Search Algorithm Idea

```
Initialize frontier list with the start node
Initialize explored list with empty list
While frontier is not empty:
    Get a node from frontier   ← How is a node selected?
    If the node is the goal node
        Return Success (solution found)
    If the node is NOT in explored
        Remove node from frontier and add to explored
        Get the neighbors of the node
        For each neighbor in neighbors
            Apply the heuristic to calculate the neighbor cost
            If the neighbor is NOT in frontier
                Add neighbor to frontier   ← How is a node added?
            Else
                Replace existing neighbor if this neighbor's cost is less
Return Failure (no solution found)
```

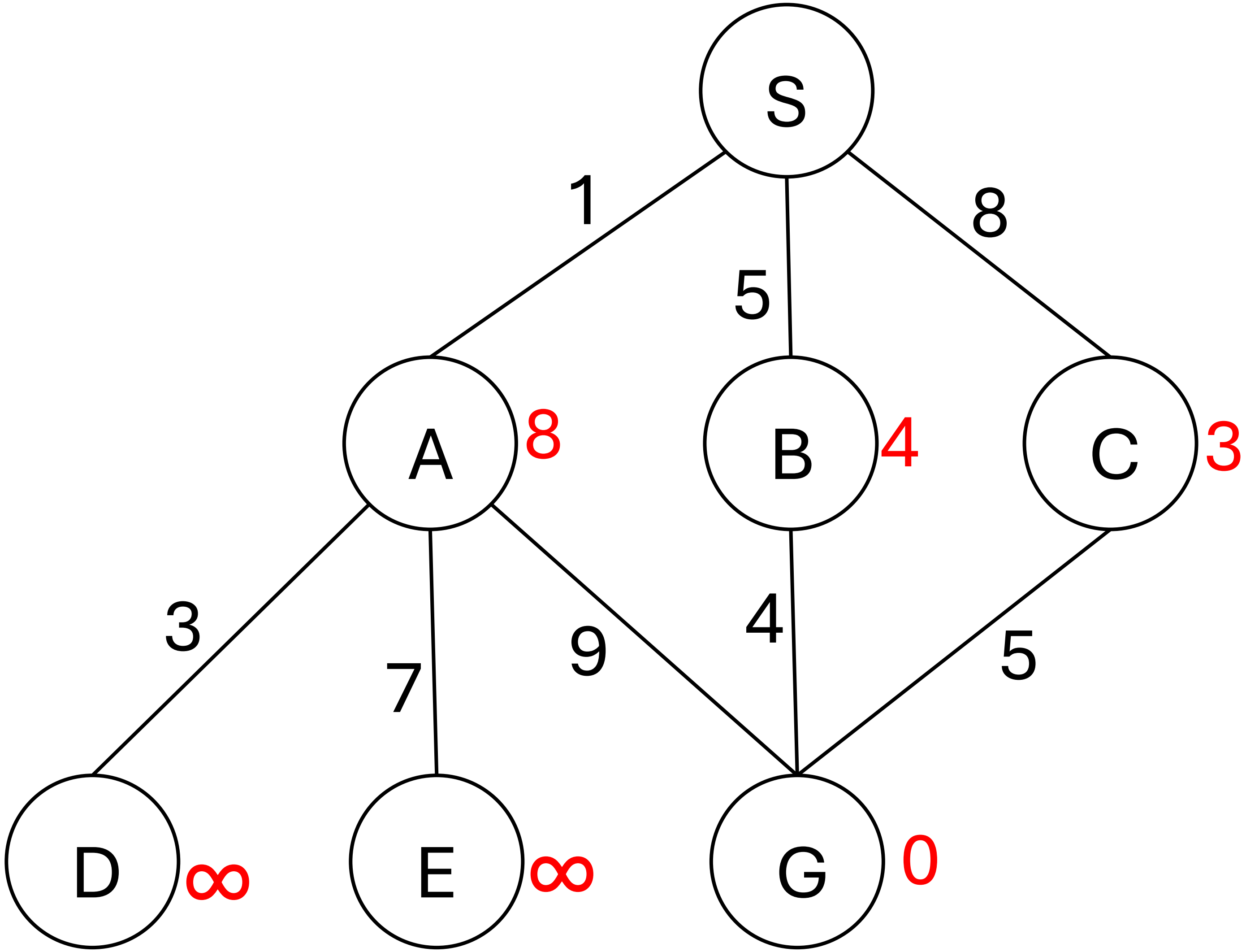
Search Algorithms

Uninformed Search

- Depth First Search (DFS)
    - Frontier list is represented by a **stack**: last in, first out
- Breadth First Search (BFS)
    - Frontier list is represented by a **queue:** first in, first out

Informed Search

- Frontier is represented by a **priority queue**: best node at the front
- Uniform Cost Search
    - Heuristic g(n): actual cost of the path from start to current node
- Best First Search (Greedy Search)
    - Heuristic h(n): best estimate cost of the path from current to goal
- A* Search
    - Heuristic: f(n) = g(n) + h(n)



Use this example to trace the

- **uninformed** and

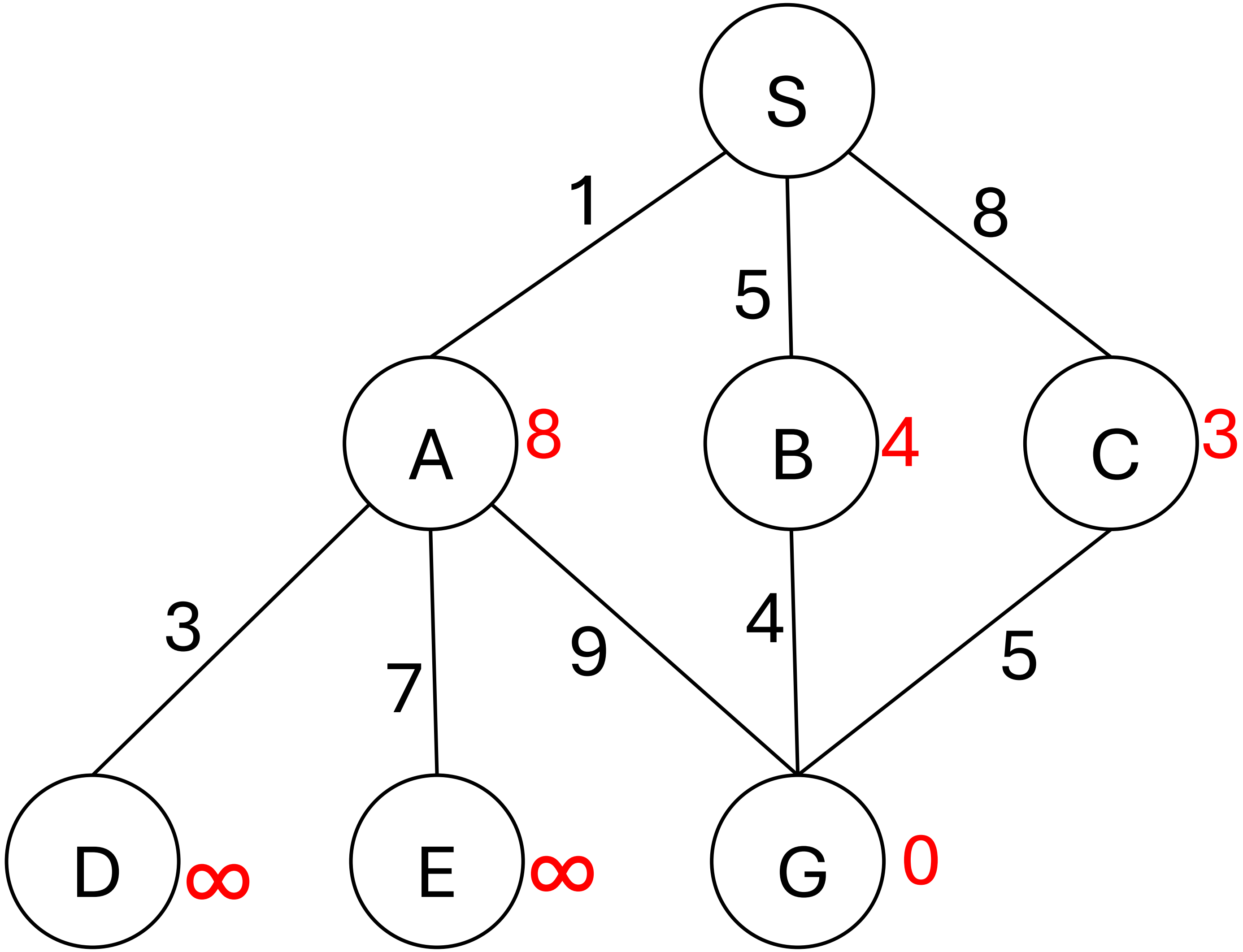
- **informed search** algorithms by

showing the content of the

- **Frontier** list
- **Explored** nodes list

Depth-First Search

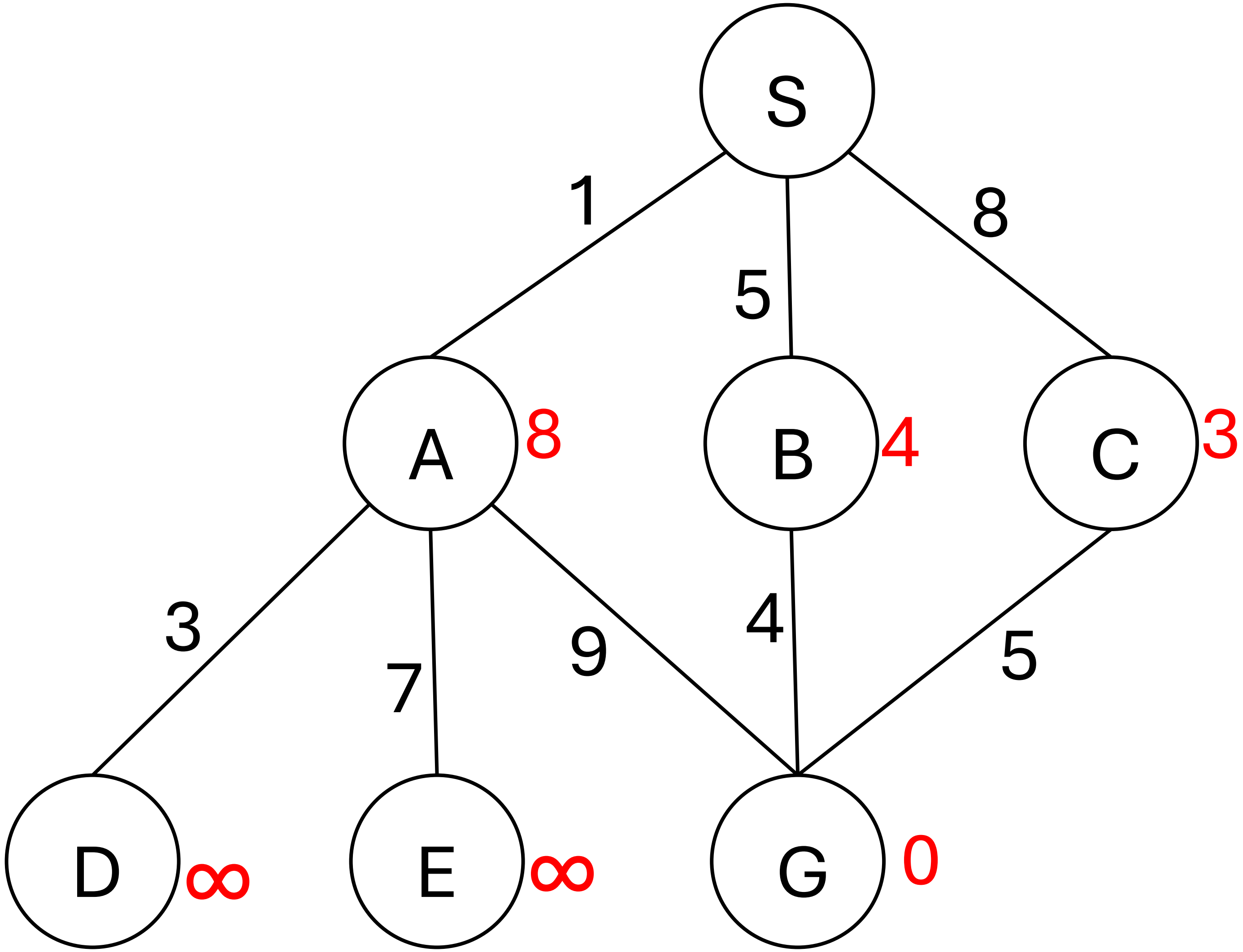
Stack frontier (last-in, first-out)



Explored	Frontier

Breadth-First Search

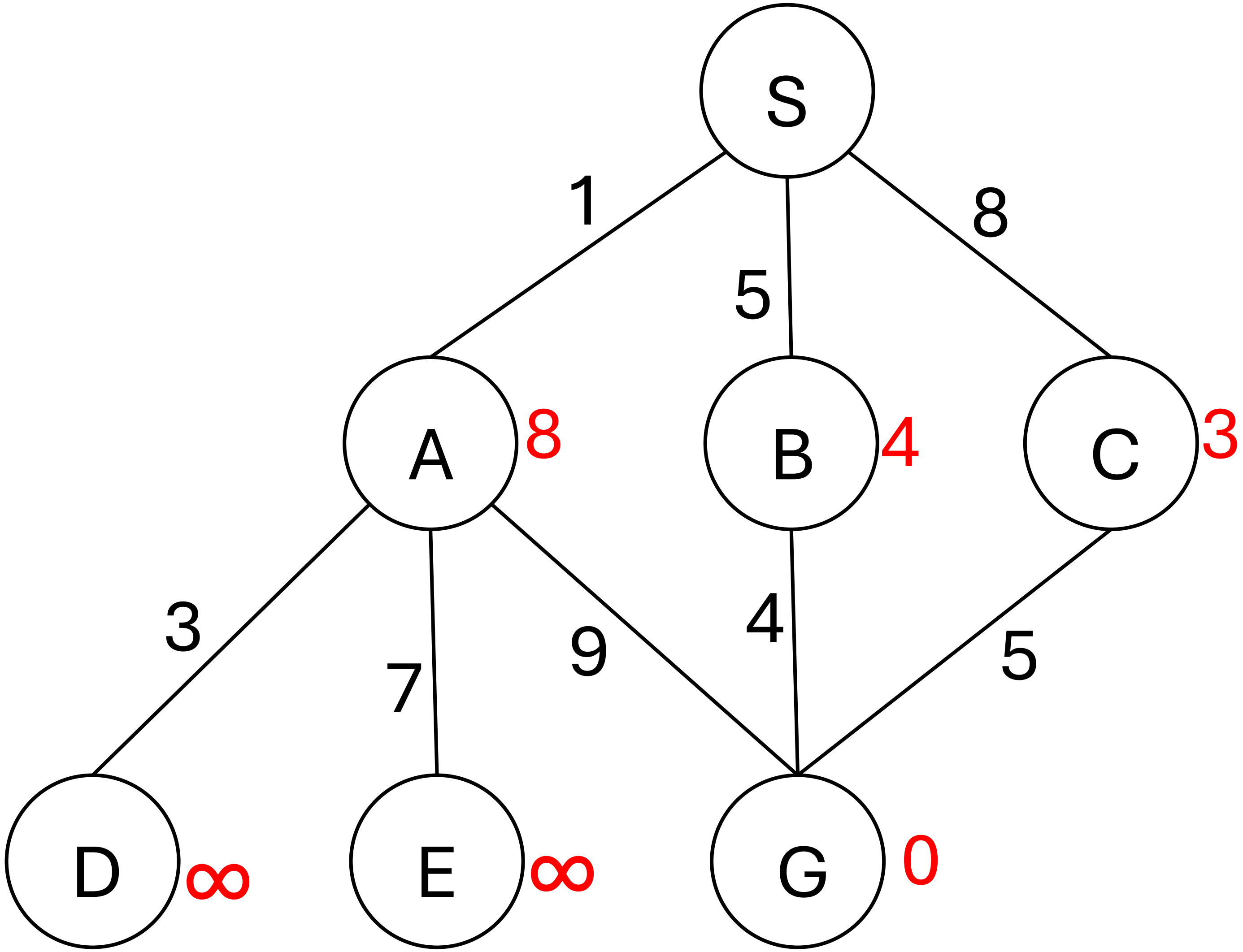
Queue frontier (first-in, first-out)



Explored | Frontier

Uniformed Cost (Best-First) Search

Priority queue of actual, g(n) costs

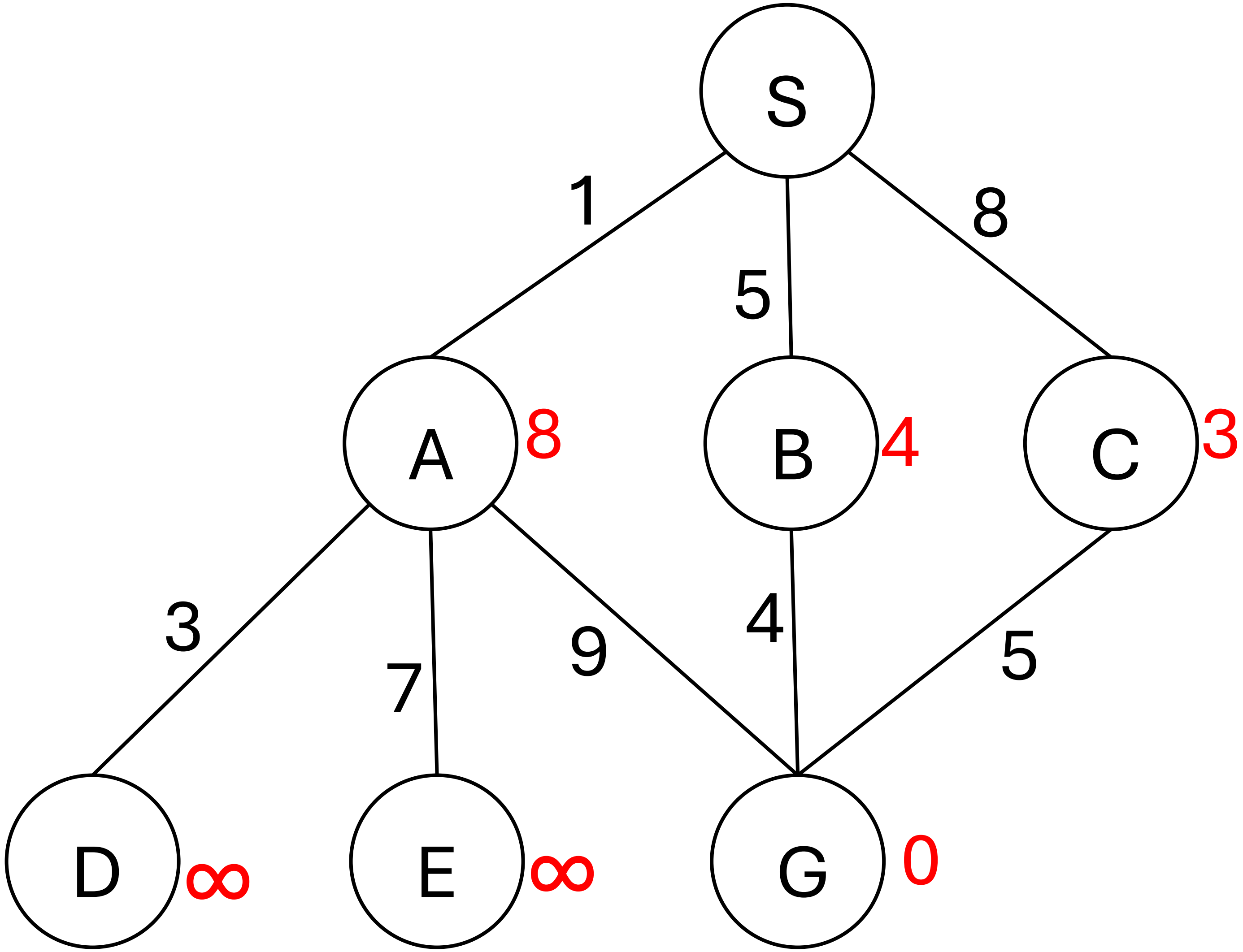


Explored

Frontier

Greedy Search

Priority queue of estimated h(n) costs



Explored

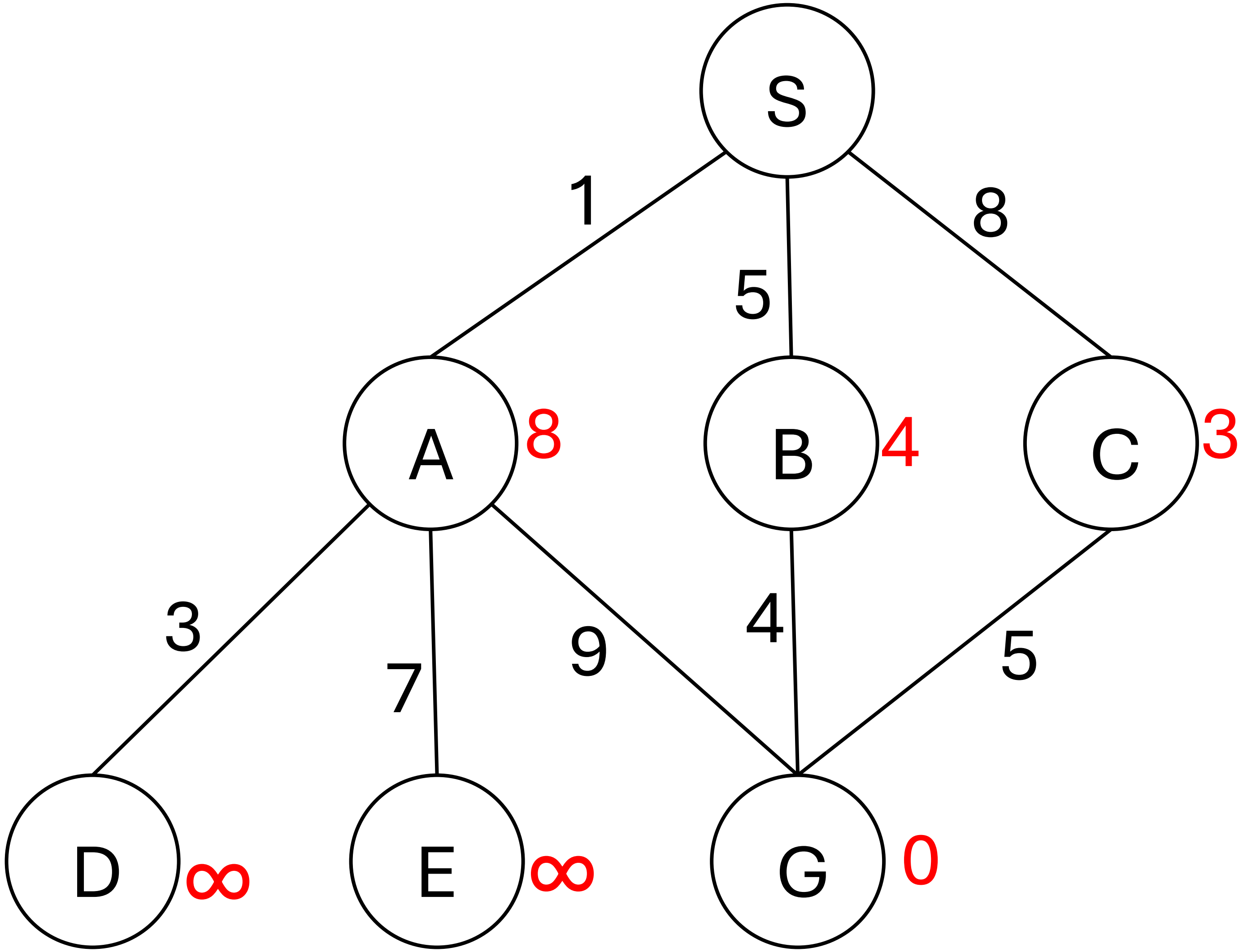
Frontier

A* Search

Priority queue of f(n) = g(n) + h(n) costs

Explored

Frontier


S
A
B
C
D
E
G
1
5
8
3
7
9
4
5
8
4
3
∞
∞
0