CS101-Quiz7-Review

Suting Chen

CS101-Quiz7-Review

Key Points

1. Disjoint set

2. Minimum Spanning Tree

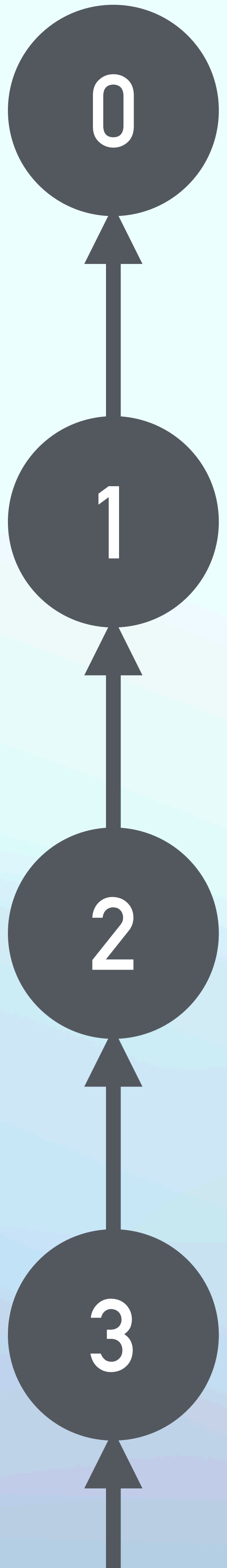
Disjoint set

Time complexity

1. No optimization
2. Union-by-rank
3. Union-by-rank and path compression

Disjoint set

Time complexity — No optimization

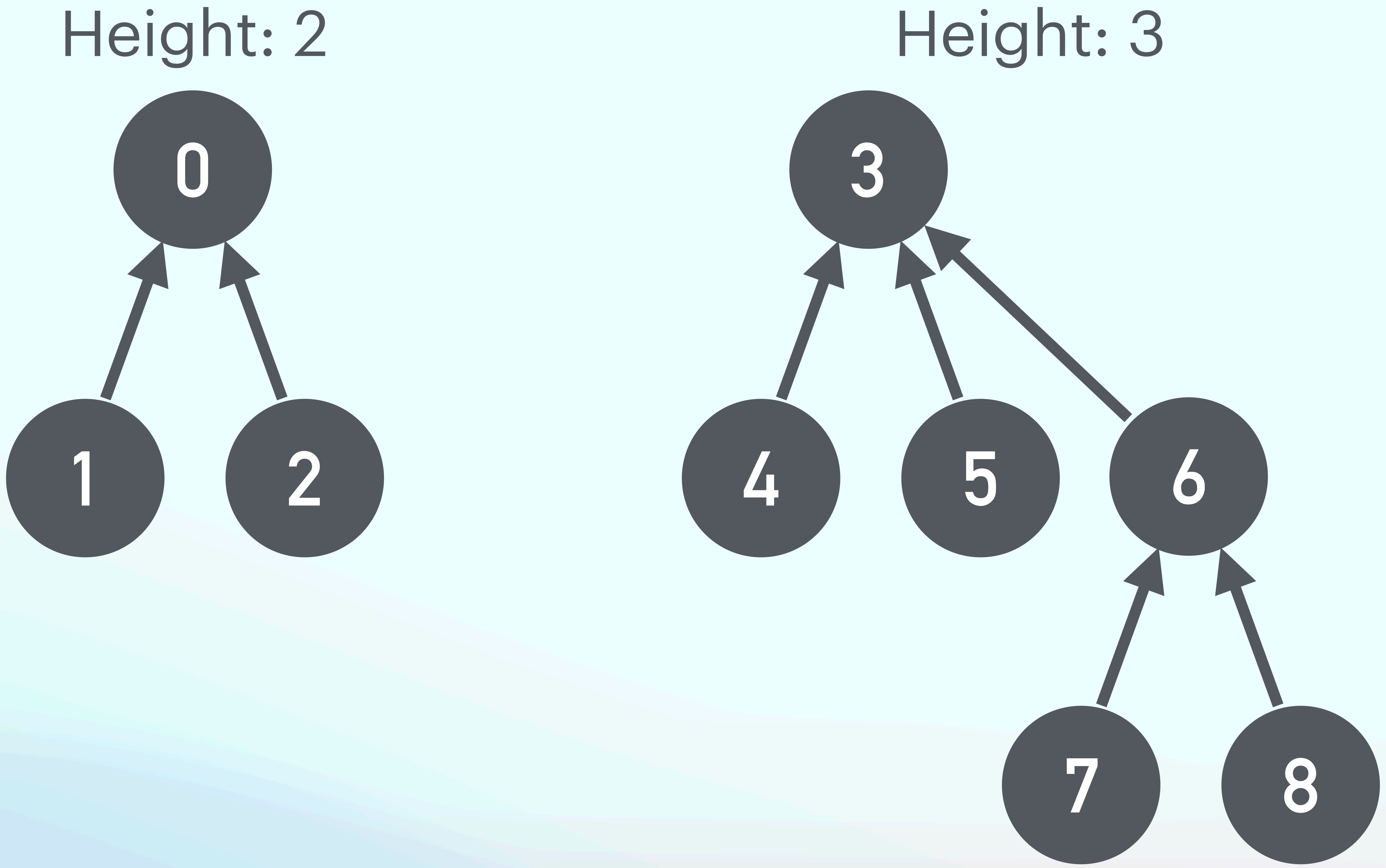


Node	0	1	2	3	4	5	6	7	8	9	10
Root	0	0	1	2	3	4	5	6	7	8	9

Obviously, every operation is $O(n)$ in the worst case.

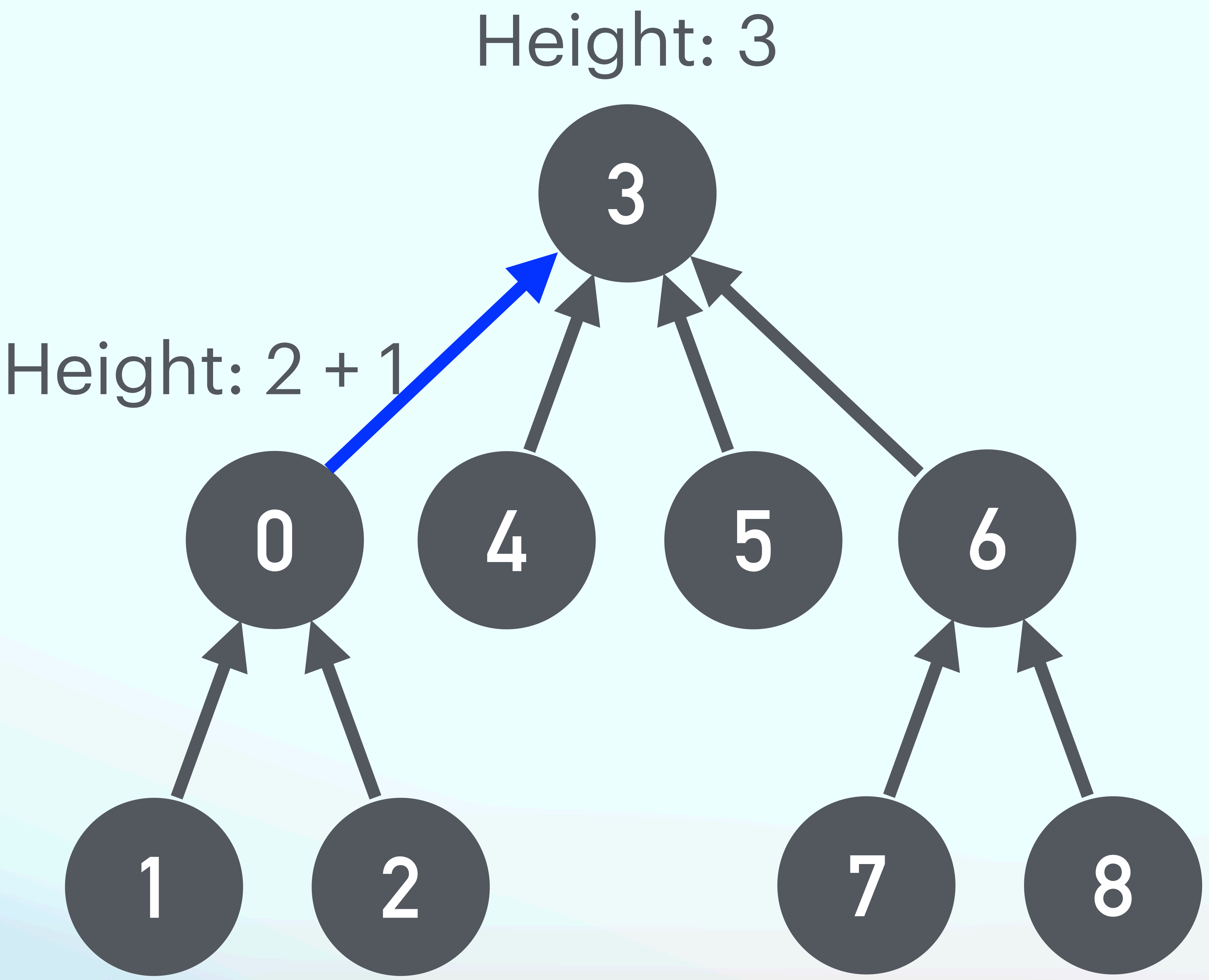
Disjoint set

Time complexity — Union-by-rank



Disjoint set

Time complexity — Union-by-rank

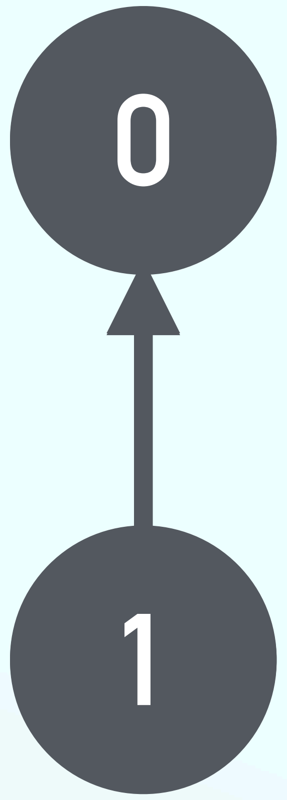


What is the worst case?

Disjoint set

Time complexity — Union-by-rank

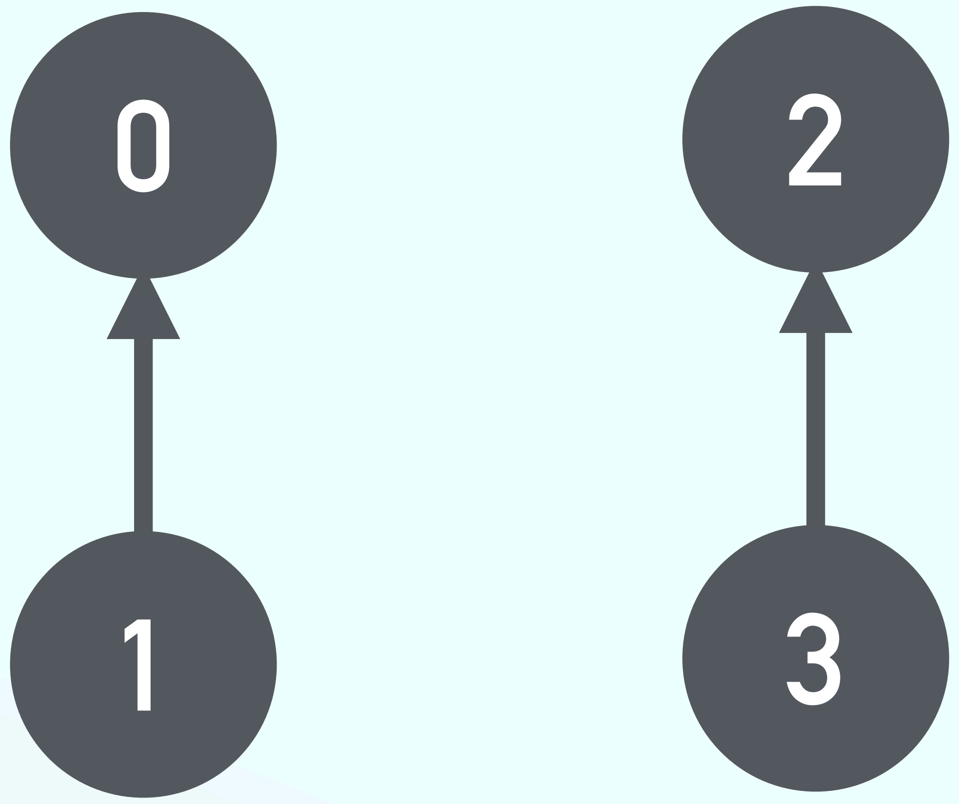
What is the worst case?



Disjoint set

Time complexity — Union-by-rank

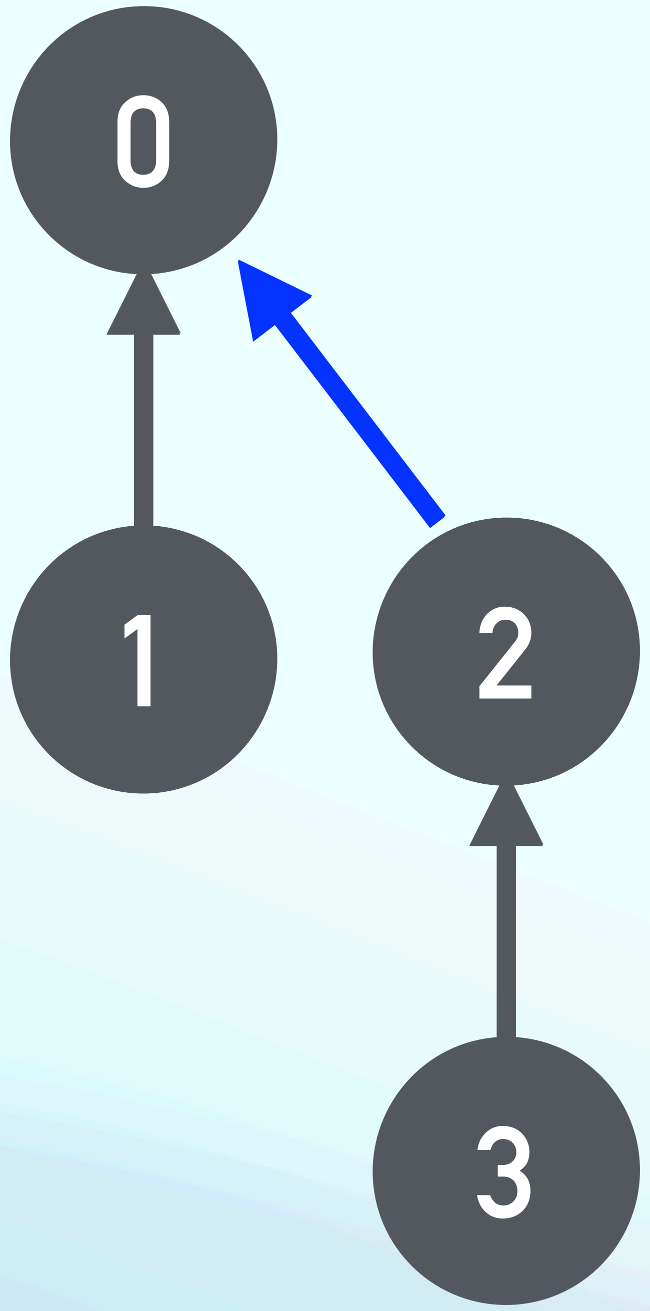
What is the worst case?



Disjoint set

Time complexity — Union-by-rank

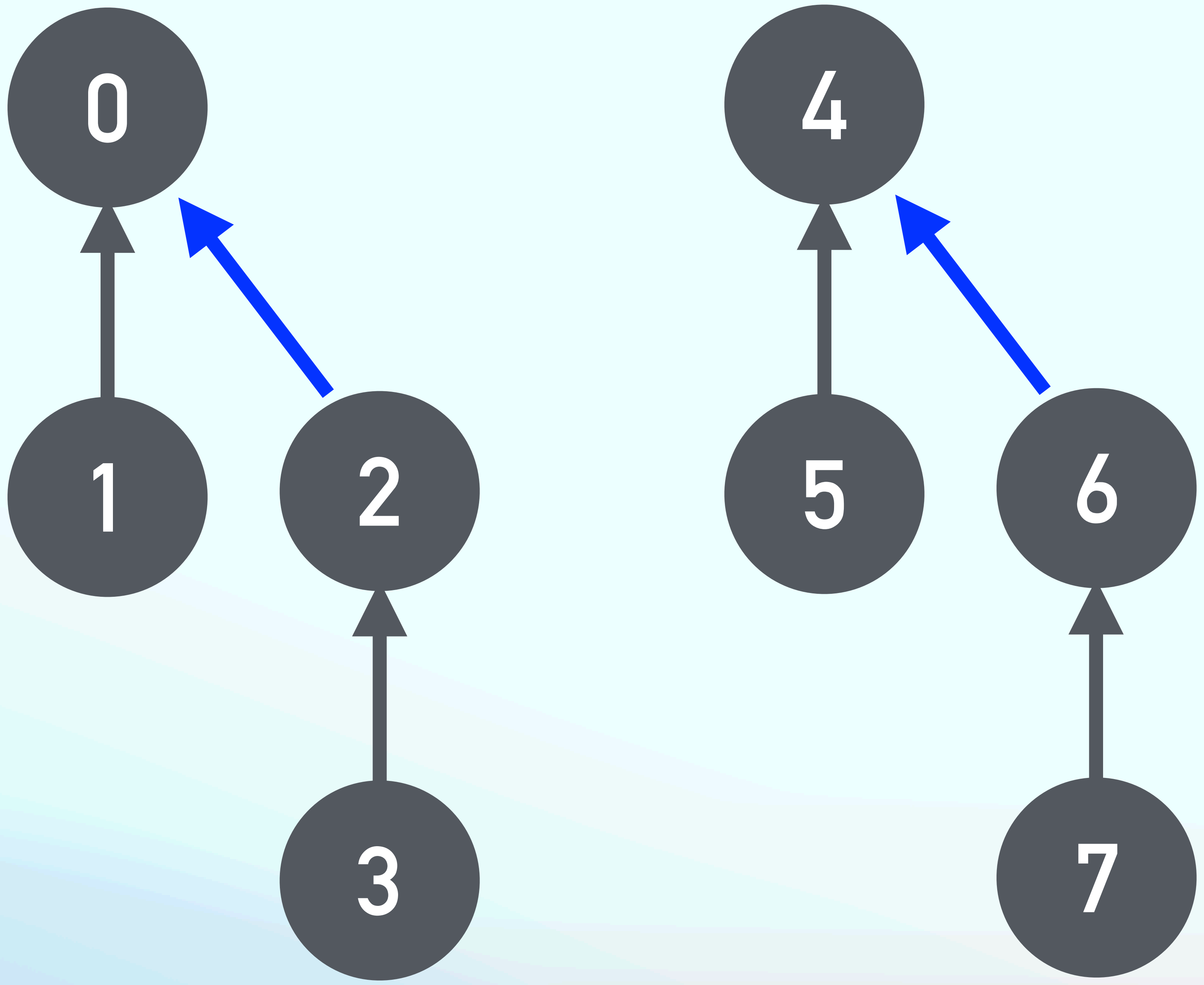
What is the worst case?



Disjoint set

Time complexity — Union-by-rank

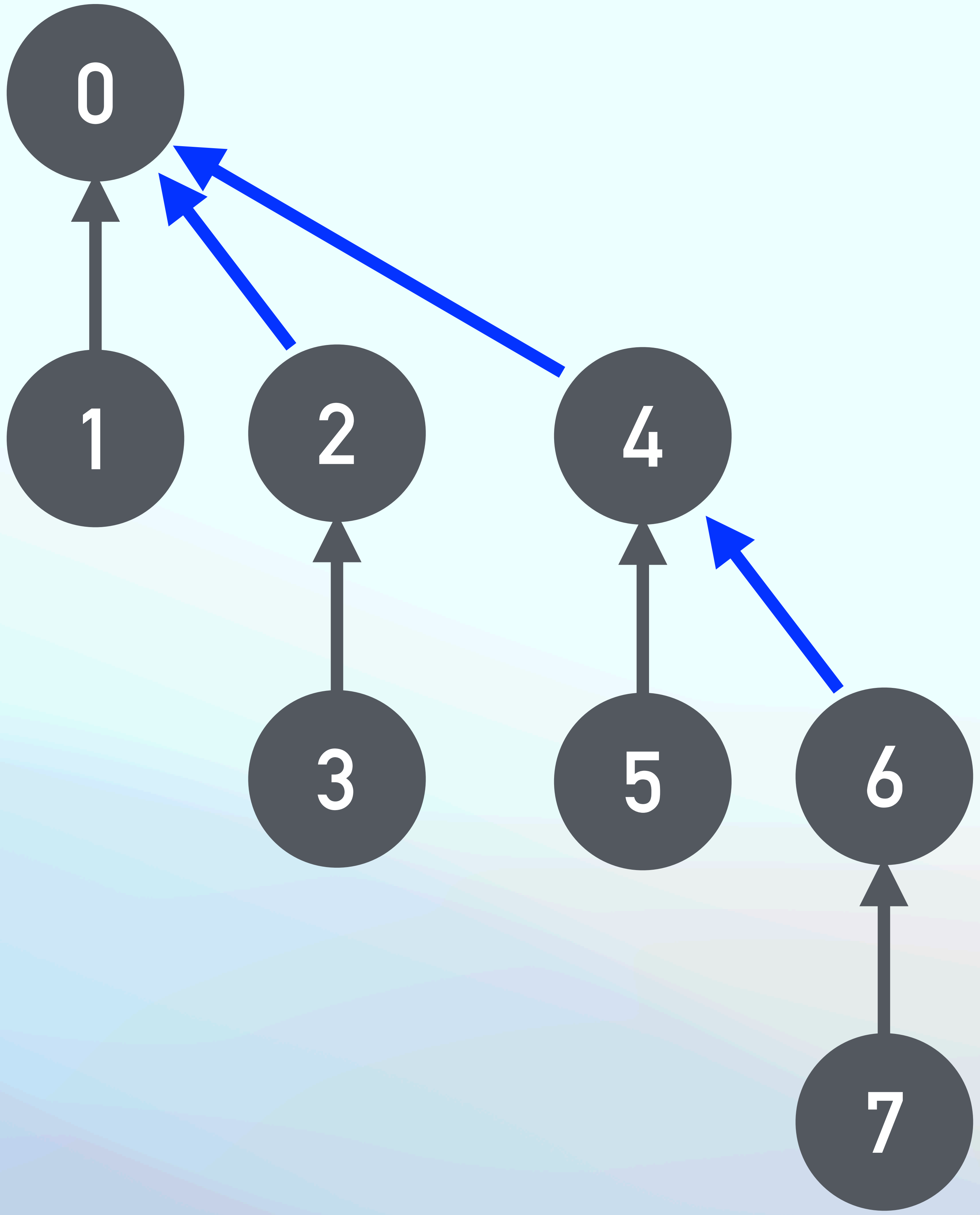
What is the worst case?



Disjoint set

Time complexity — Union-by-rank

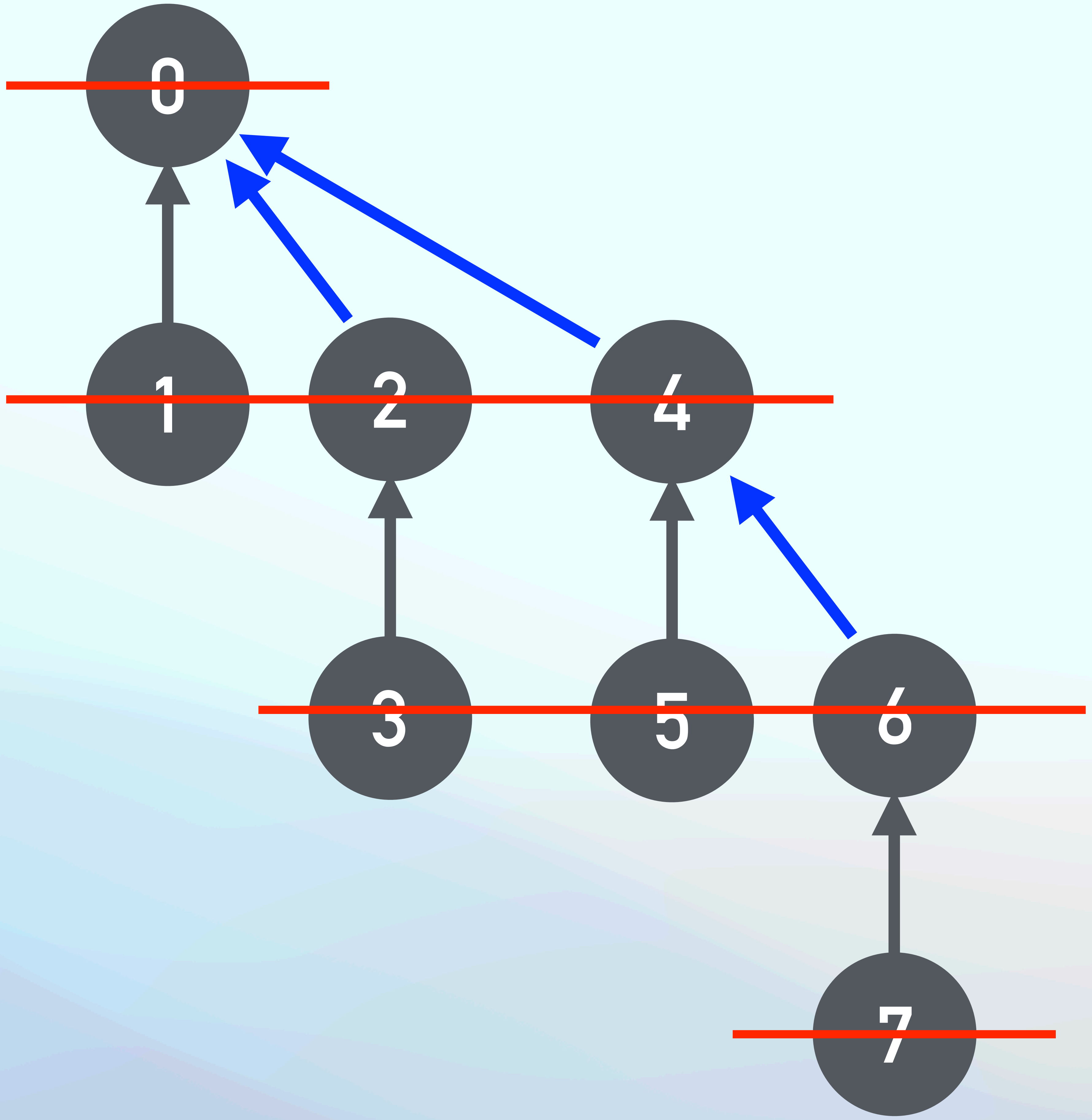
What is the worst case?



Disjoint set

Time complexity — Union-by-rank

What is the worst case?



1

1 1

1 2 1

~~1 3 3 1~~

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

Disjoint set

Time complexity — Union-by-rank — Worst case

Average depth of nodes: $\dfrac{\displaystyle\sum_{k=0}^{h} k \binom{h}{k}}{\displaystyle\sum_{k=0}^{h} \binom{h}{k}} = \dfrac{h2^{h-1}}{2^h} = \dfrac{h}{2}$

Operations are $O(\log n)$

Disjoint set

Time complexity — Union-by-rank + **path compression**

1. Compress EVERY node on the path to the root.

2. Time complexity is $O\left(\alpha(n)\right)$, where $\alpha(n)$ is the **inverse Ackermann function.**

3. However, if we ask you to implement …

Why Union-by-Rank instead of Union-by-Height?

CS101-Quiz7-Review

Key Points

1. Disjoint set

2. **Minimum Spanning Tree**

Minimum Spanning Tree

Cut property — in short

1. A vertex is not connected.

2. MST must contain the edge with the smallest weight.

Minimum Spanning Tree

Cycle property — in short

1. Any cycle in the graph.

2. The edge with the largest weight is not in MST.

Minimum Spanning Tree

Prim's

1. Add the "nearest" vertex.

2. Prove by cut property.

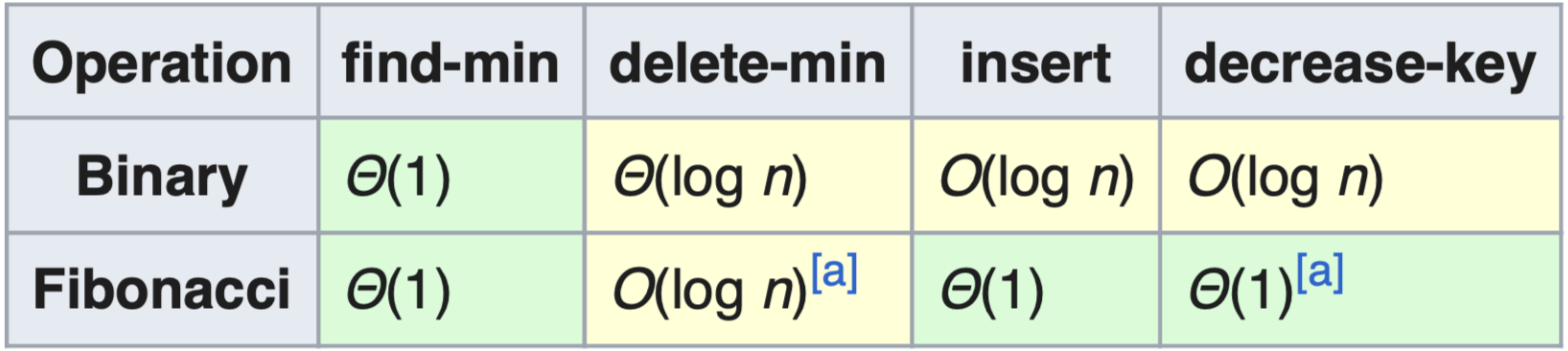
Minimum Spanning Tree

Prim's — Time complexity

Adjacency Matrix	$O(V^2)$
Adjacency List + Binary Heap	$O\left((V+E)\log V\right)$
Adjacency List + Fibonacci Heap	$O(E + V\log V)$

Minimum Spanning Tree

Prim's — Time complexity



Operation	find-min	delete-min	insert	decrease-key
Binary	$\Theta(1)$	$\Theta(\log n)$	$O(\log n)$	$O(\log n)$
Fibonacci	$\Theta(1)$	$O(\log n)$[a]	$\Theta(1)$	$\Theta(1)$[a]

Adjacency List + Fibonacci Heap	$O(E + V \log V)$

Minimum Spanning Tree

Kruskal's

1. Add the shortest edge.

2. Prove by cycle property.

Minimum Spanning Tree

Kruskal's — Time complexity

$$\underset{\text{sort}}{O(E\log E)} + \underset{\text{Disjoint set}}{O(E\alpha(V))}$$

What if the edges are already sorted?