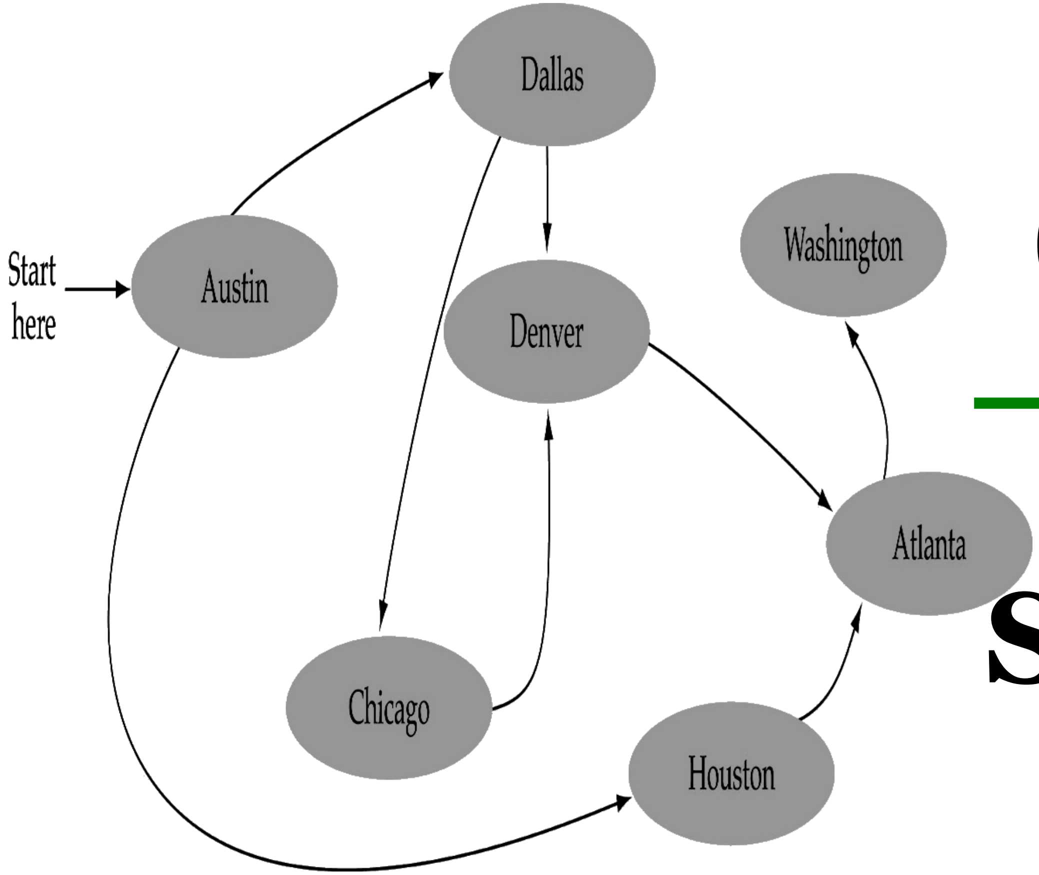


Graphs

Shortest Path

Dijkstra's Shortest Path Algorithm

- Dijkstra's algorithm finds the shortest path from one vertex v_o to each other vertex in a digraph.
- When it has finished, the length of the shortest distance from v_o to v is stored in the vertex v, and the shortest path from v_o to v is recorded in the back pointers of v and the other vertices along that path.
- The algorithm uses a priority queue, initializing it with all the vertices and then dequeueing one vertex on each iteration.

Dijkstra's Shortest Path Algorithm

- (Precondition: G = (V,w) is a weighted graph with initial vertex vo.)
- (Postcondition: Each vertex v in V stores the shortest distance from vo to v and a back reference to the preceding vertex along that shortest path.)

1. Initialize the distance field to 0 for v_o and to ∞ for each of the other vertices.
2. Enqueue all the vertices into a priority queue Q with highest priority being the lowest distance field value.
3. Repeat steps 4 to 10 until Q is empty.

Dijkstra’s Shortest Path Algorithm

4. (Invariant: The distance and back reference fields of every vertex that is not in Q are correct.)

5. Dequeue the highest priority vertex into x.

6. Do steps 7 to 10 for each vertex y that is adjacent to x and in the priority queue.

7. Let s be the sum of the x’s distance field plus the weight of the edge from x to y.

8. If s is less than y’s distance field, do steps 9 10; otherwise go back to Step 3.

9. Assign s to y’s distance field.

10. Assign x to y’s back reference field.

Dijkstra's Shortest Path Algorithm

Pseudocode without using priority queue

```
function Dijkstra(Graph, source):
        create vertex set Q
        for each vertex v in Graph:
                dist[v] ← INFINITY
                prev[v] ← UNDEFINED
                add v to Q
        dist[source] ← 0
        while Q is not empty:
                u ← vertex in Q with min dist[u]
                remove u from Q
                for each neighbor v of u:
                        alt ← dist[u] + length(u, v)
                        if alt < dist[v]:
                                dist[v] ← alt
                                prev[v] ← u
        return dist[], prev[]
```

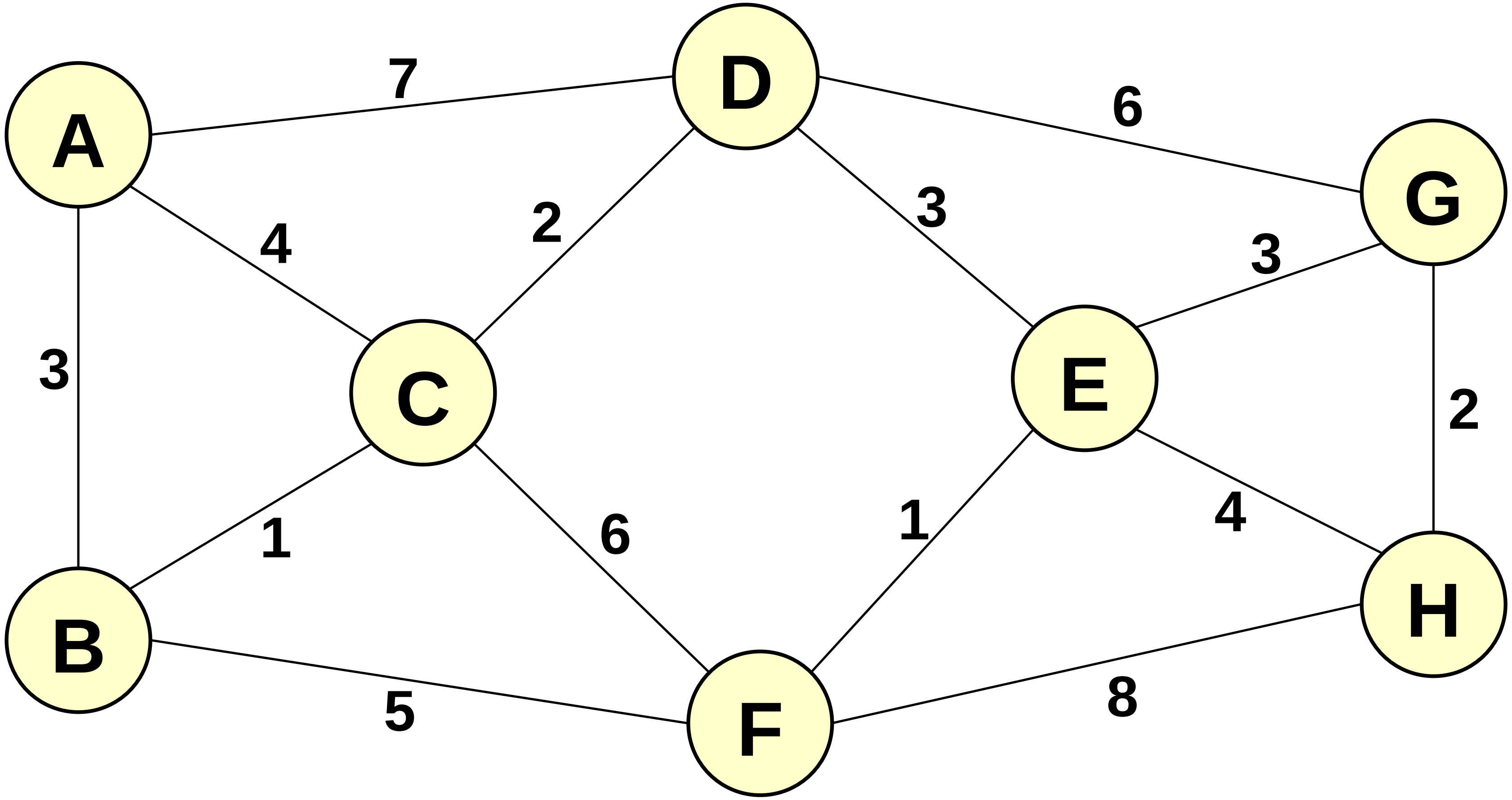
Dijkstra's Shortest Path Algorithm

- If we are only interested in a shortest path between vertices *source* and *target*, we can terminate the search after line 10 if u = *target*.
- Now we can read the shortest path from ***source*** to ***target*** by reverse iteration:

```
S ← empty sequence
u ← target
if prev[u] is defined or u = source:      // only if the vertex is reachable
    while u is defined:        // Construct the shortest path with a stack S
        insert u at the beginning of S      // Push the vertex onto the stack
        u ← prev[u]             // Traverse from target to source
```

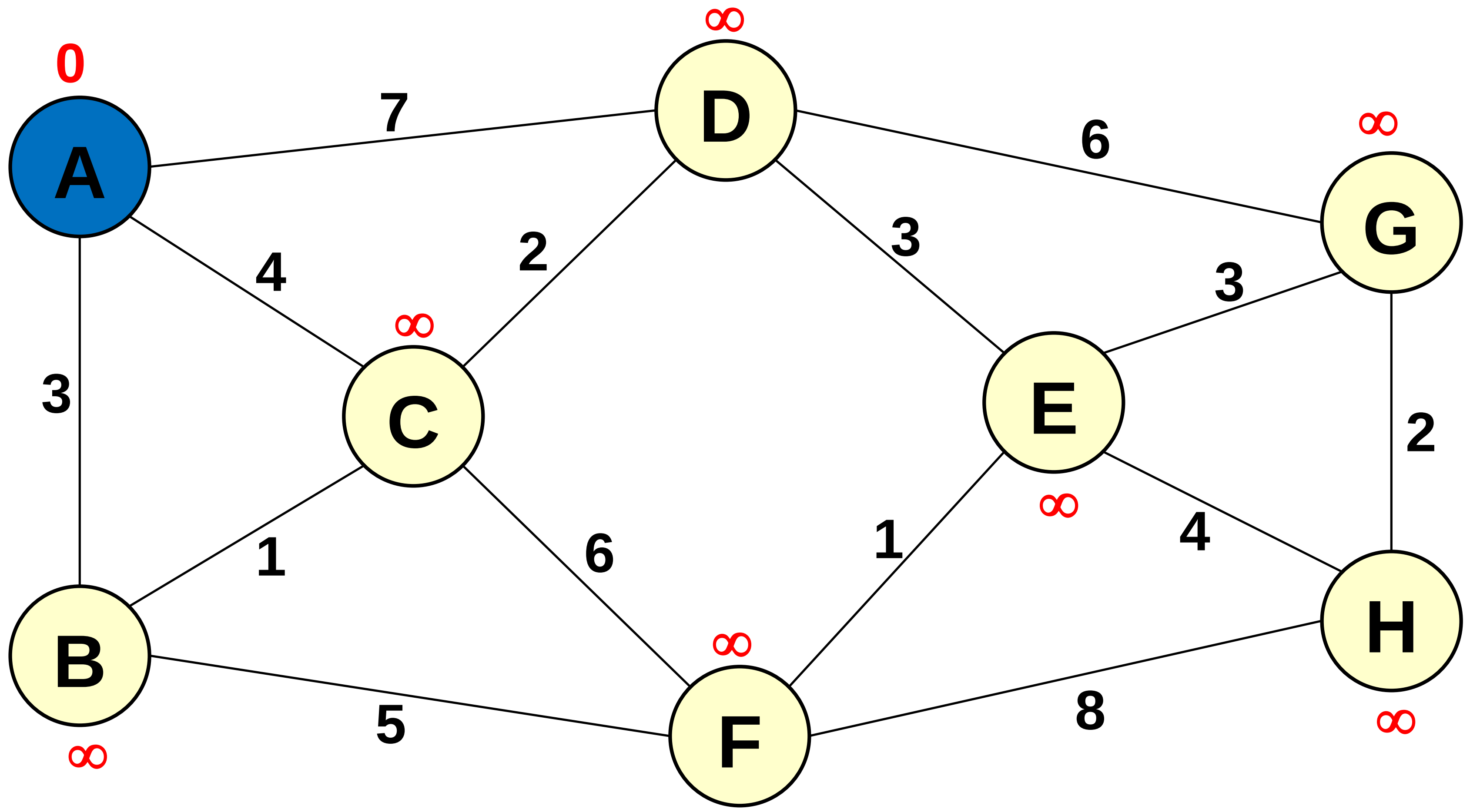
Dijkstra’s Shortest Path Algorithm

Example 1:



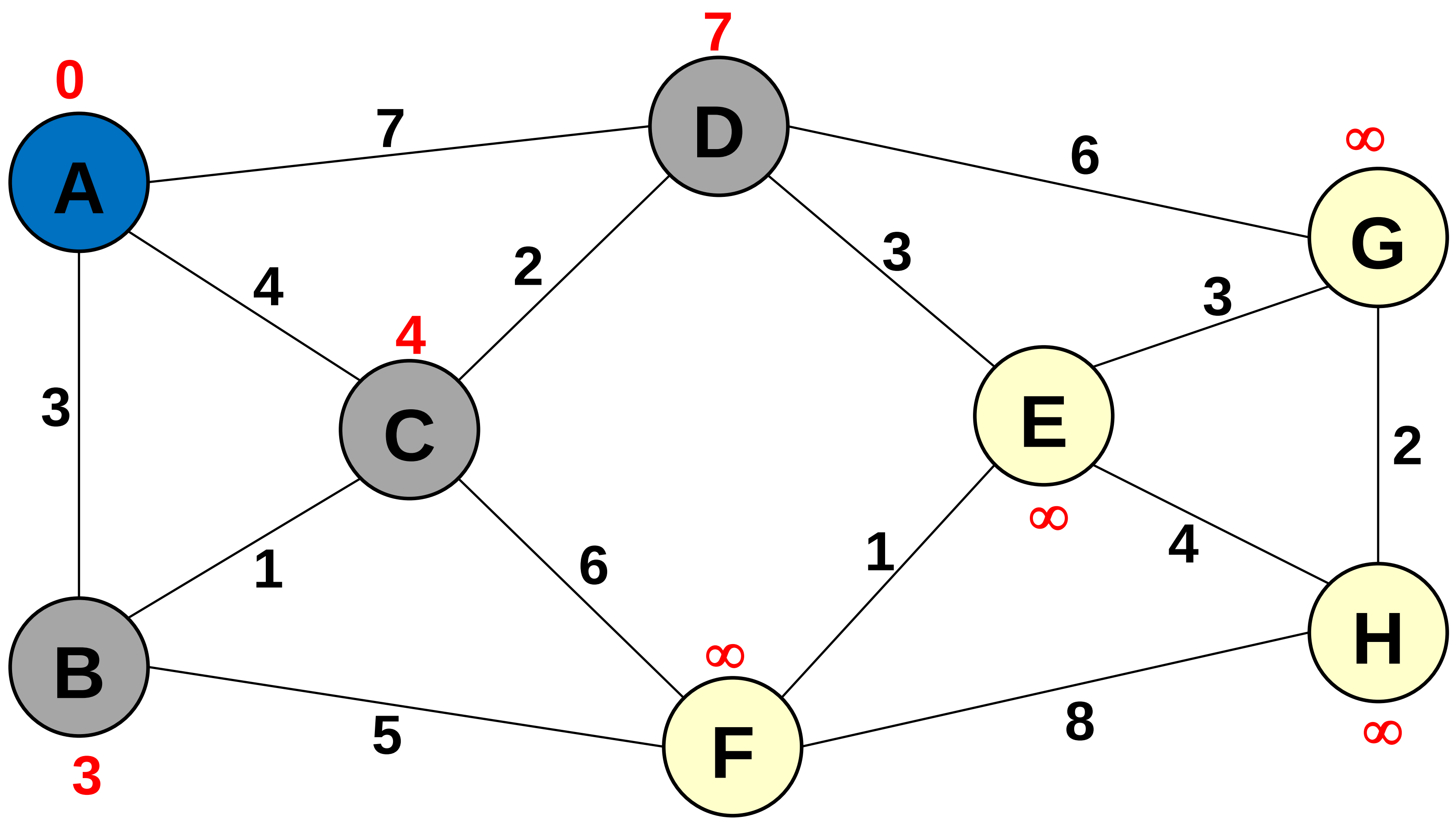
Dijkstra's Shortest Path Algorithm

Example 1:



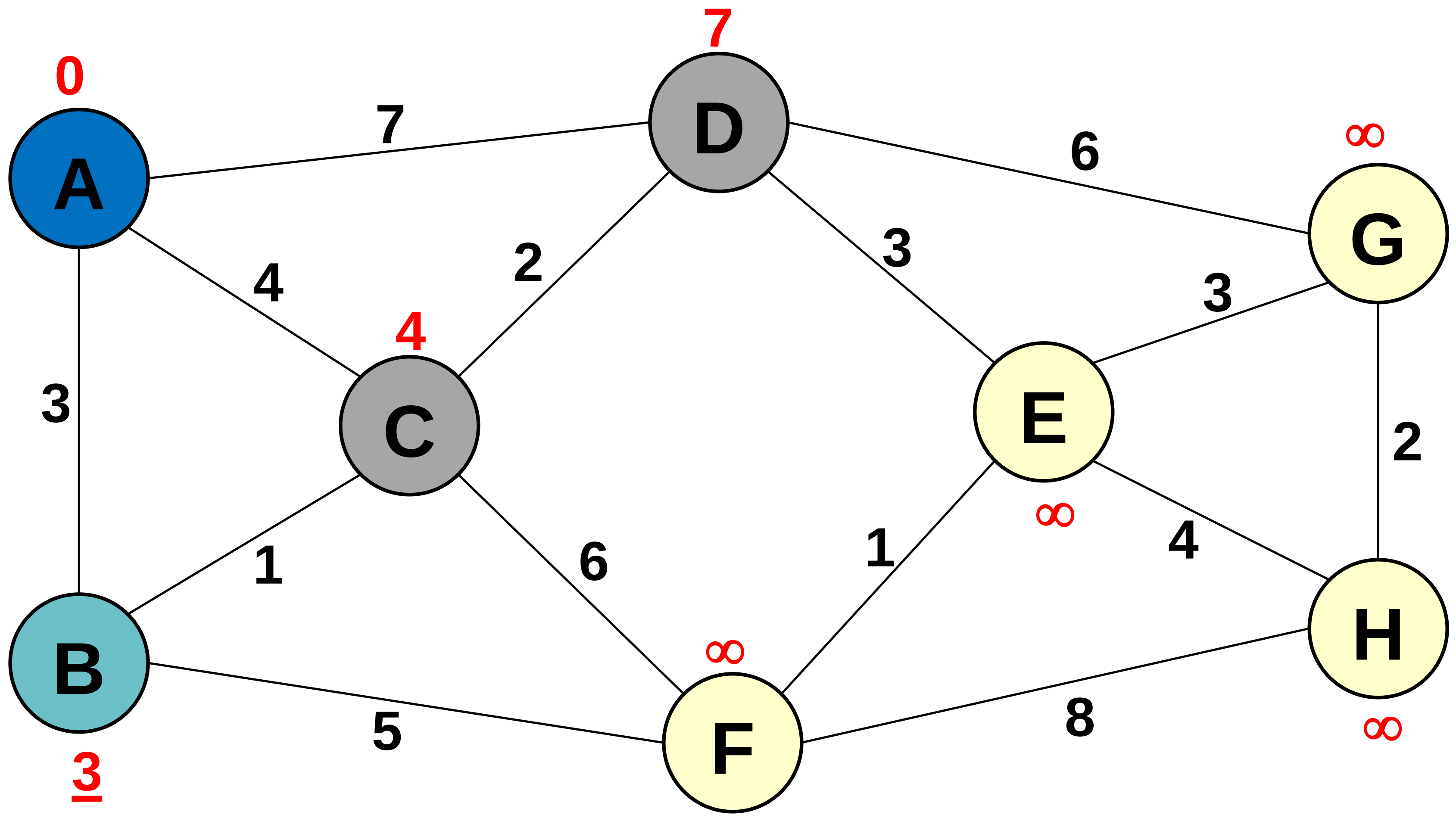
Dijkstra's Shortest Path Algorithm

Example 1:



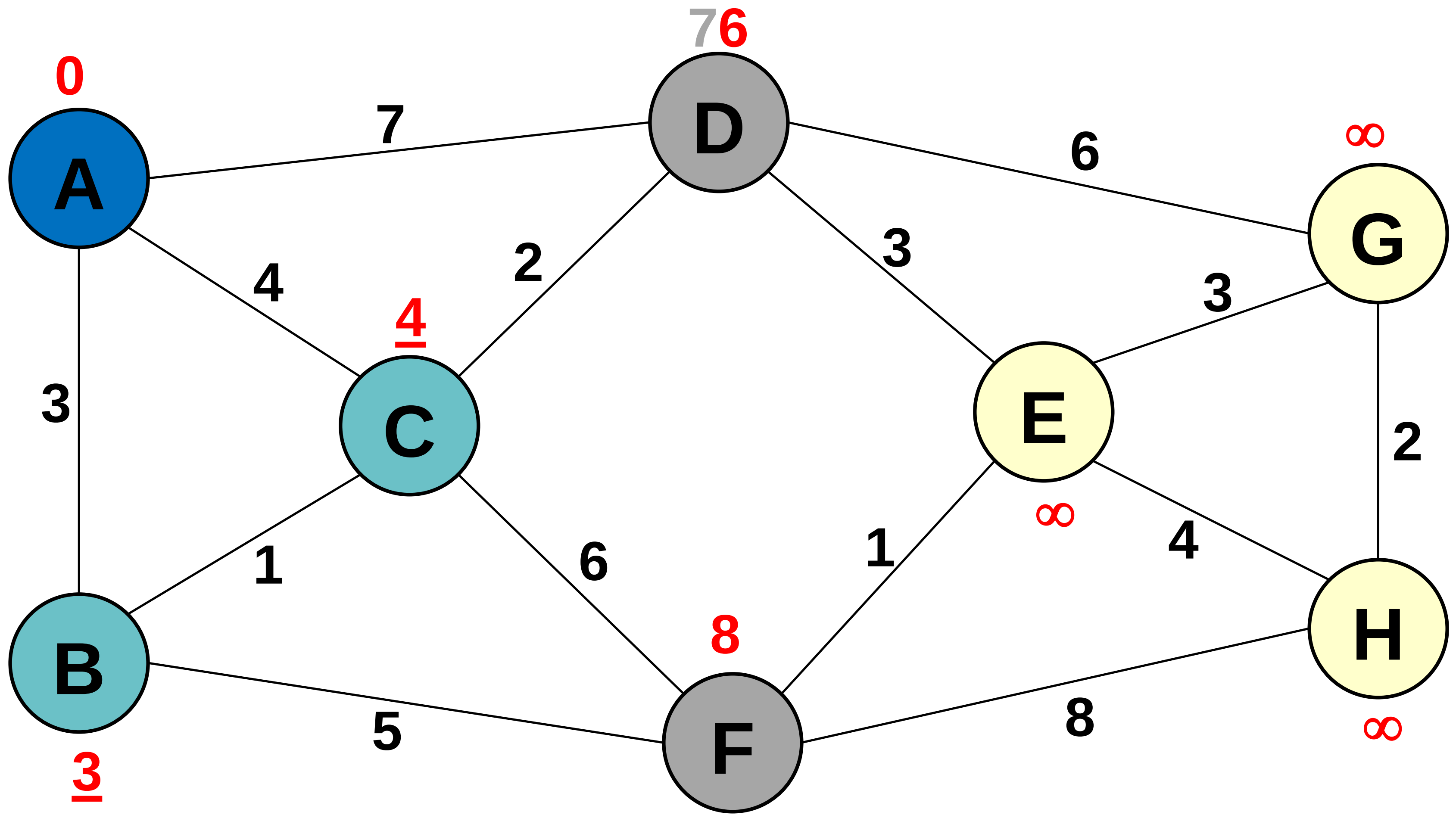
Dijkstra’s Shortest Path Algorithm

Example 1:



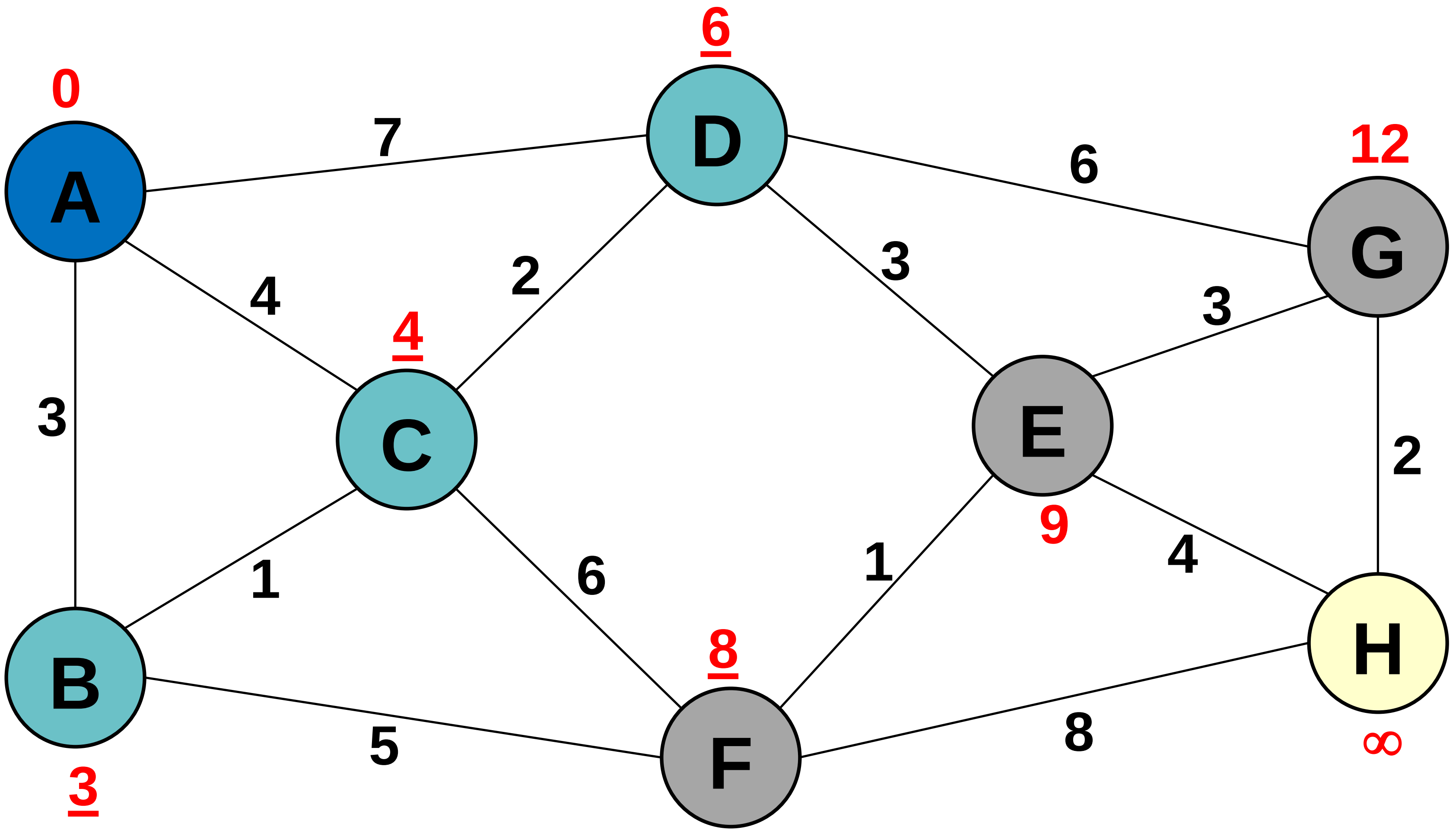
Dijkstra's Shortest Path Algorithm

Example 1:



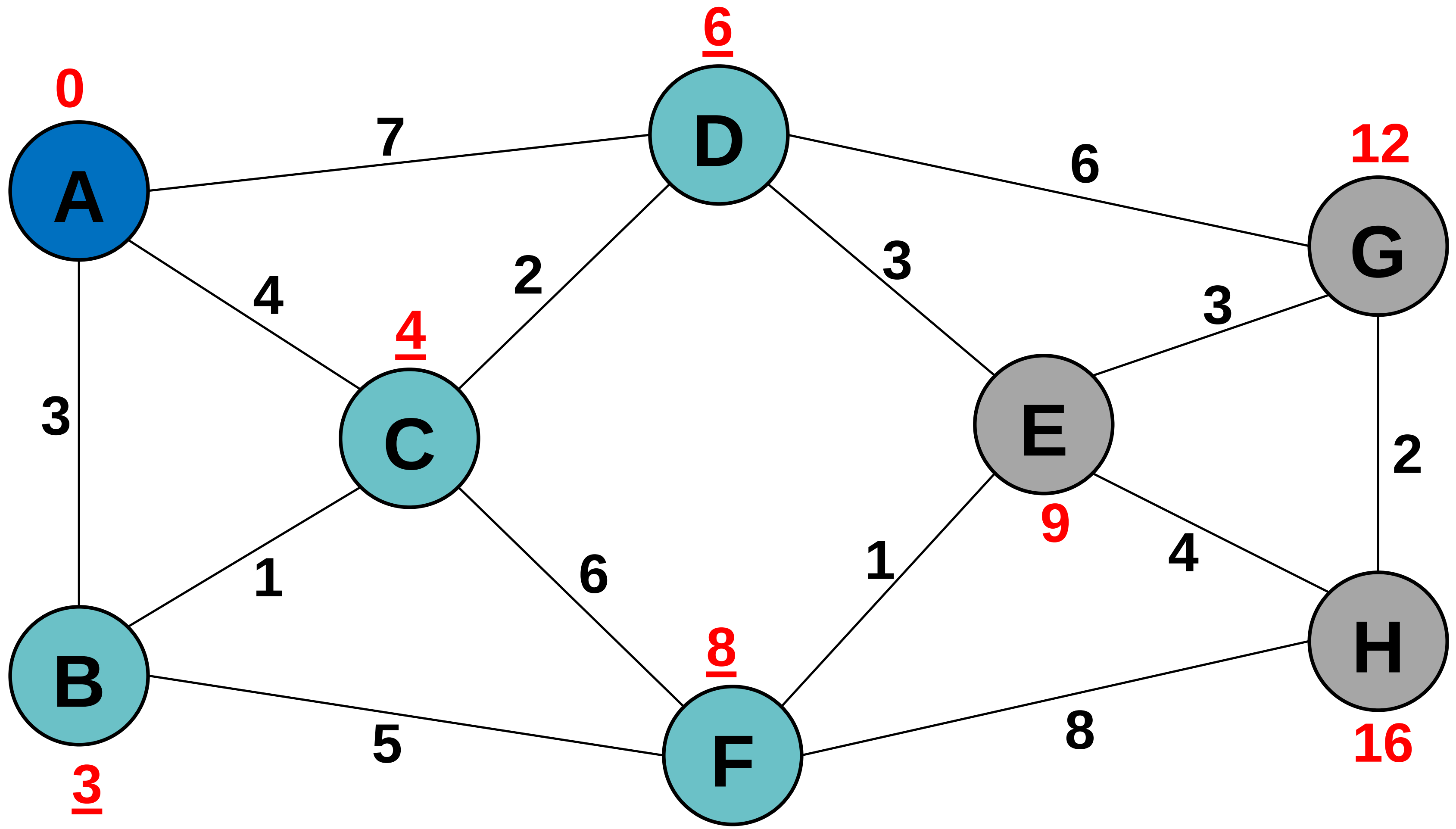
Dijkstra's Shortest Path Algorithm

Example 1:



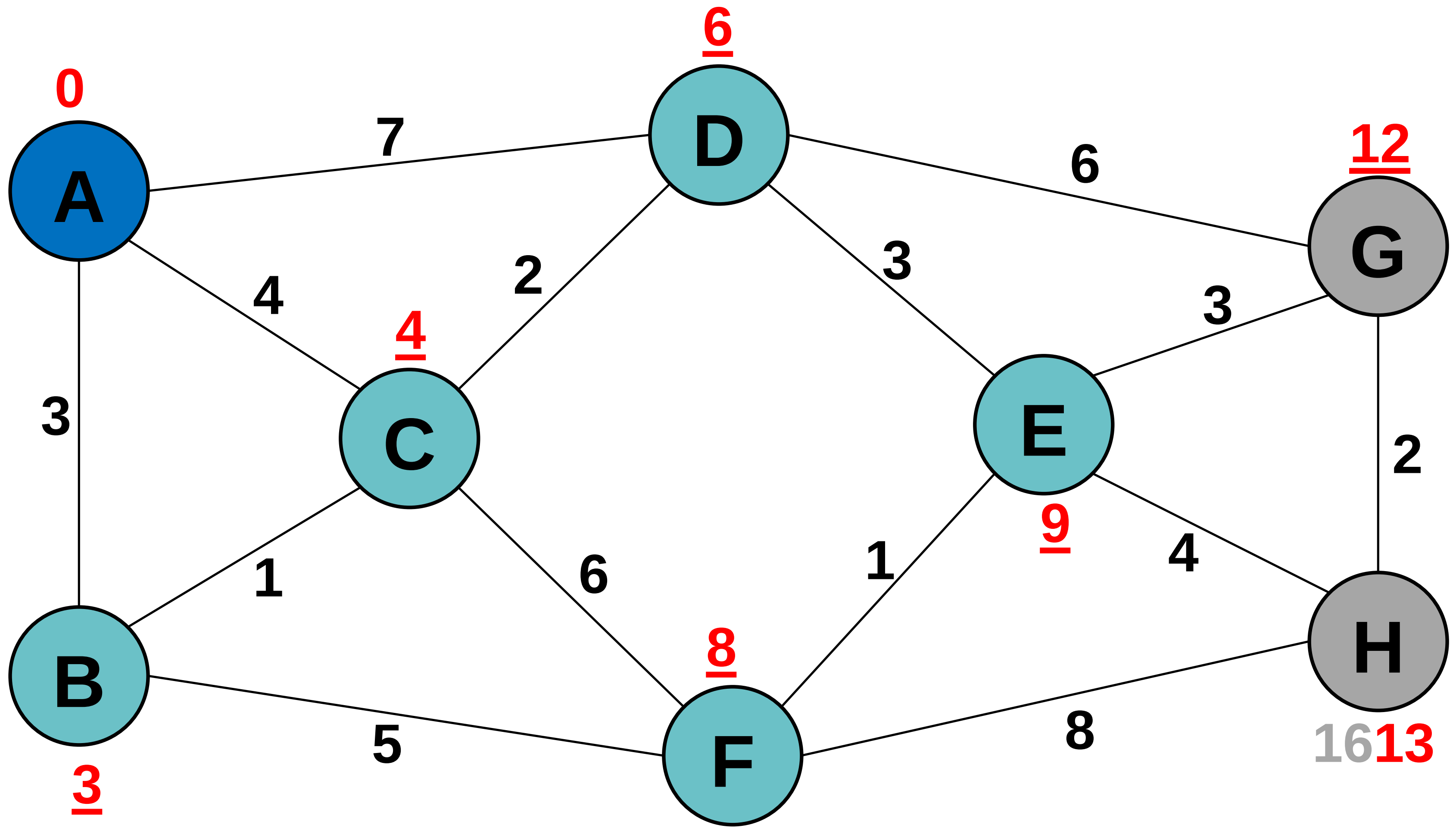
Dijkstra's Shortest Path Algorithm

Example 1:



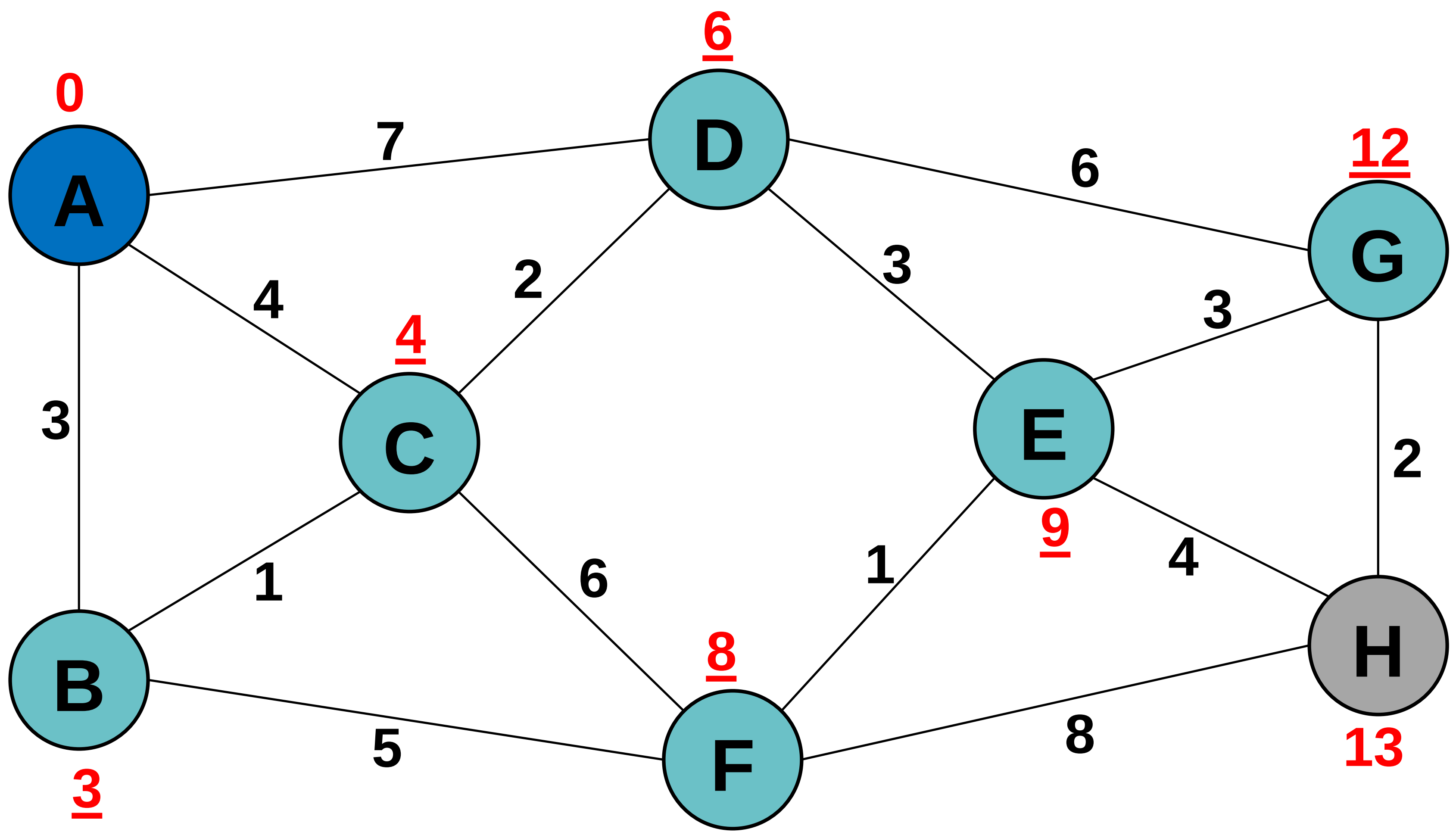
Dijkstra’s Shortest Path Algorithm

Example 1:



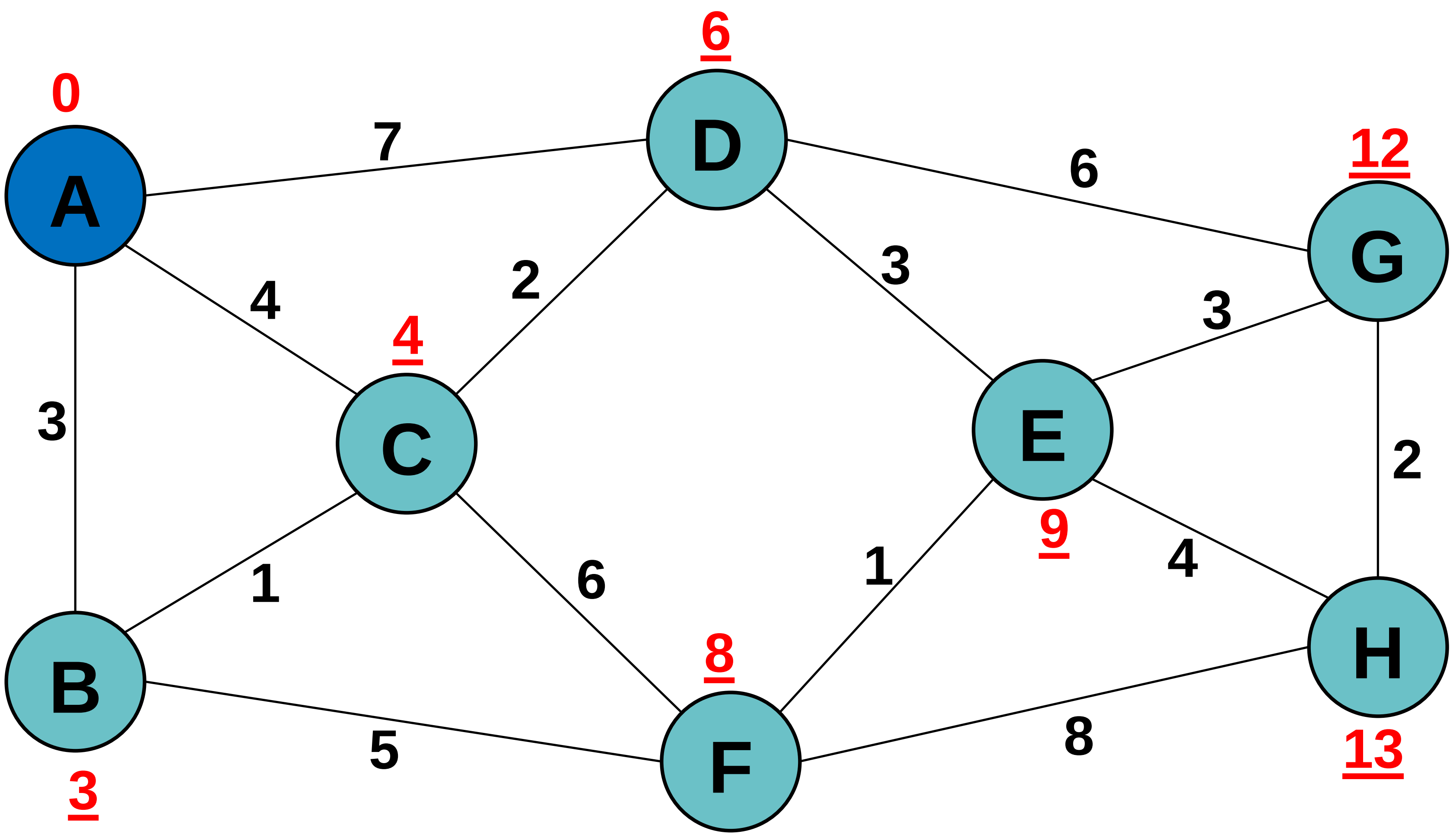
Dijkstra's Shortest Path Algorithm

Example 1:



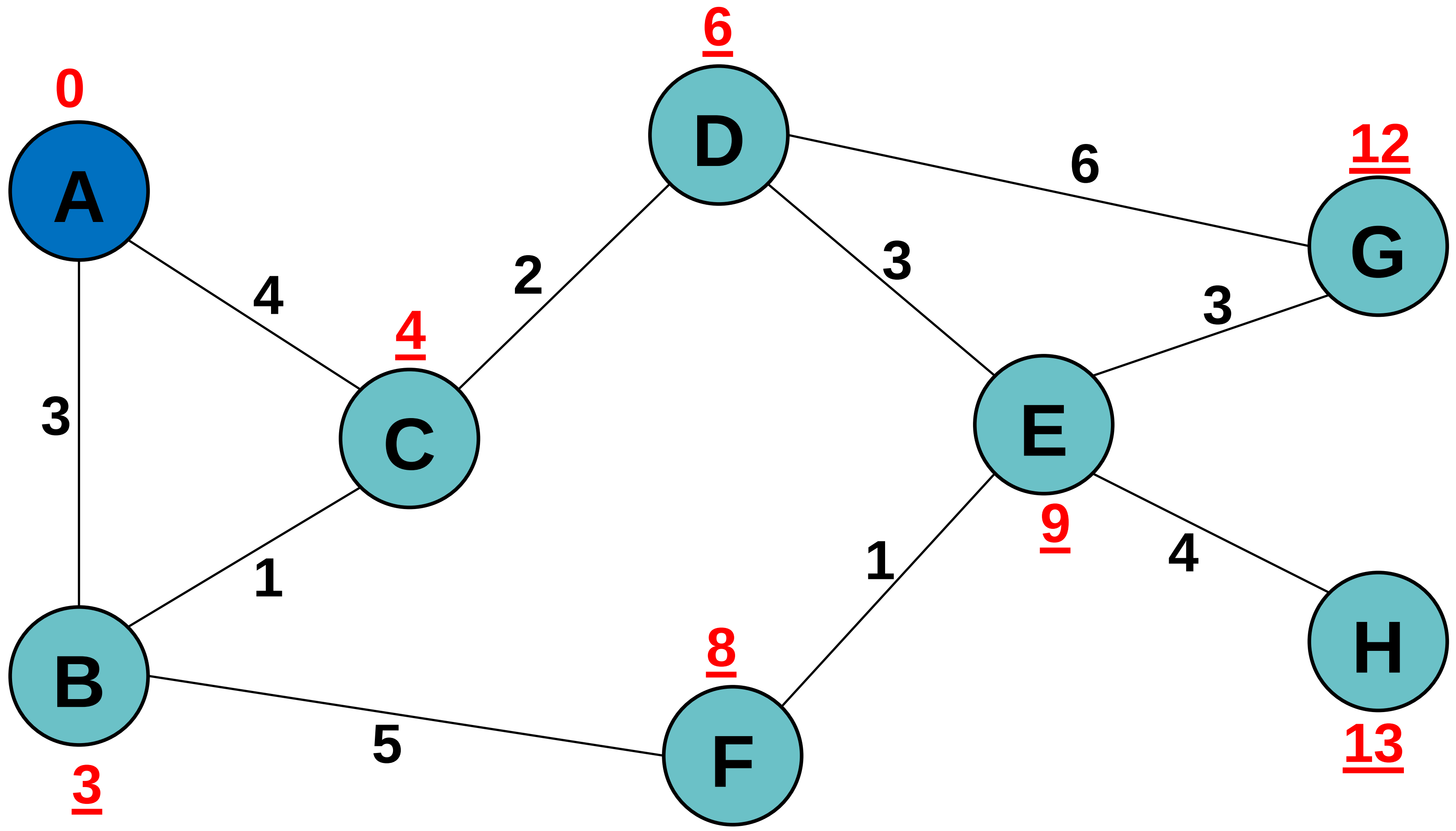
Dijkstra's Shortest Path Algorithm

Example 1:



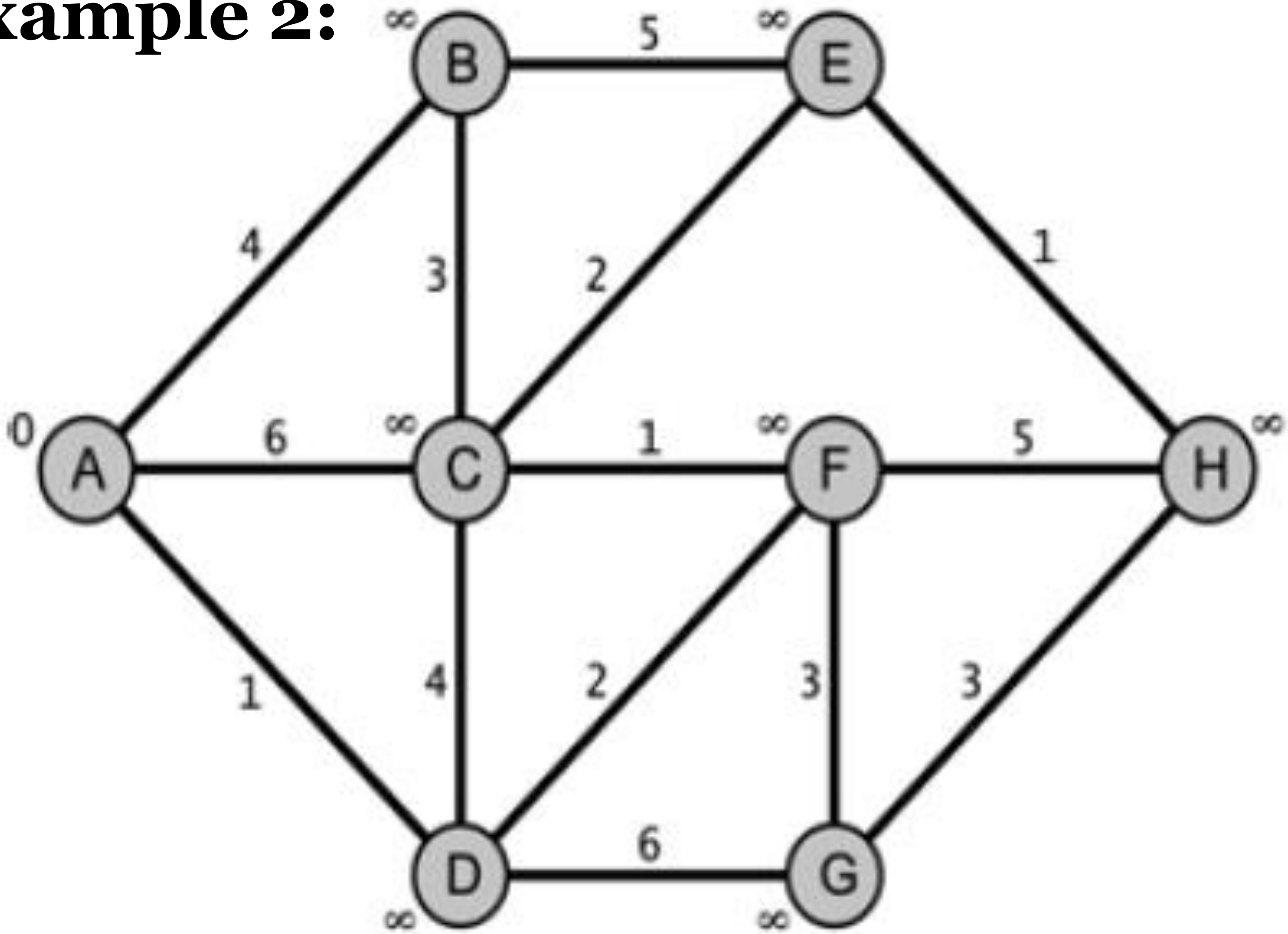
Dijkstra's Shortest Path Algorithm

Example 1: Only shortest paths are shown



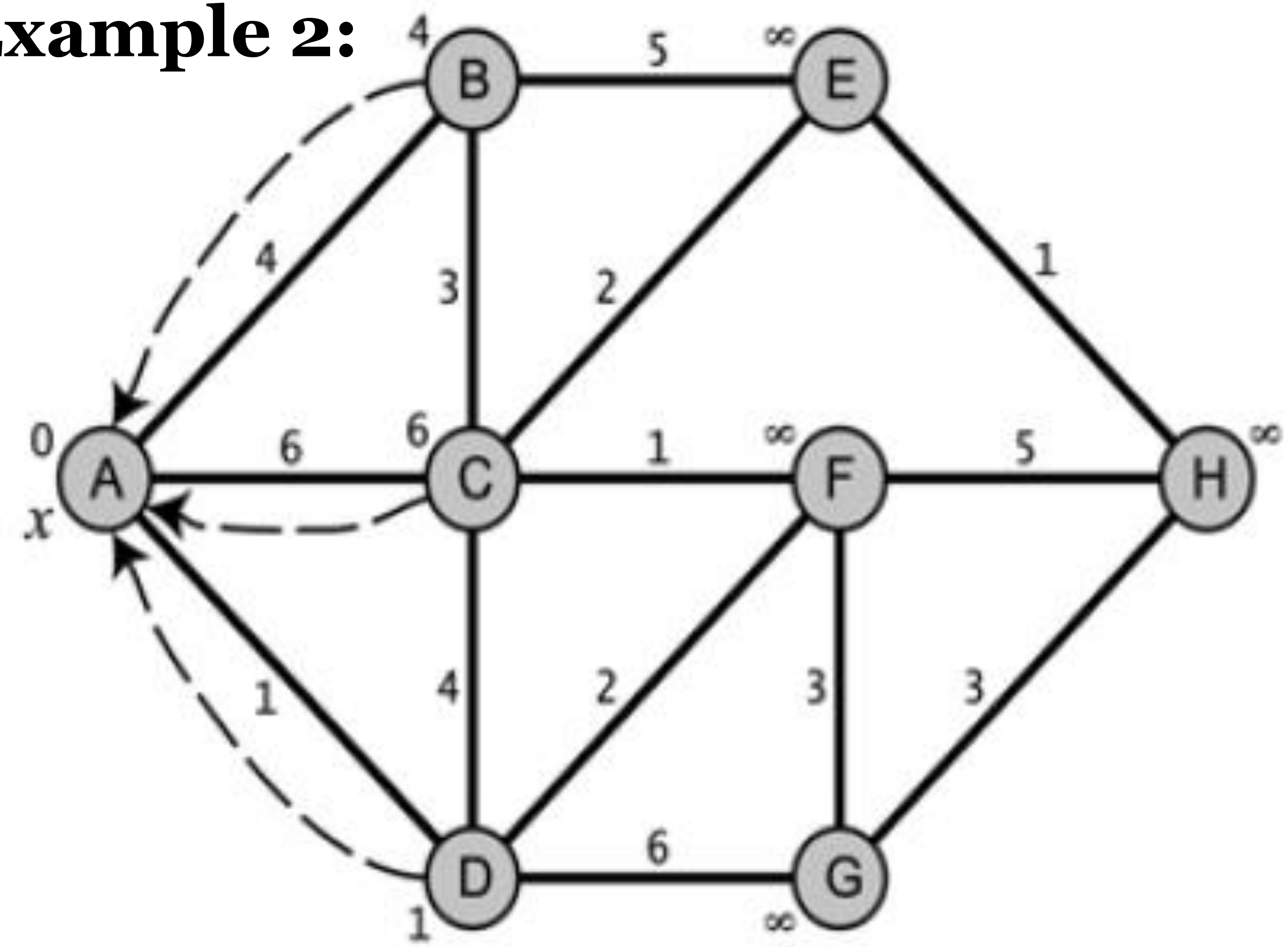
Dijkstra’s Shortest Path Algorithm

Example 2:



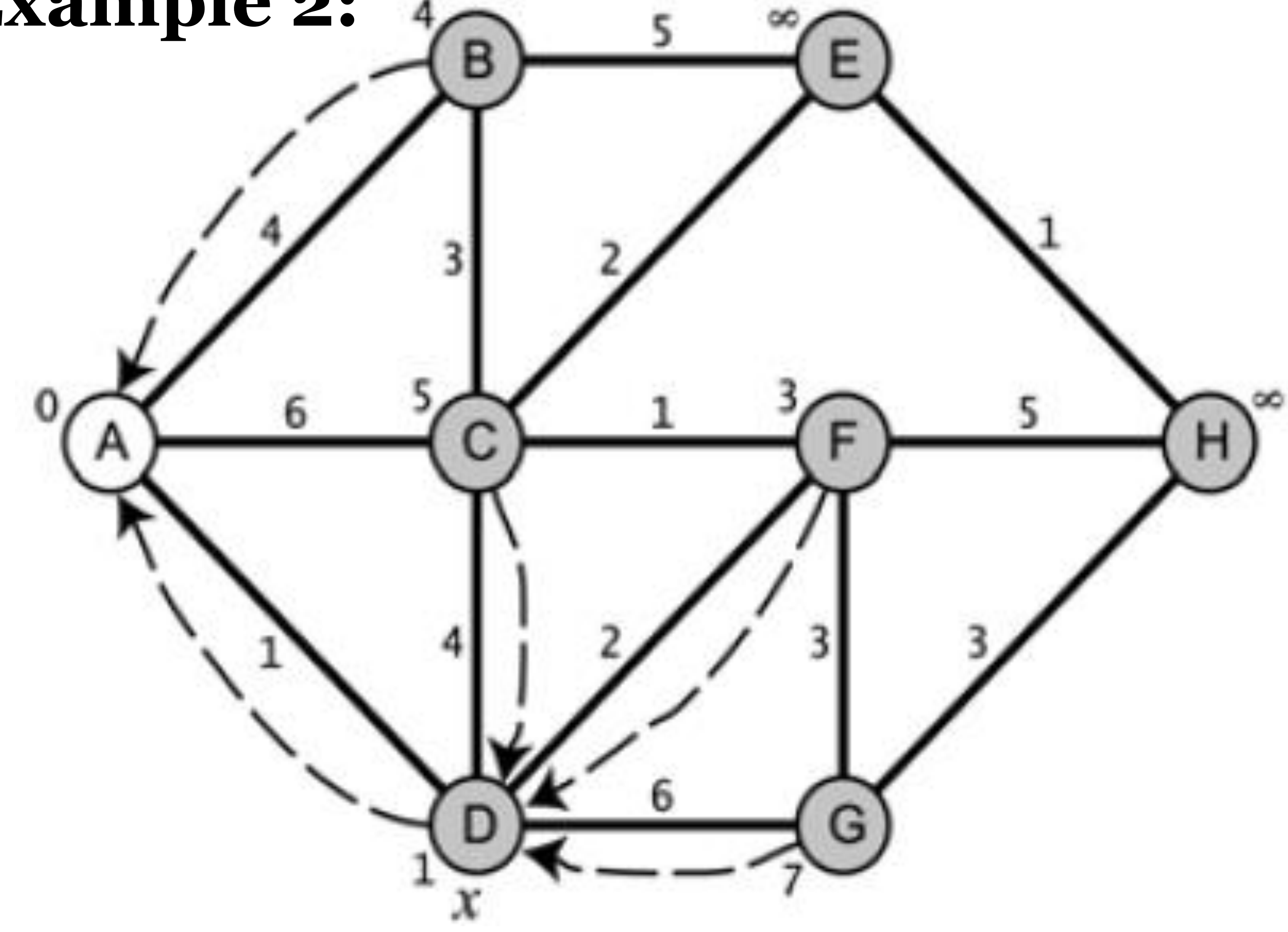
Dijkstra’s Shortest Path Algorithm

Example 2:



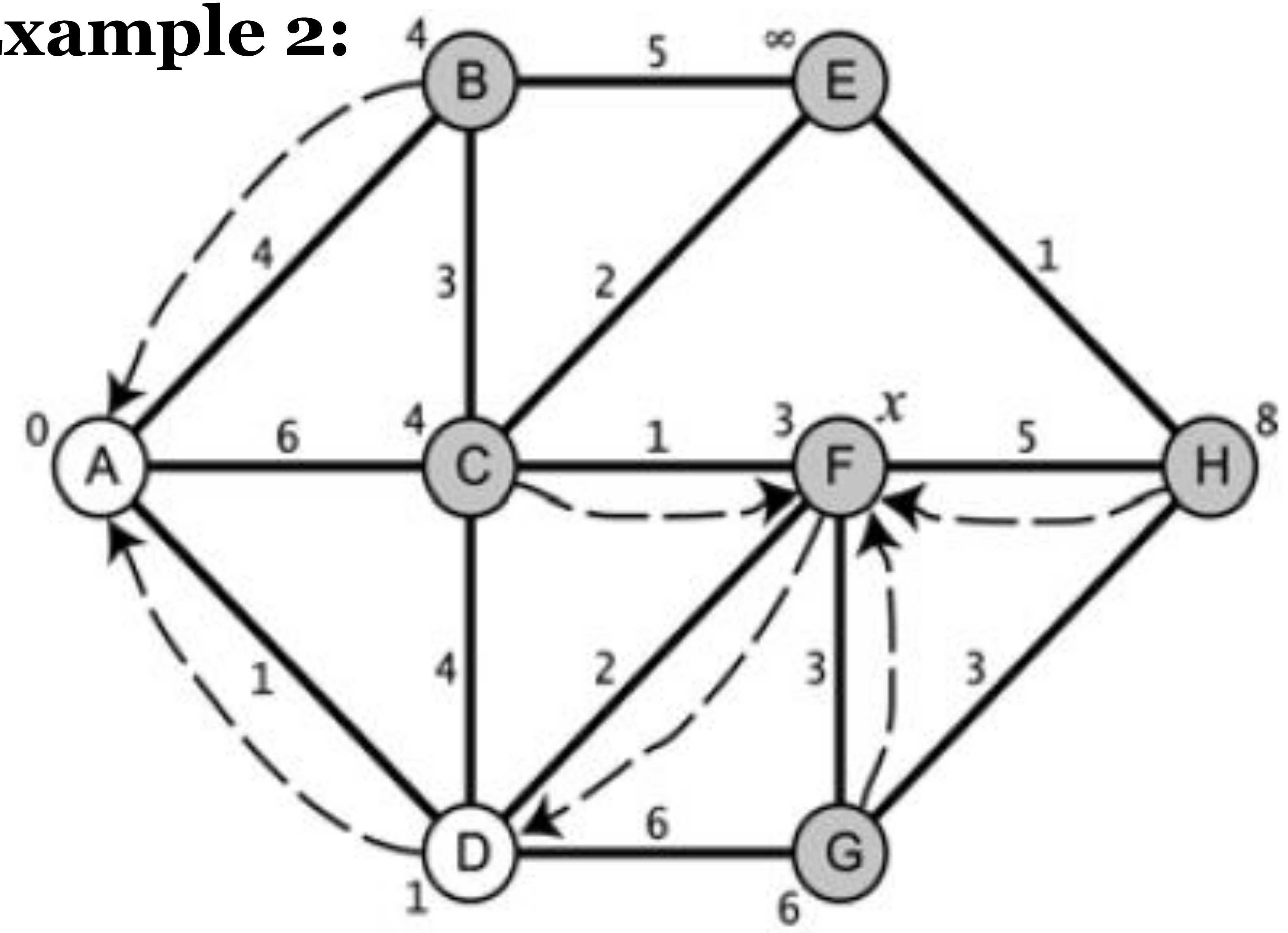
Dijkstra's Shortest Path Algorithm

Example 2:



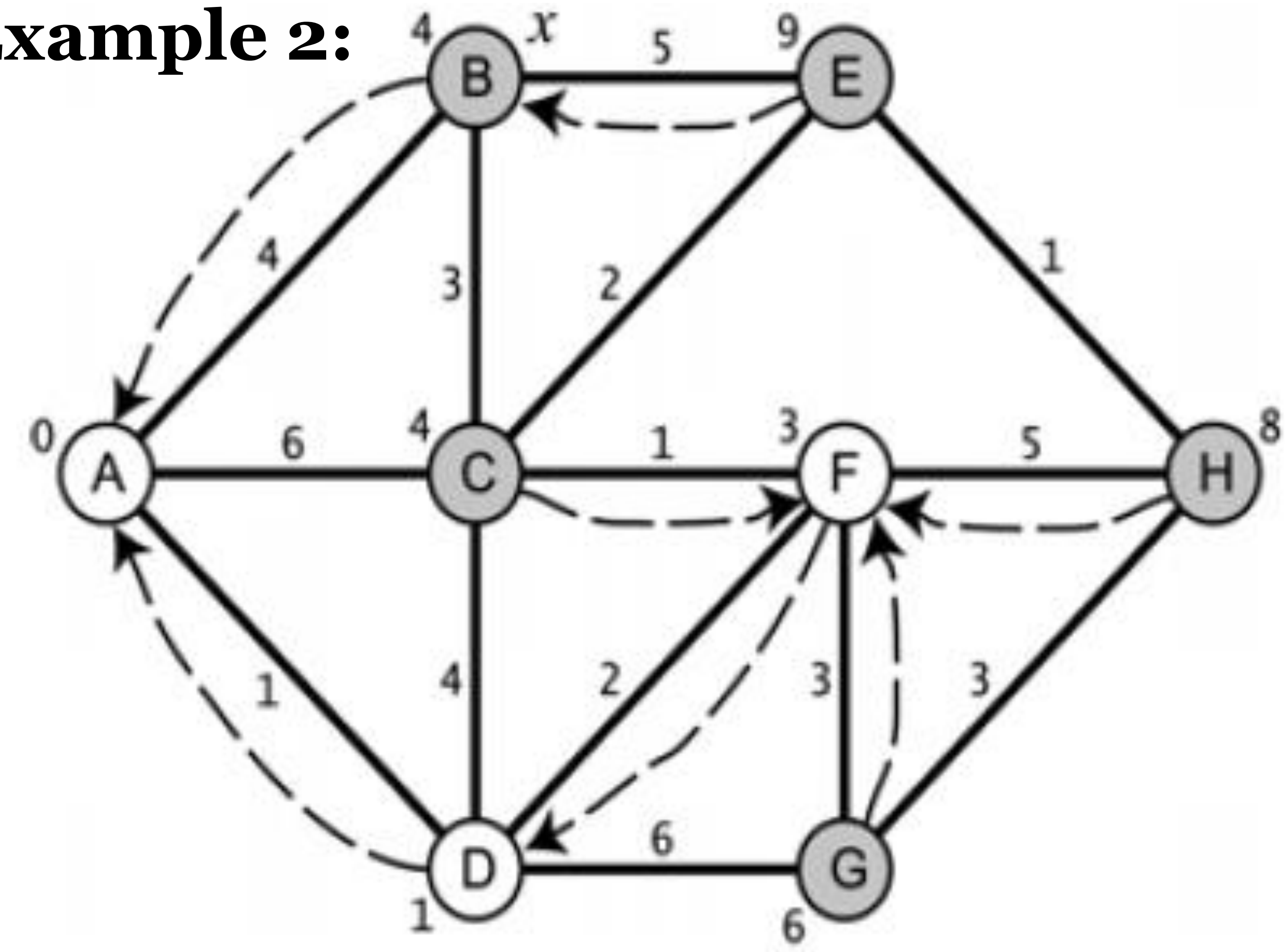
Dijkstra's Shortest Path Algorithm

Example 2:



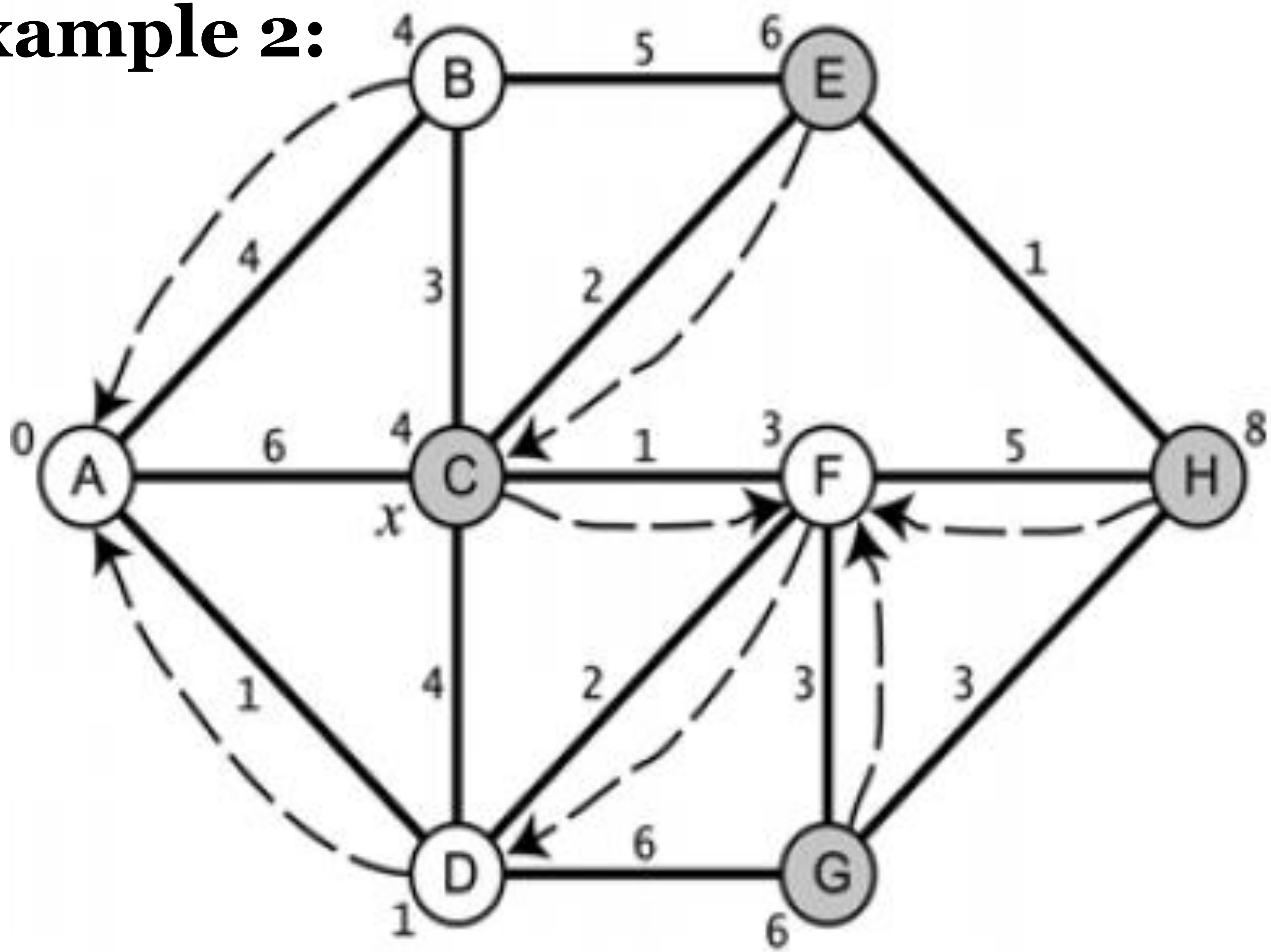
Dijkstra's Shortest Path Algorithm

Example 2:



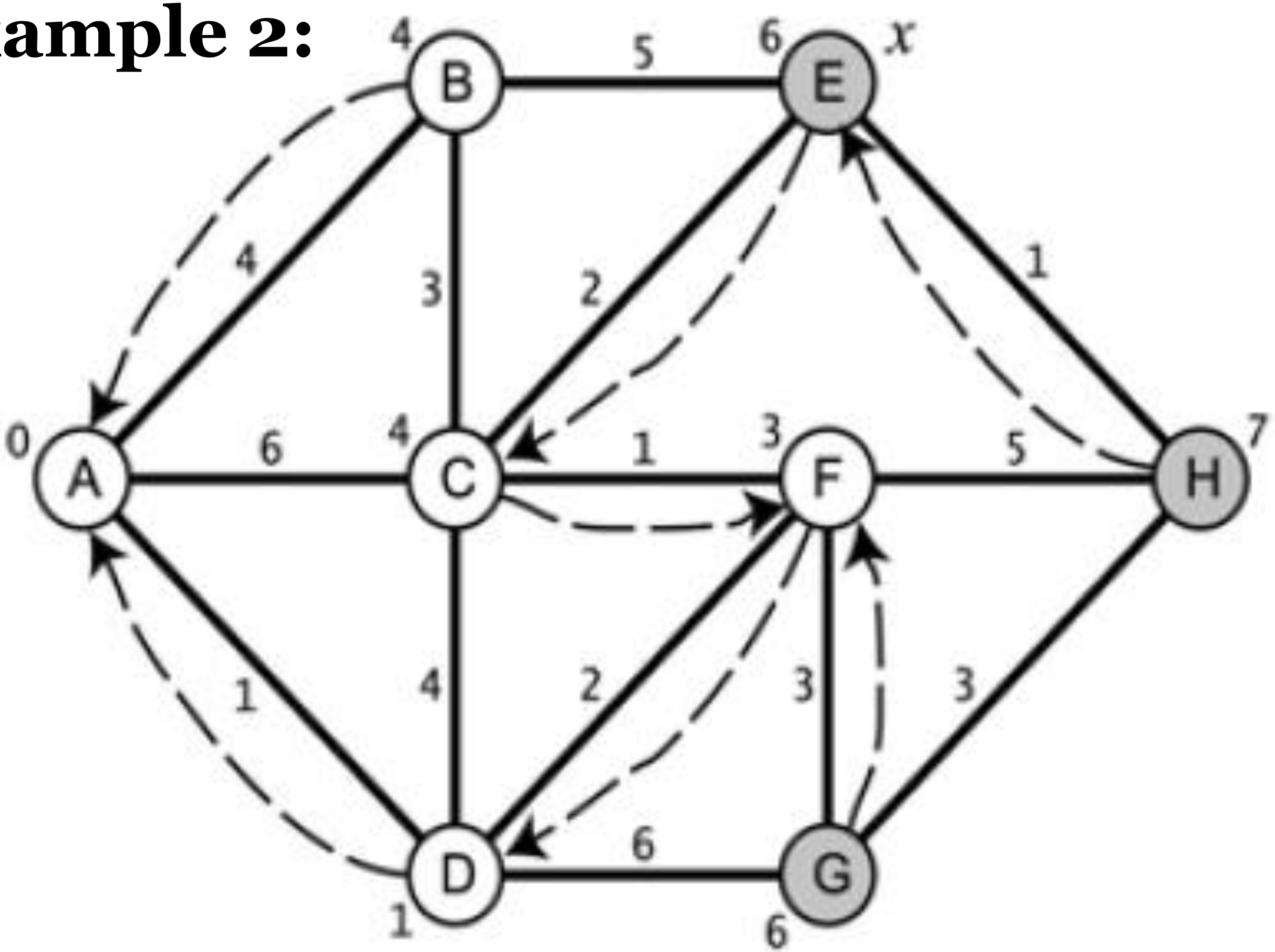
Dijkstra’s Shortest Path Algorithm

Example 2:



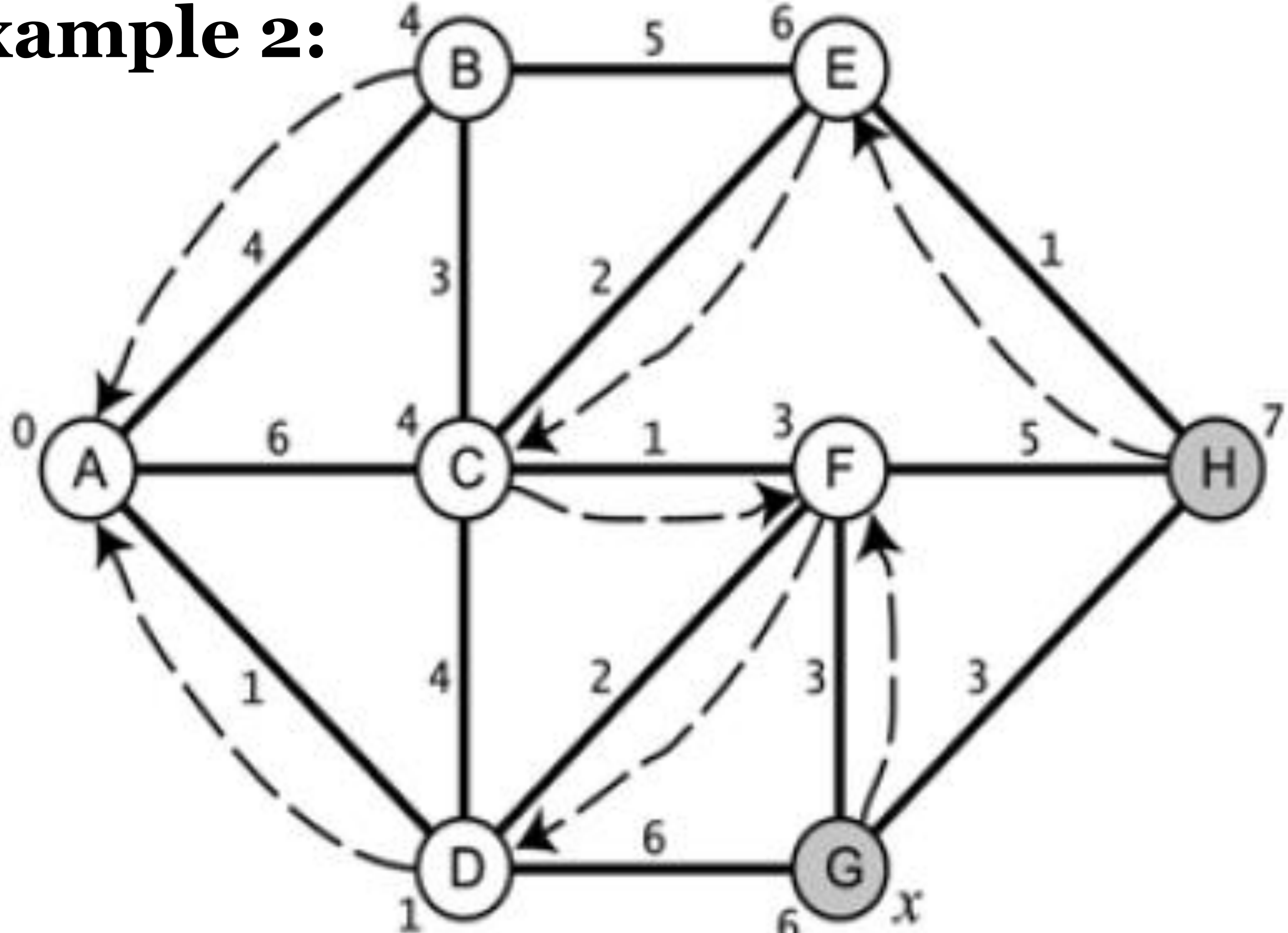
Dijkstra’s Shortest Path Algorithm

Example 2:



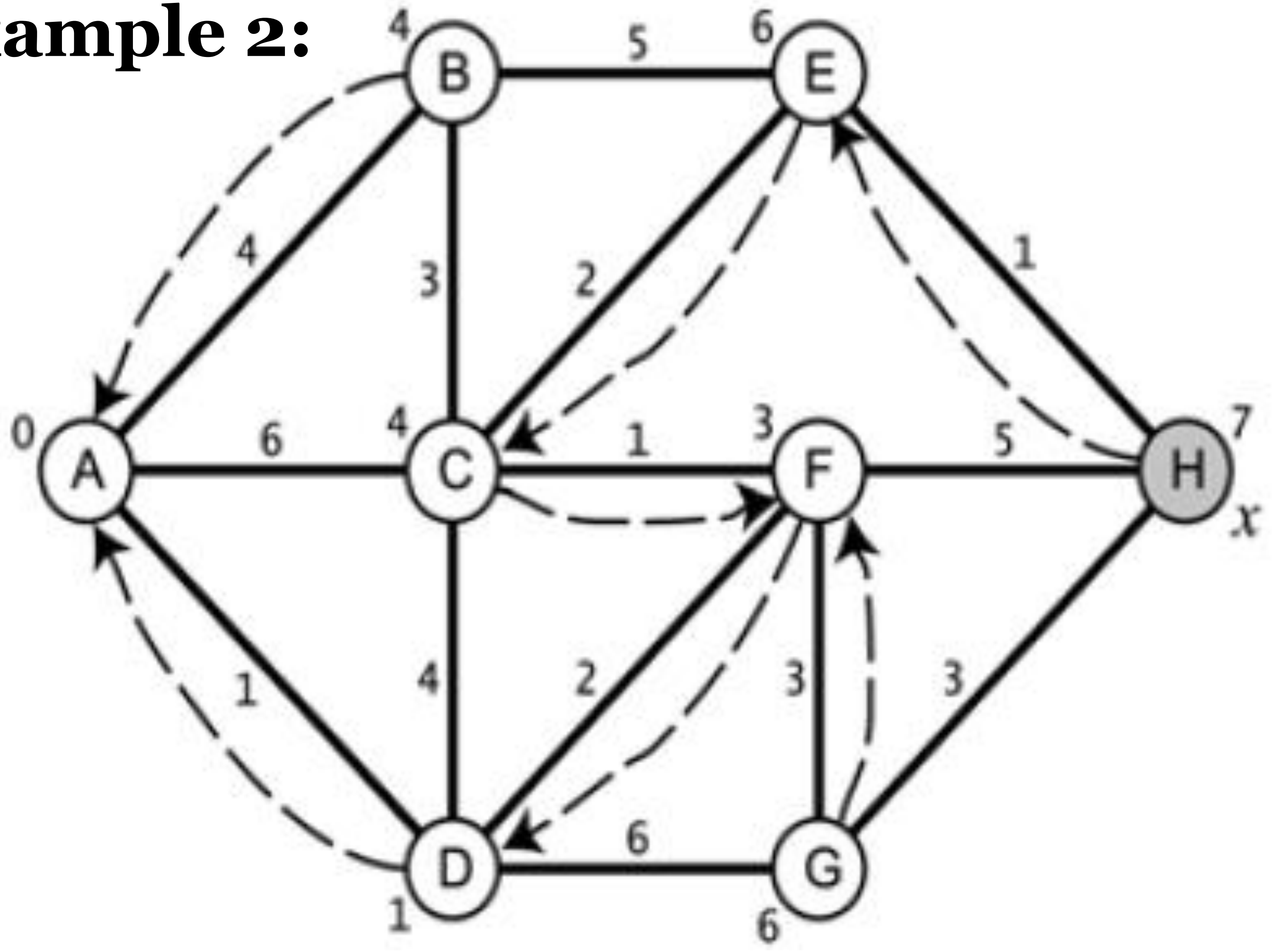
Dijkstra's Shortest Path Algorithm

Example 2:



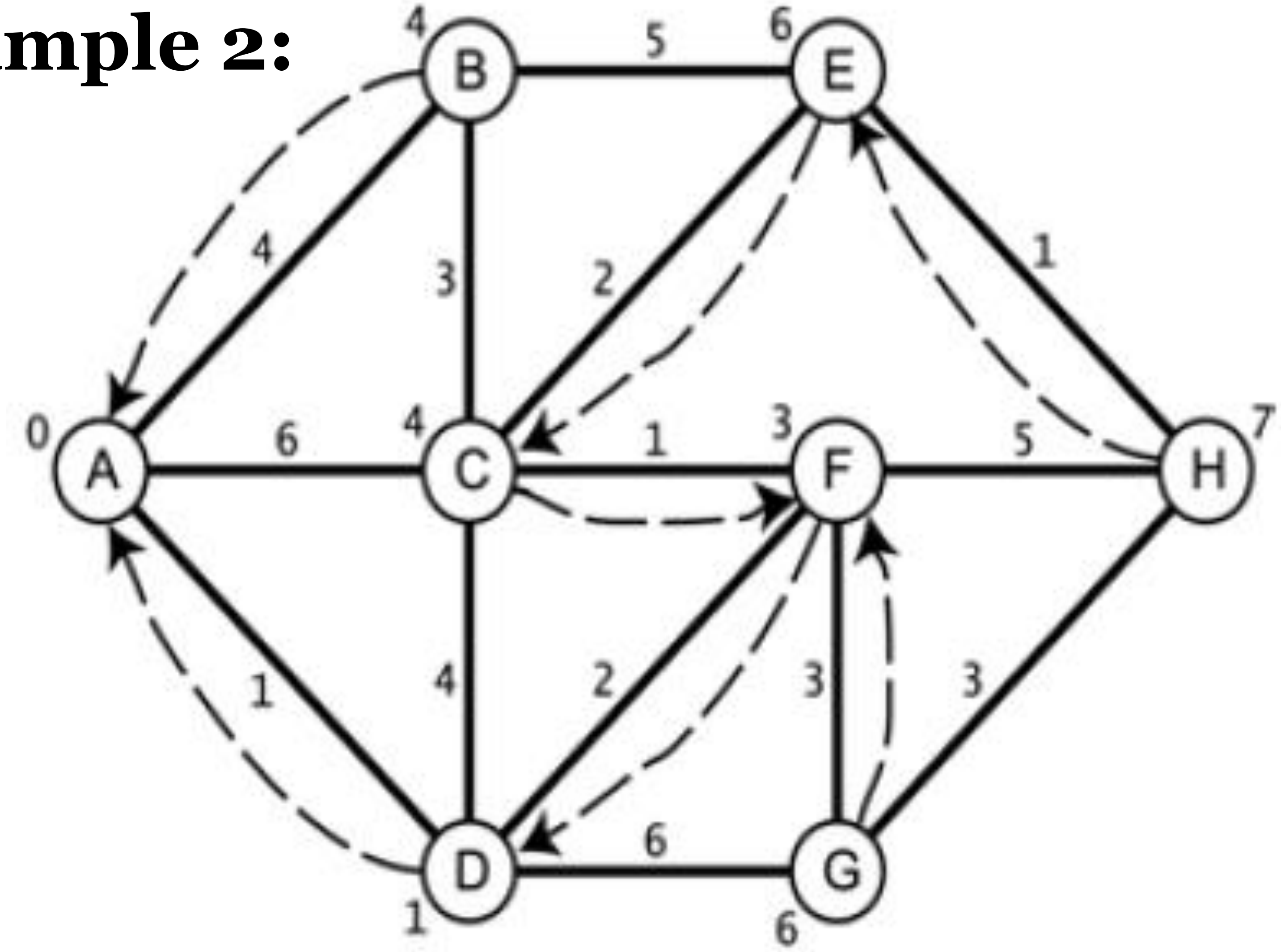
Dijkstra's Shortest Path Algorithm

Example 2:



Dijkstra’s Shortest Path Algorithm

Example 2:



for example, that the shortest path from A to E is ADFCE with length 6.

Dijkstra’s SPA- Time Complexity

- If the input graph is implemented using **adjacency matrix**, it is **$O(V^2)$**.

- If the input graph is represented using **adjacency list**, it can be reduced to **$O(E \log V)$** with the help of **binary heap** (**priority queue**).

Thank you!