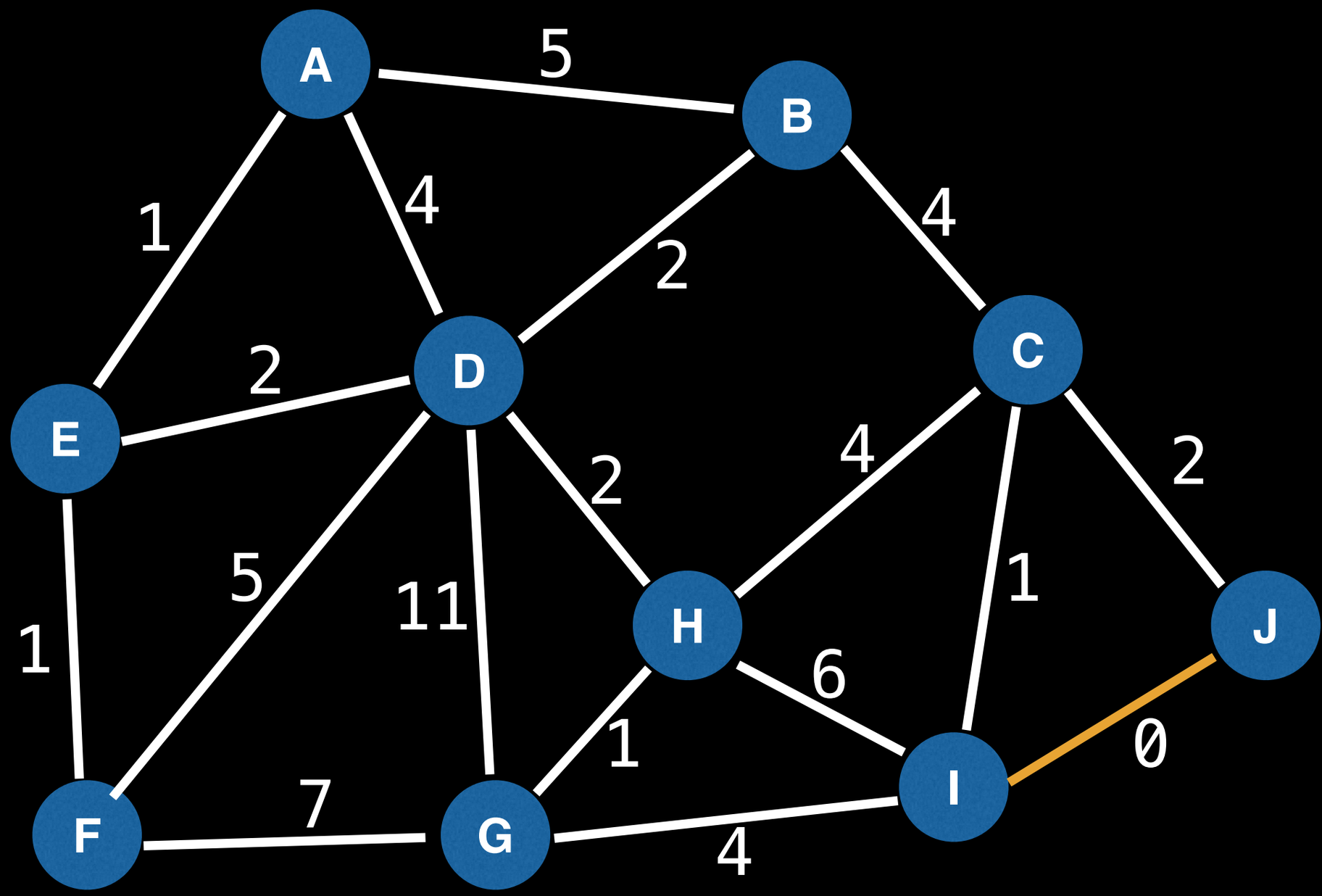
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



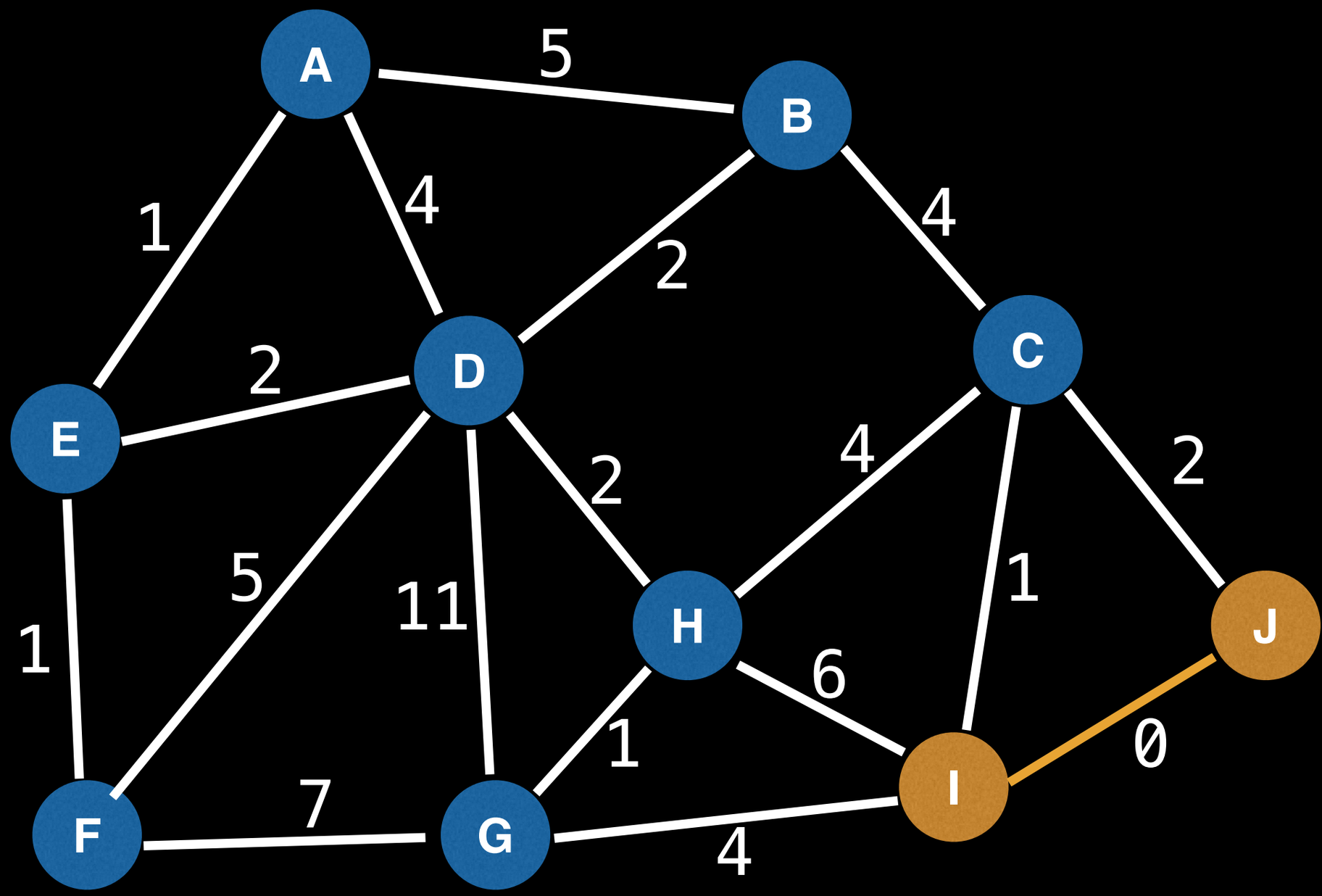
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



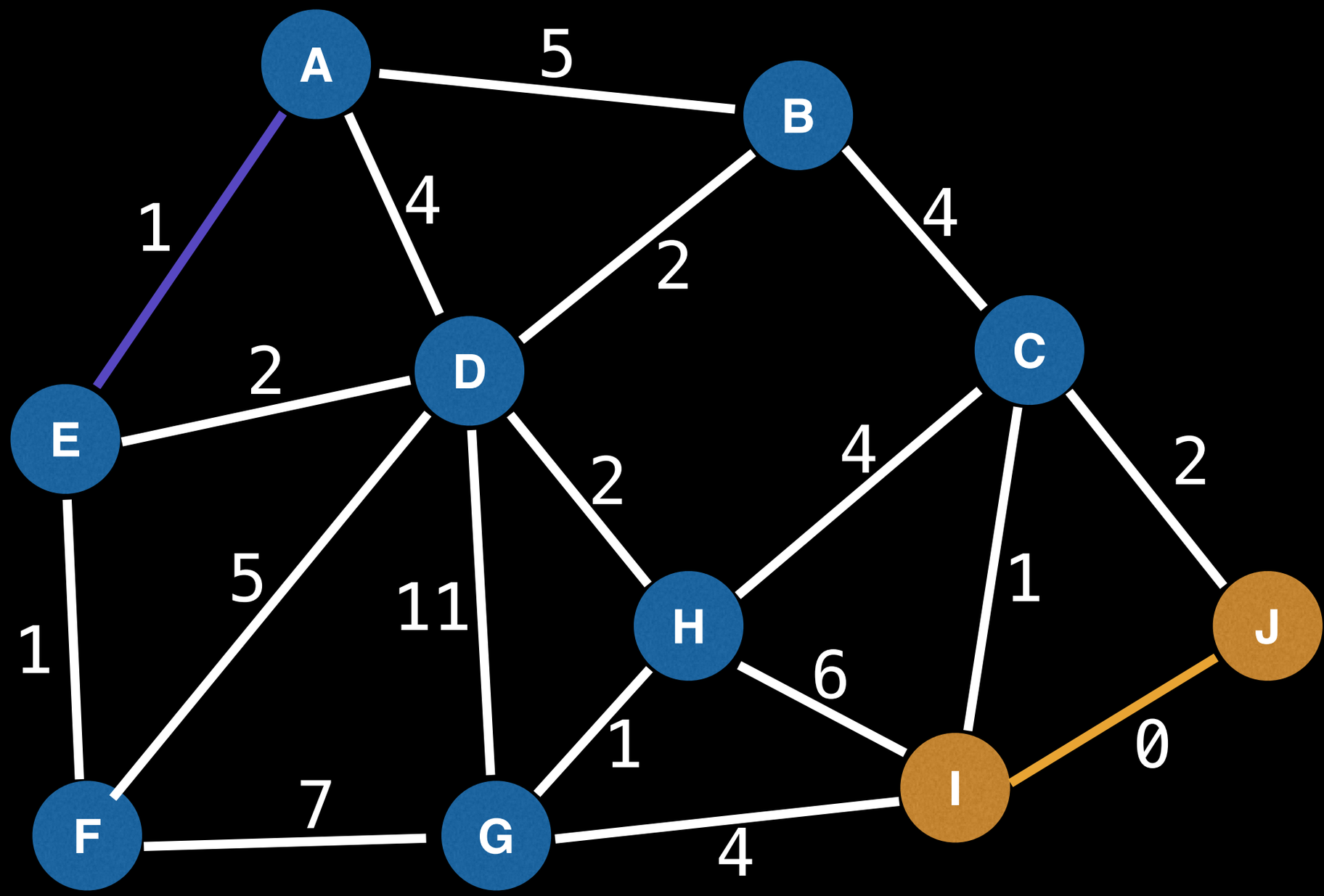
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



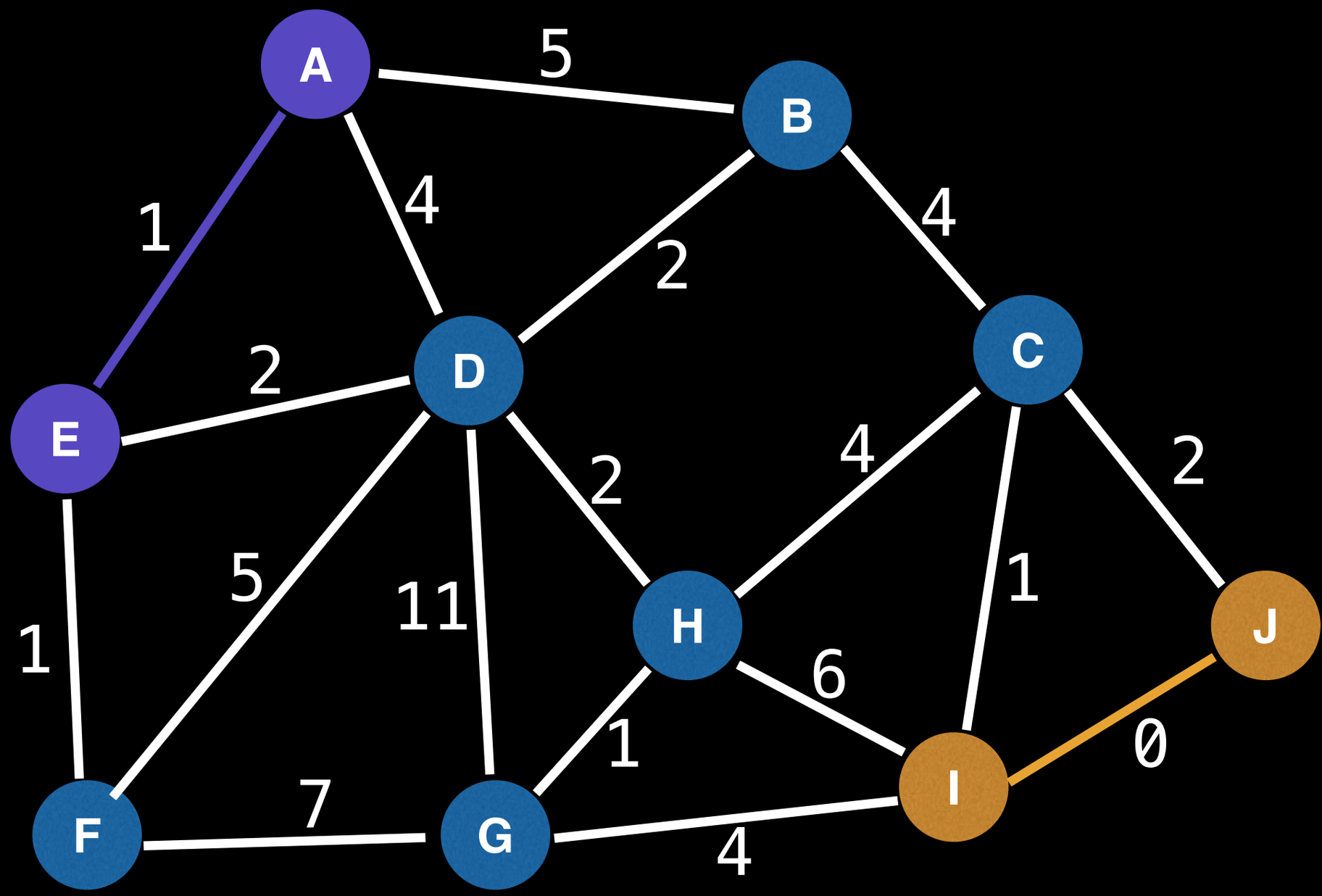
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



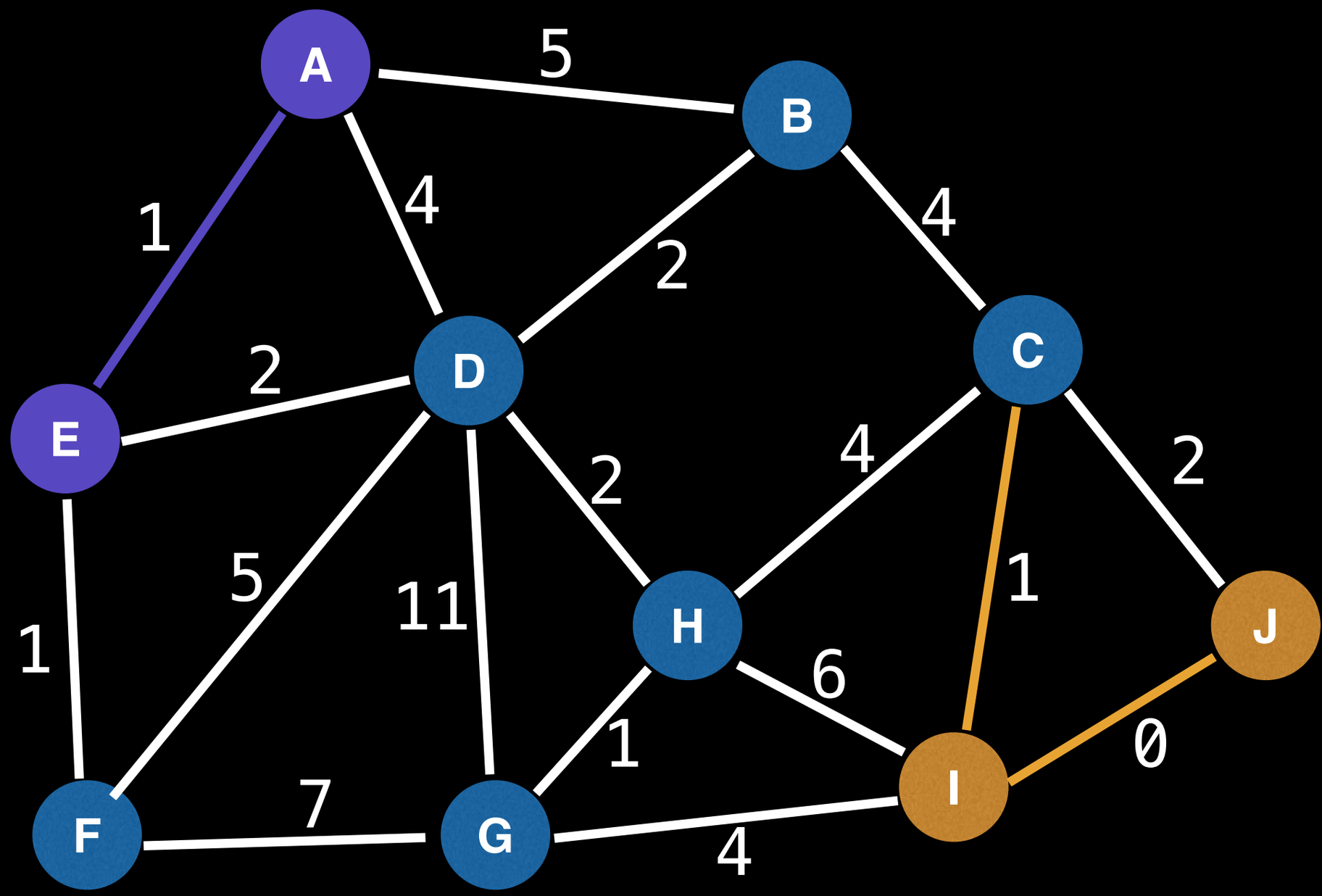
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



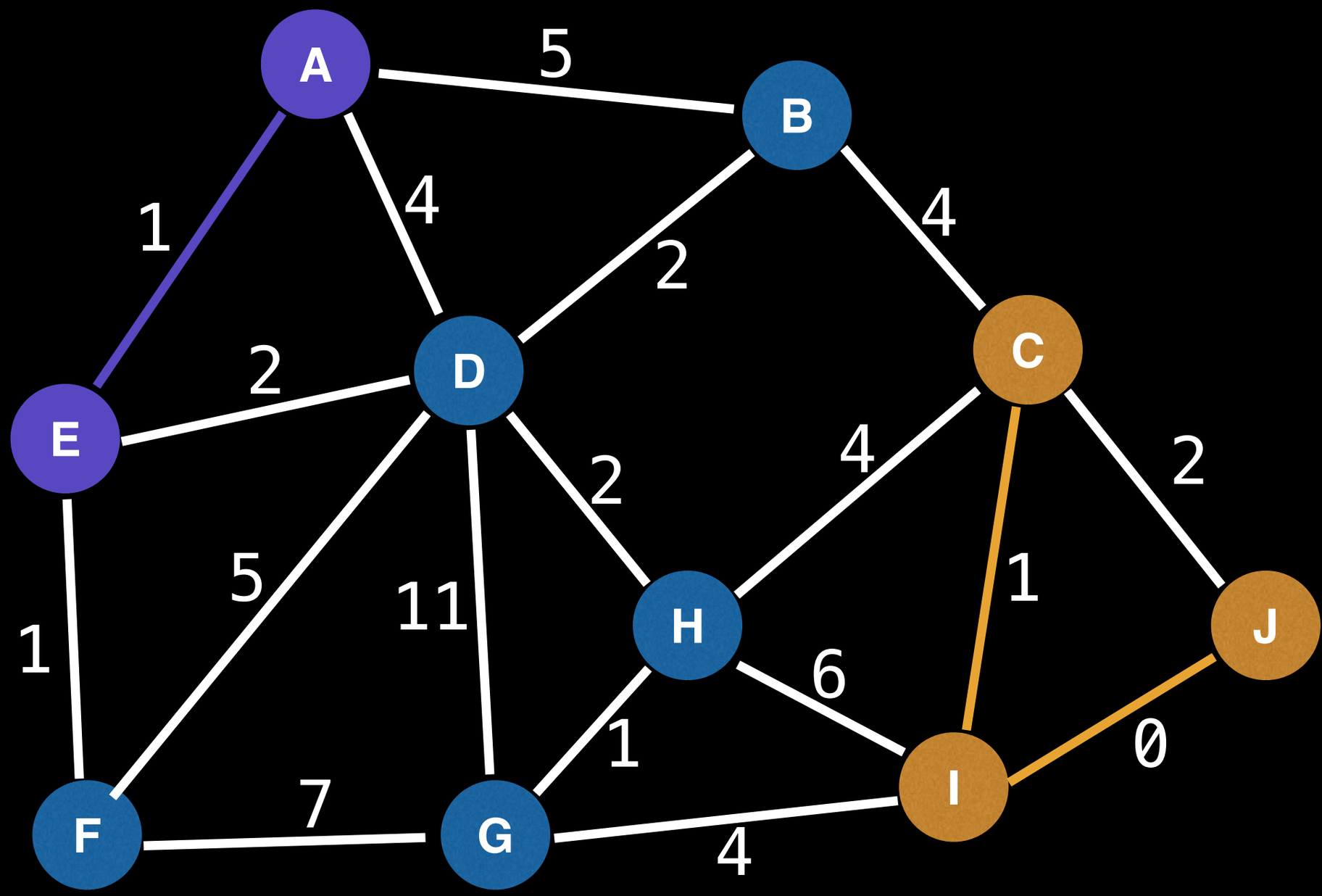
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



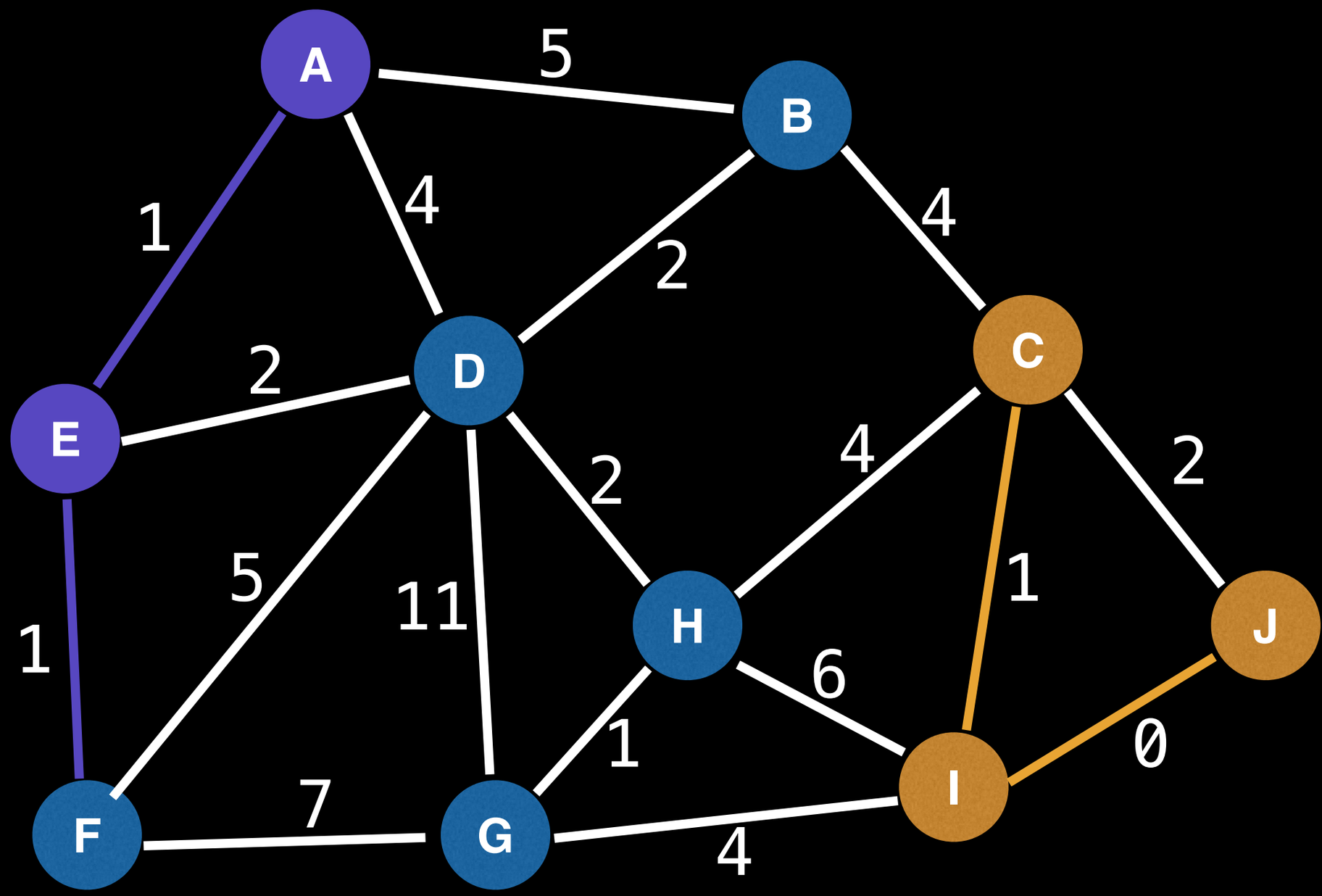
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



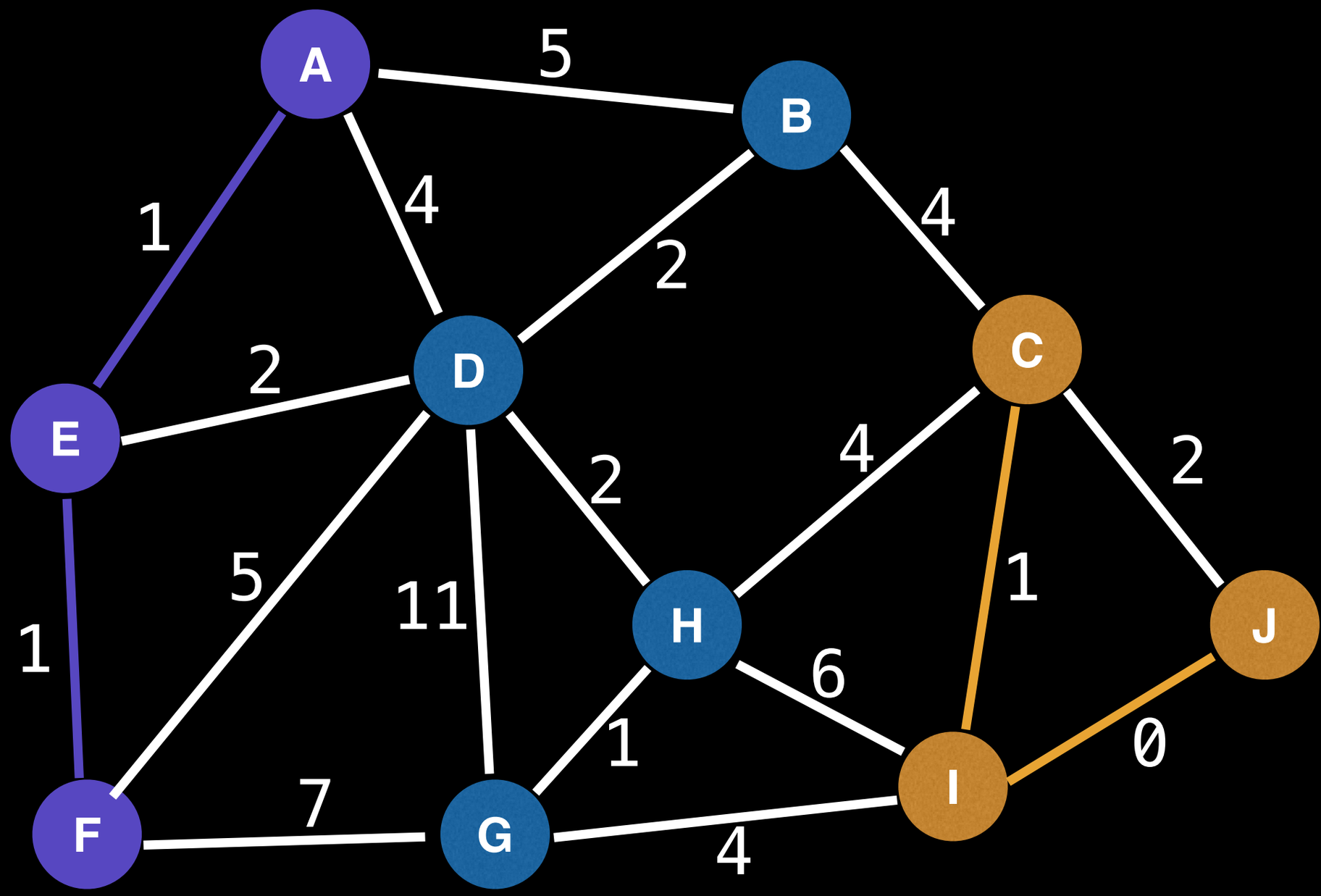
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



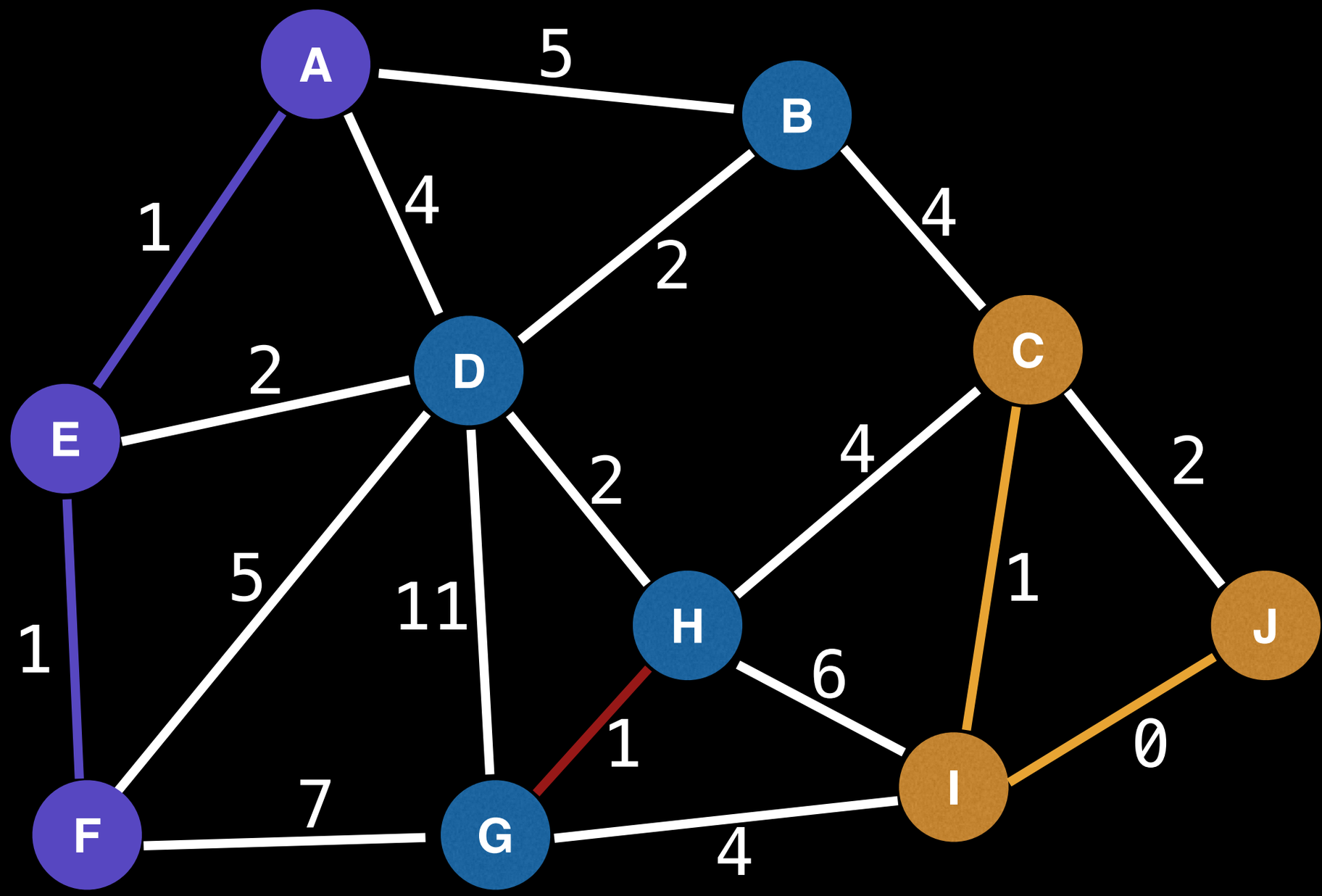
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



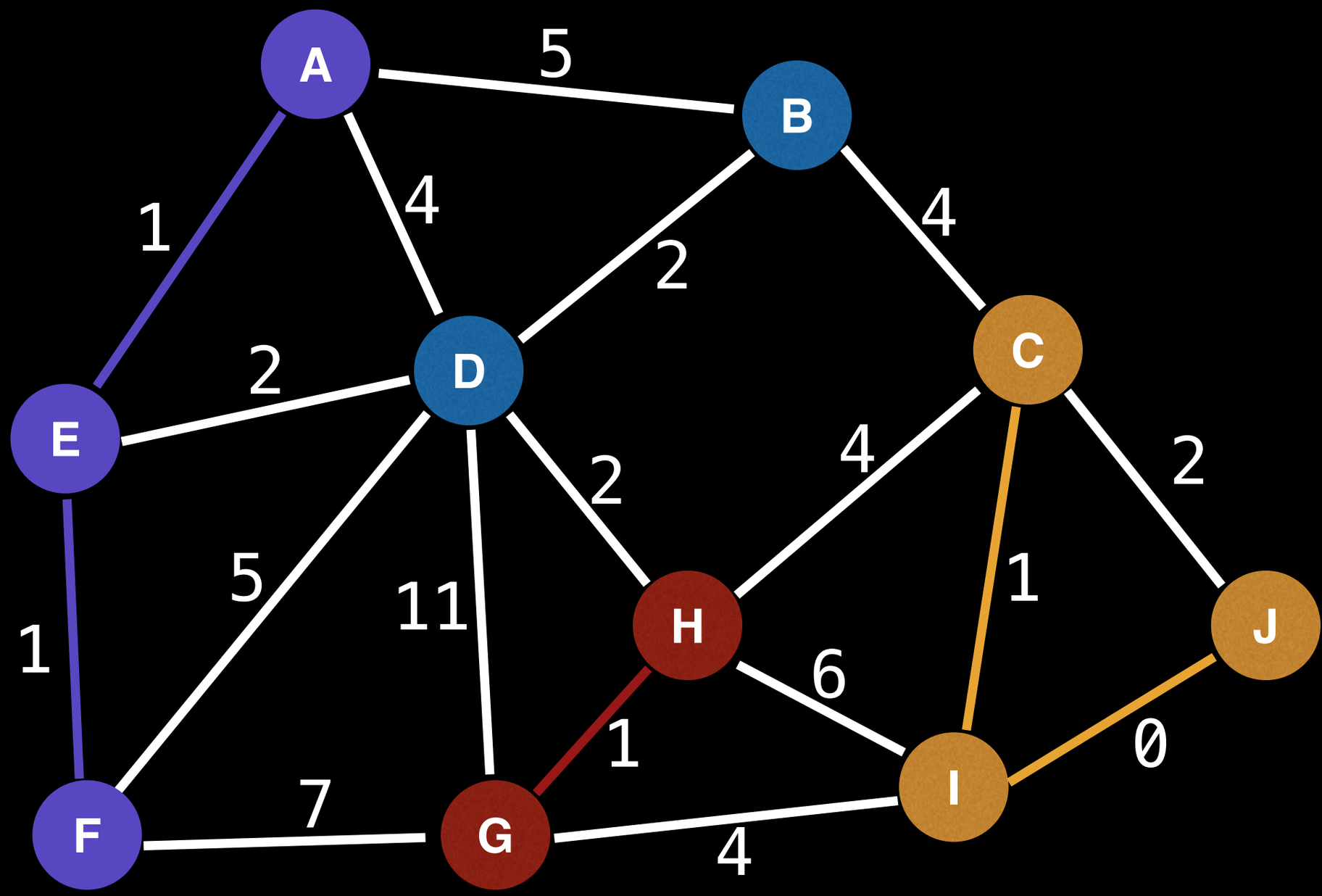
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



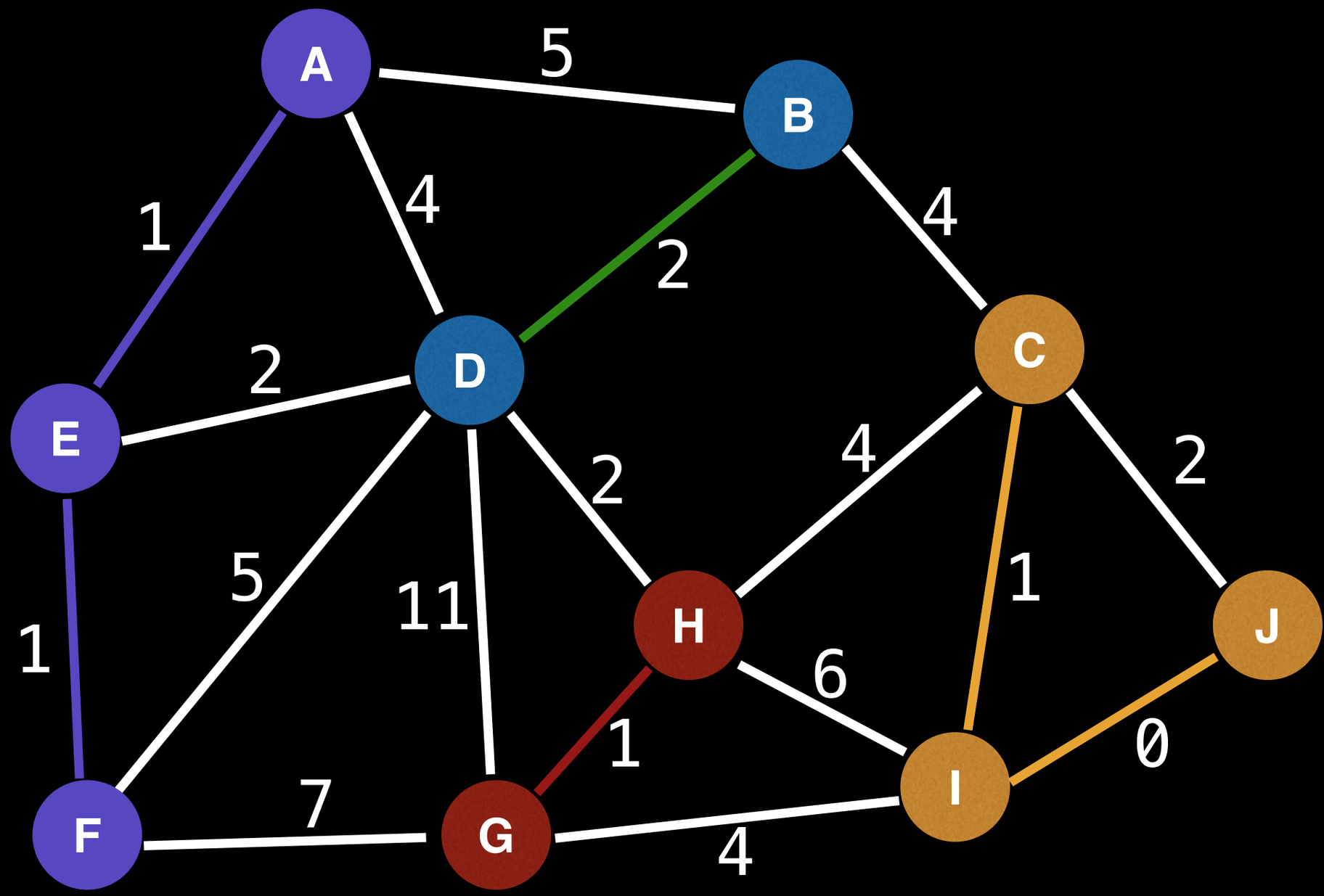
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



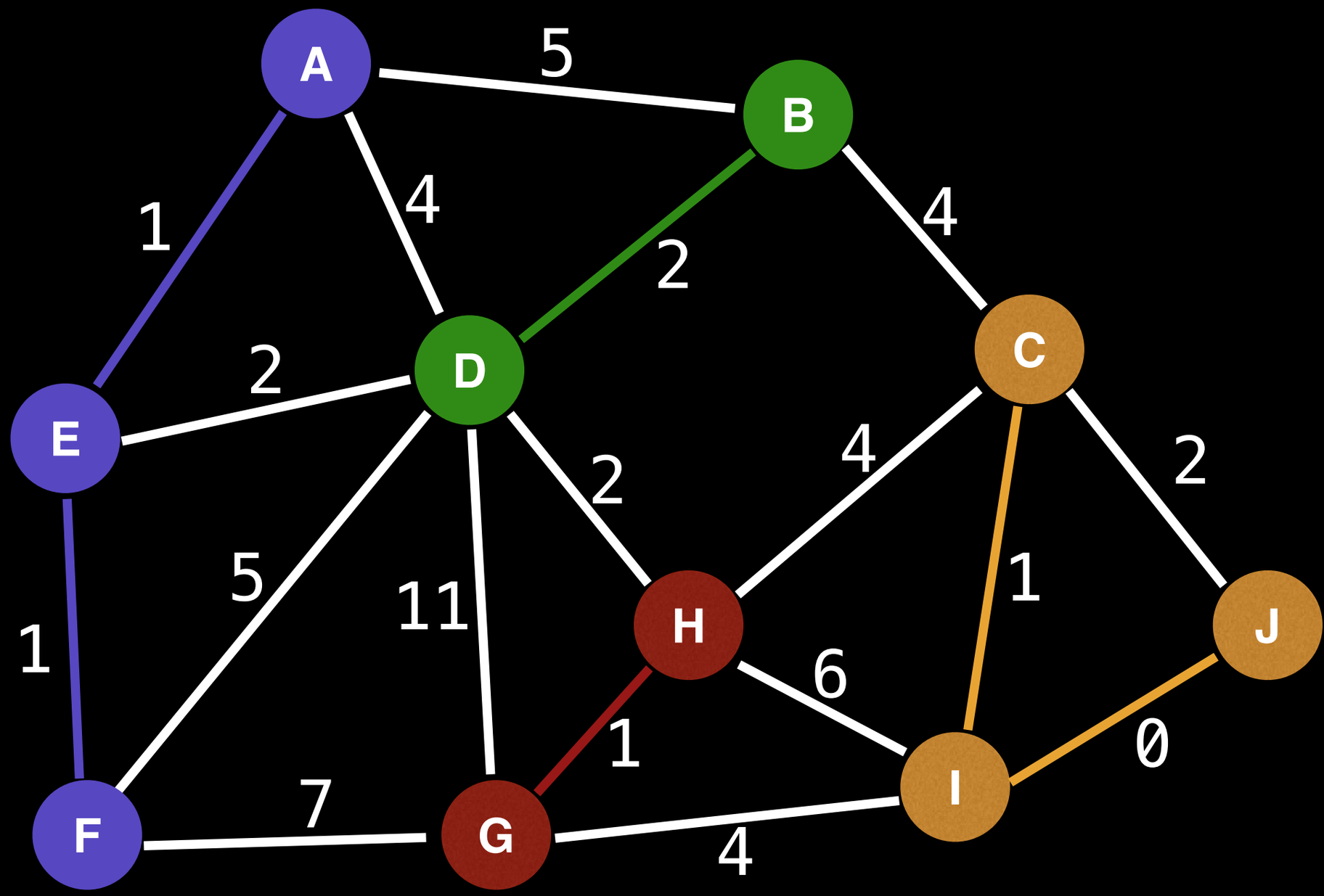
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



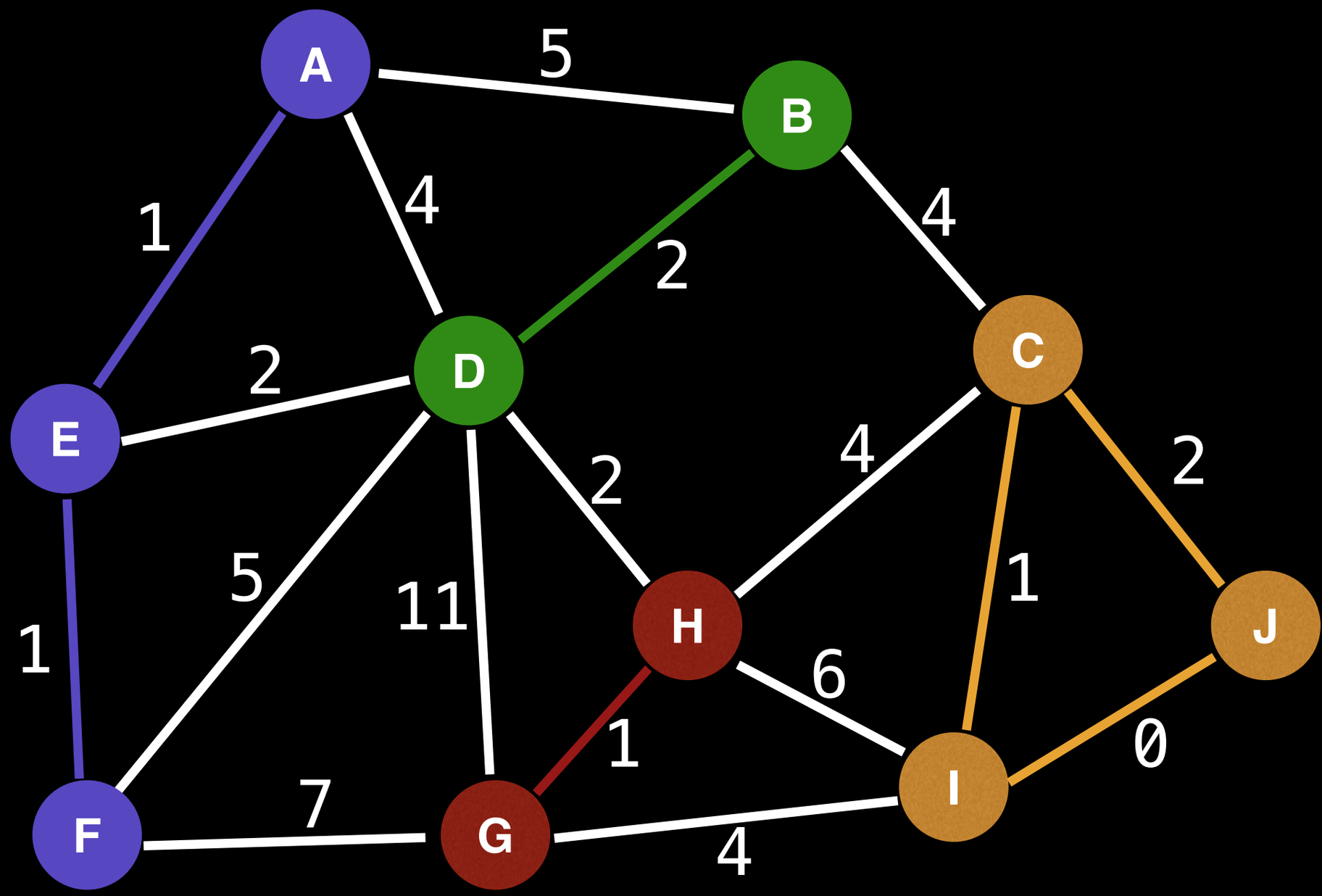
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



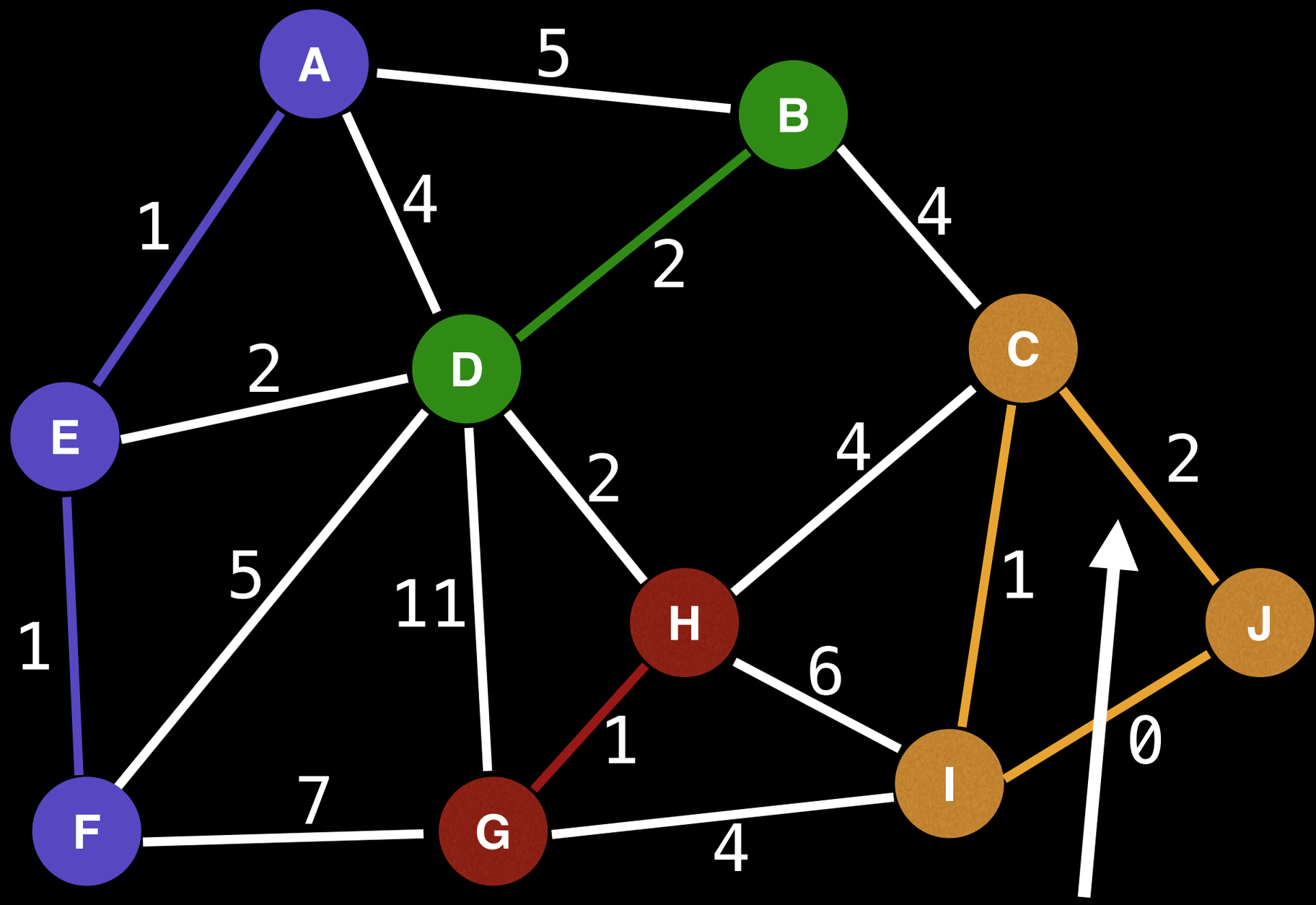
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



Union Find application: Kruskal's Minimum Spanning Tree

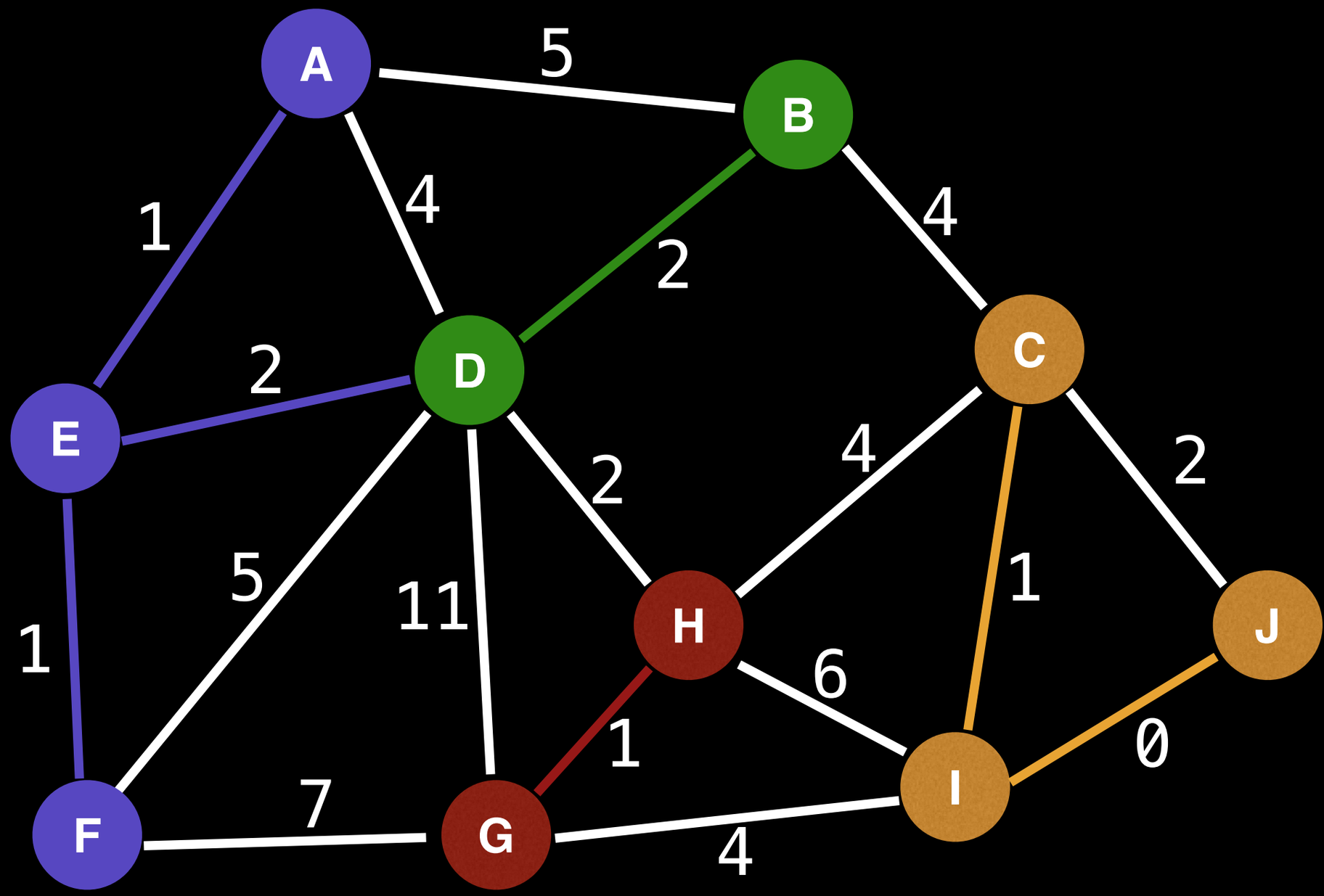
I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



Nodes C,J are already connected in yellow group. This creates a cycle

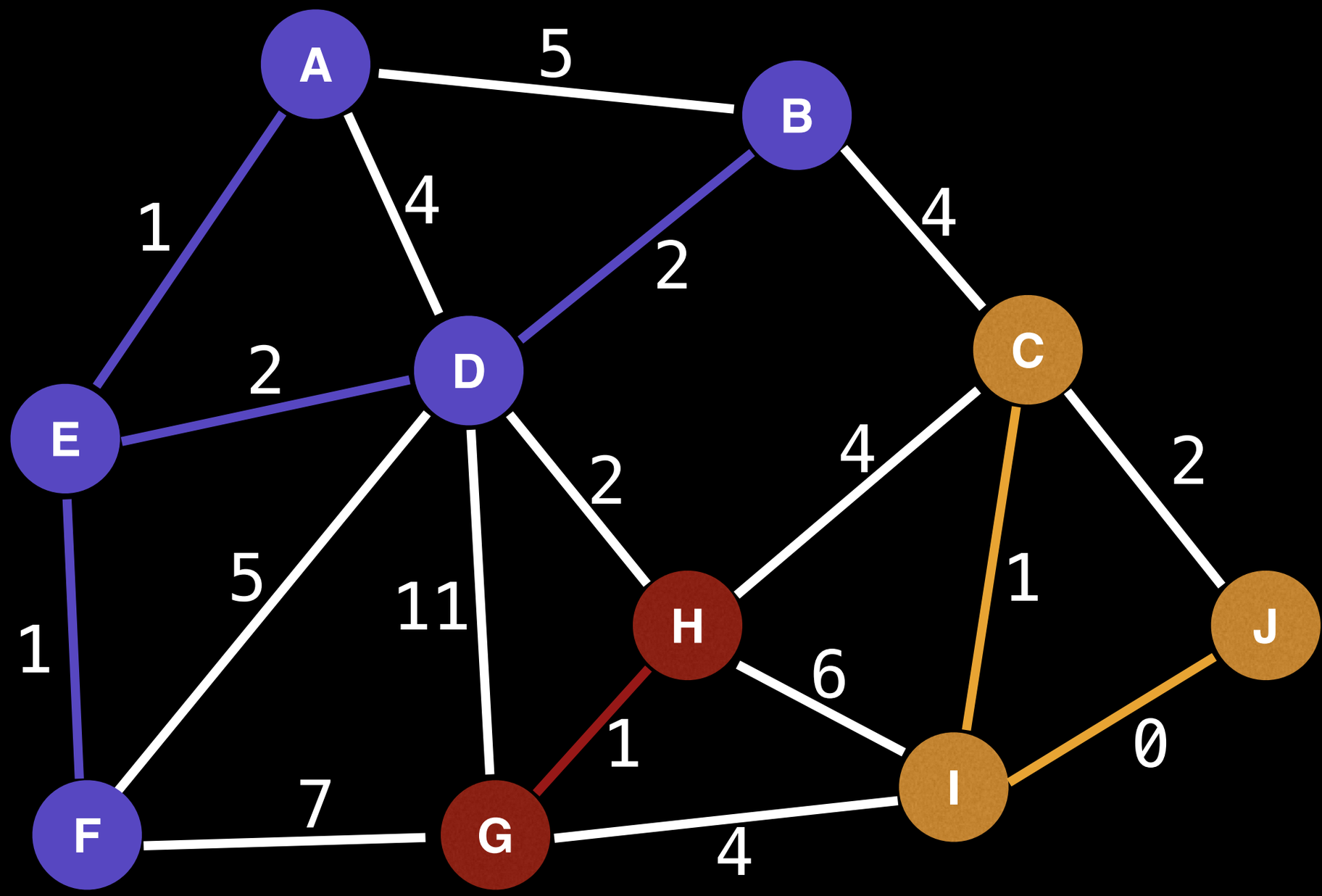
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



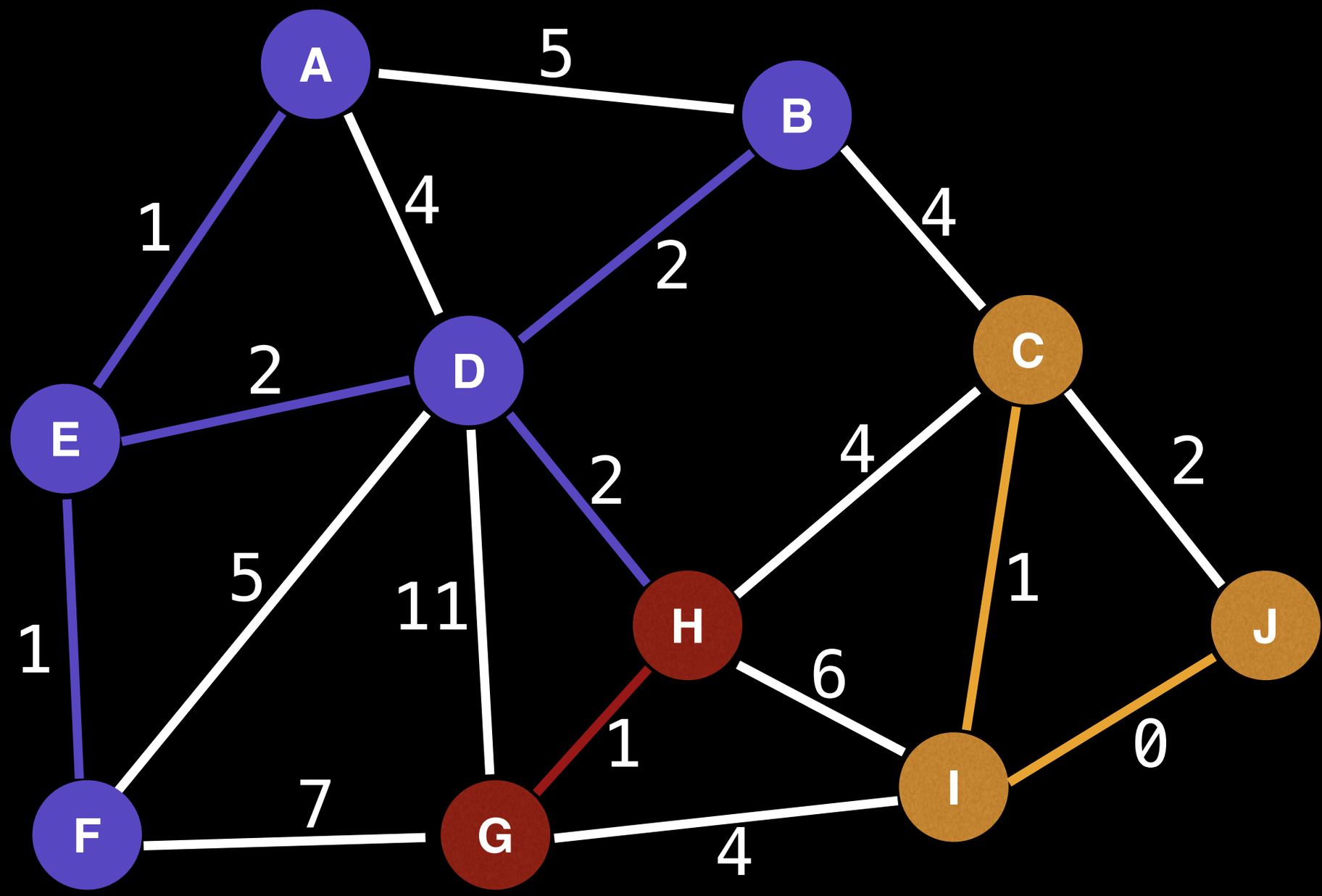
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



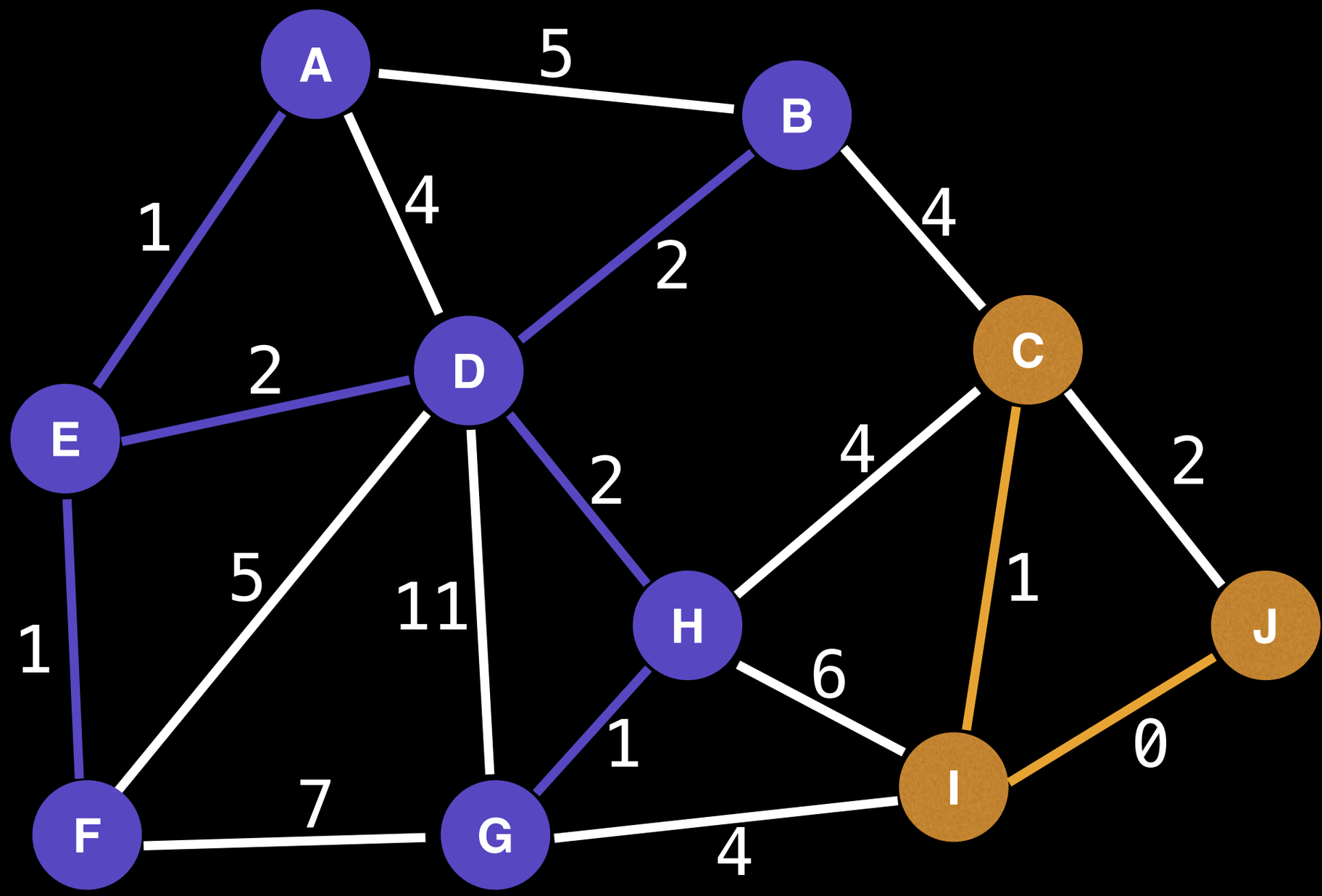
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



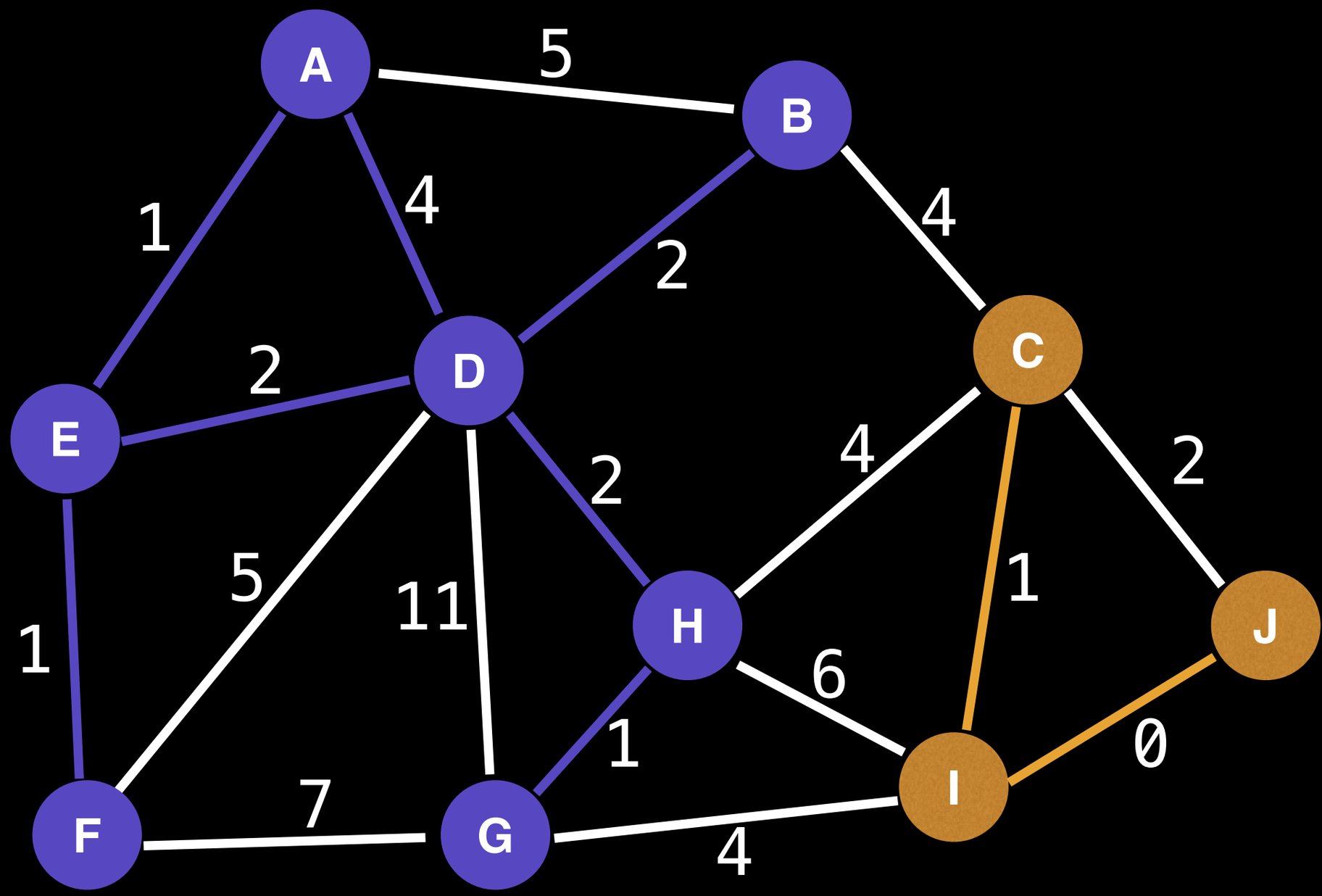
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



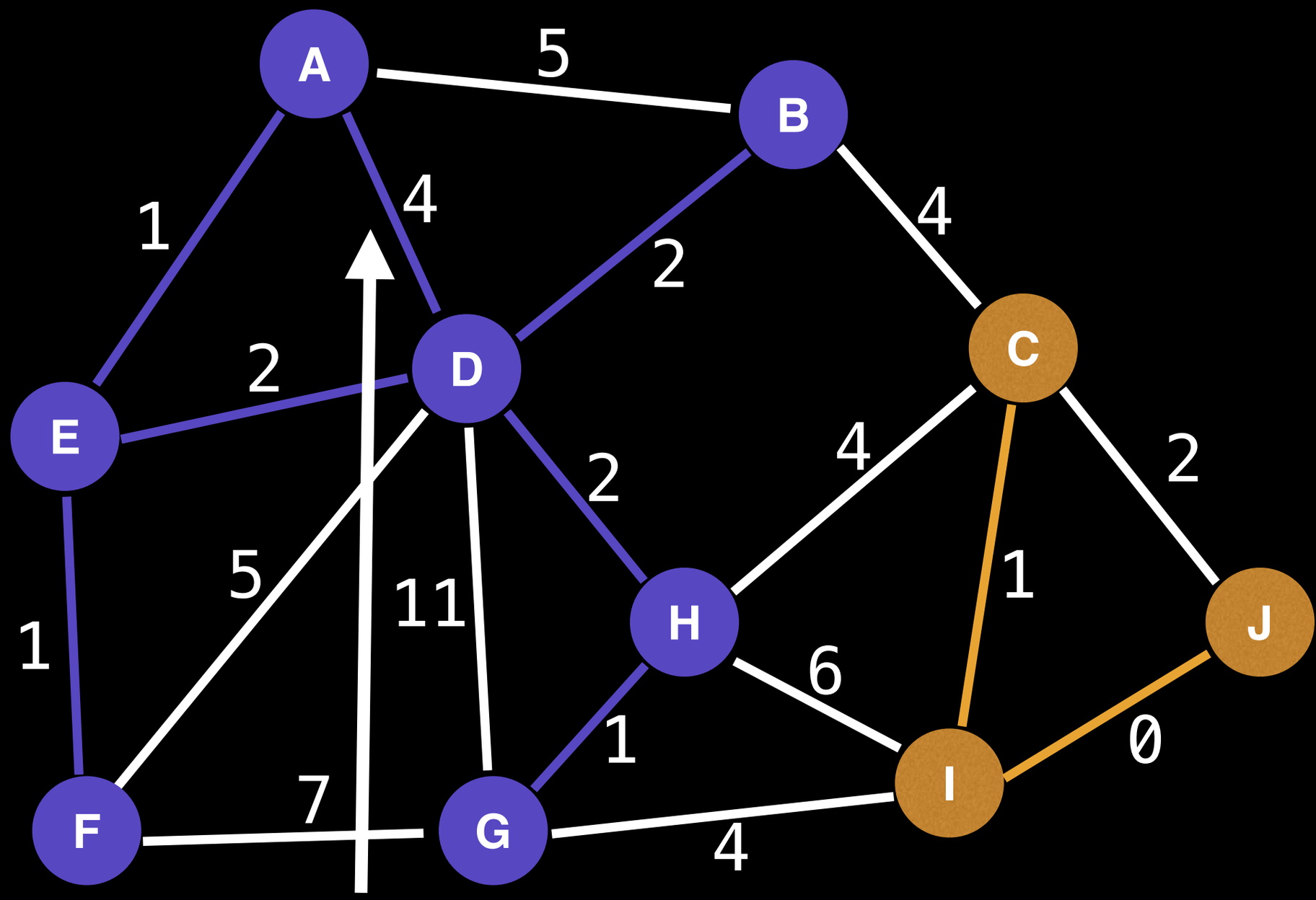
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



Union Find application: Kruskal's Minimum Spanning Tree

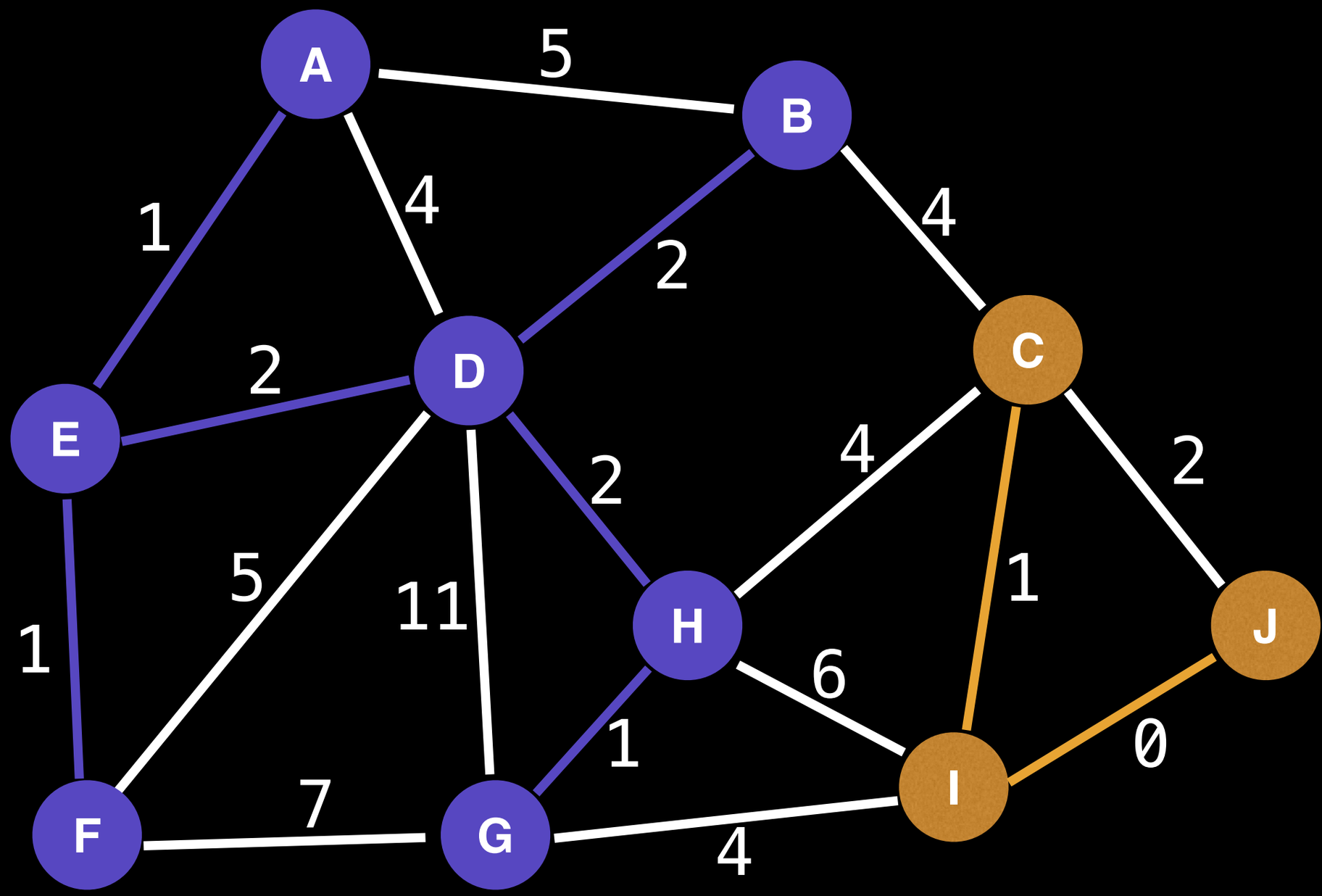
I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



Nodes A,D are already connected in purple group. This creates a cycle

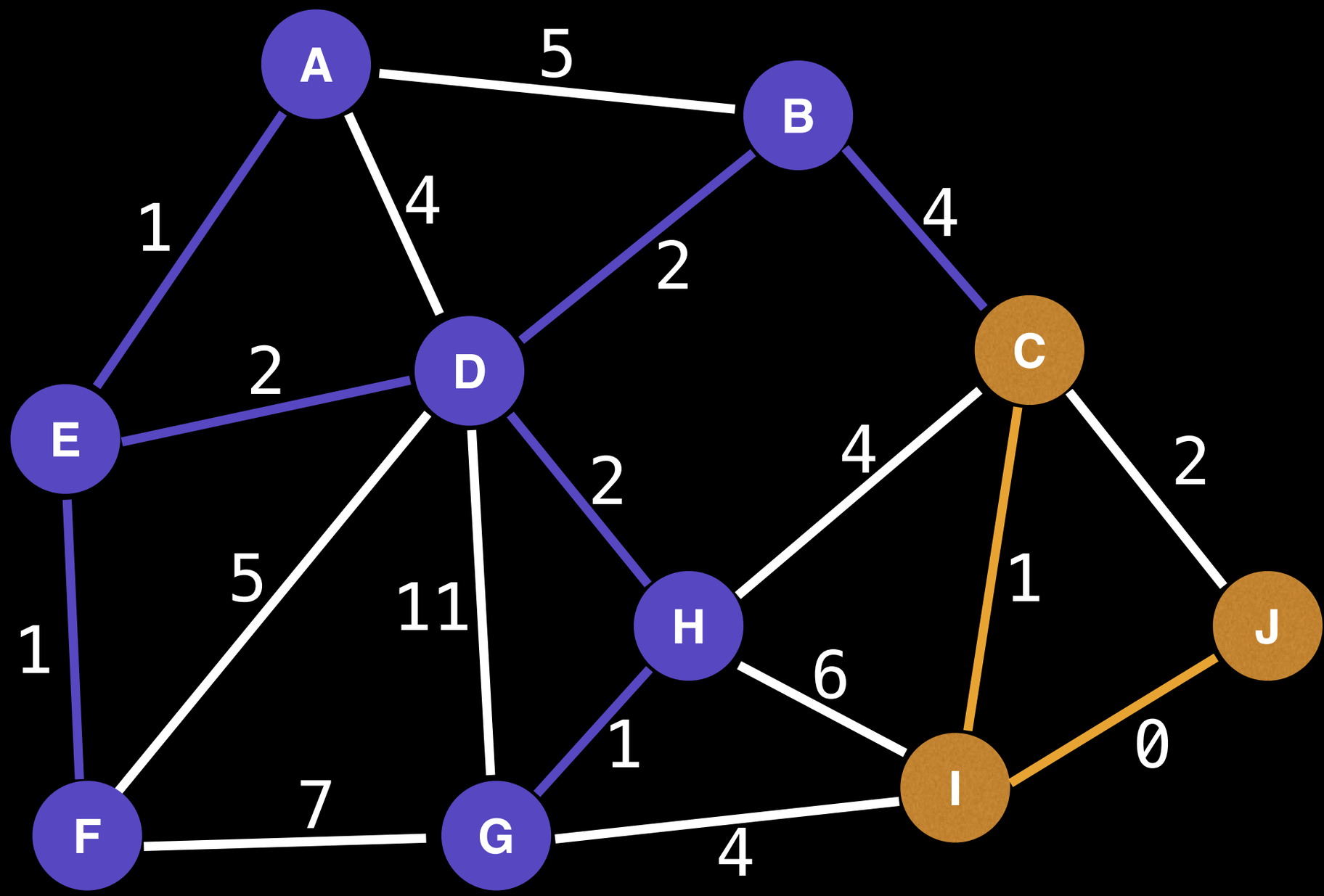
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



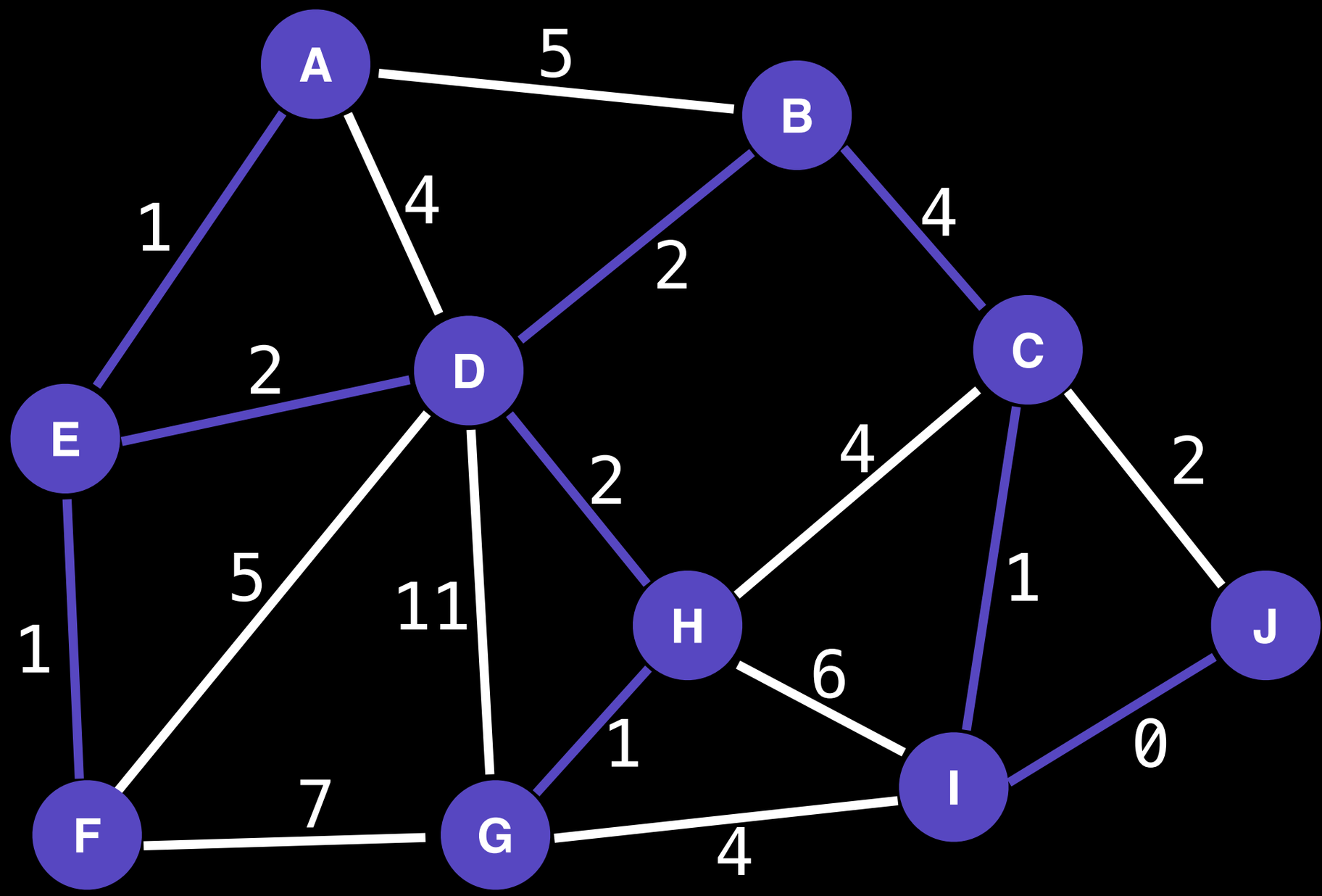
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



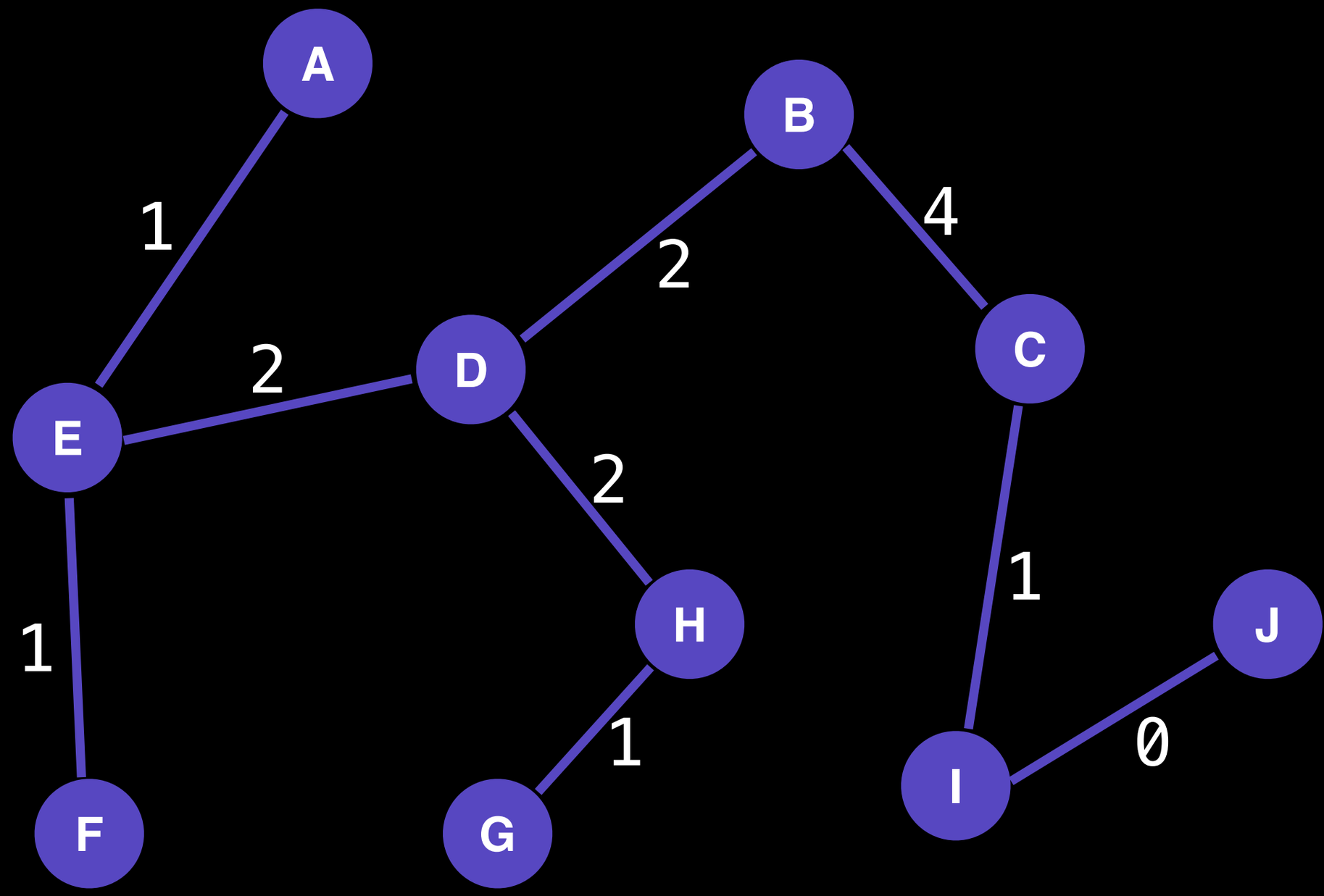
Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11

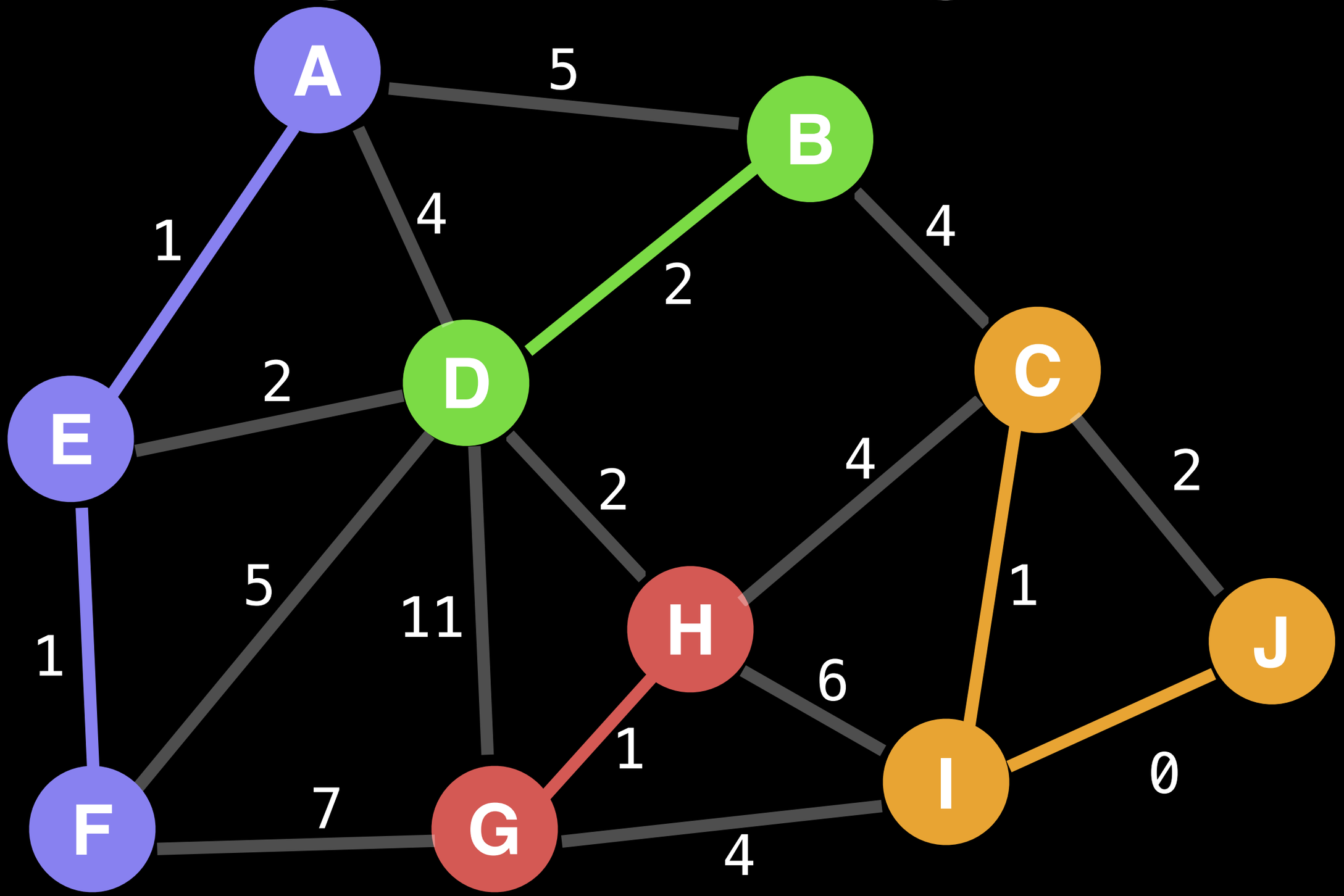


Union Find application: Kruskal's Minimum Spanning Tree

I to J = 0
A to E = 1
C to I = 1
E to F = 1
G to H = 1
B to D = 2
C to J = 2
D to E = 2
D to H = 2
A to D = 4
B to C = 4
C to H = 4
G to I = 4
A to B = 5
D to F = 5
H to I = 6
F to G = 7
D to G = 11



Kruskal's Minimum Spanning Tree Algorithm



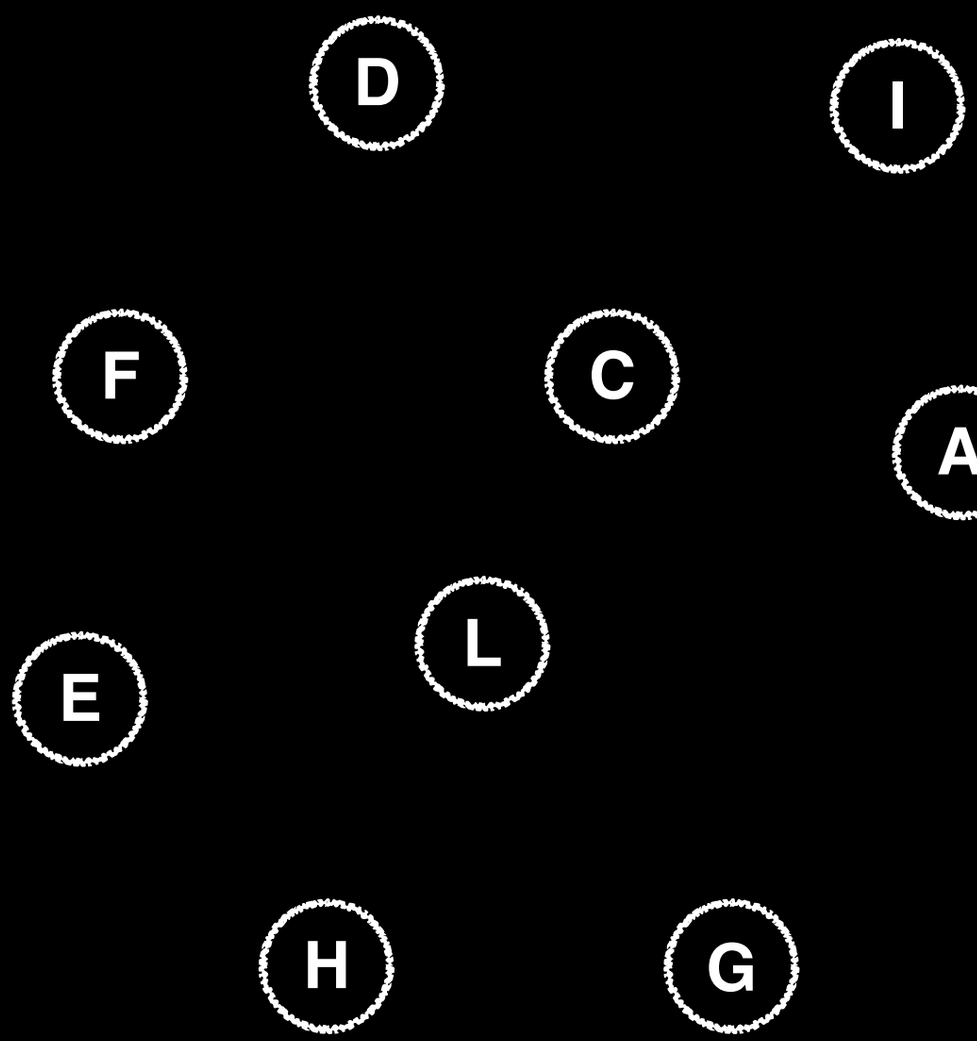
Union and Find Operations

Creating Union Find

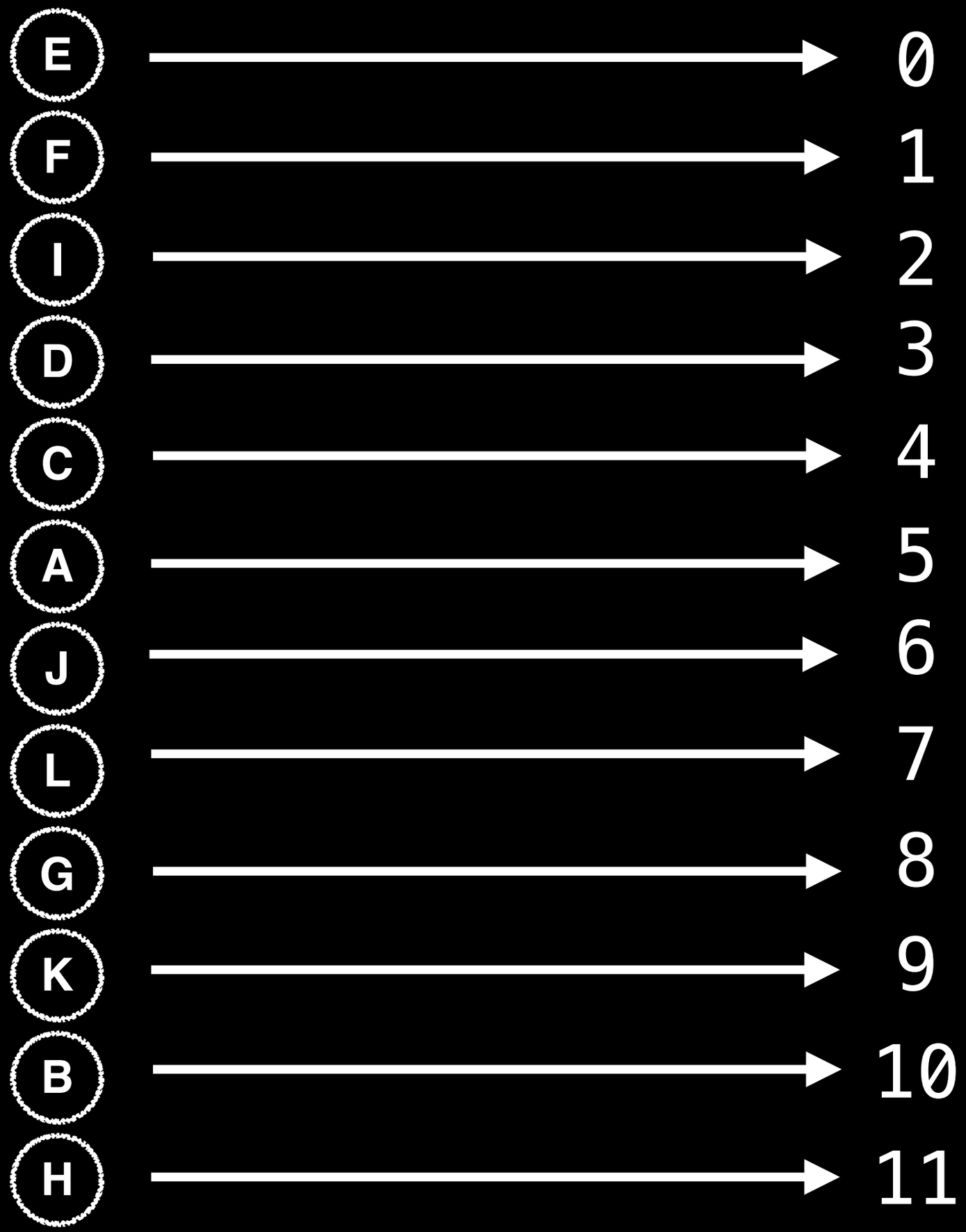
To begin using Union Find, first construct a **bijection** (a mapping) between your objects and the integers in the range [0, n).

NOTE: This step is not necessary in general, but it will allow us to construct an array-based union find.

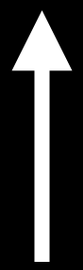
Randomly assign a mapping between the objects and the integers on the right.



0
1
2
3
4
5
6
7
8
9
10
11


E
F
I
D
C
A
J
L
G
K
B
H
0
1
2
3
4
5
6
7
8
9
10
11

E	F	I	D	C	A	J	L	G	K	B	H
0	1	2	3	4	5	6	7	8	9	10	11
0	1	2	3	4	5	6	7	8	9	10	11

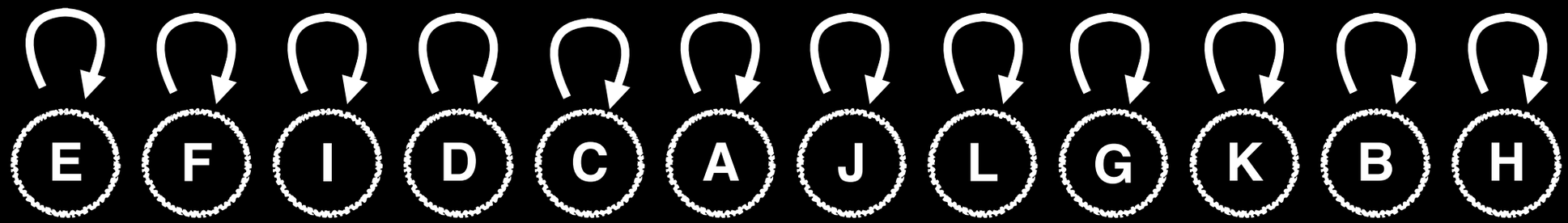


Store Union Find information in an array. Each index has an associated object (letter in this example) we can lookup through our mapping.

E	F	I	D	C	A	J	L	G	K	B	H
0	1	2	3	4	5	6	7	8	9	10	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

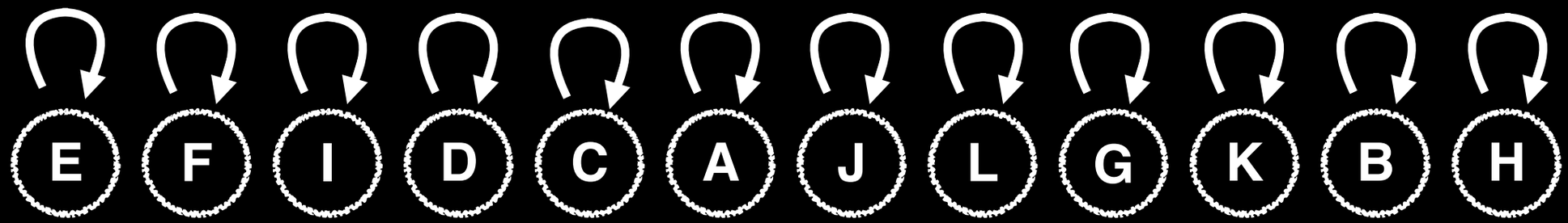


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	1	2	3	4	5	6	7	8	9	10	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

Union(C,K)←
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)

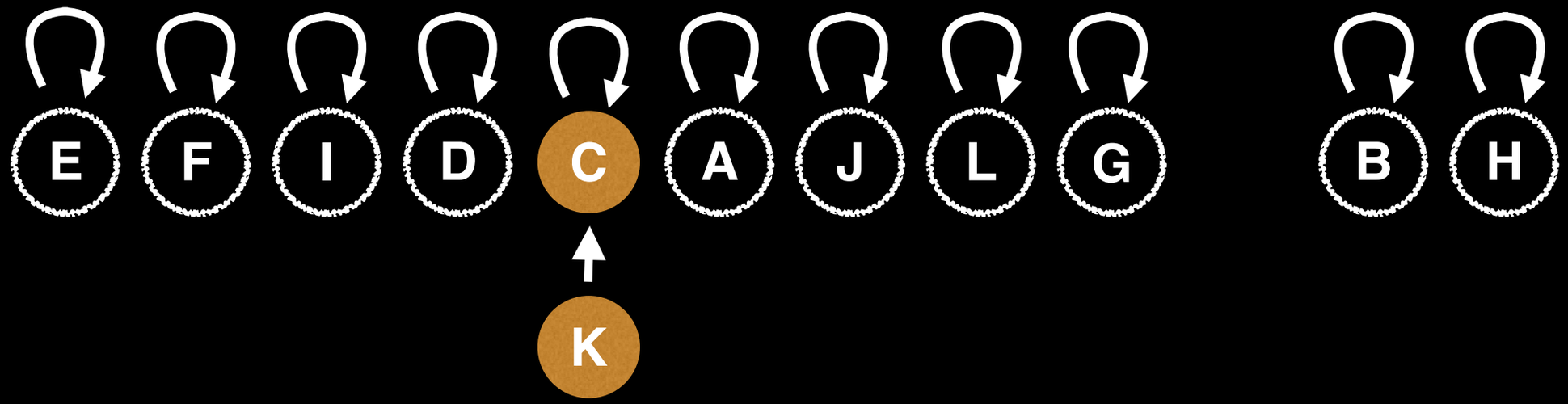


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	1	2	3	4	5	6	7	8	4	10	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

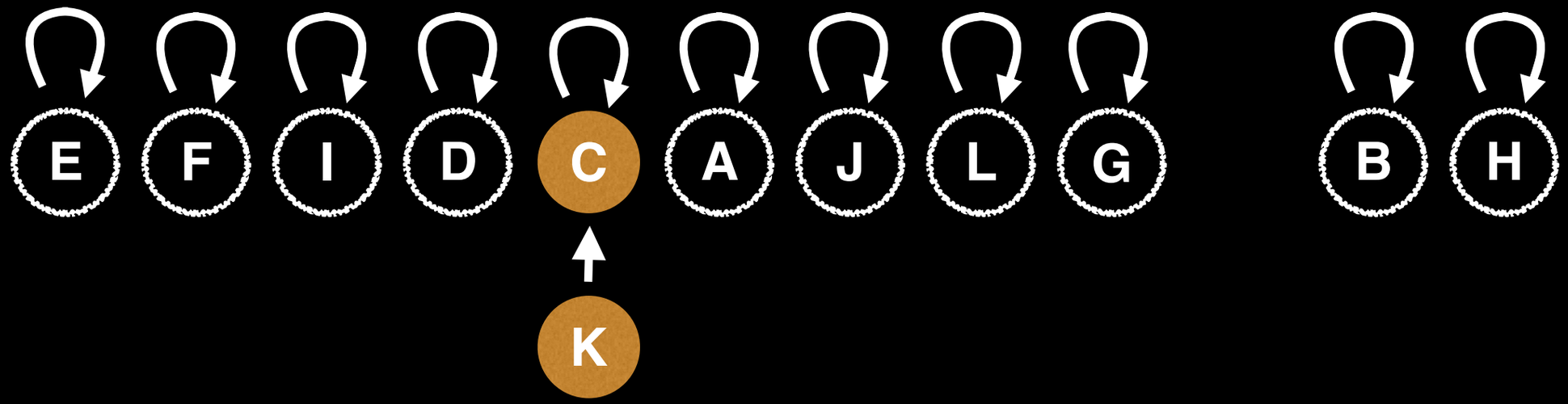


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	1	2	3	4	5	6	7	8	4	10	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)←
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

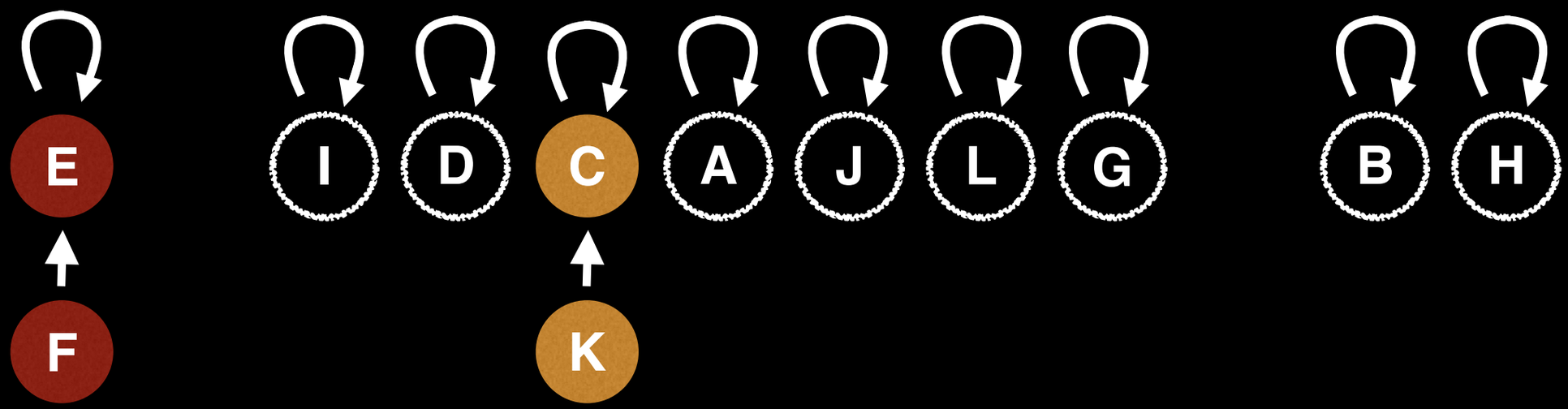


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	3	4	5	6	7	8	4	10	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

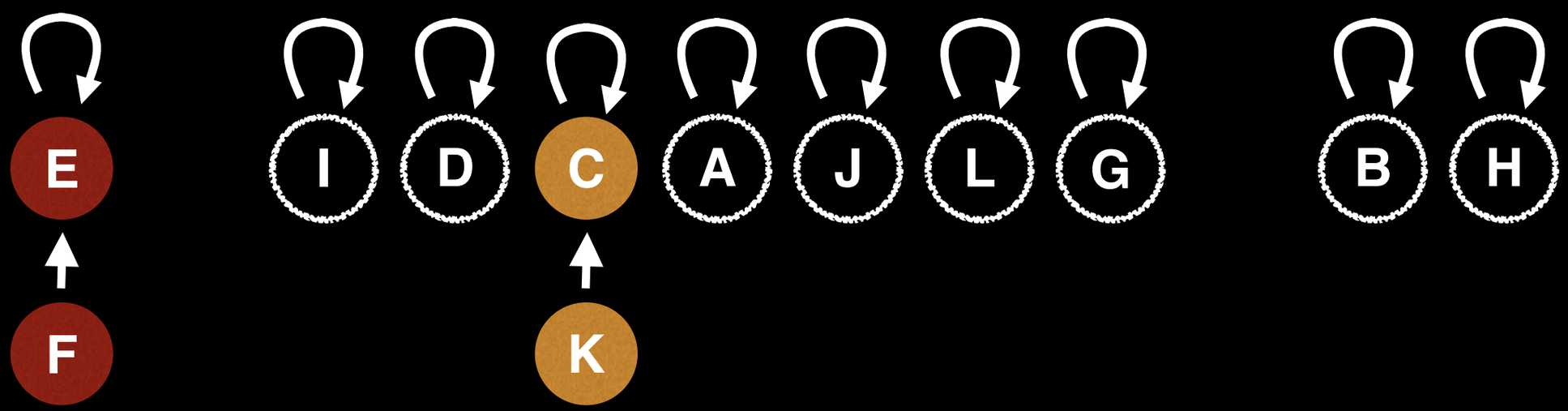


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	3	4	5	6	7	8	4	10	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

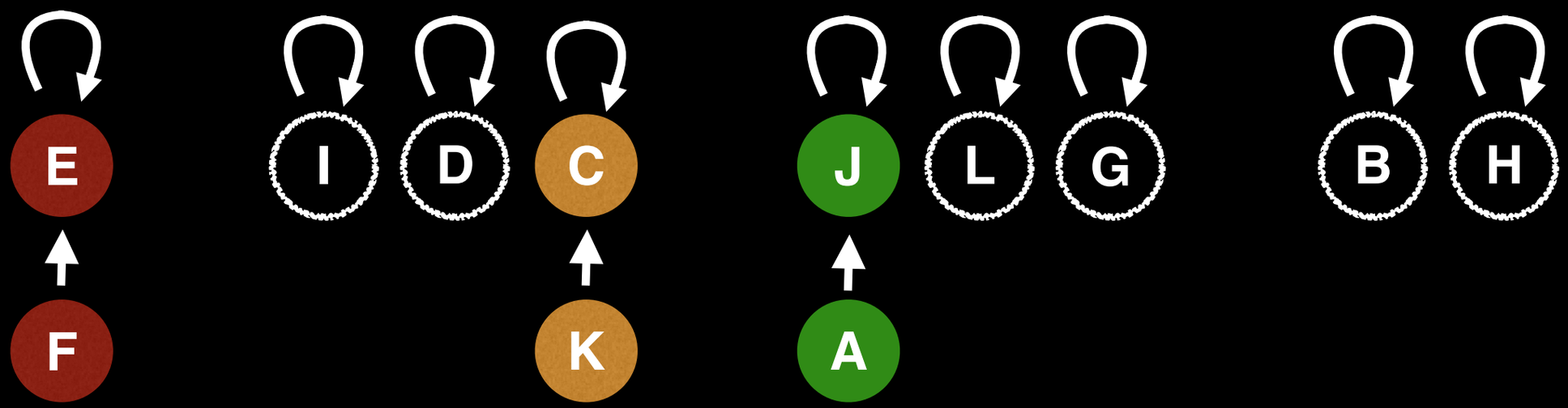


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	3	4	6	6	7	8	4	10	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

Union(C,K)
Union(F,E)
Union(A,J) ←
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)

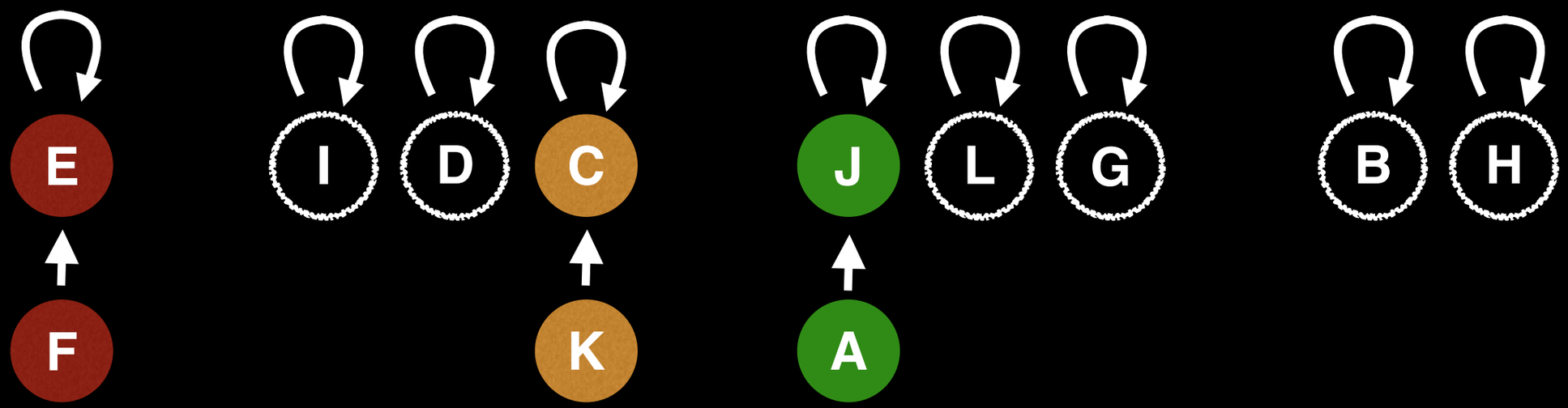


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	3	4	6	6	7	8	4	10	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B) ←
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)

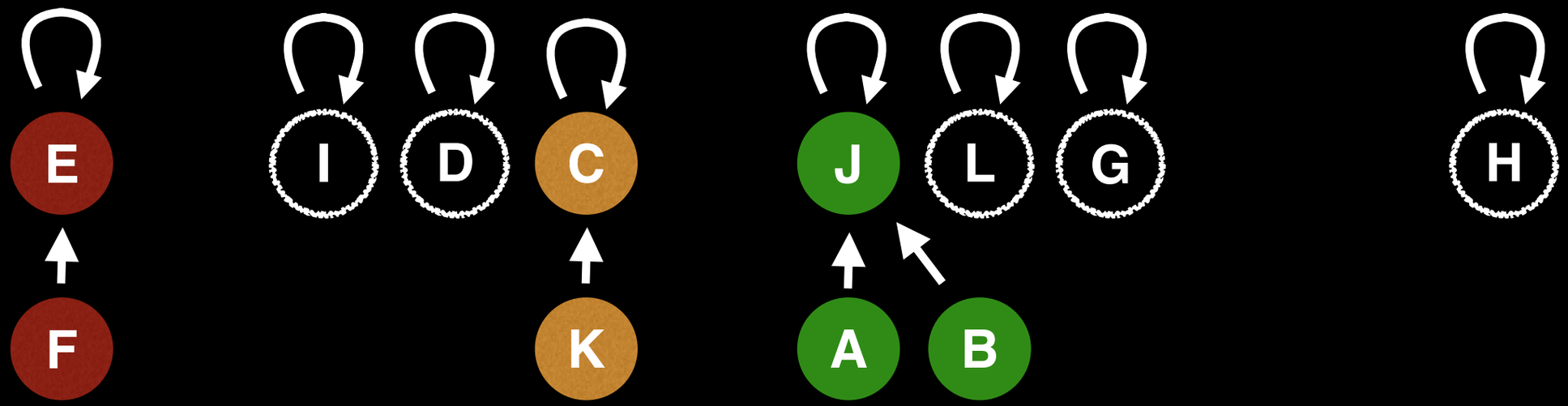


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	3	4	6	6	7	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

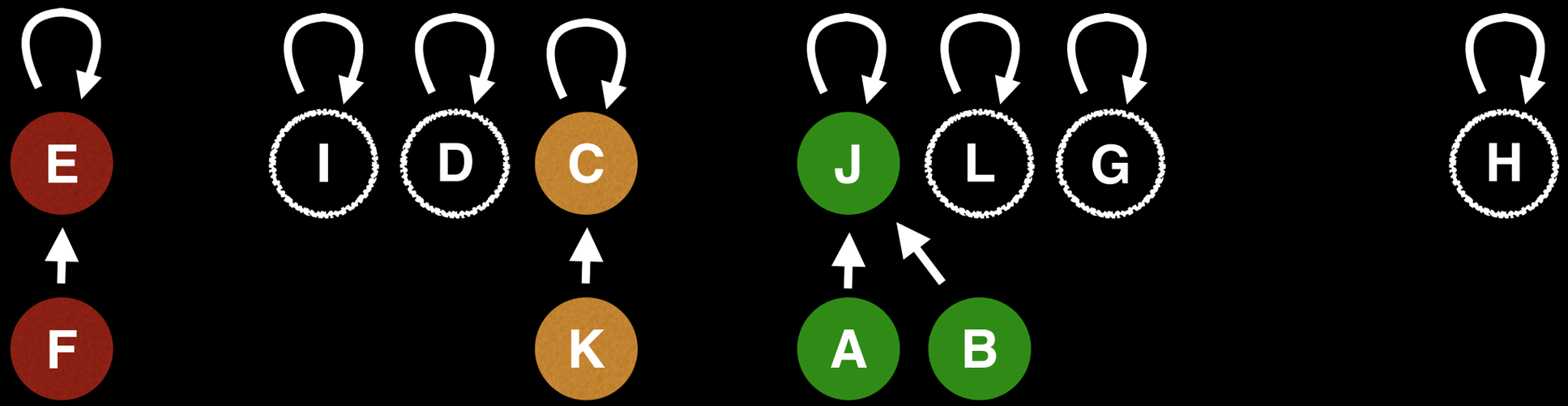


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	3	4	6	6	7	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

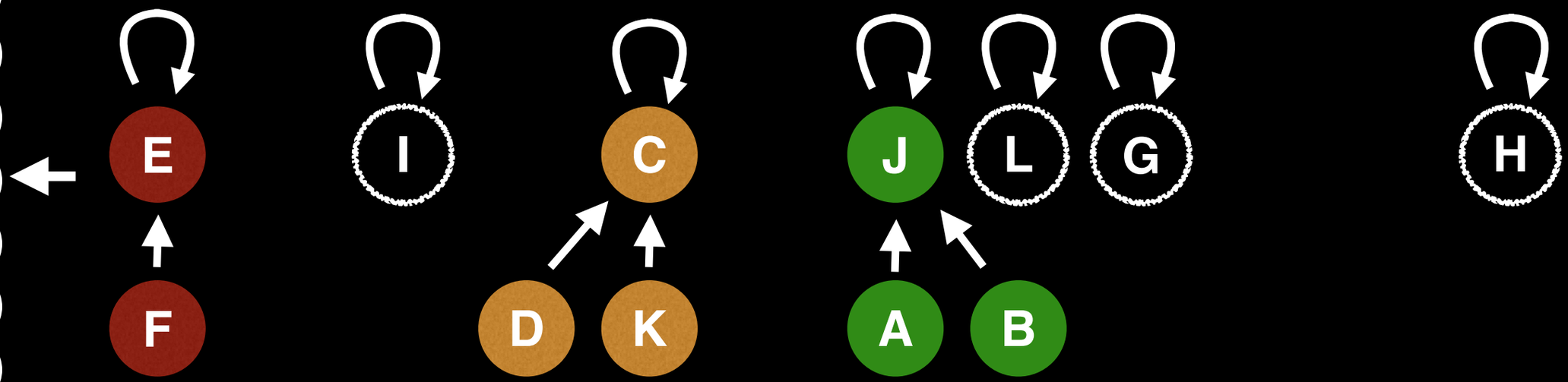


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	4	4	6	6	7	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

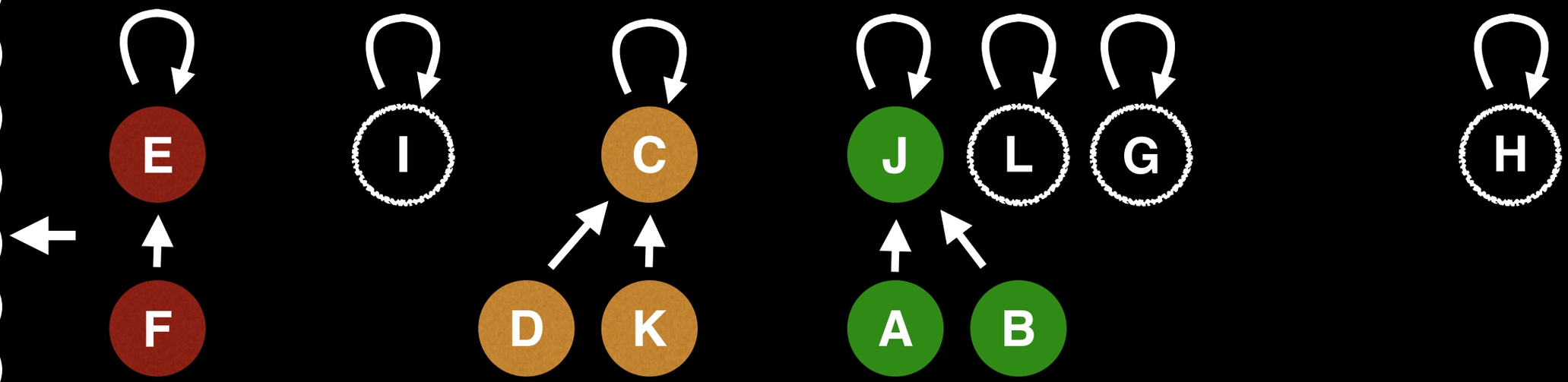


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	2	4	4	6	6	7	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

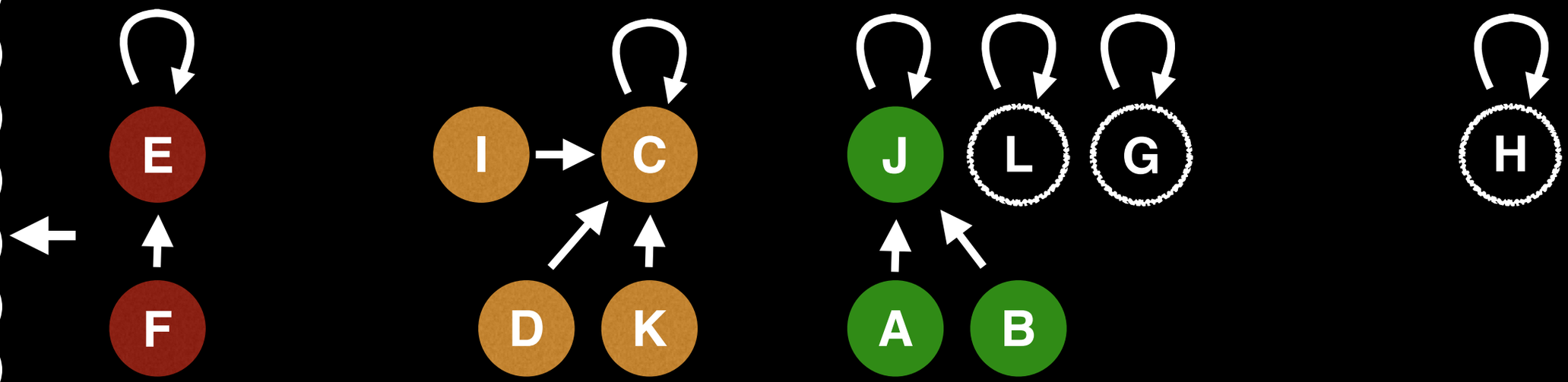


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	6	7	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

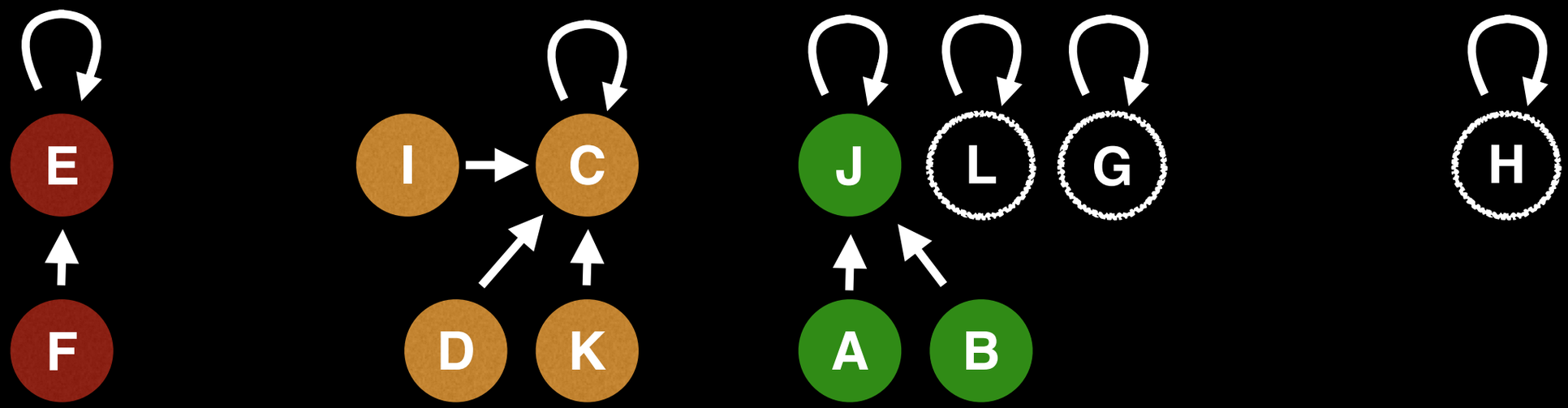


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	6	7	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

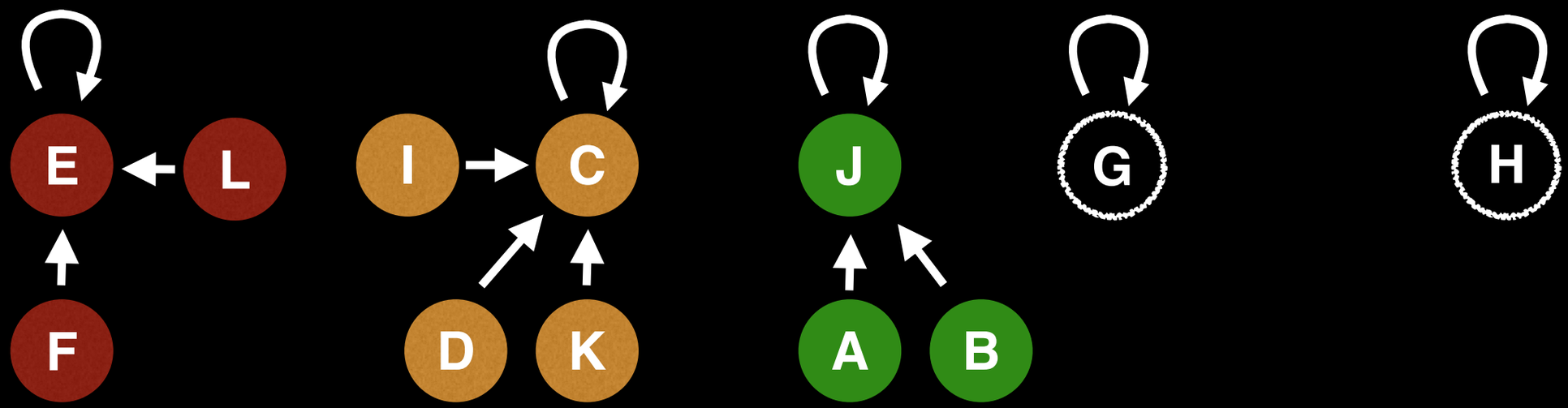


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	6	0	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

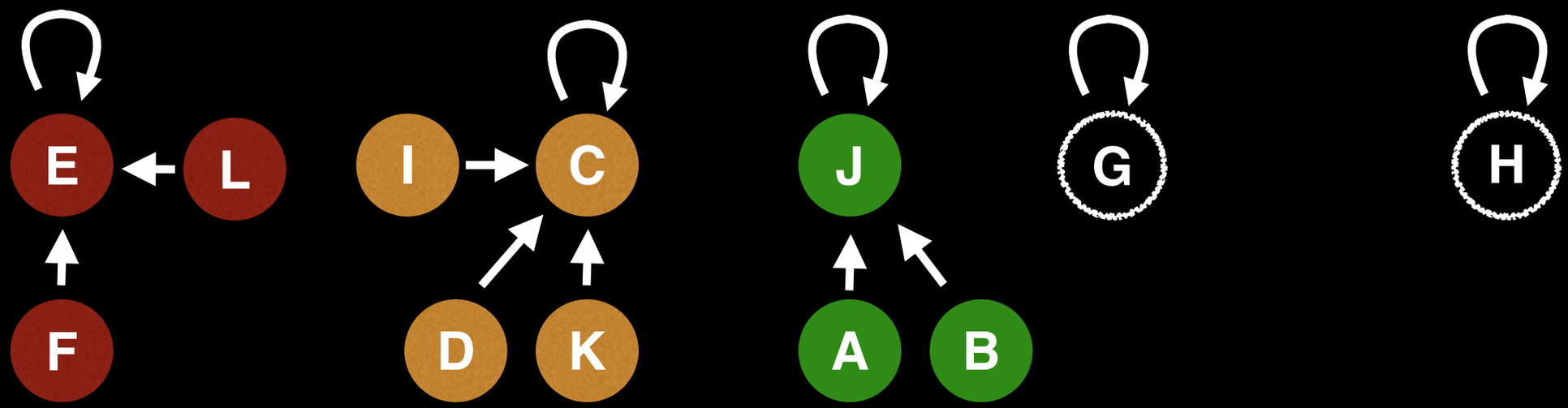


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	6	0	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

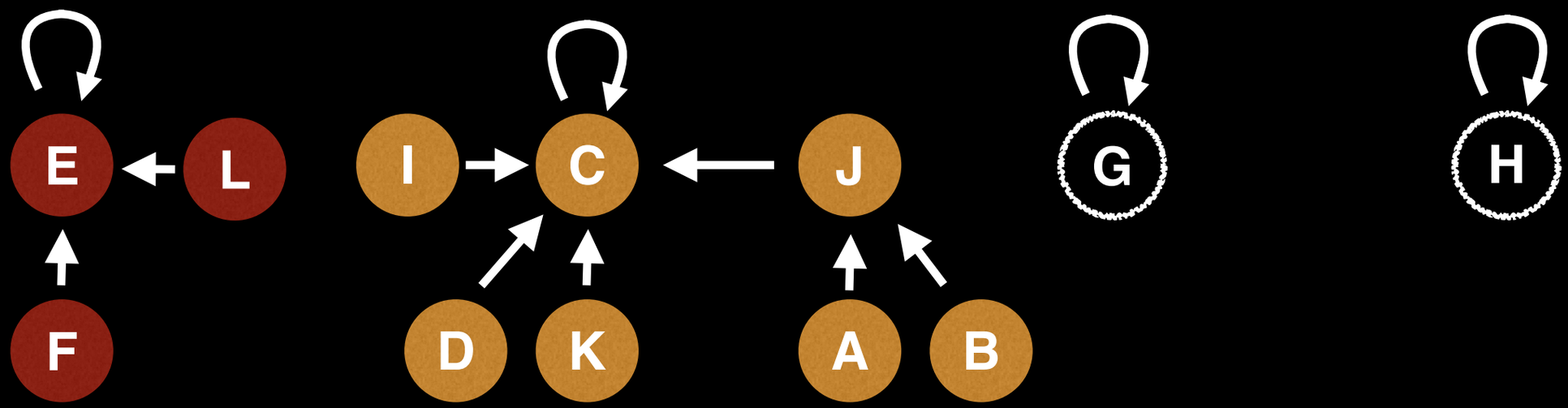


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	4	0	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

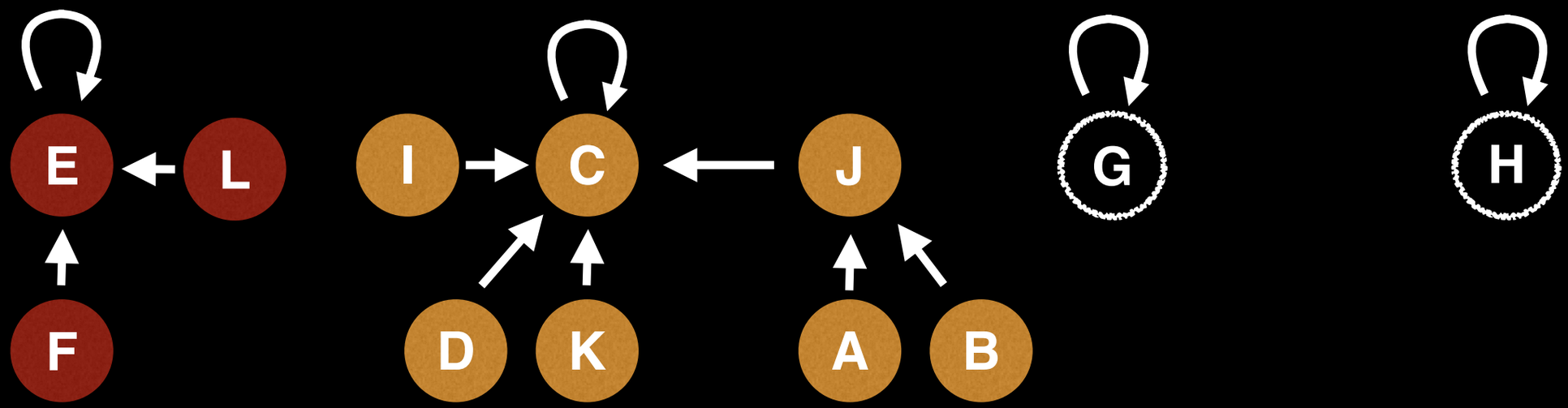


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	4	0	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B) ←
Union(H,G)
Union(H,F)
Union(H,B)

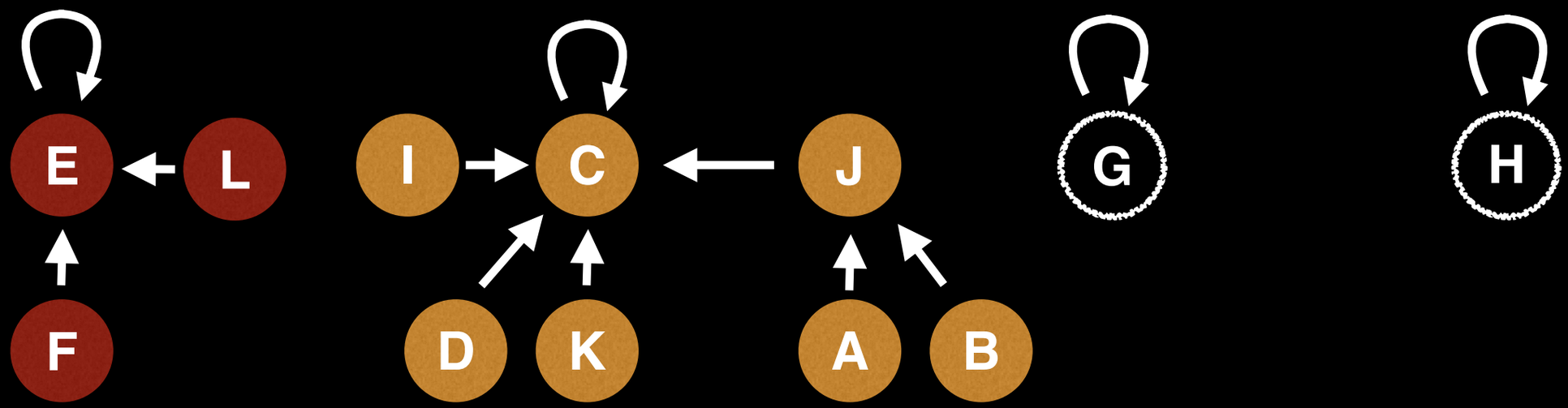


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	4	0	8	4	6	11
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

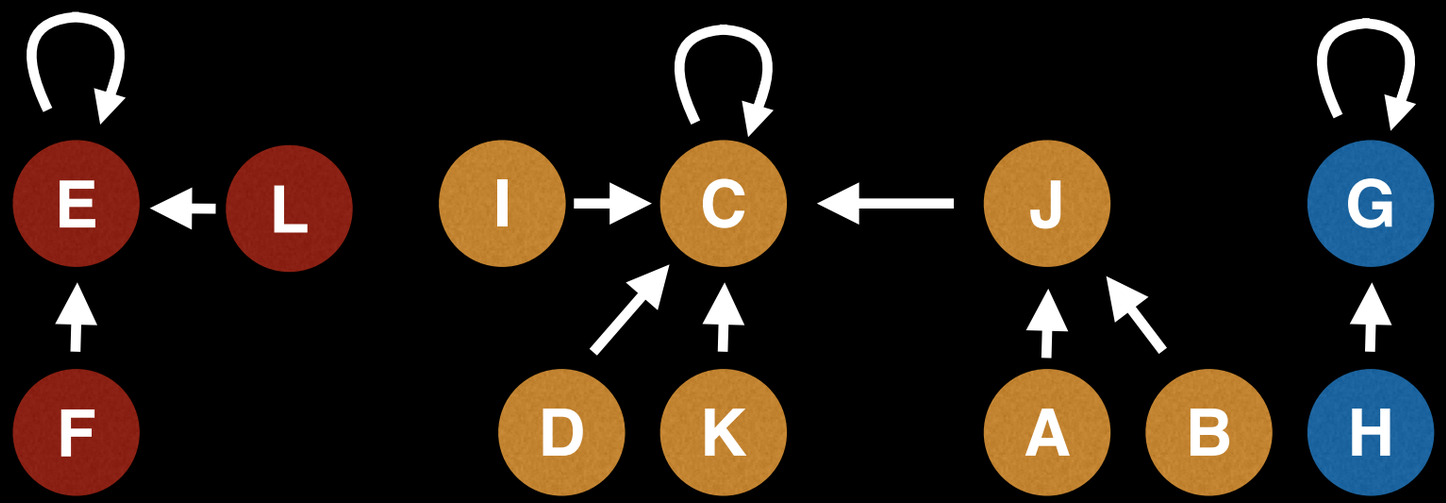


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	4	0	8	4	6	8
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

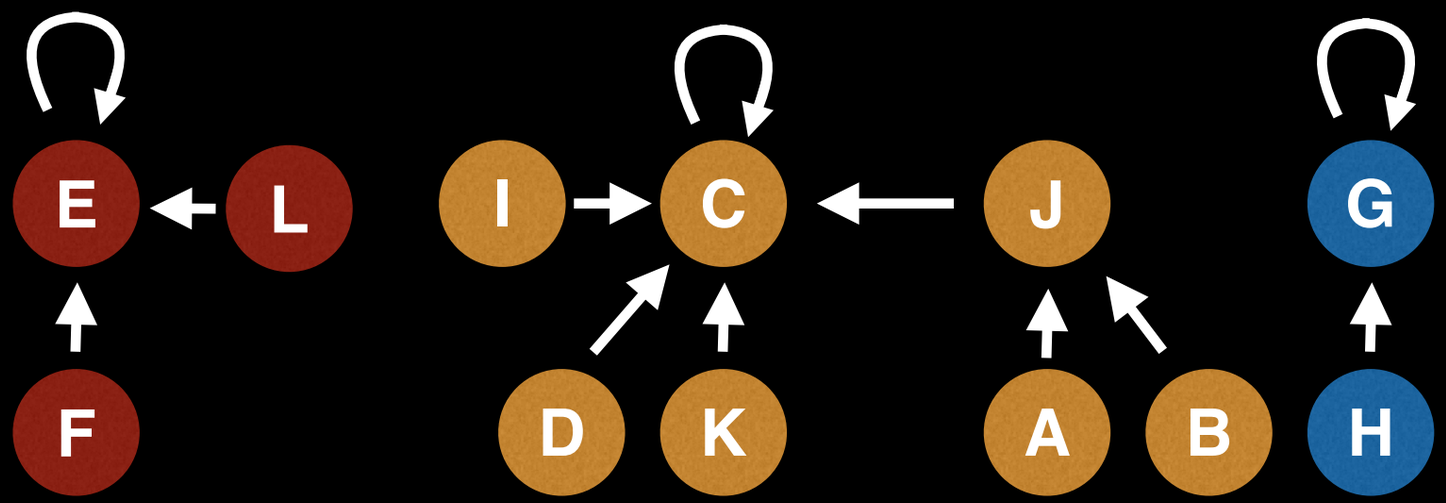


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	4	0	8	4	6	8
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)◀
Union(H,B)
```

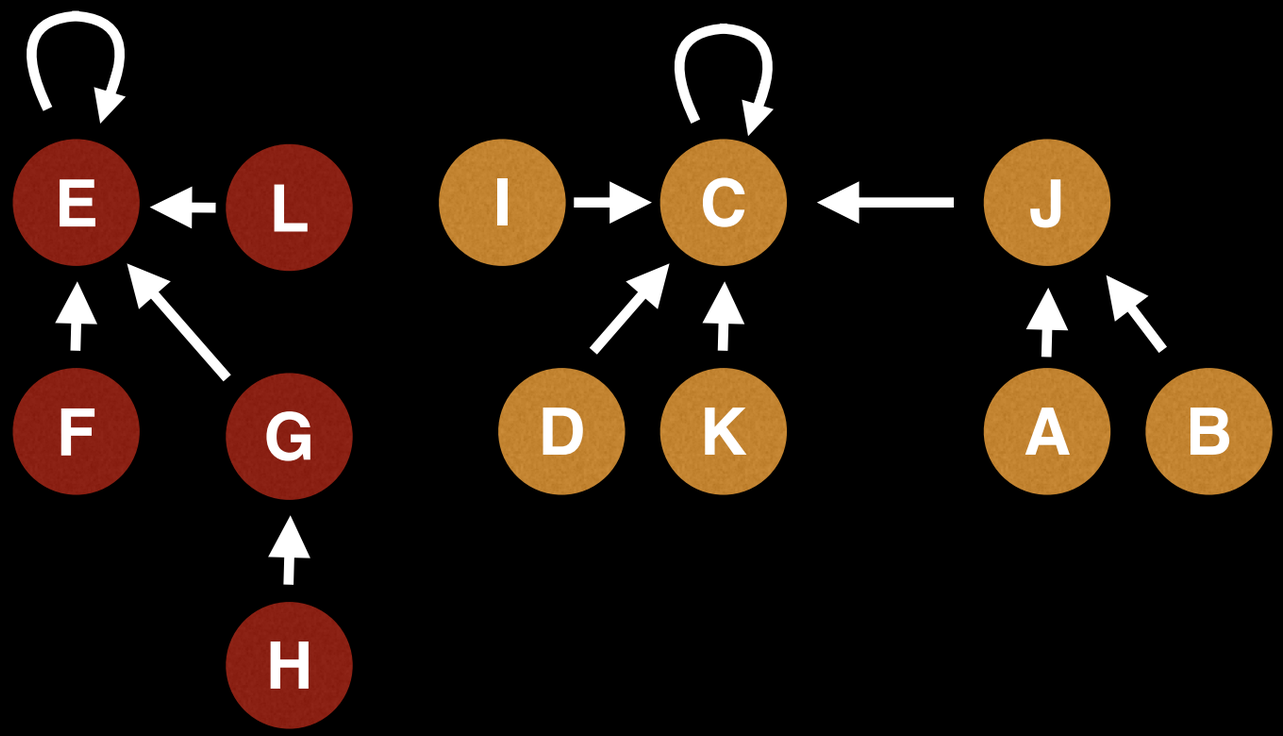


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	4	0	0	4	6	8
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F) ←
Union(H,B)
```

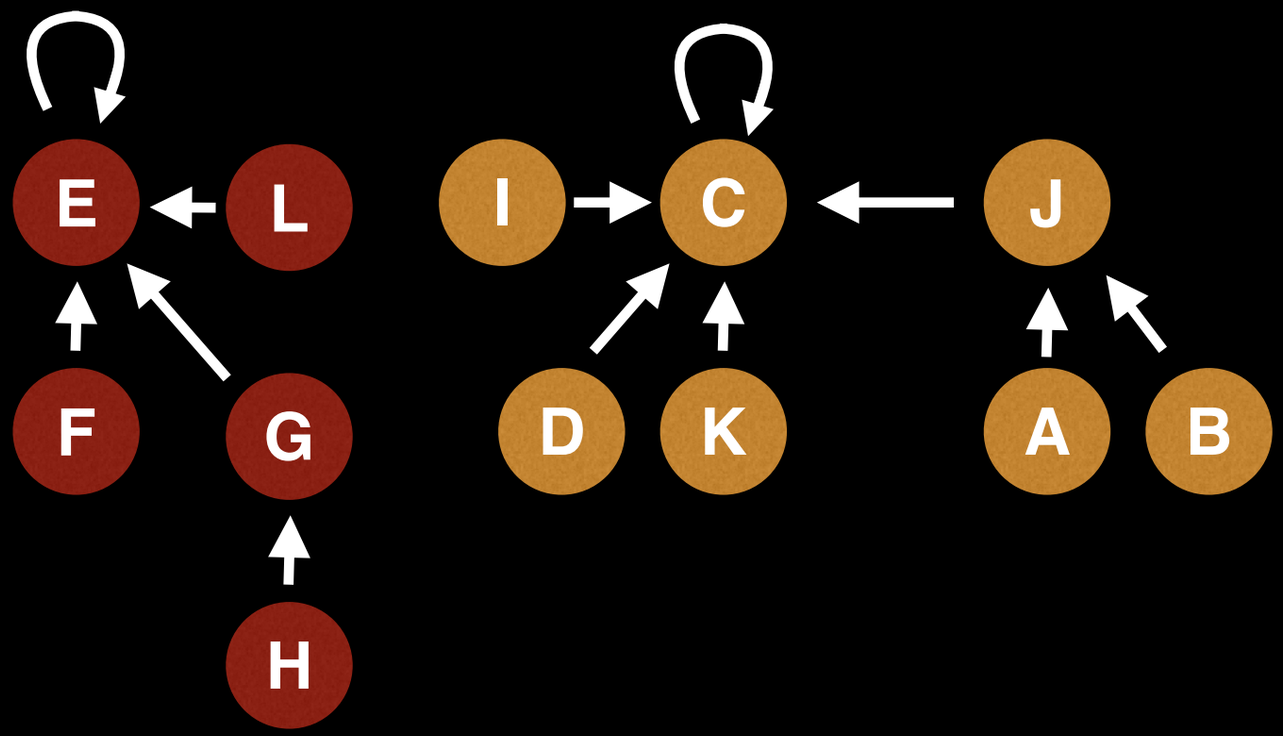


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
0	0	4	4	4	6	4	0	0	4	6	8
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

```
Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)
```

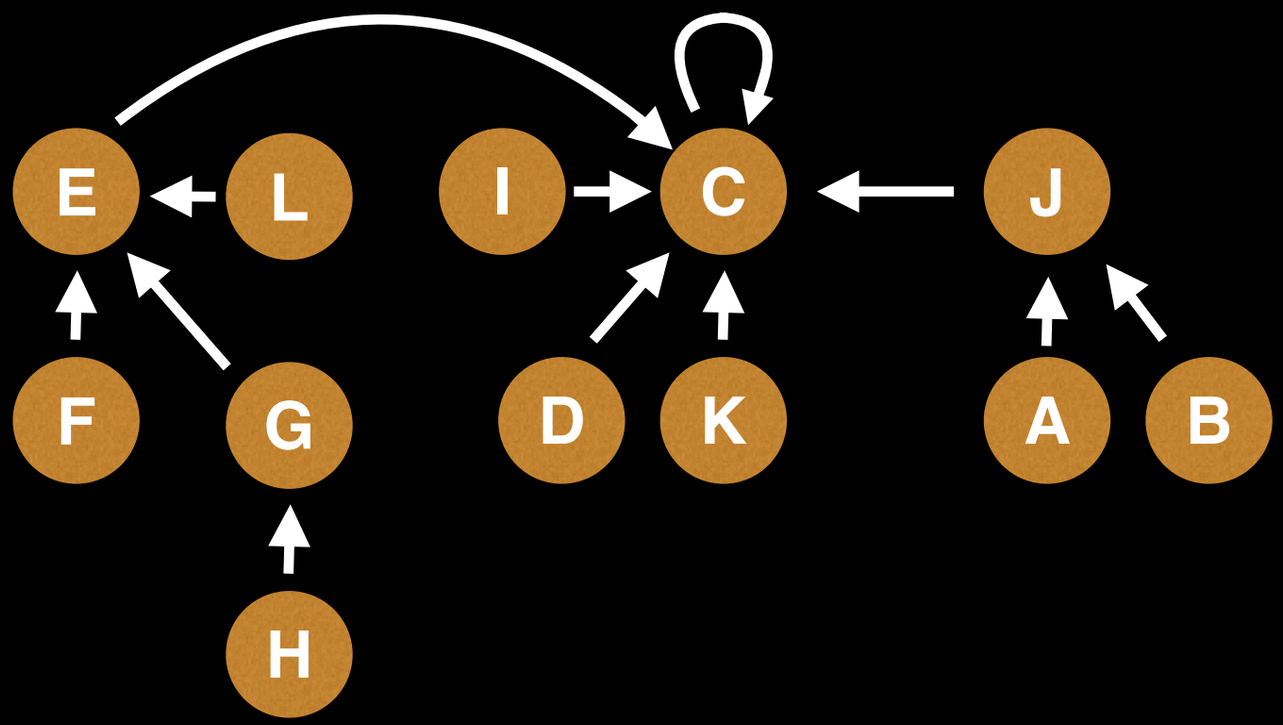


(This example does not use path compression)

E	F	I	D	C	A	J	L	G	K	B	H
4	0	4	4	4	6	4	0	0	4	6	8
0	1	2	3	4	5	6	7	8	9	10	11

Instructions:

Union(C,K)
Union(F,E)
Union(A,J)
Union(A,B)
Union(C,D)
Union(D,I)
Union(L,F)
Union(C,A)
Union(A,B)
Union(H,G)
Union(H,F)
Union(H,B)←



(This example does not use path compression)

Summary

Find Operation

To **find** which component a particular element belongs to find the root of that component by following the parent nodes until a self loop is reached (a node who's parent is itself)

Union Operation

To **unify** two elements find which are the root nodes of each component and if the root nodes are different make one of the root nodes be the parent of the other.

Remarks

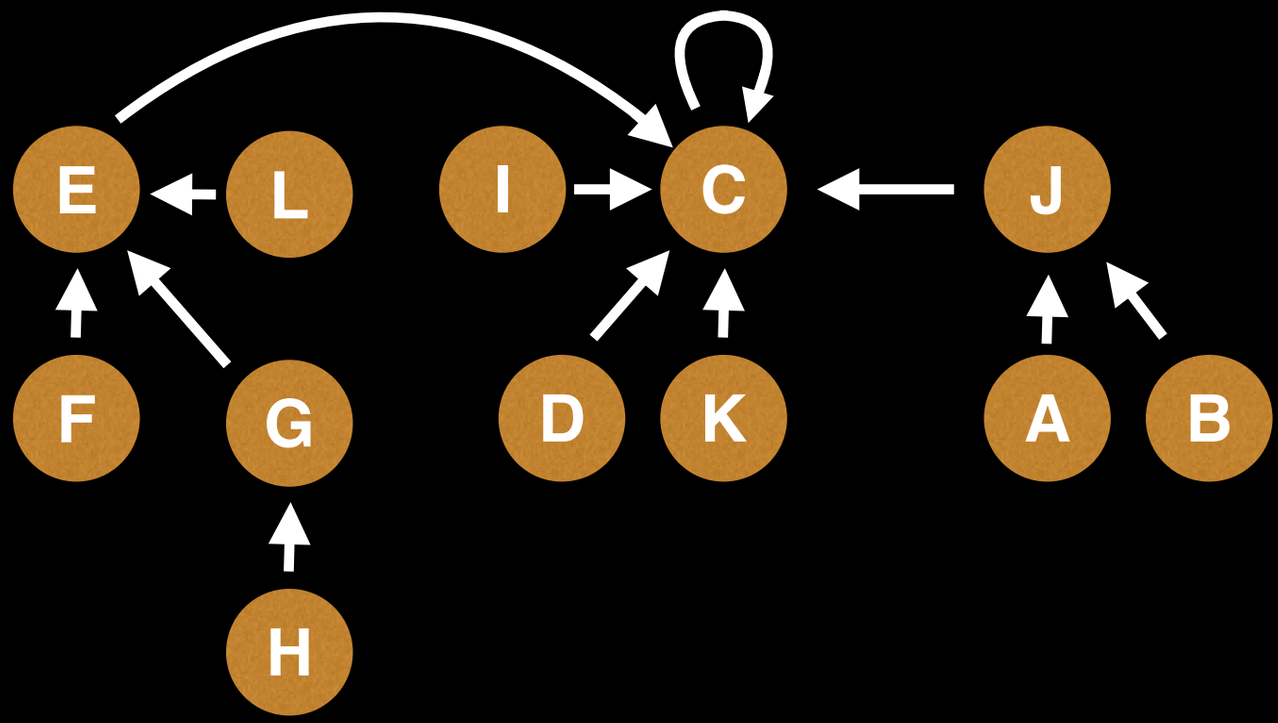
In this data structure, we do not “un-union” elements. In general, this would be very inefficient to do since we would have to update all the children of a node.

The number of components is equal to the number of roots remaining. Also, remark that the number of root nodes never increases.

Remarks

Our current version of Union Find does not support the nice $\alpha(n)$ time complexity we want.

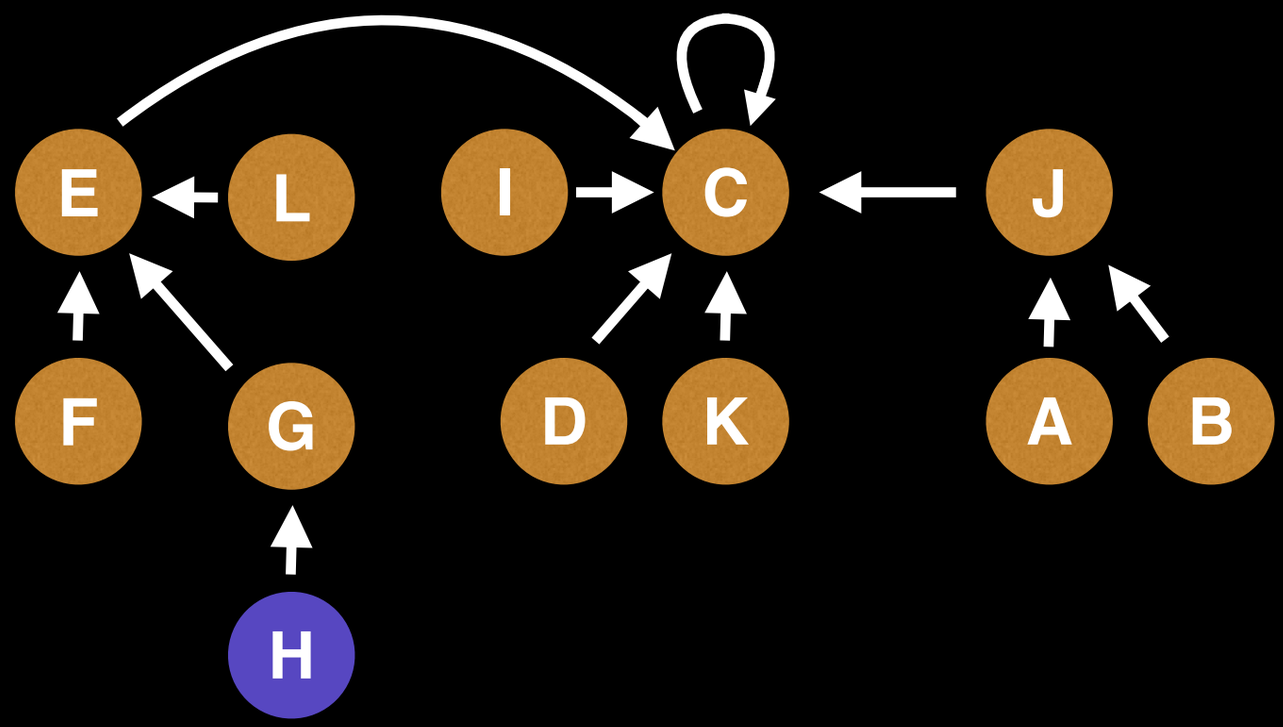
Checking if H and B belong to the same group takes five hops and in the worst case this is potentially much more.



Remarks

Our current version of Union Find does not support the nice $\alpha(n)$ time complexity we want.

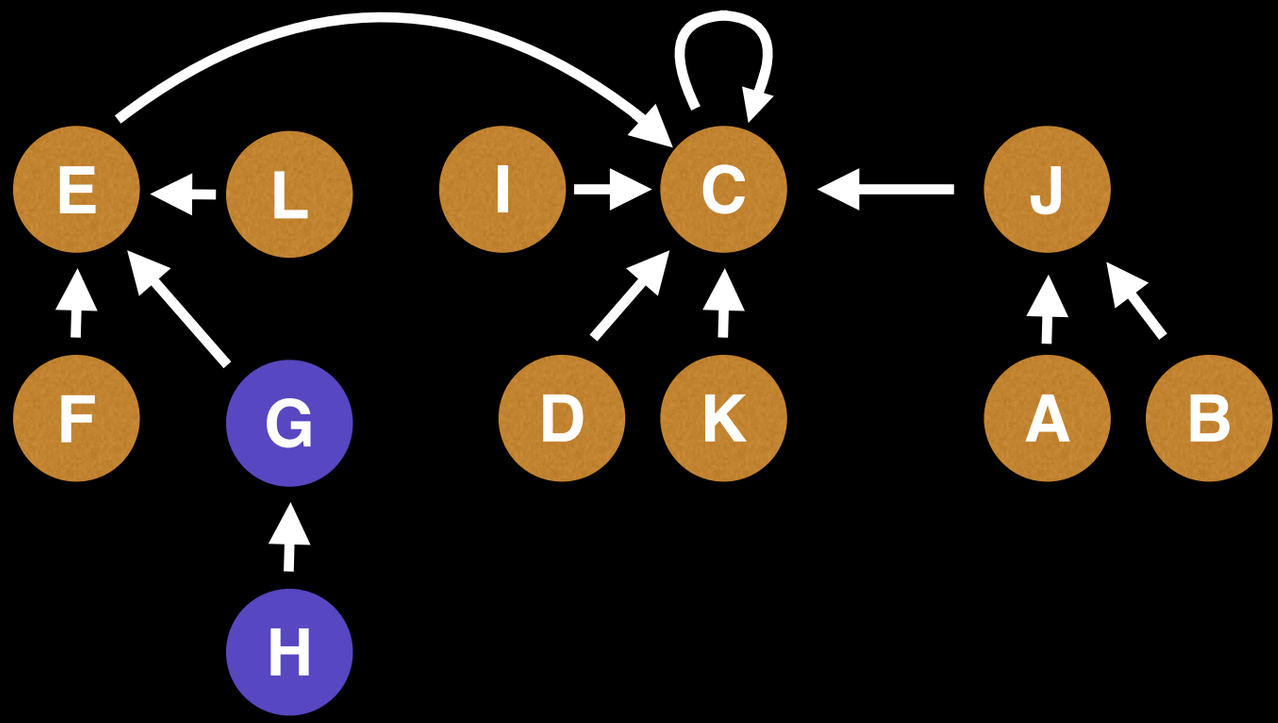
Checking if H and B belong to the same group takes five hops and in the worst case this is potentially much more.



Remarks

Our current version of Union Find does not support the nice $\alpha(n)$ time complexity we want.

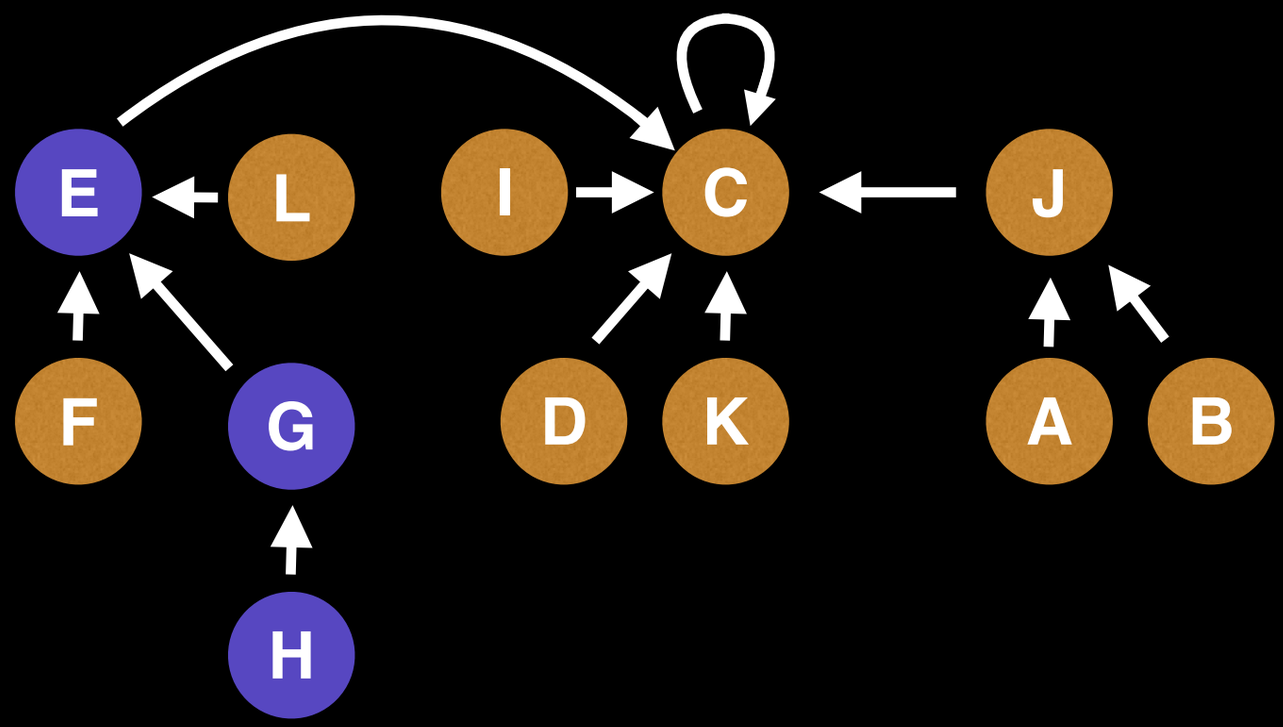
Checking if H and B belong to the same group takes five hops and in the worst case this is potentially much more.



Remarks

Our current version of Union Find does not support the nice $\alpha(n)$ time complexity we want.

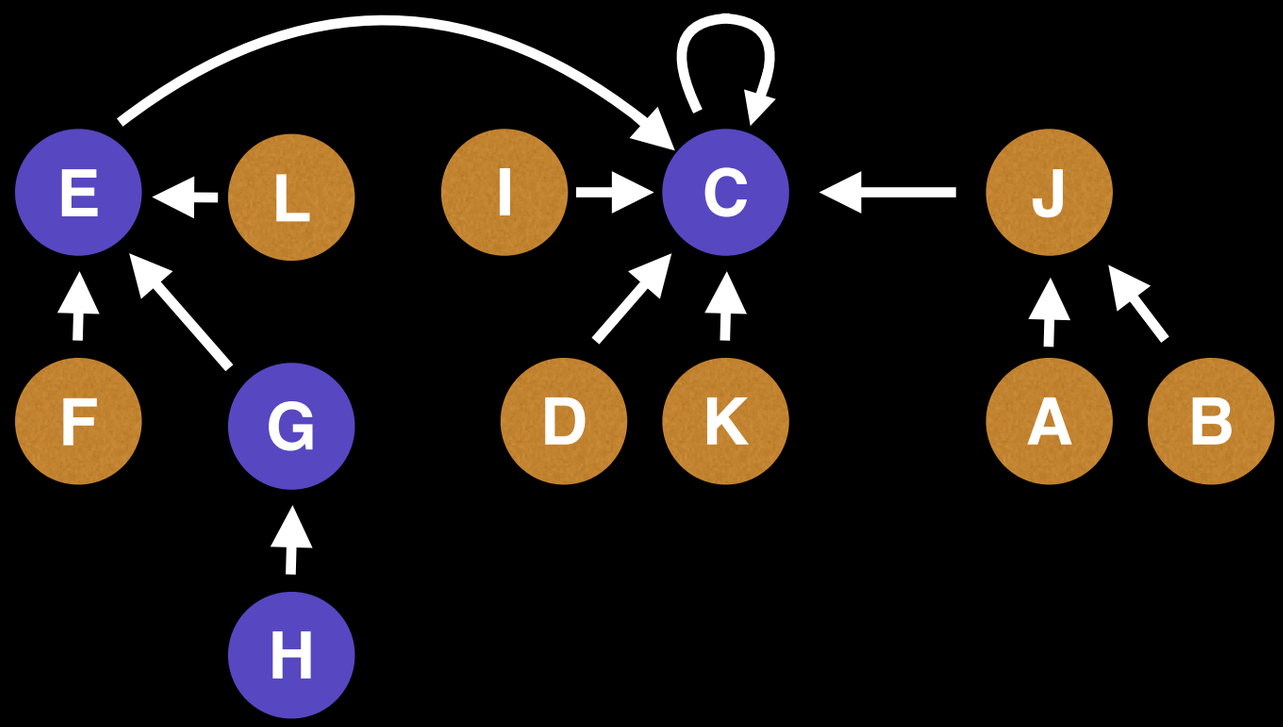
Checking if H and B belong to the same group takes five hops and in the worst case this is potentially much more.



Remarks

Our current version of Union Find does not support the nice $\alpha(n)$ time complexity we want.

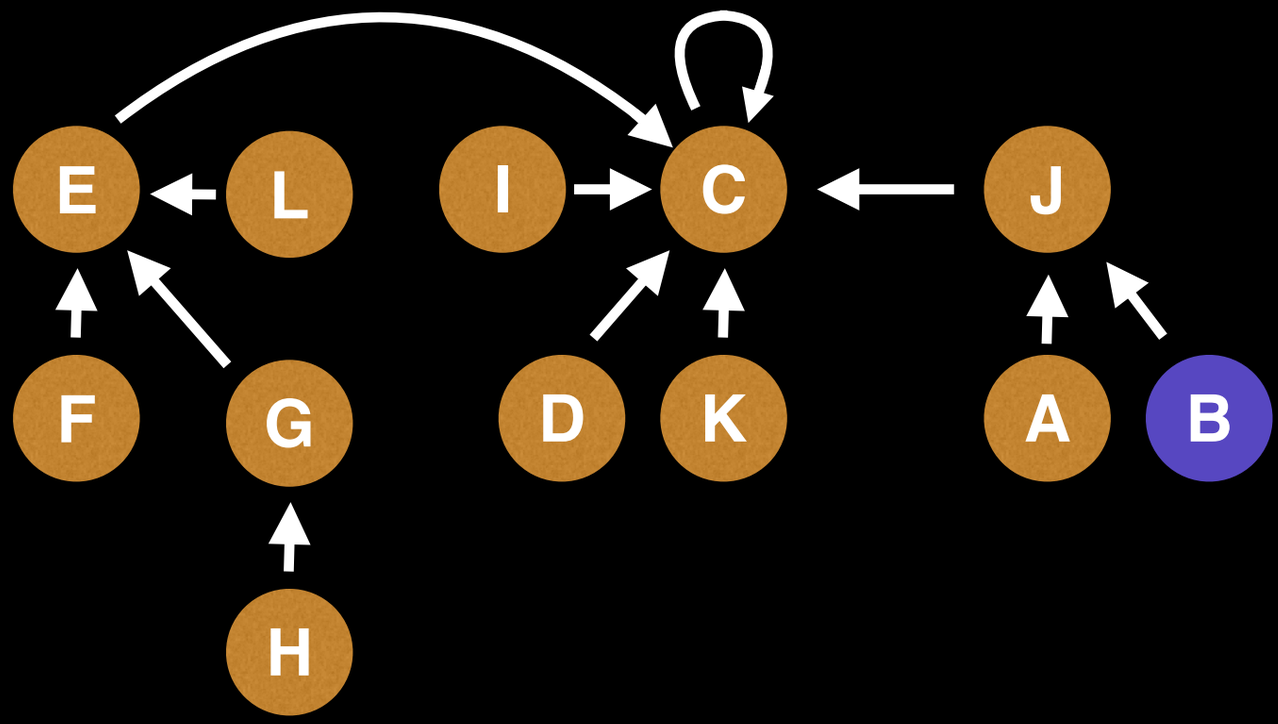
Checking if H and B belong to the same group takes five hops and in the worst case this is potentially much more.



Remarks

Our current version of Union Find does not support the nice $\alpha(n)$ time complexity we want.

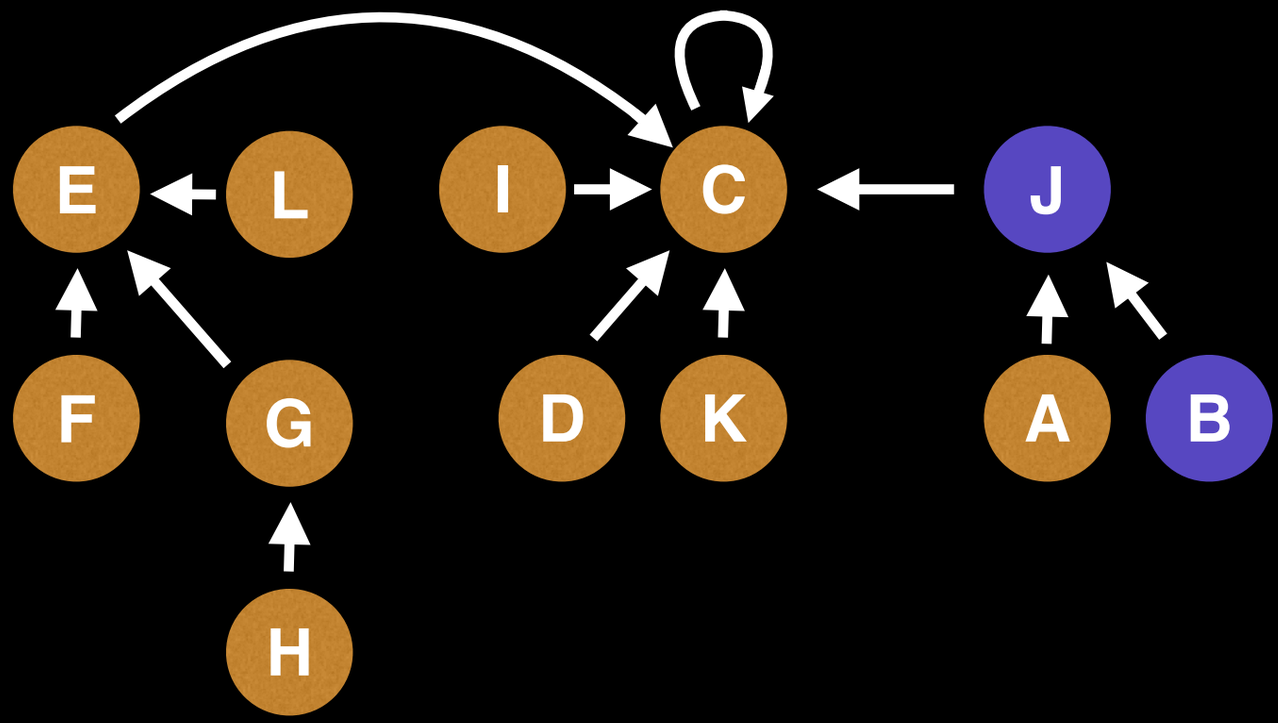
Checking if H and B belong to the same group takes five hops and in the worst case this is potentially much more.



Remarks

Our current version of Union Find does not support the nice $\alpha(n)$ time complexity we want.

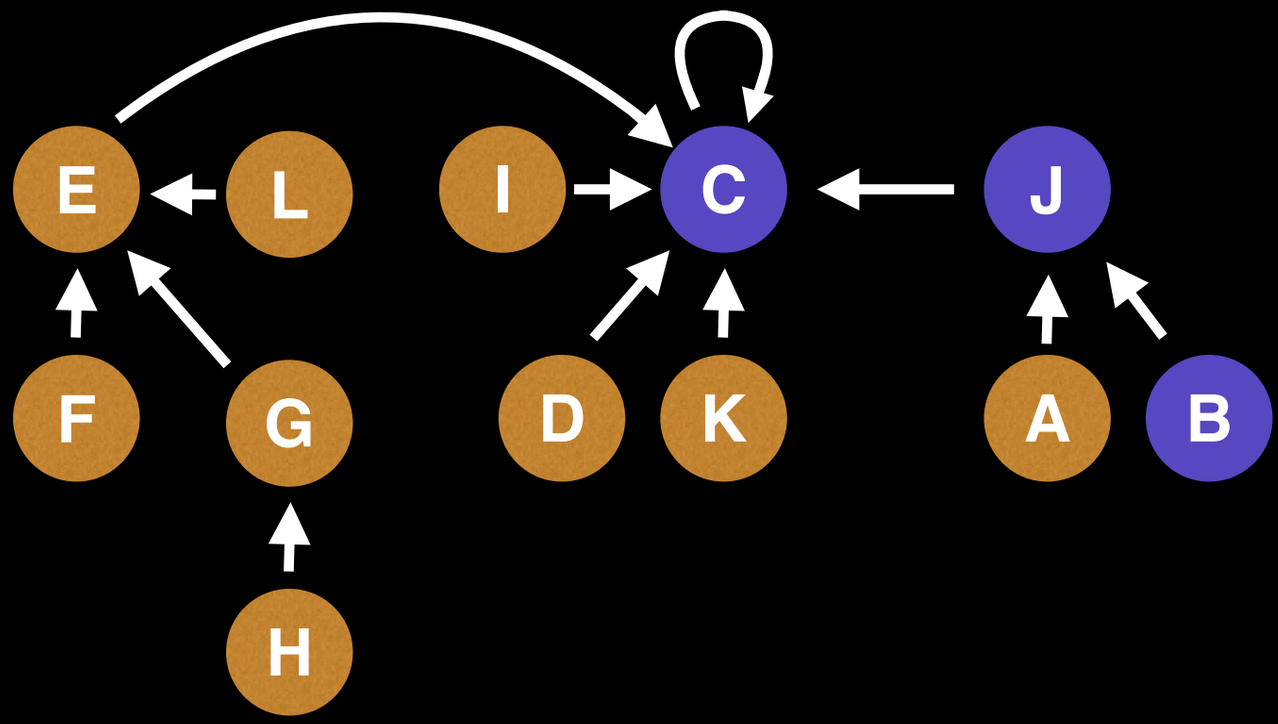
Checking if H and B belong to the same group takes five hops and in the worst case this is potentially much more.



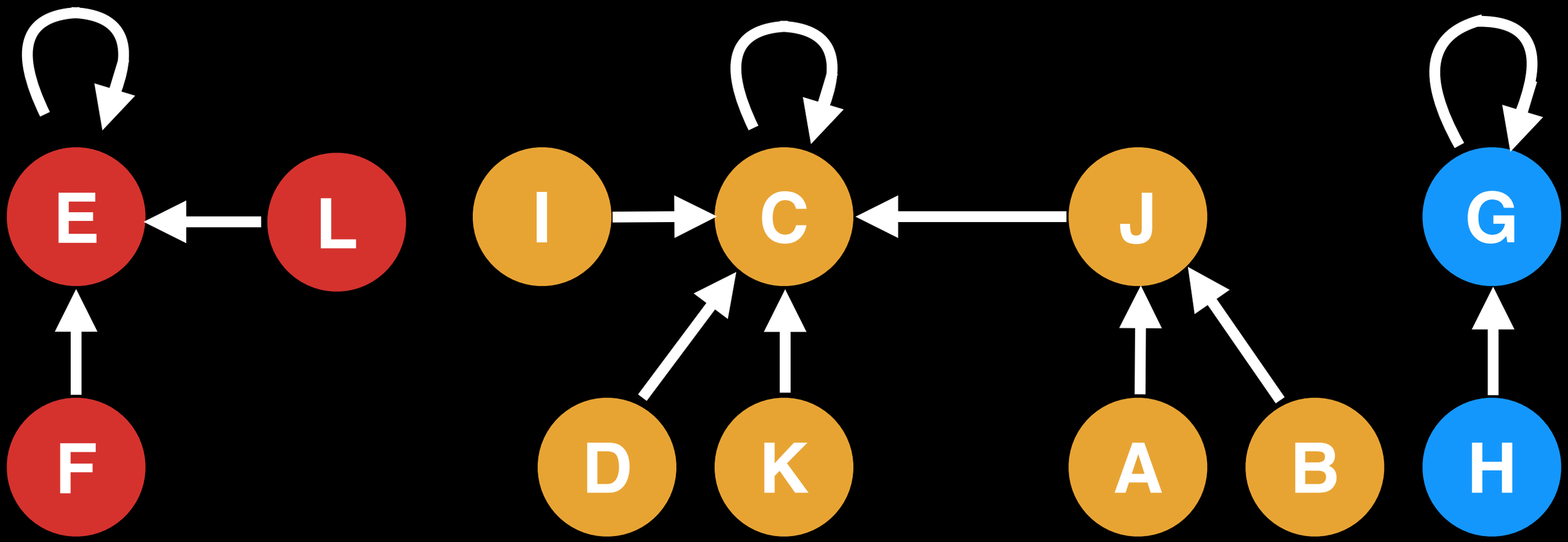
Remarks

Our current version of Union Find does not support the nice $\alpha(n)$ time complexity we want.

Checking if H and B belong to the same group takes five hops and in the worst case this is potentially much more.



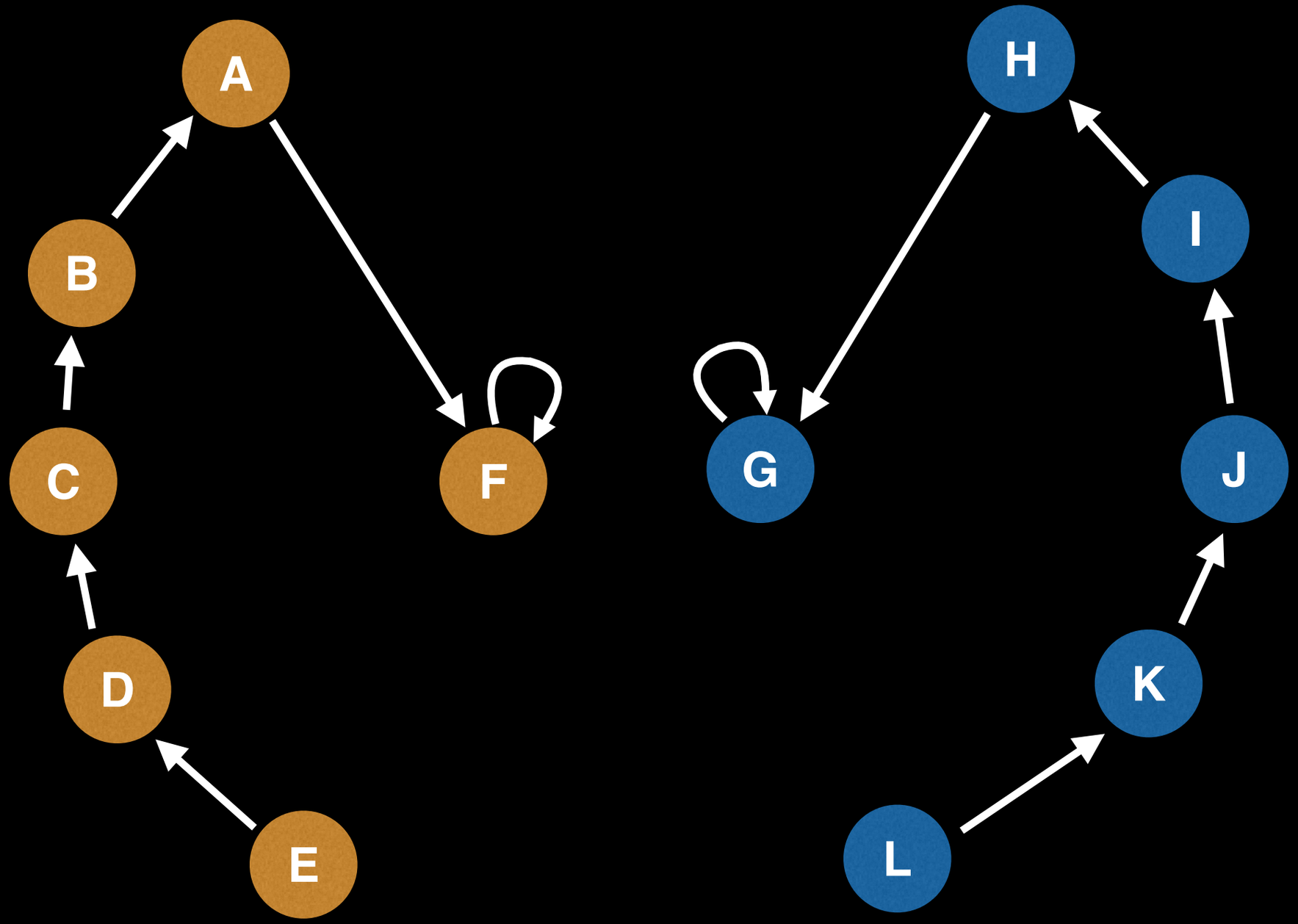
Union Find: Union & Find Operations



Union Find

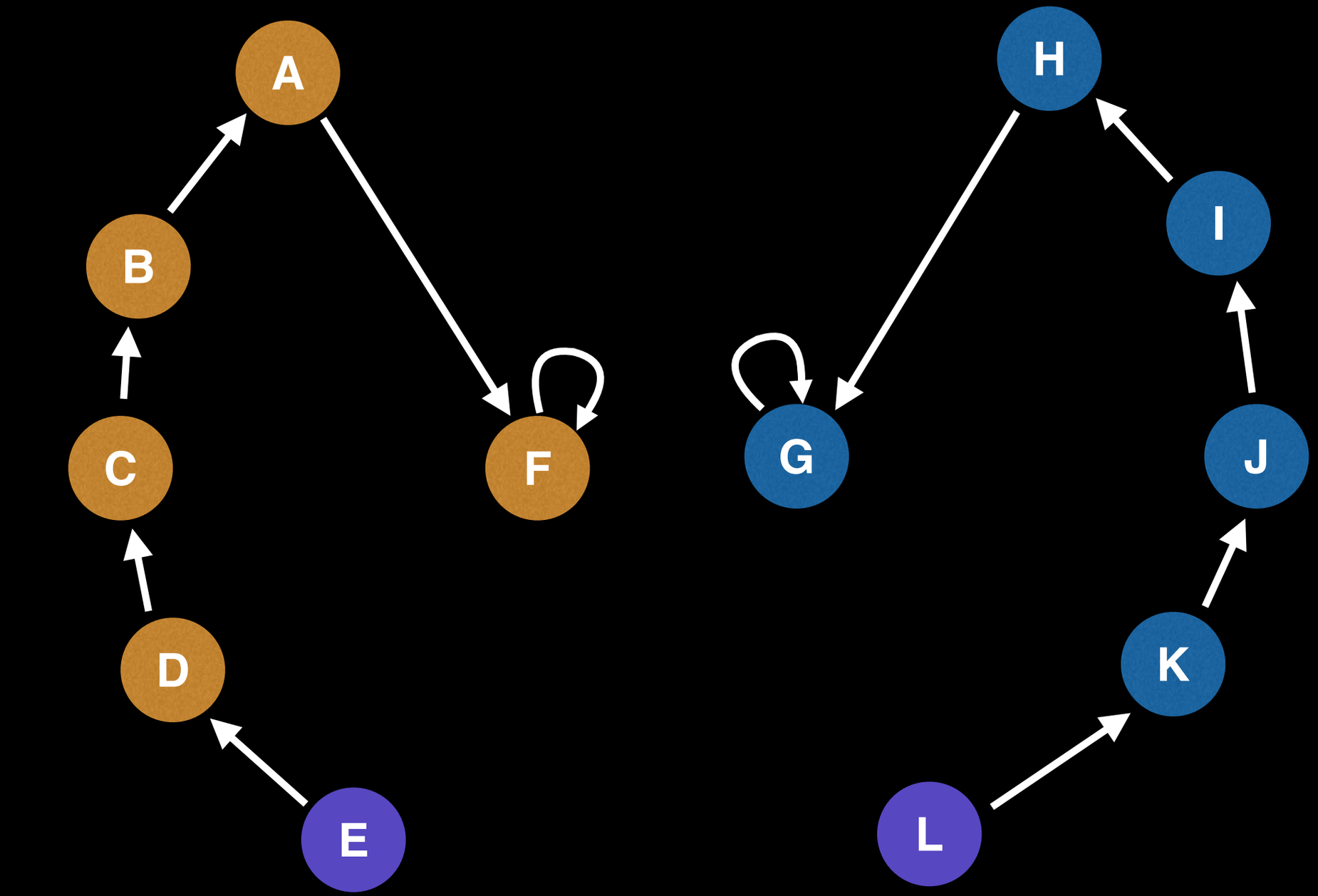
Path Compression

Hypothetical Union Find path compression example



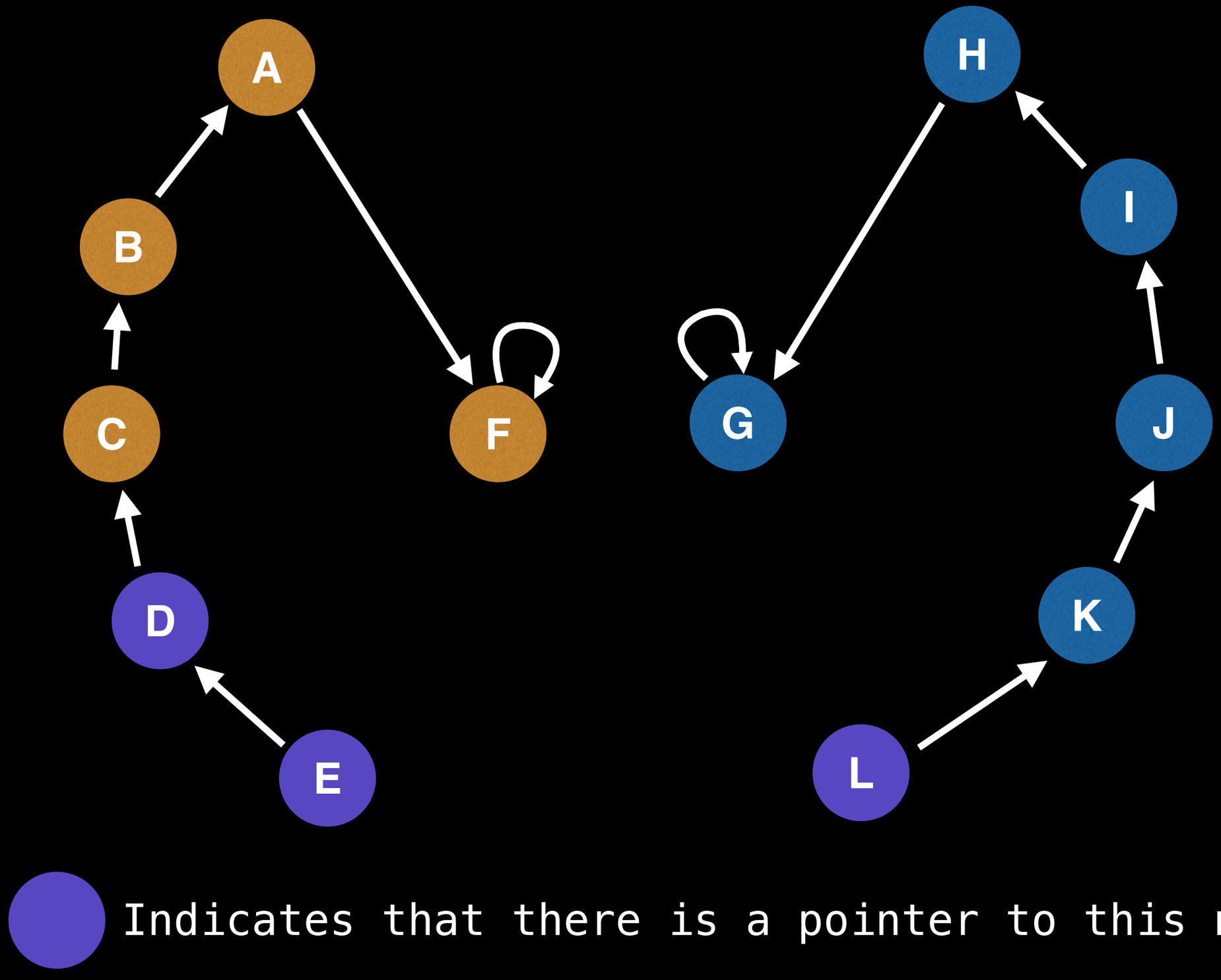
Operation: Take the union of **E** and **L**

Hypothetical Union Find path compression example



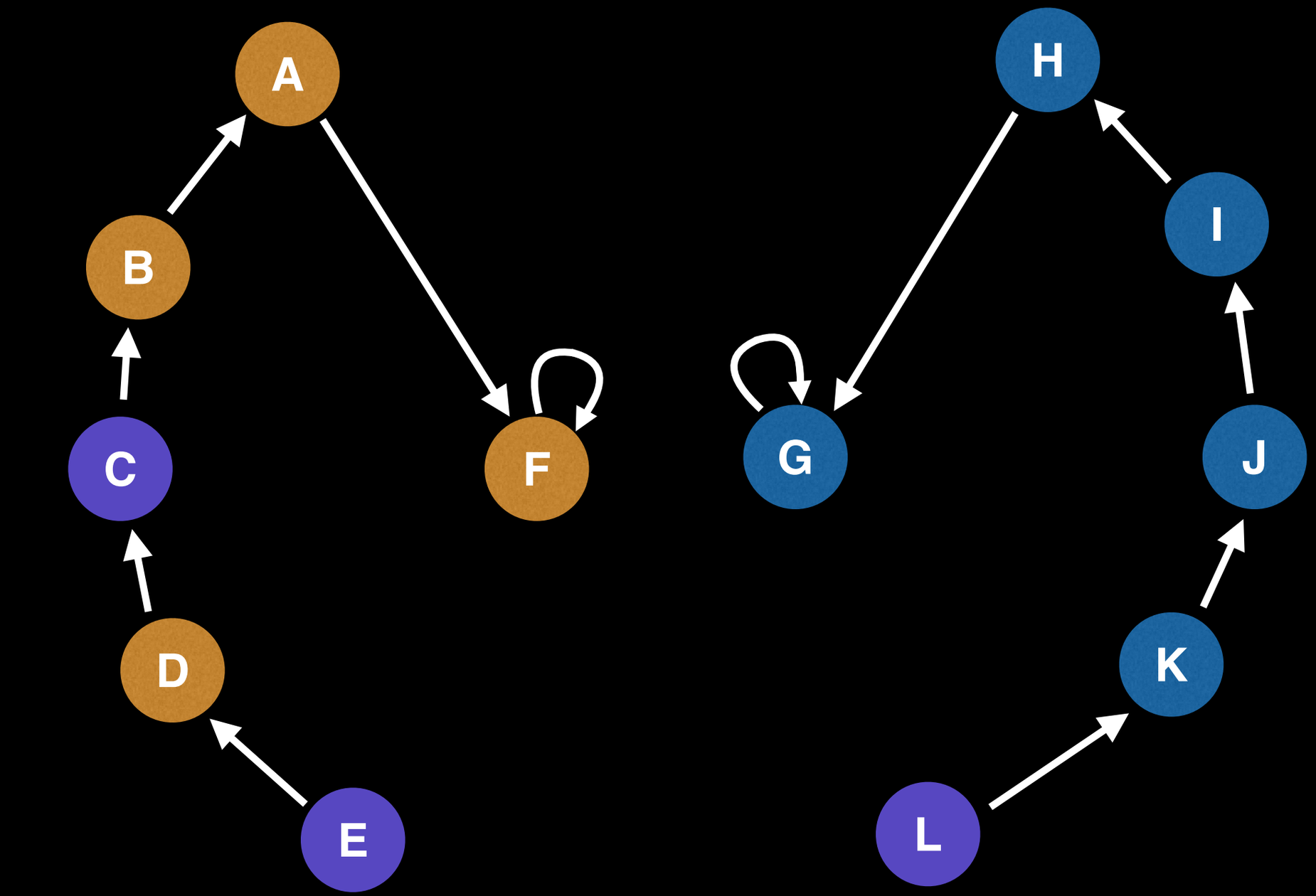
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



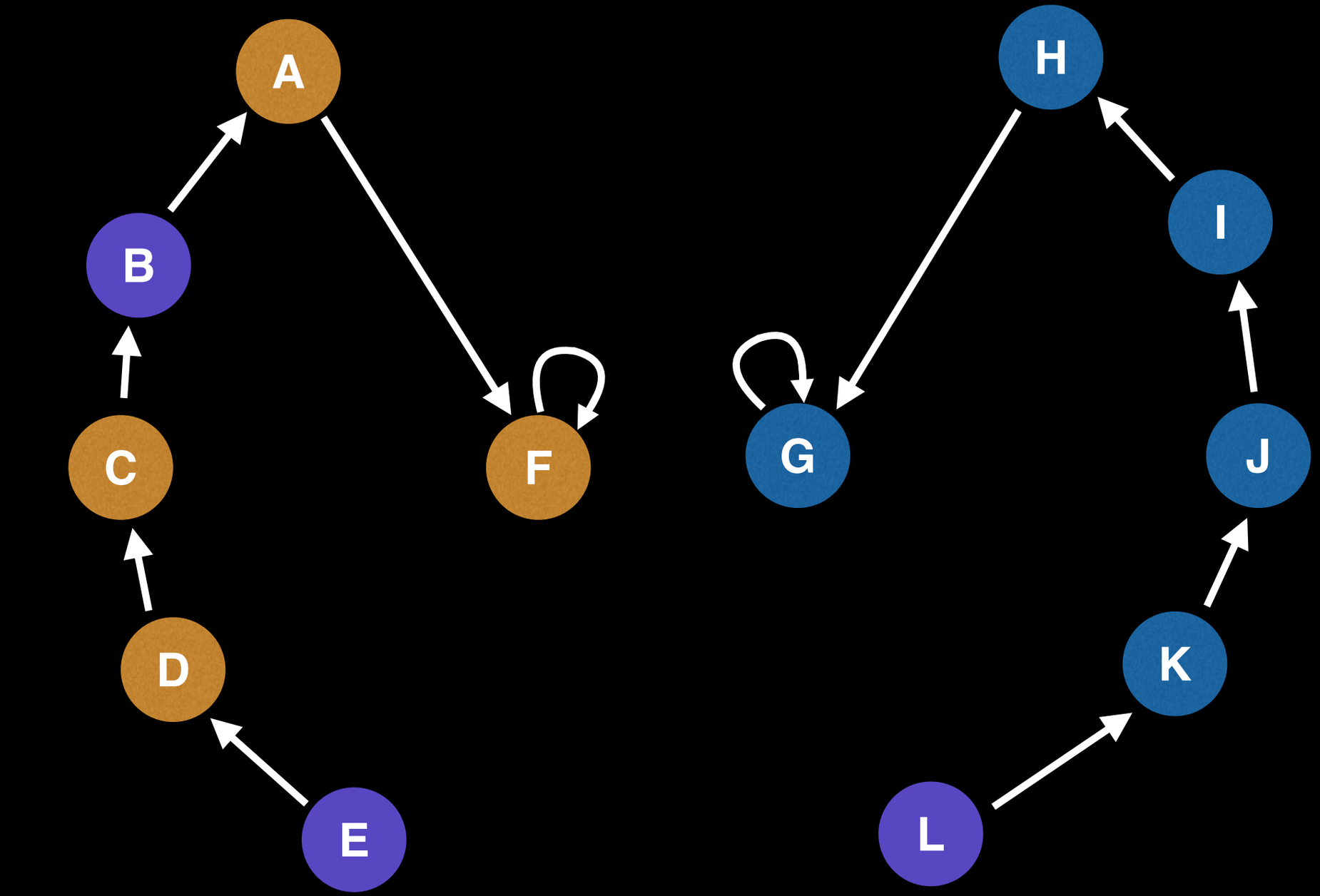
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



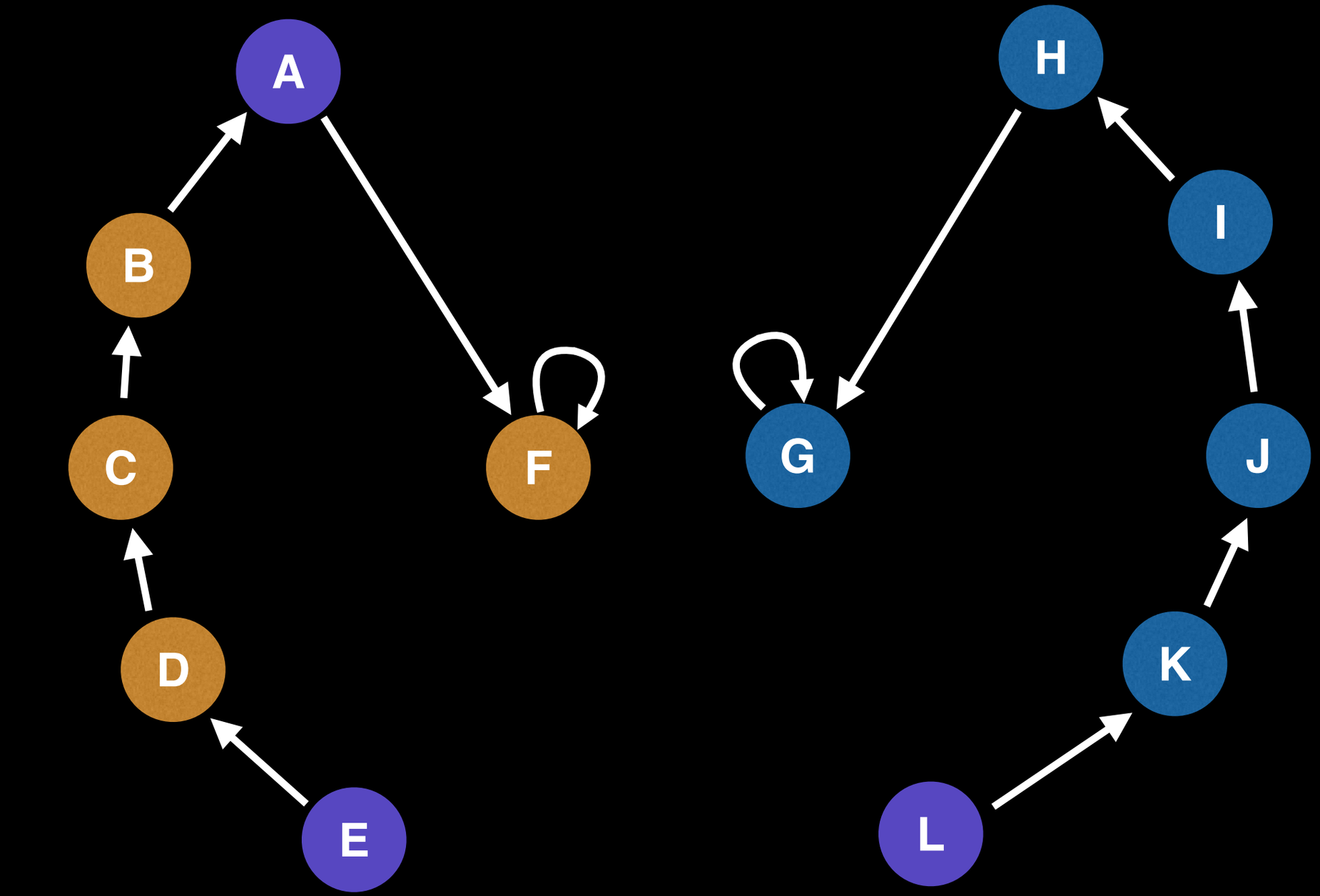
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



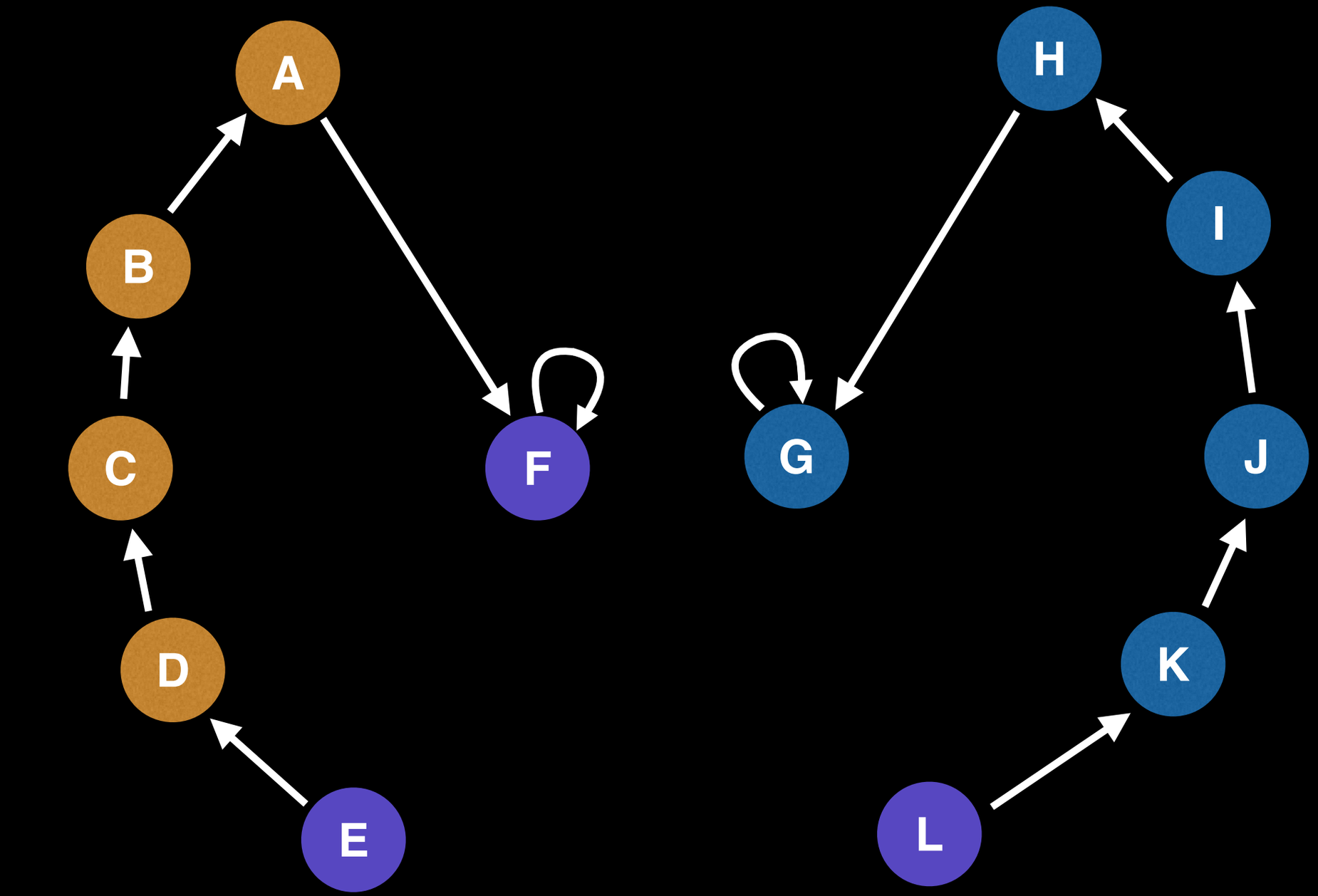
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



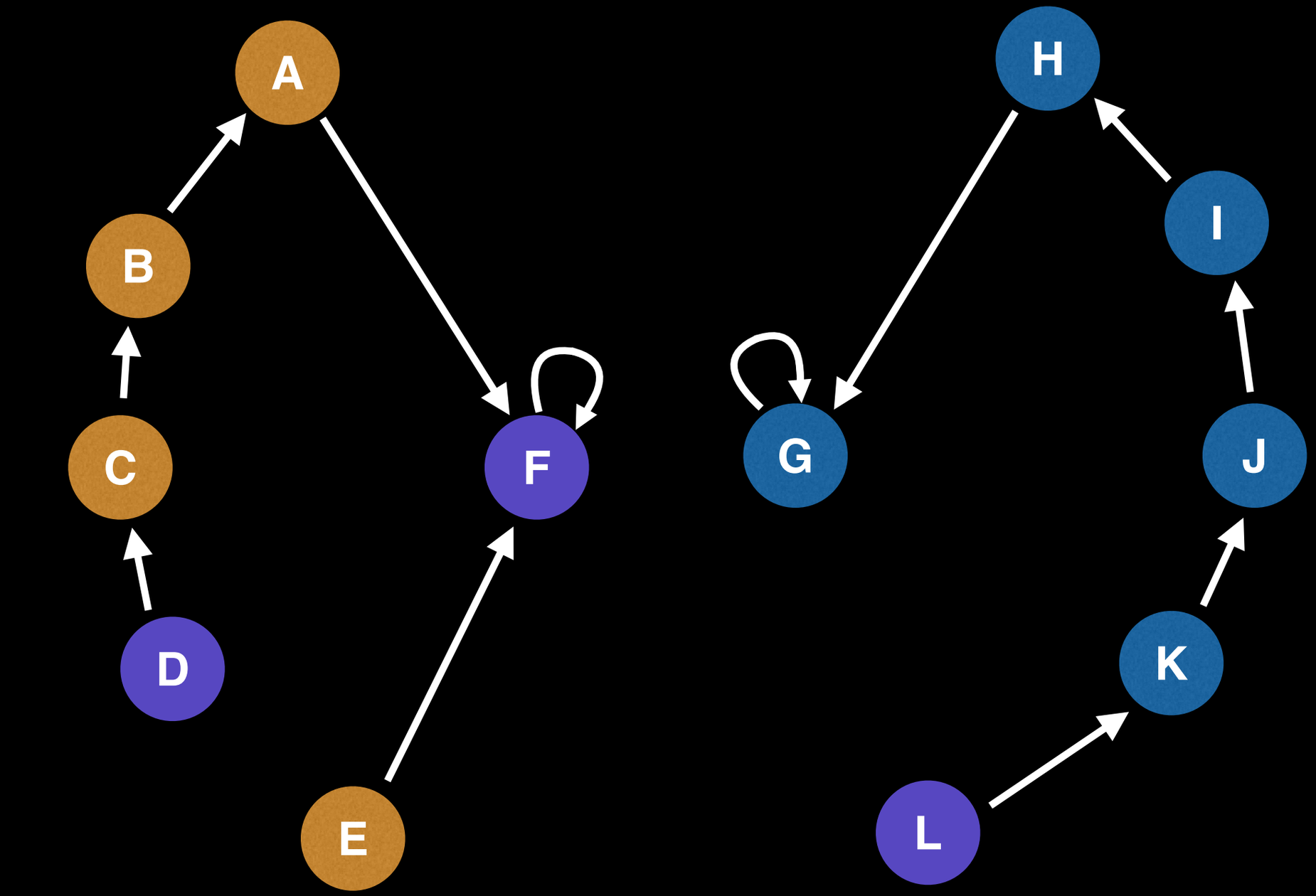
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



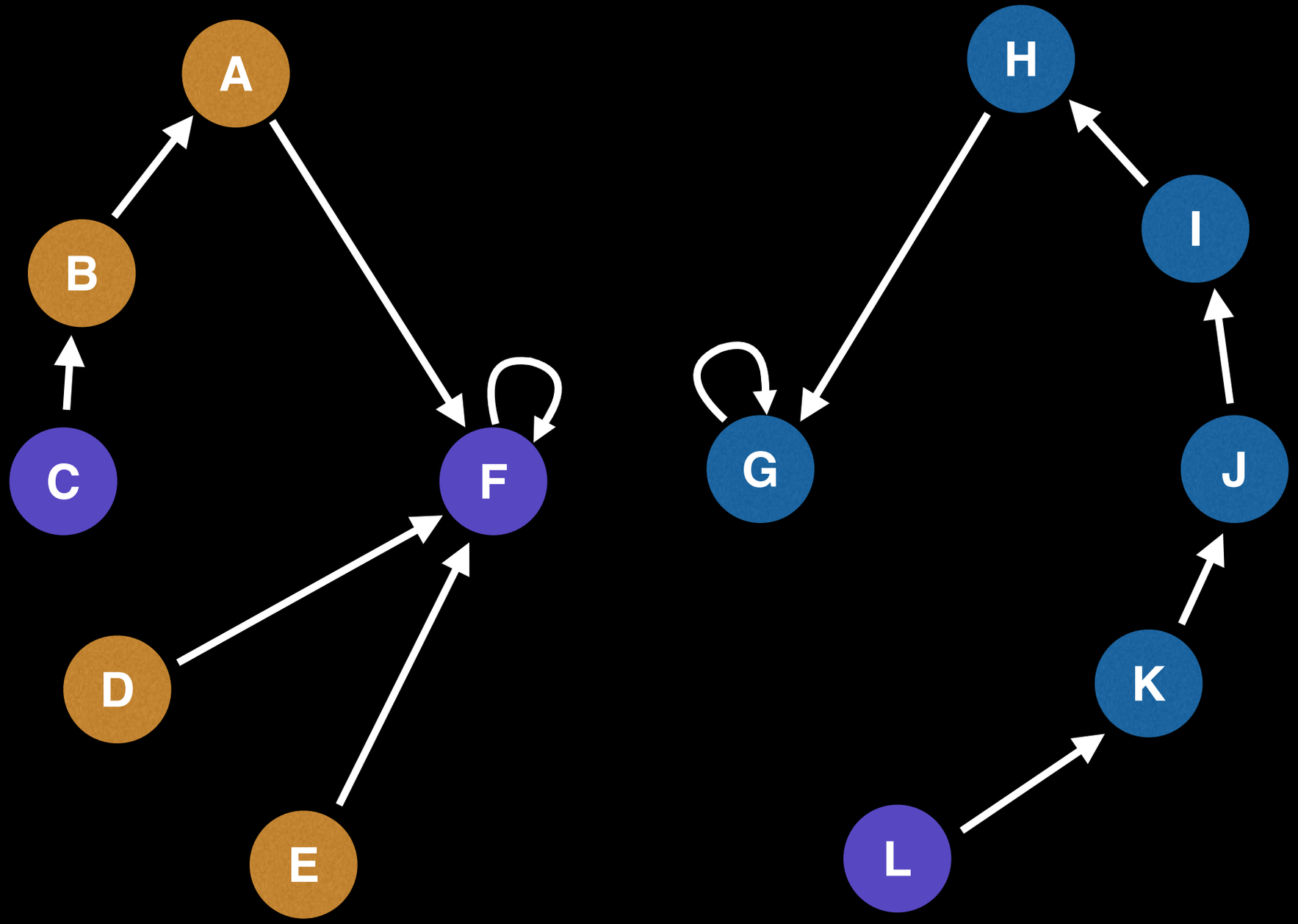
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



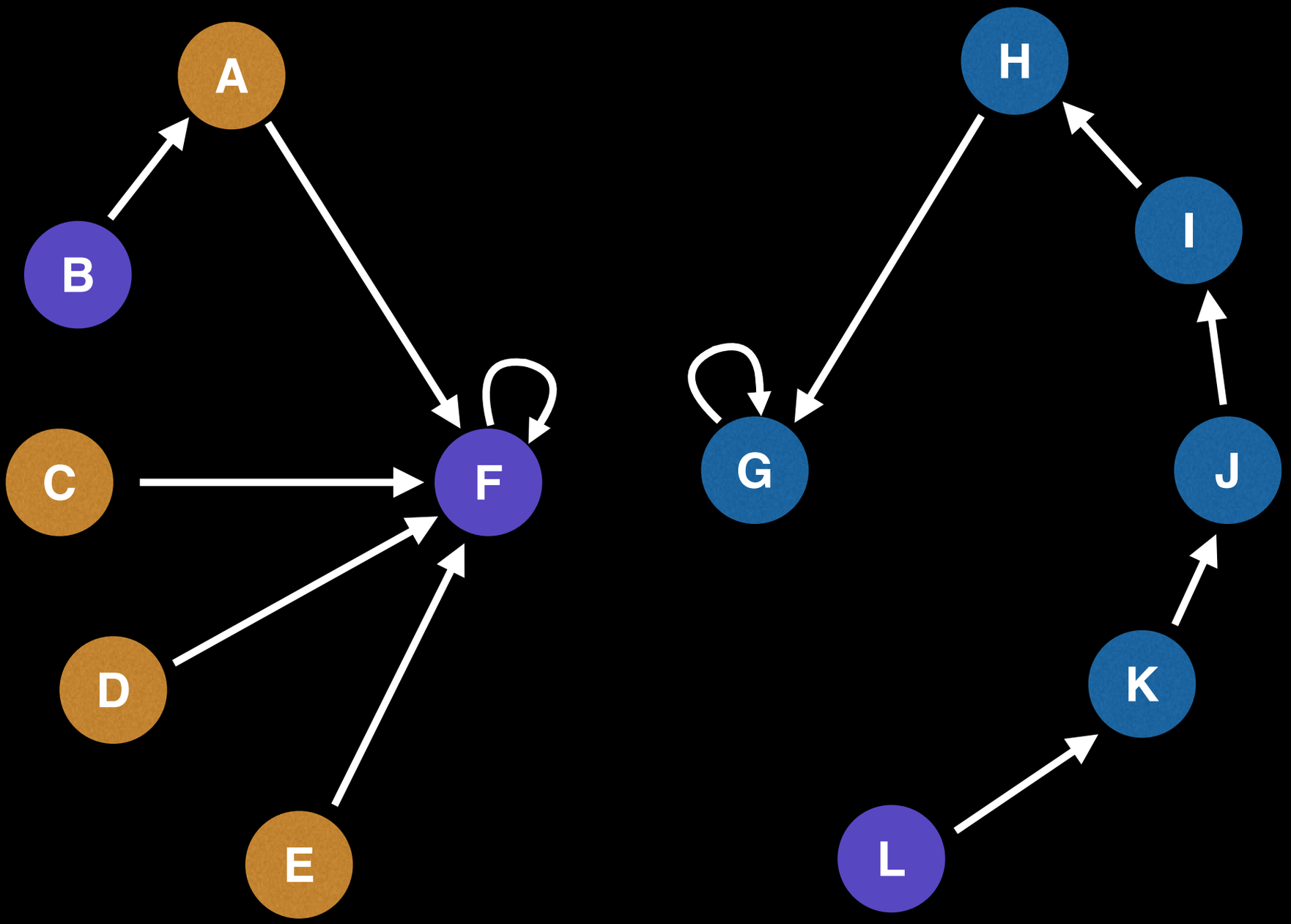
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



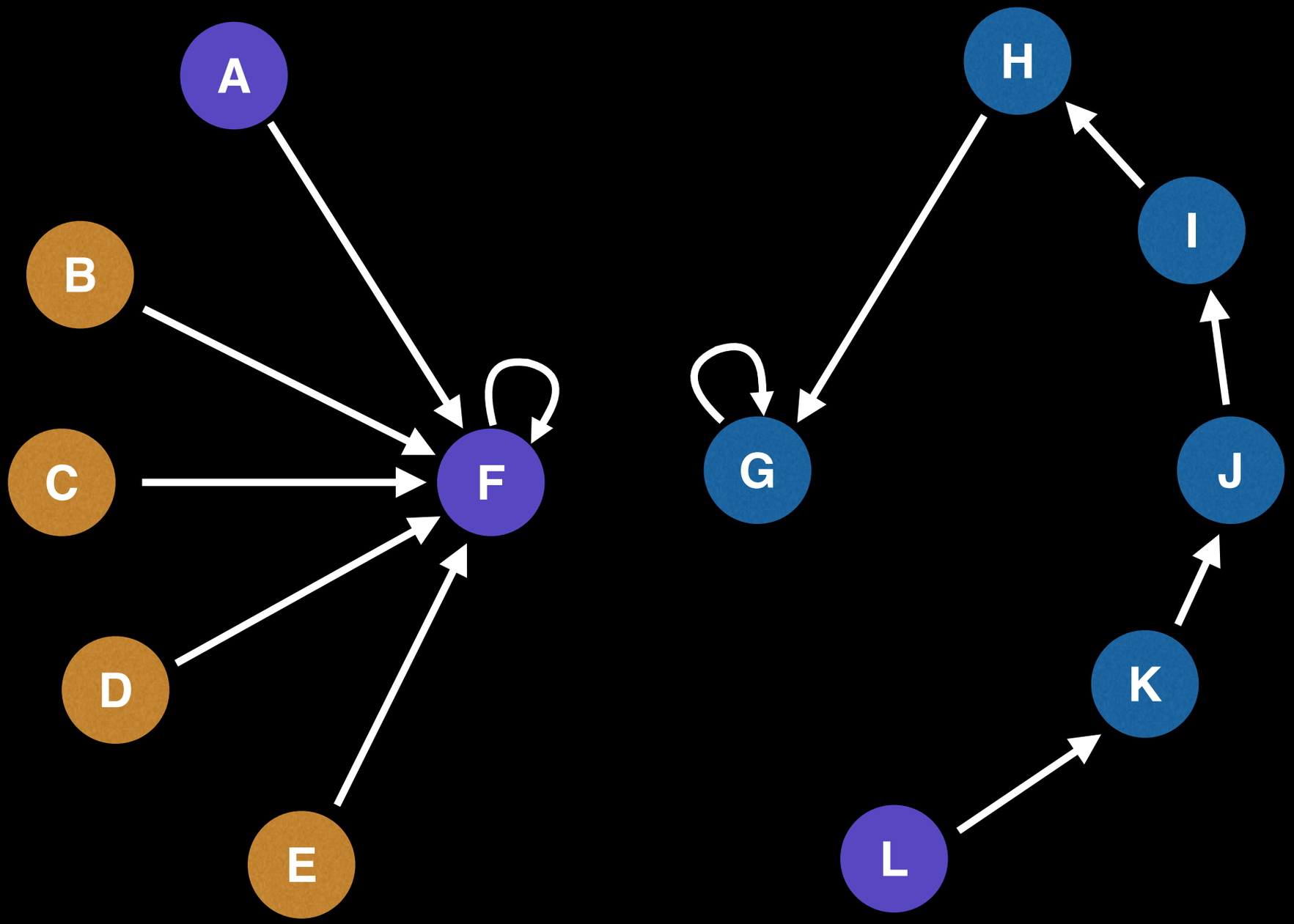
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



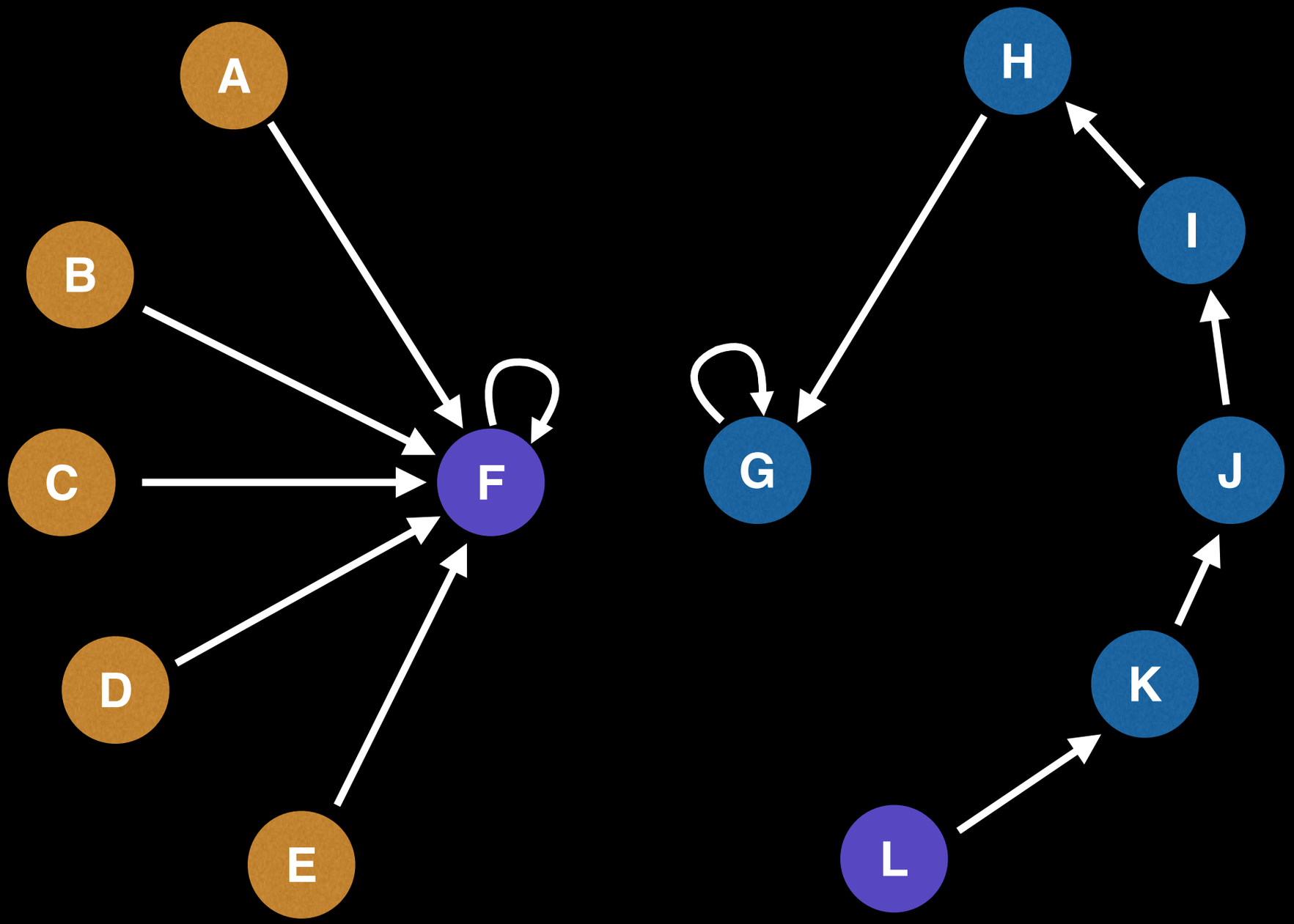
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



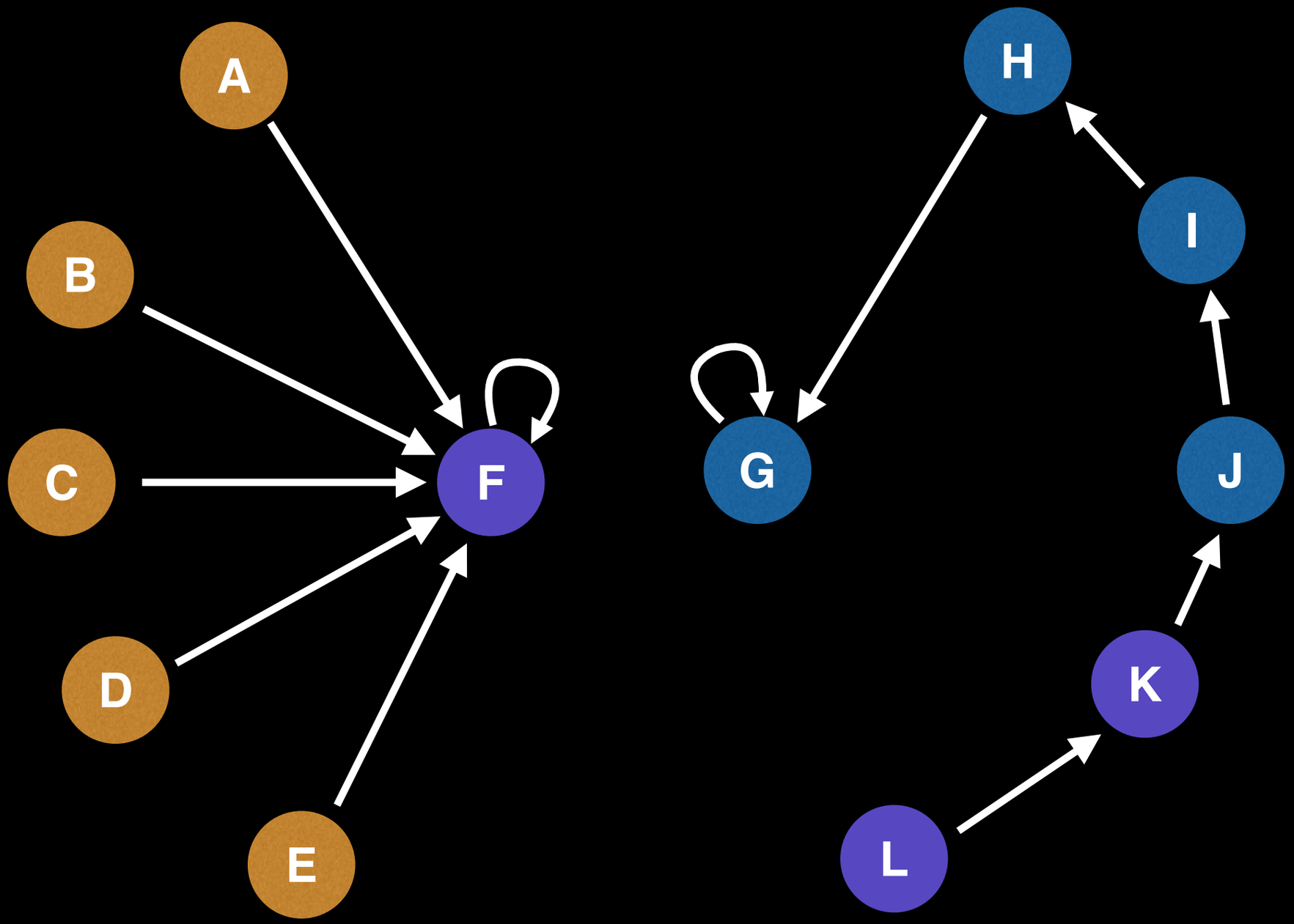
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



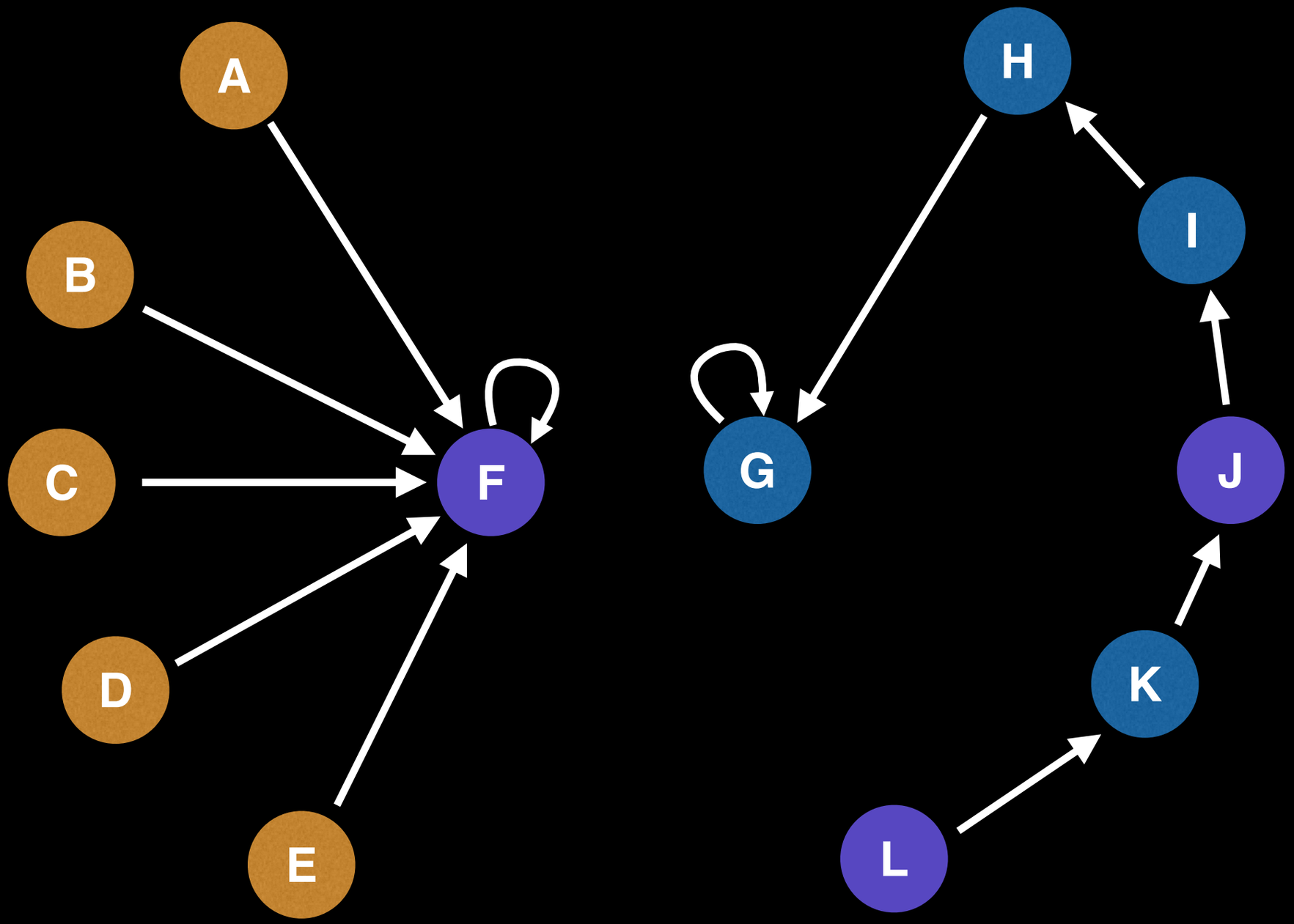
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



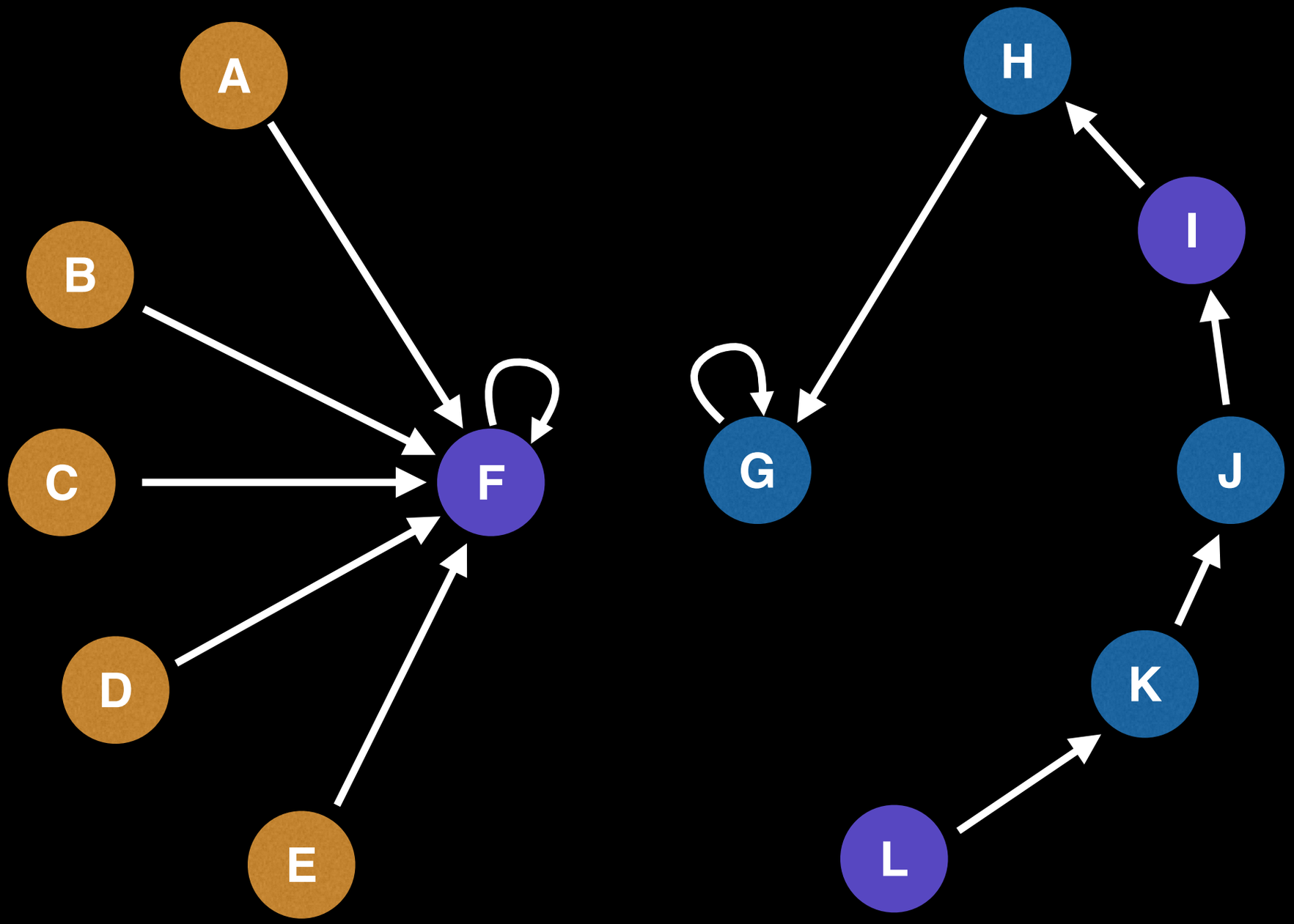
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



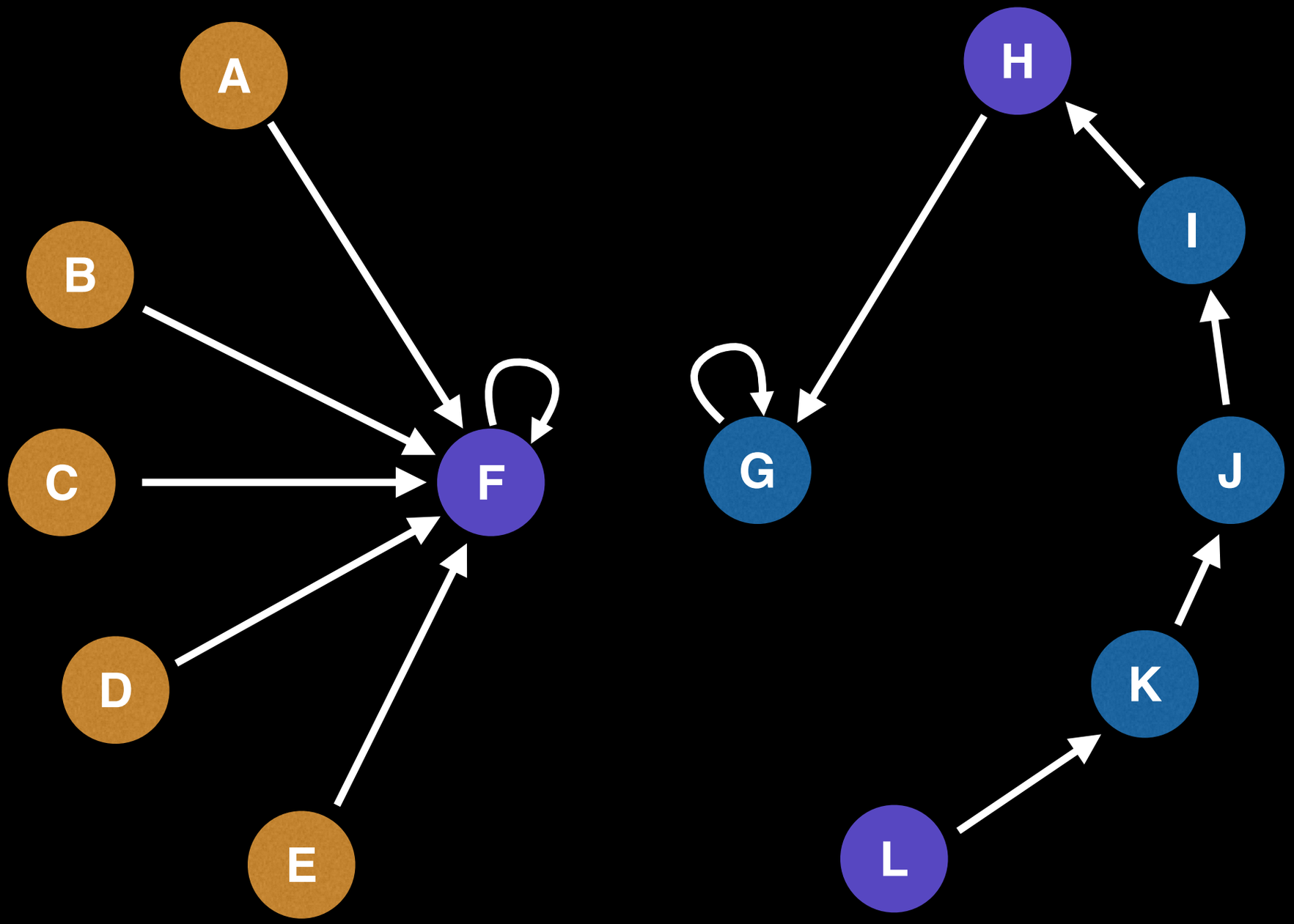
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



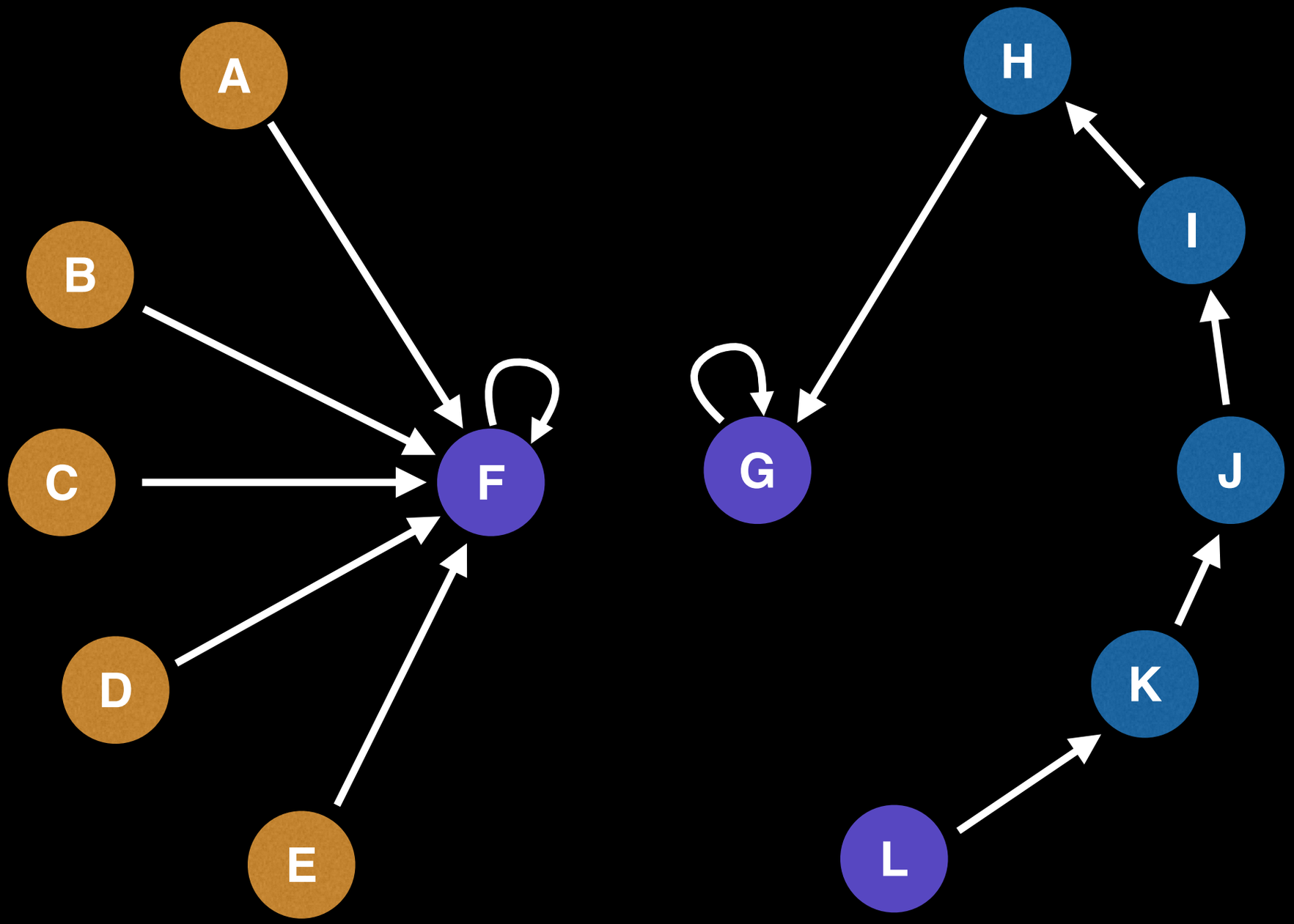
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



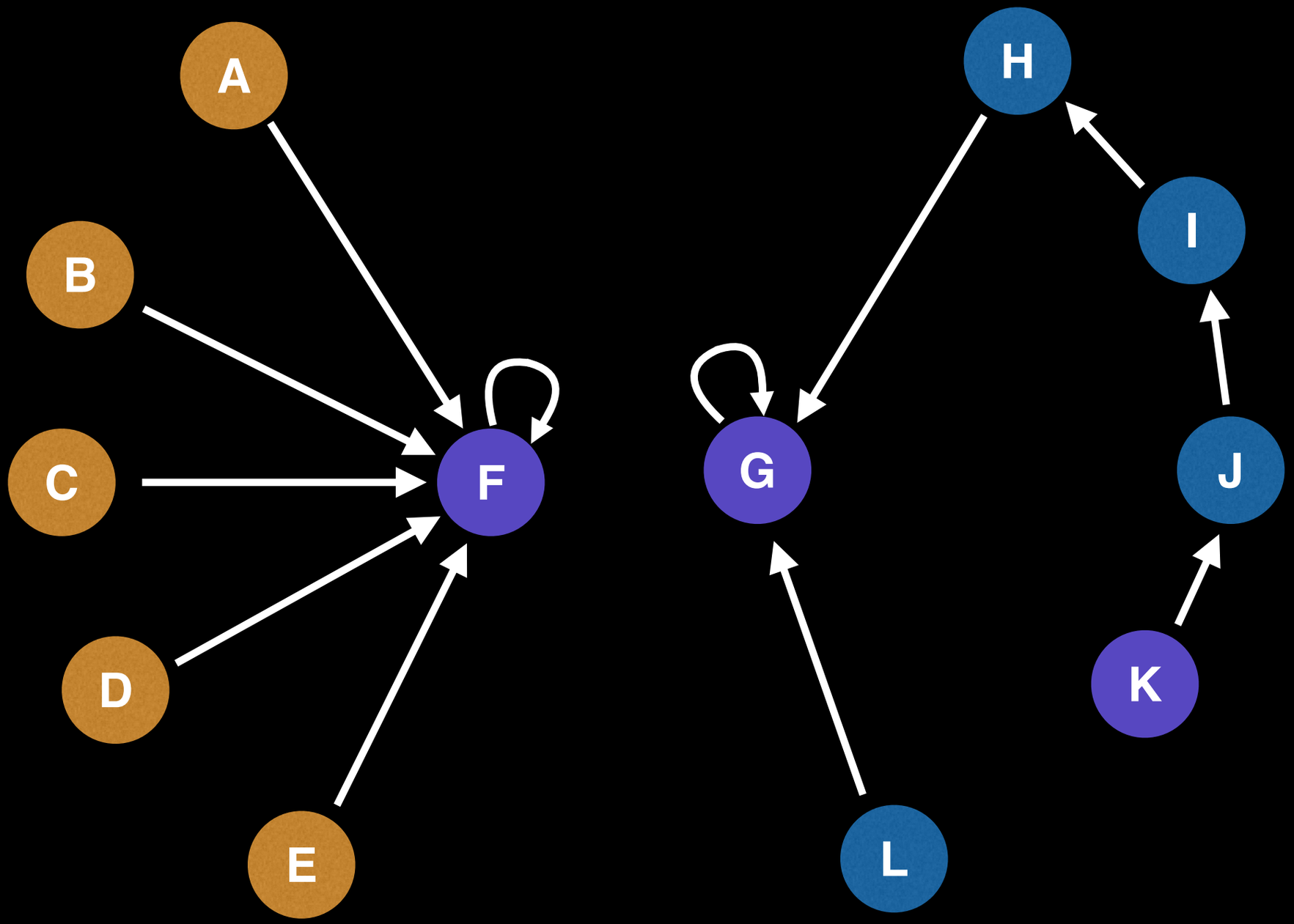
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



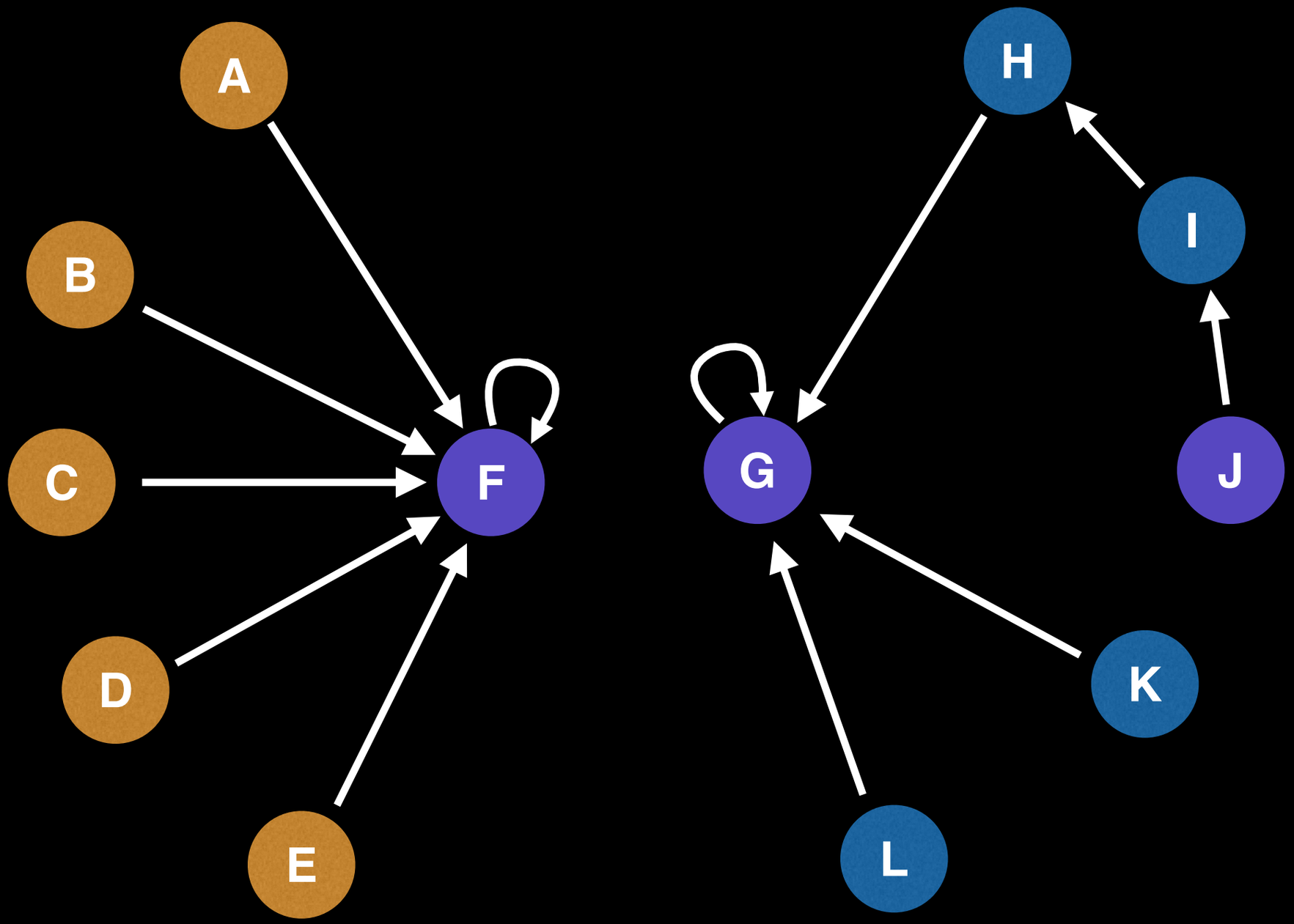
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



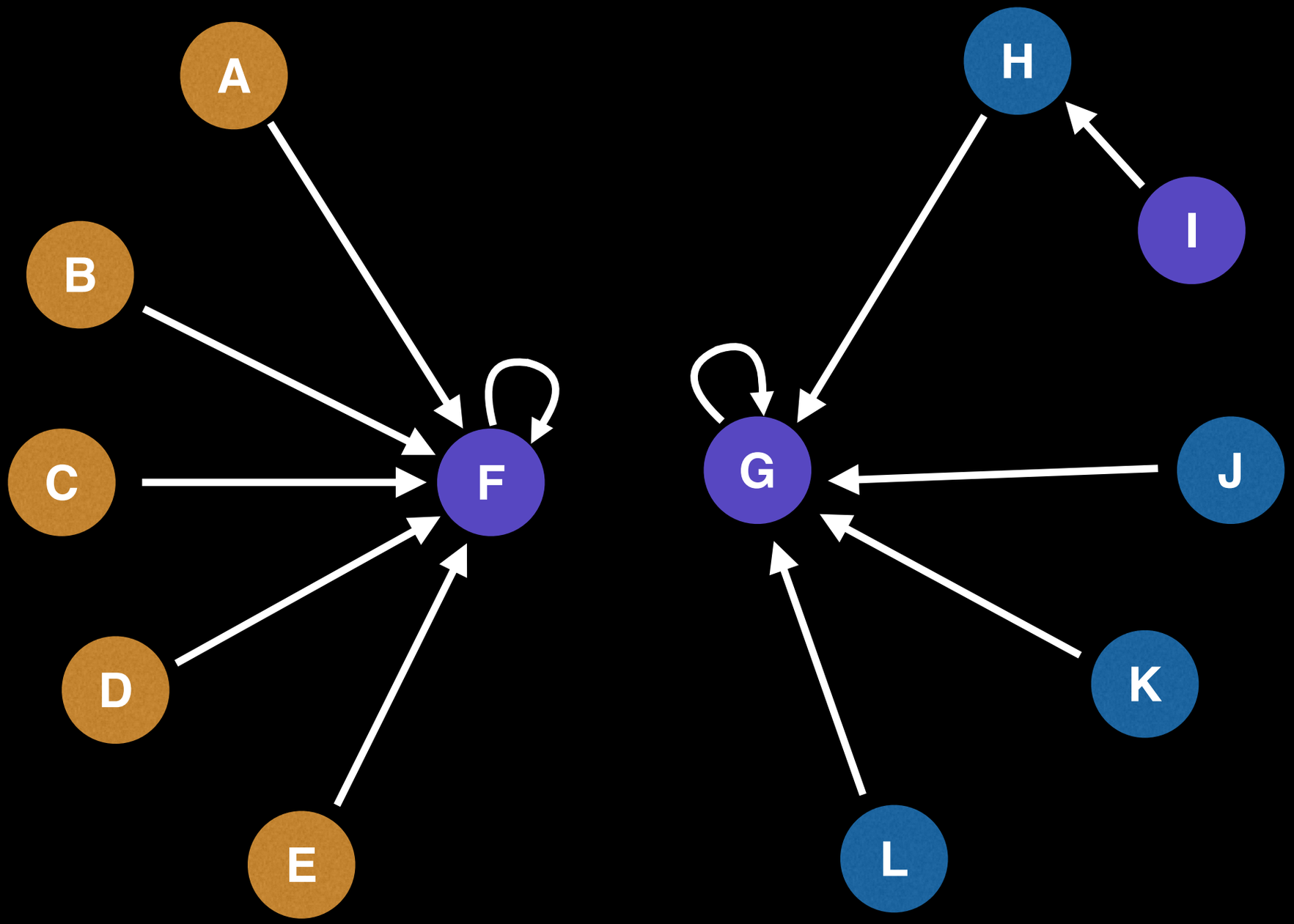
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



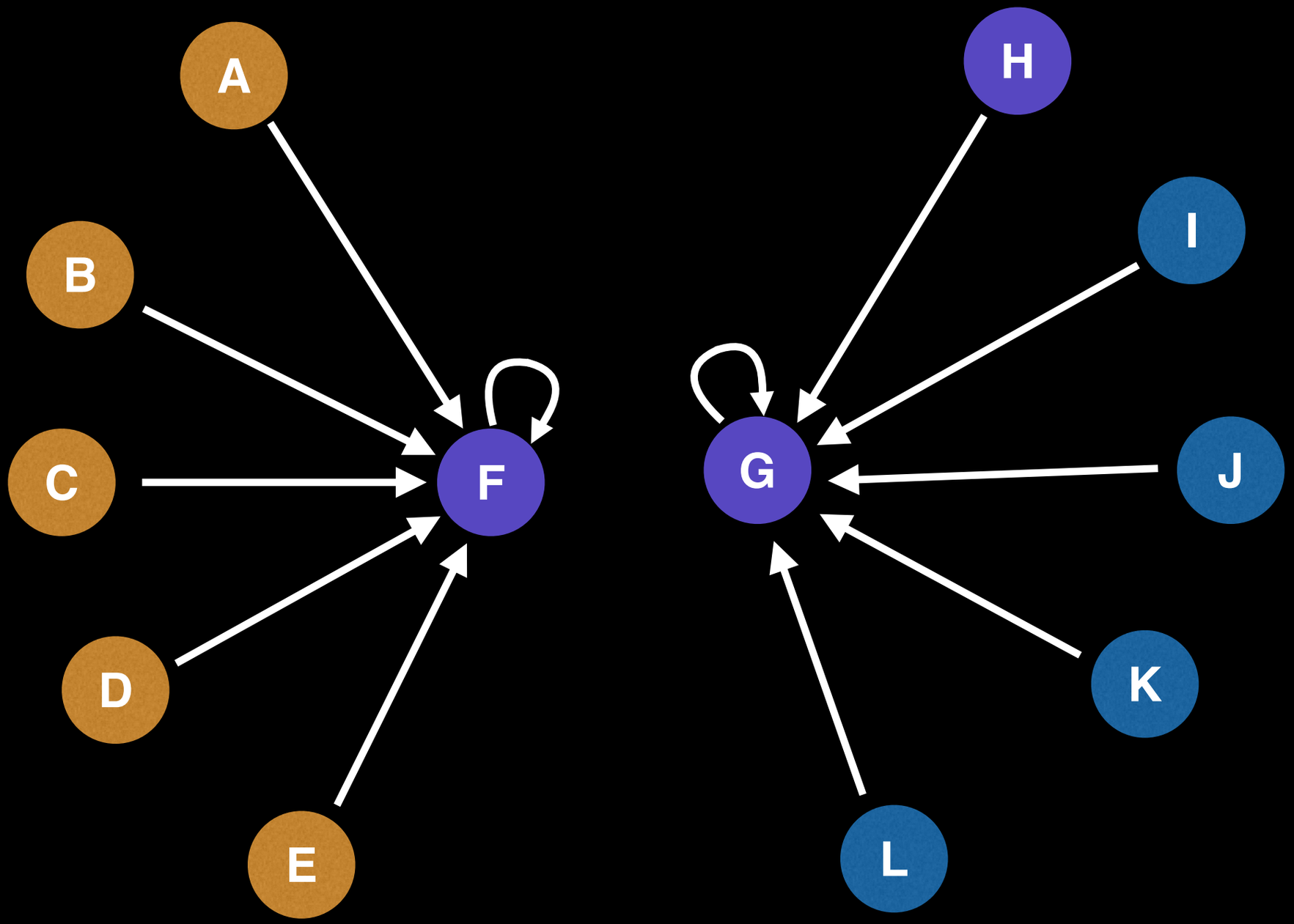
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



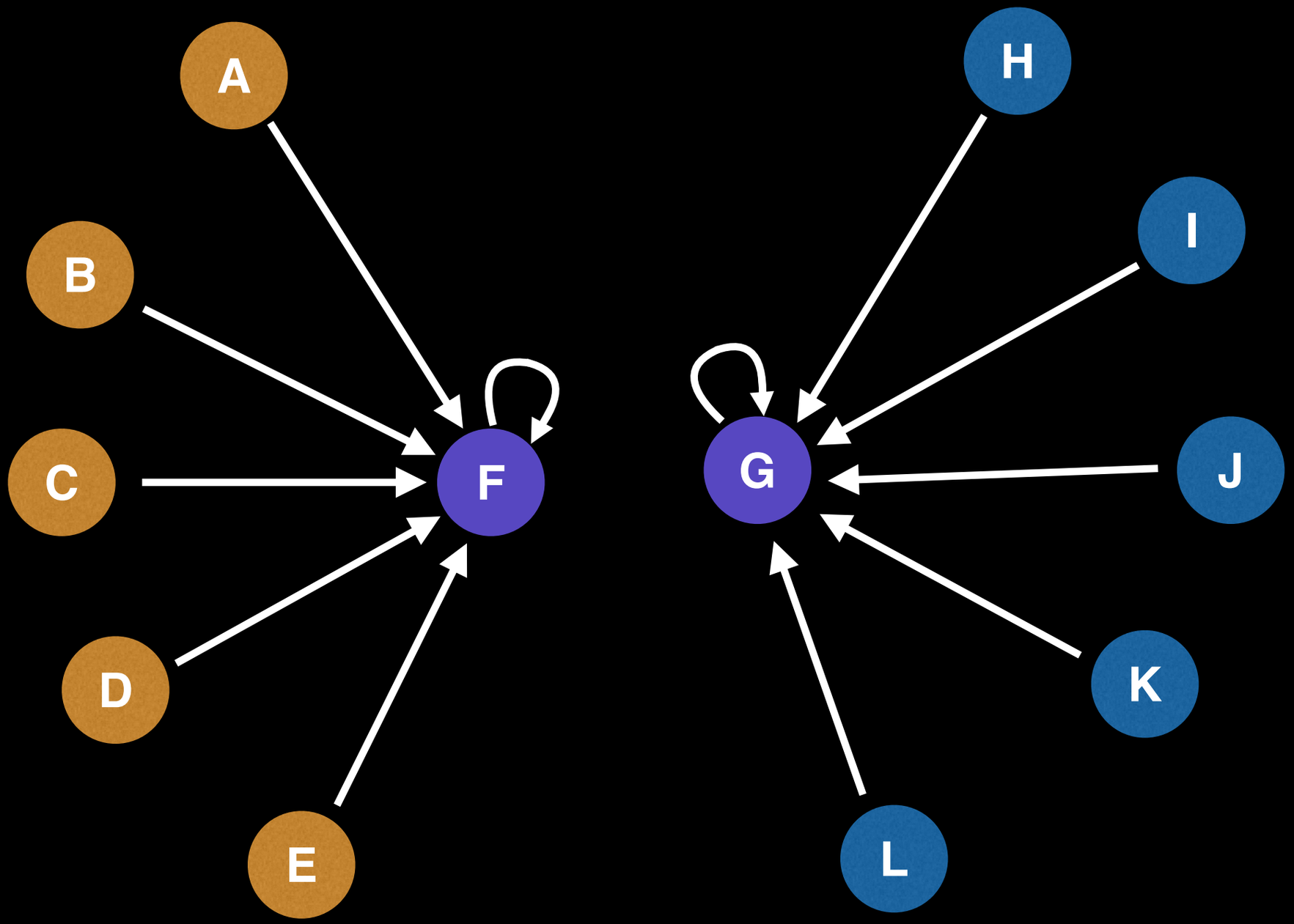
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



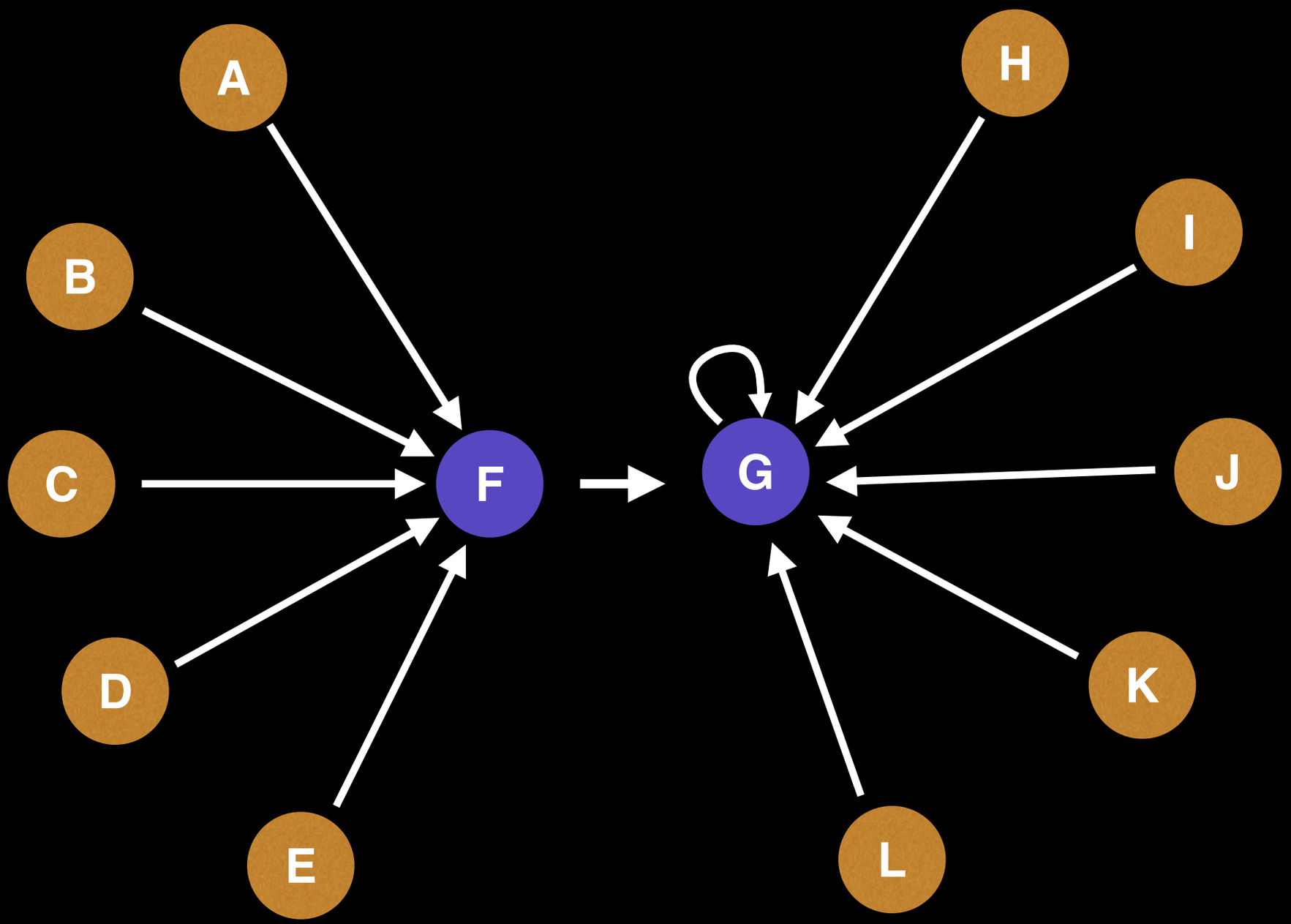
Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



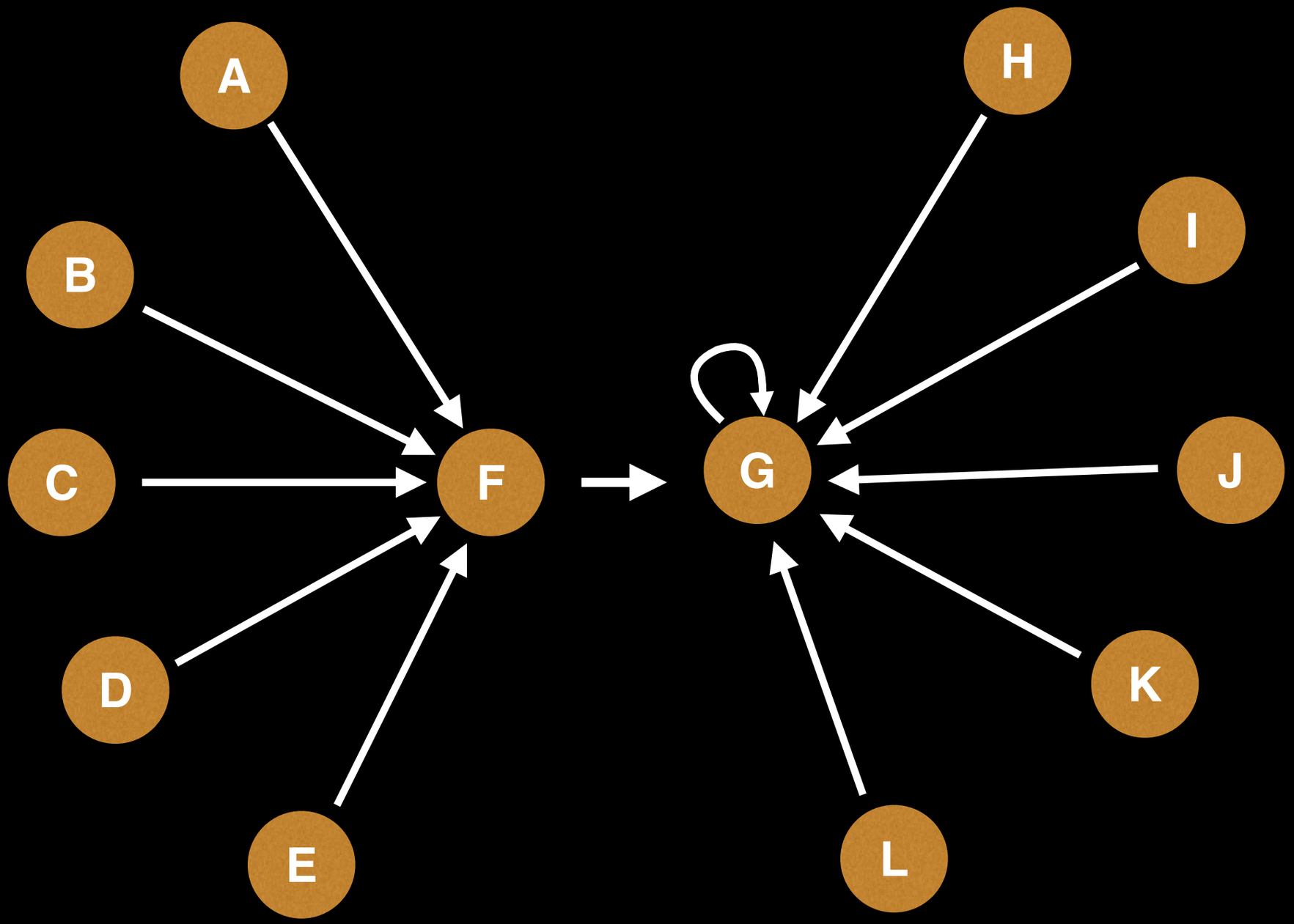
Indicates that there is a pointer to this node

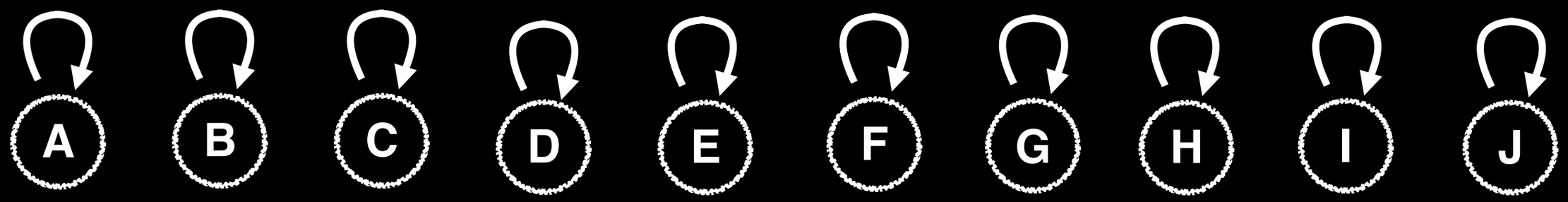
Hypothetical Union Find path compression example



Indicates that there is a pointer to this node

Hypothetical Union Find path compression example



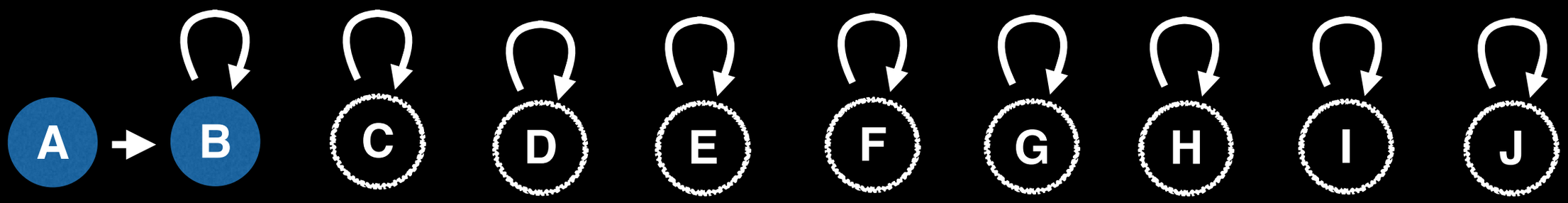


Using regular union find method

Instructions:

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

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

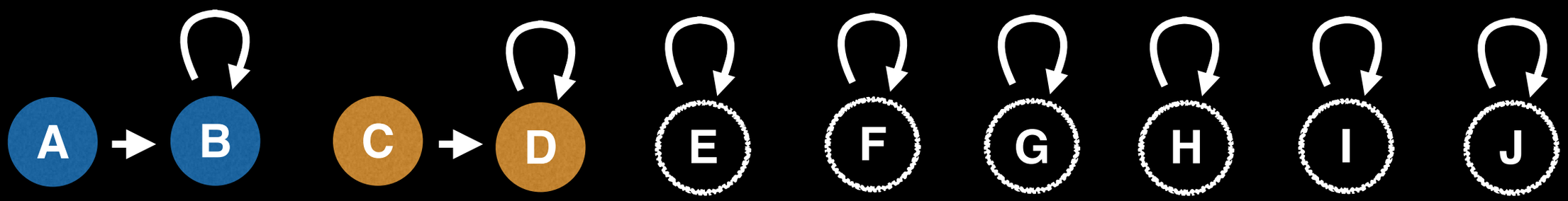


Using regular union find method

Instructions:

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

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

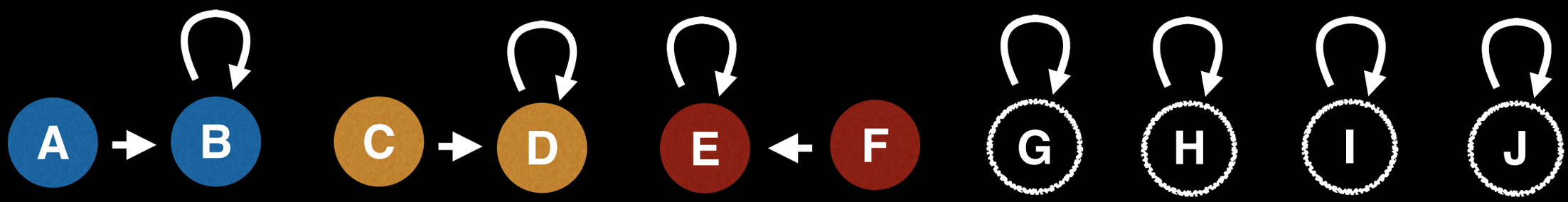


Using regular union find method

Instructions:

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

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

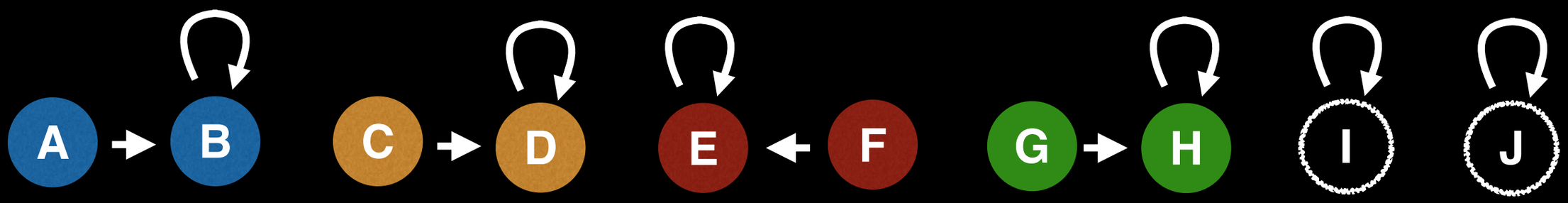


Using regular union find method

Instructions:

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

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

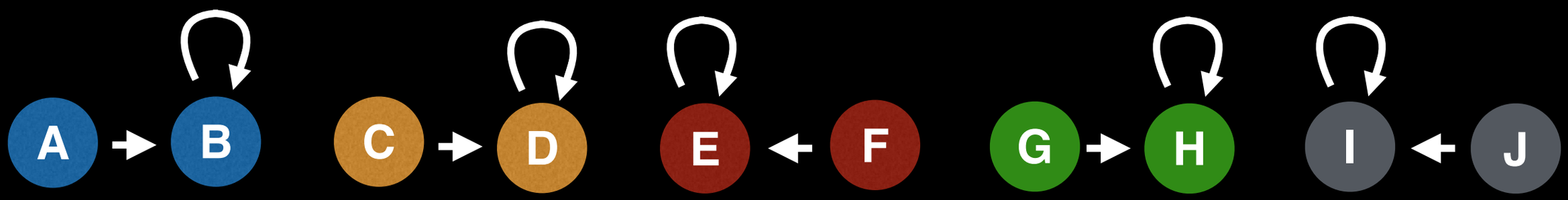


Using regular union find method

Instructions:

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

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

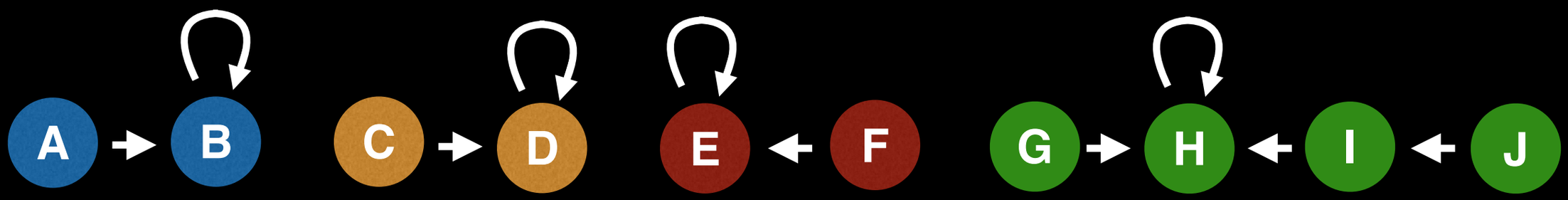


Using regular union find method

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

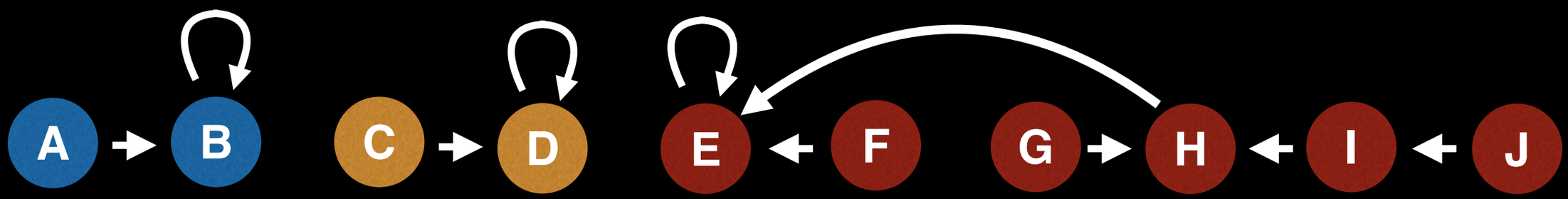


Using regular union find method

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

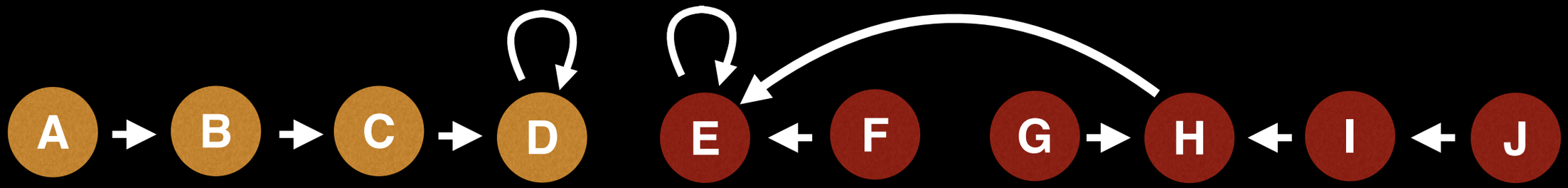


Using regular union find method

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

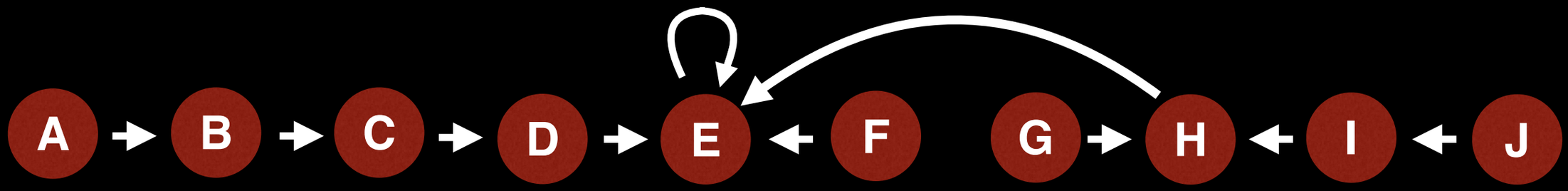


Using regular union find method

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

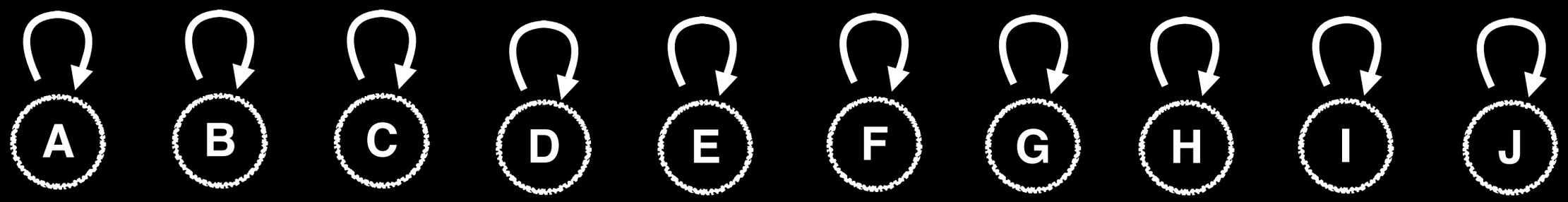


Using regular union find method

Instructions:

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

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

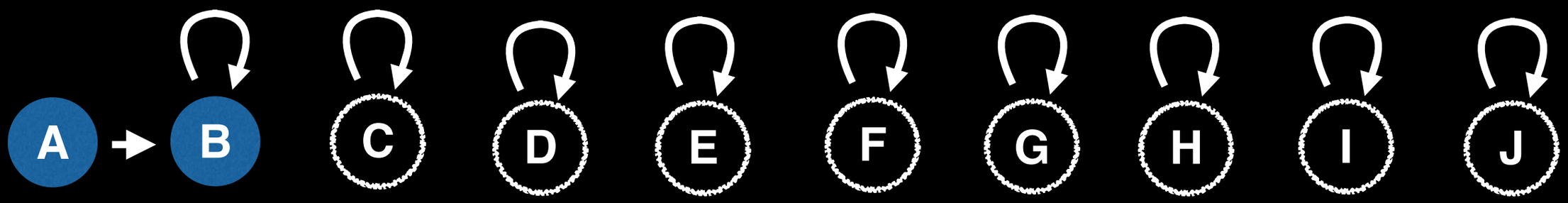


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

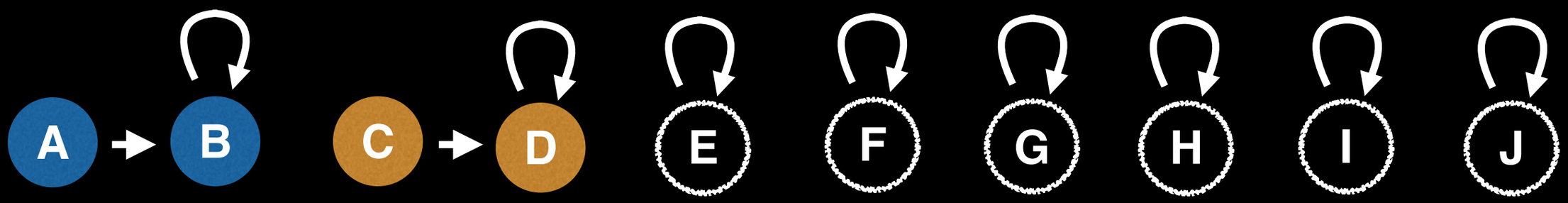


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

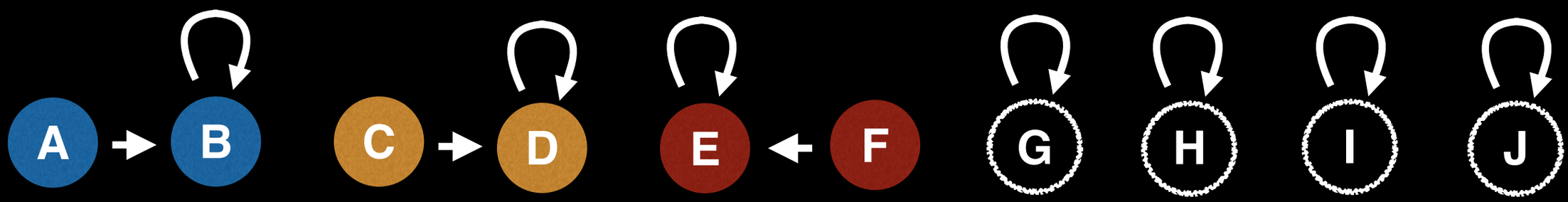


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

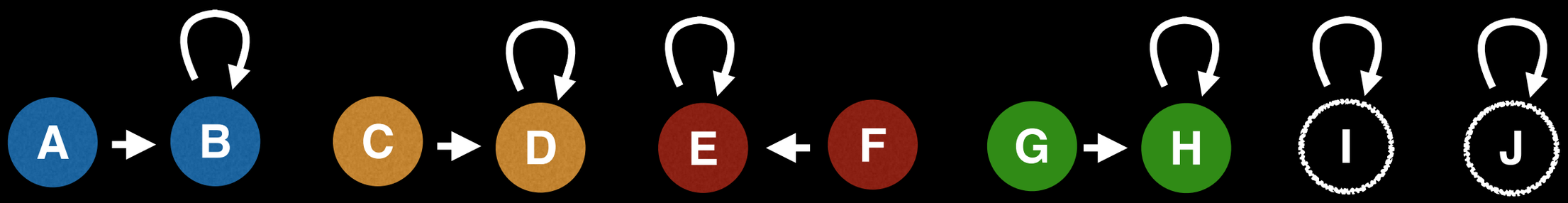


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

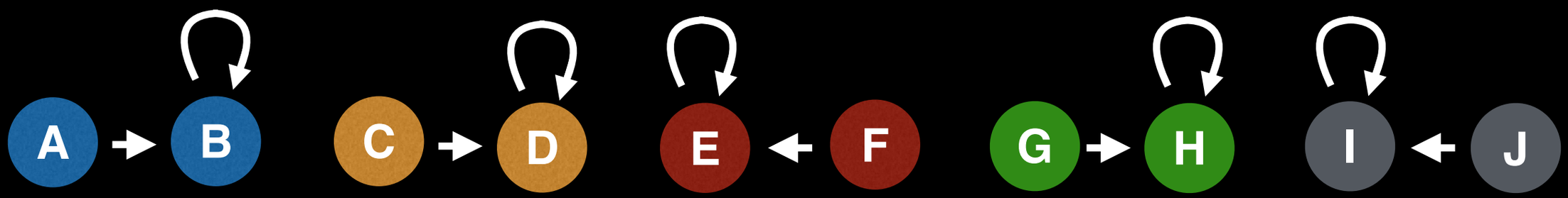


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

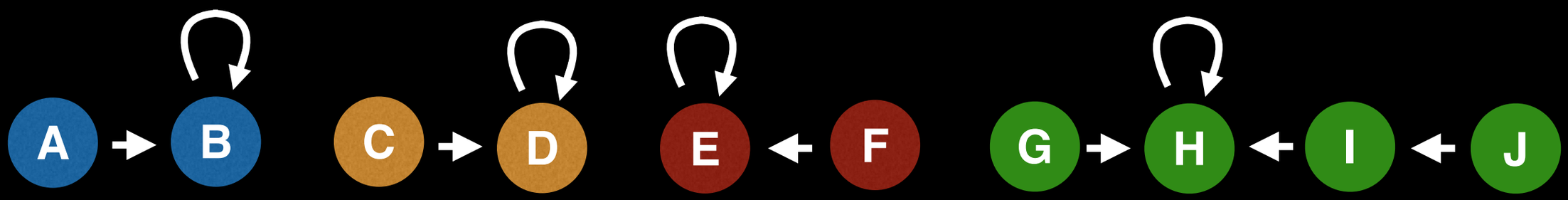


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

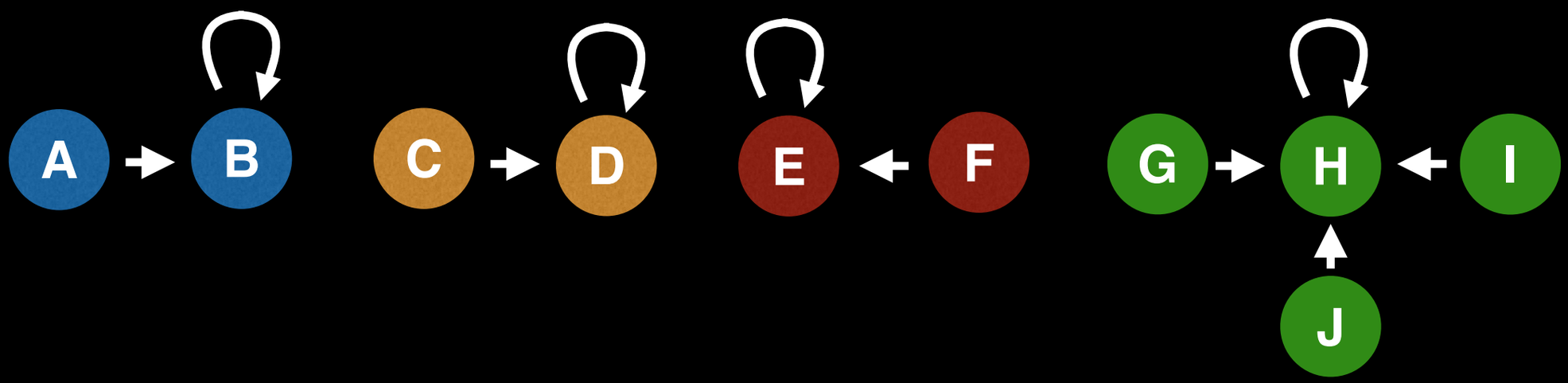


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

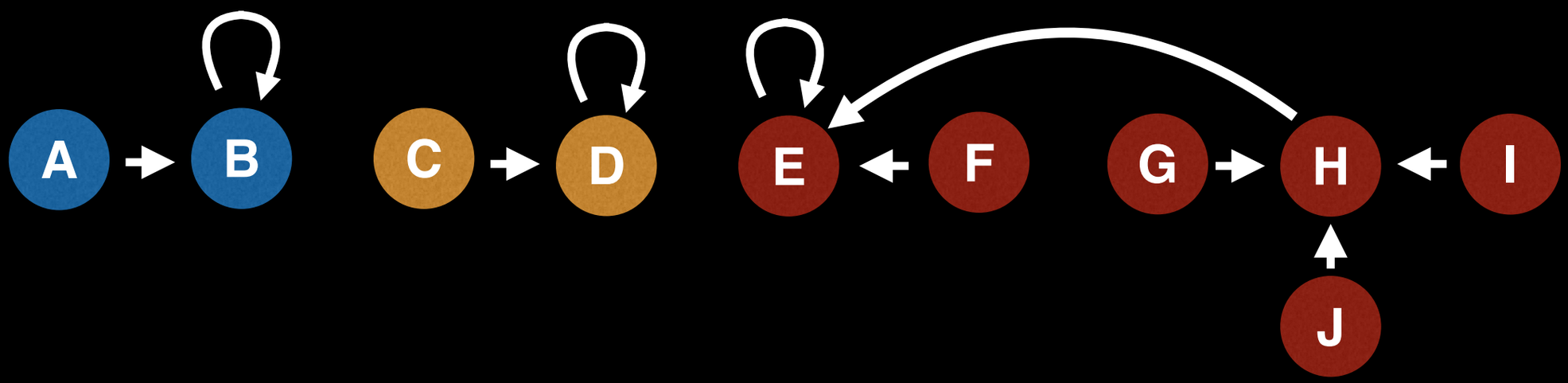


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

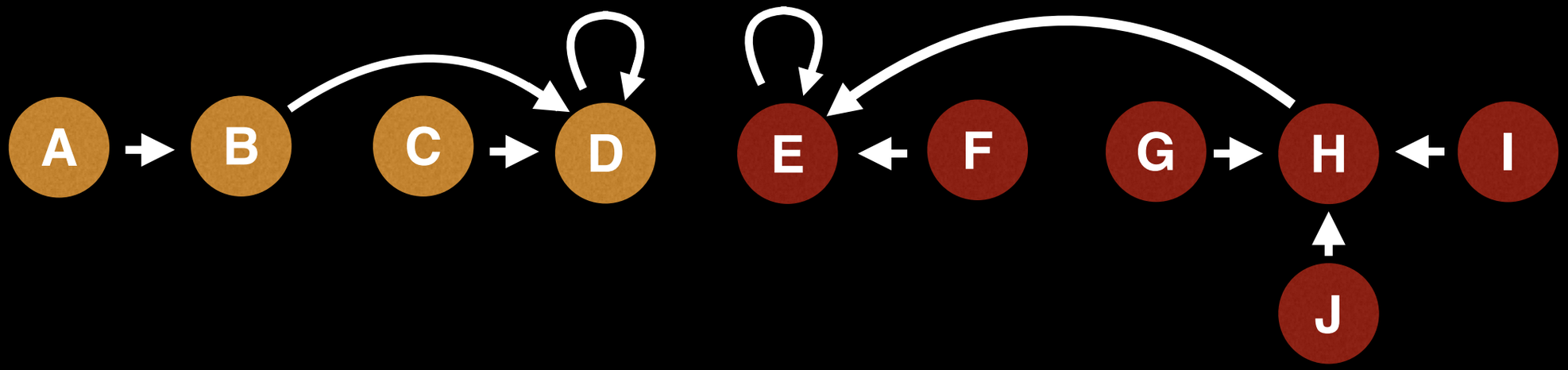


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

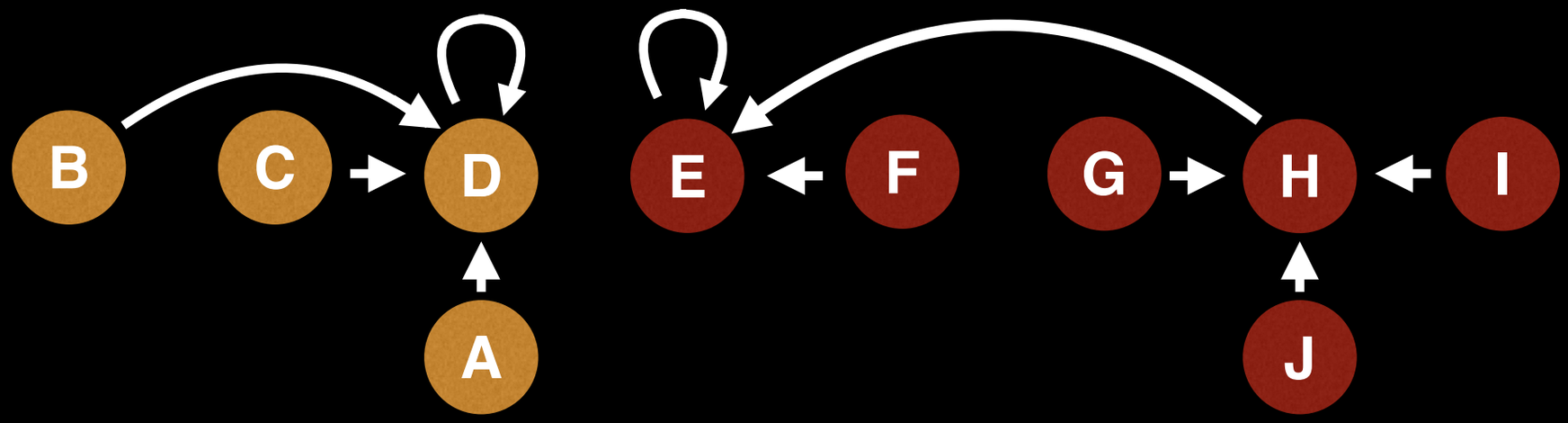


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

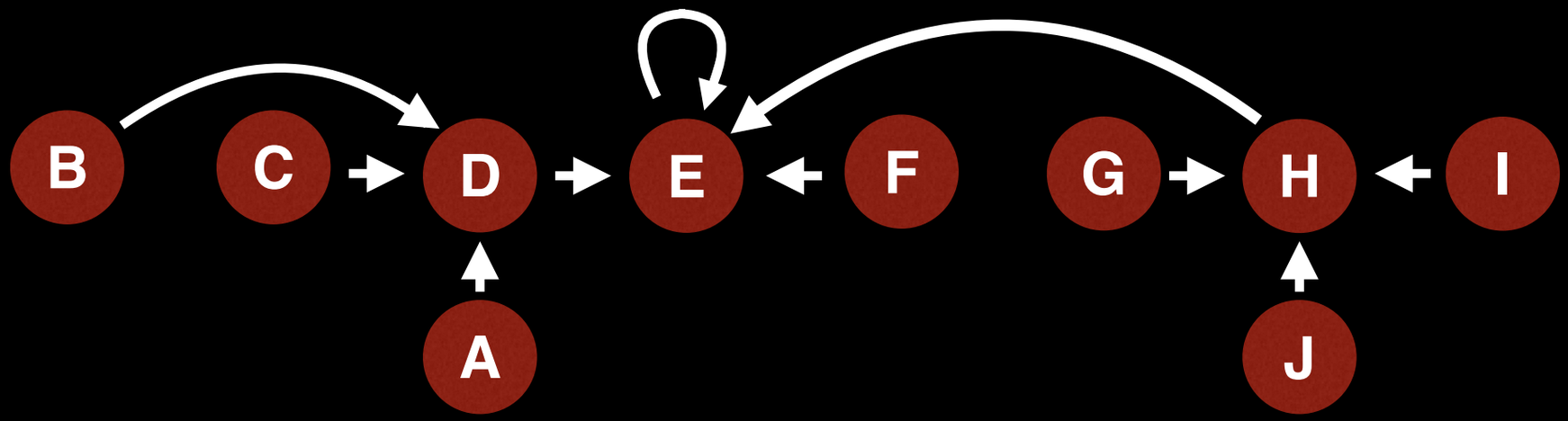


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

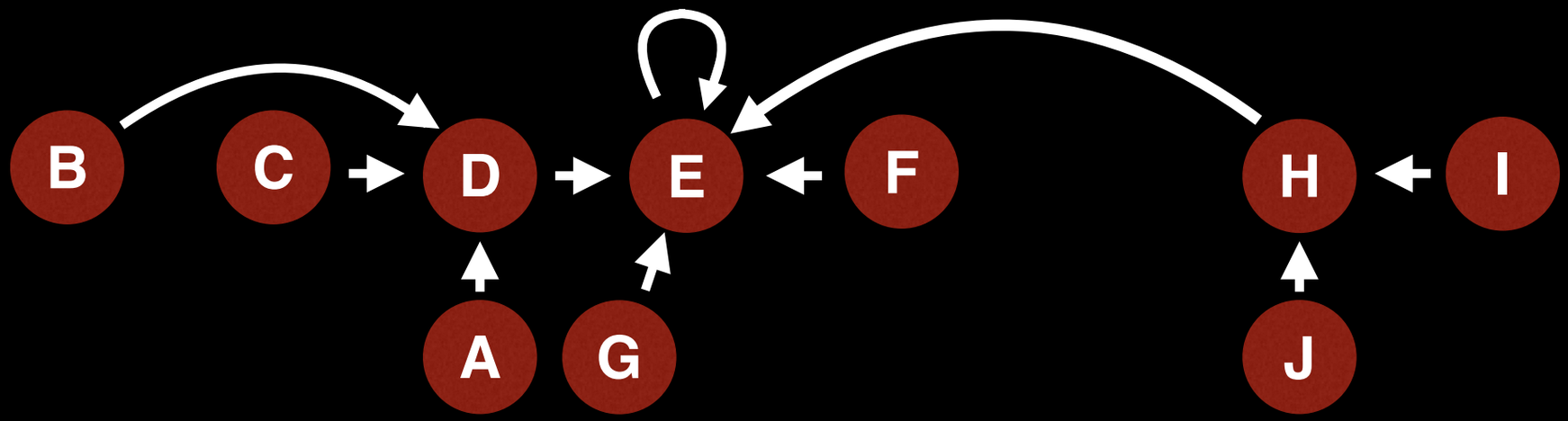


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

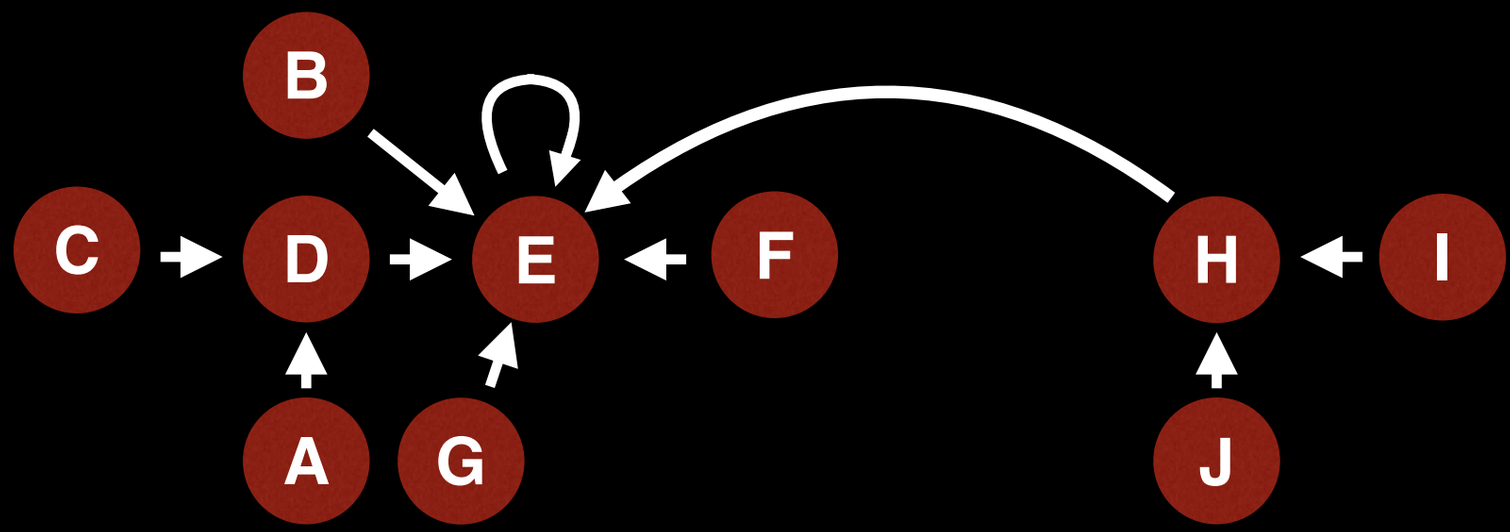


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

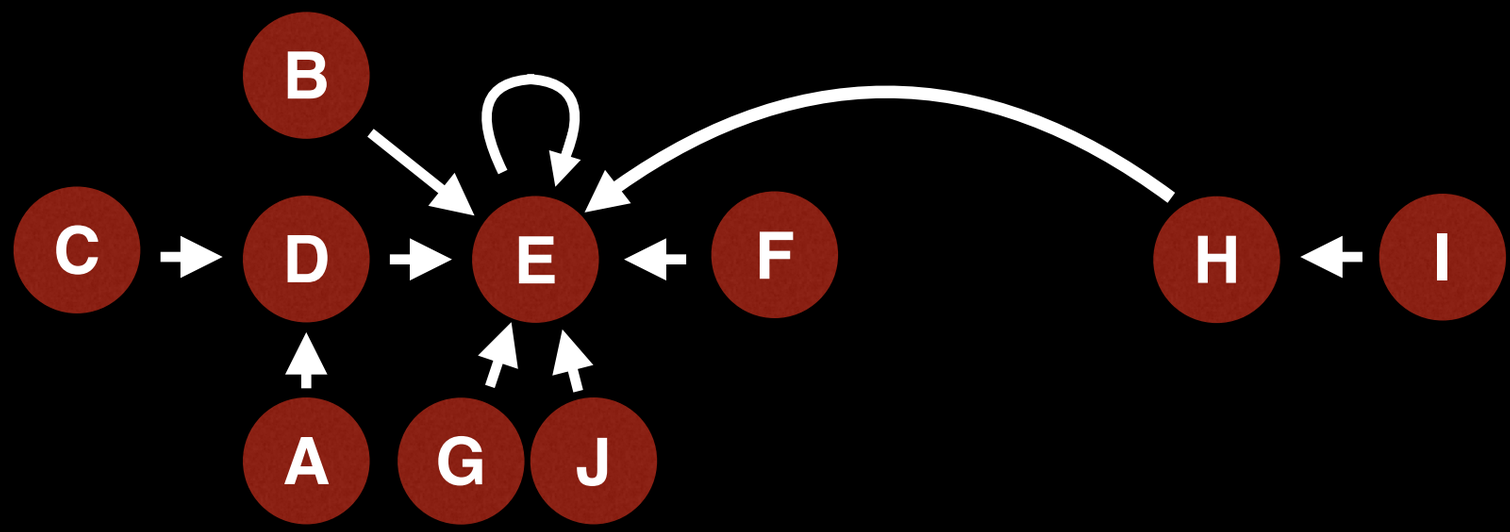


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

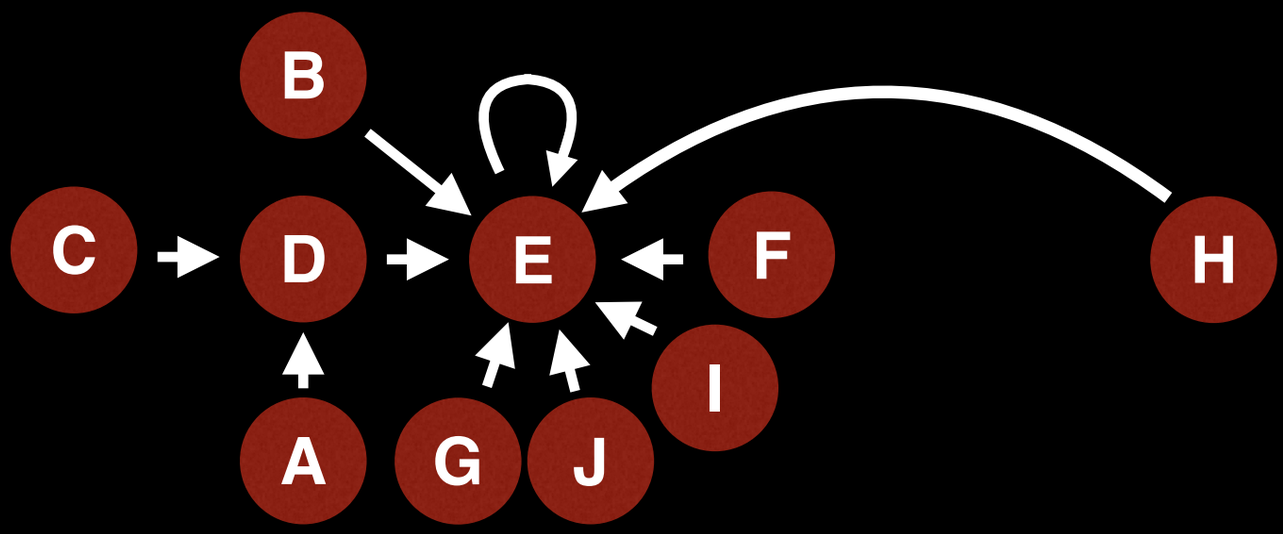


Using **path compression**

Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```



Instructions:

```
Union(A,B)
Union(C,D)
Union(E,F)
Union(G,H)
Union(I,J)
```

```
Union(J,G)
Union(H,F)
Union(A,C)
Union(D,E)
Union(G,B)
Union(I,J)
```

Union Find: Path Compression