Faculty of Computer Science
Data Science and Business Analytics (DSBA)

Algorithms and Data Structures

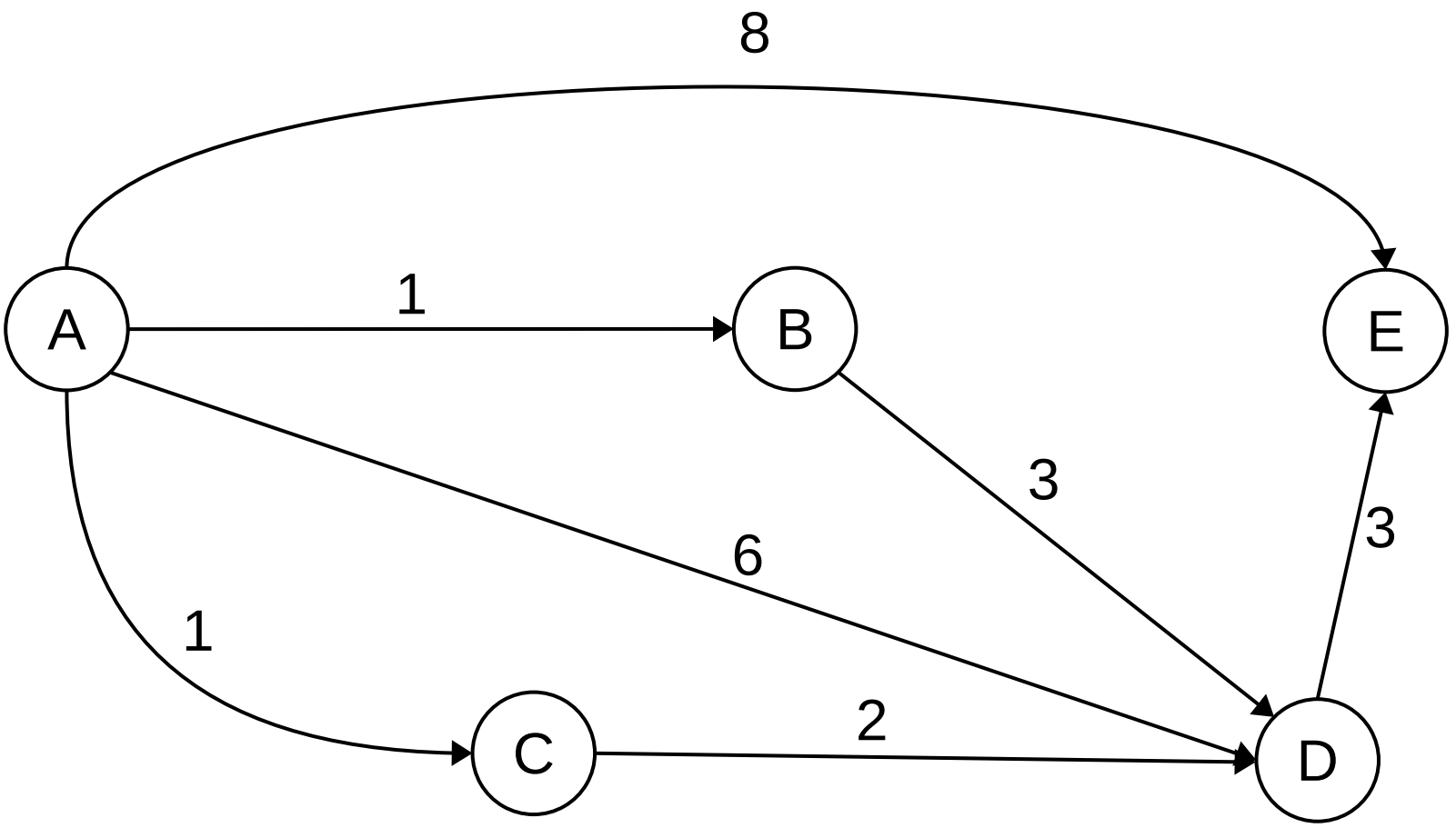
Seminar 8 – Module 3

Dijkstra's Algorithm

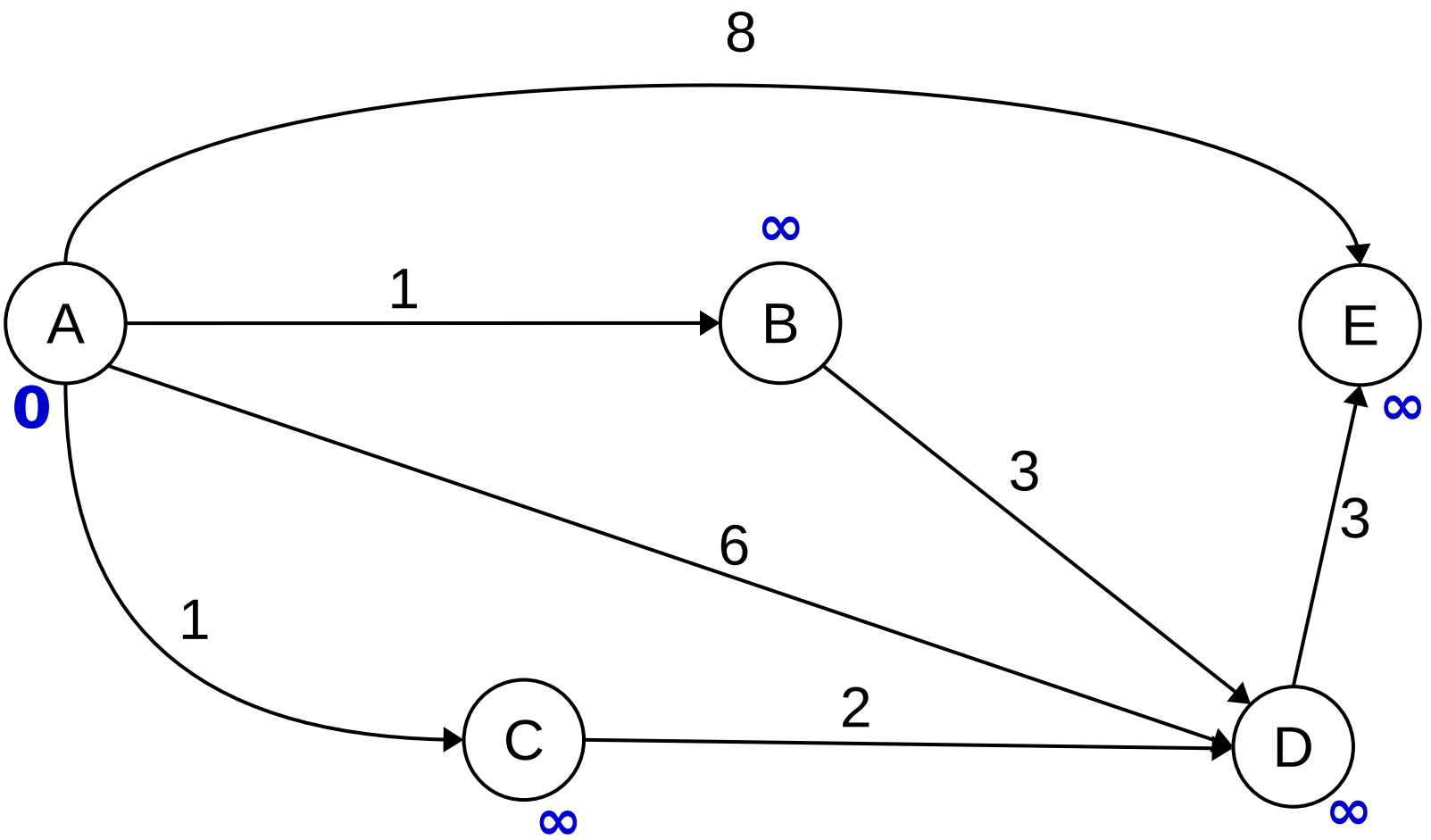
February 2022

J.C. Carrasquel

Dijkstra’s Algorithm

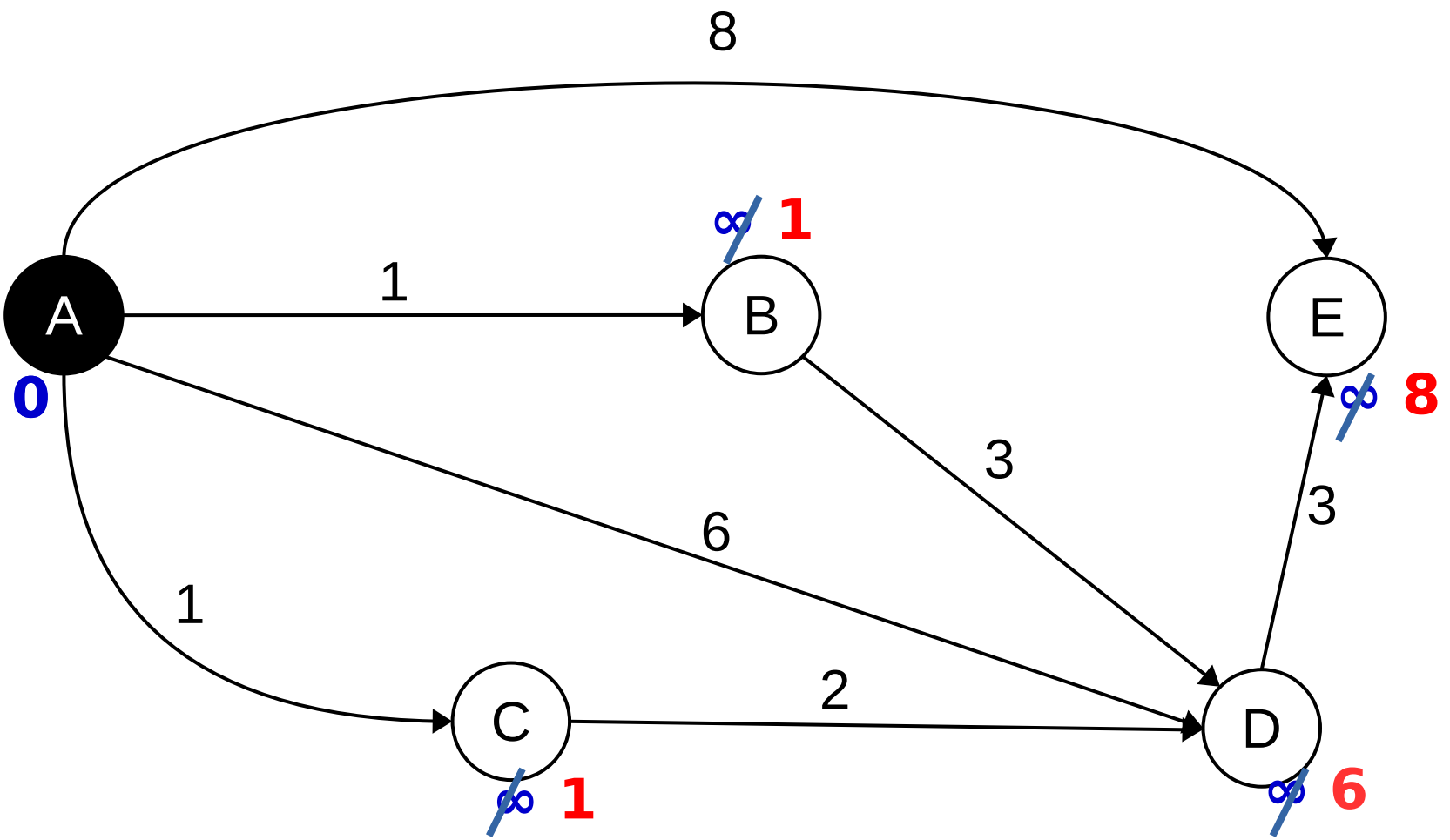


Dijkstra's Algorithm



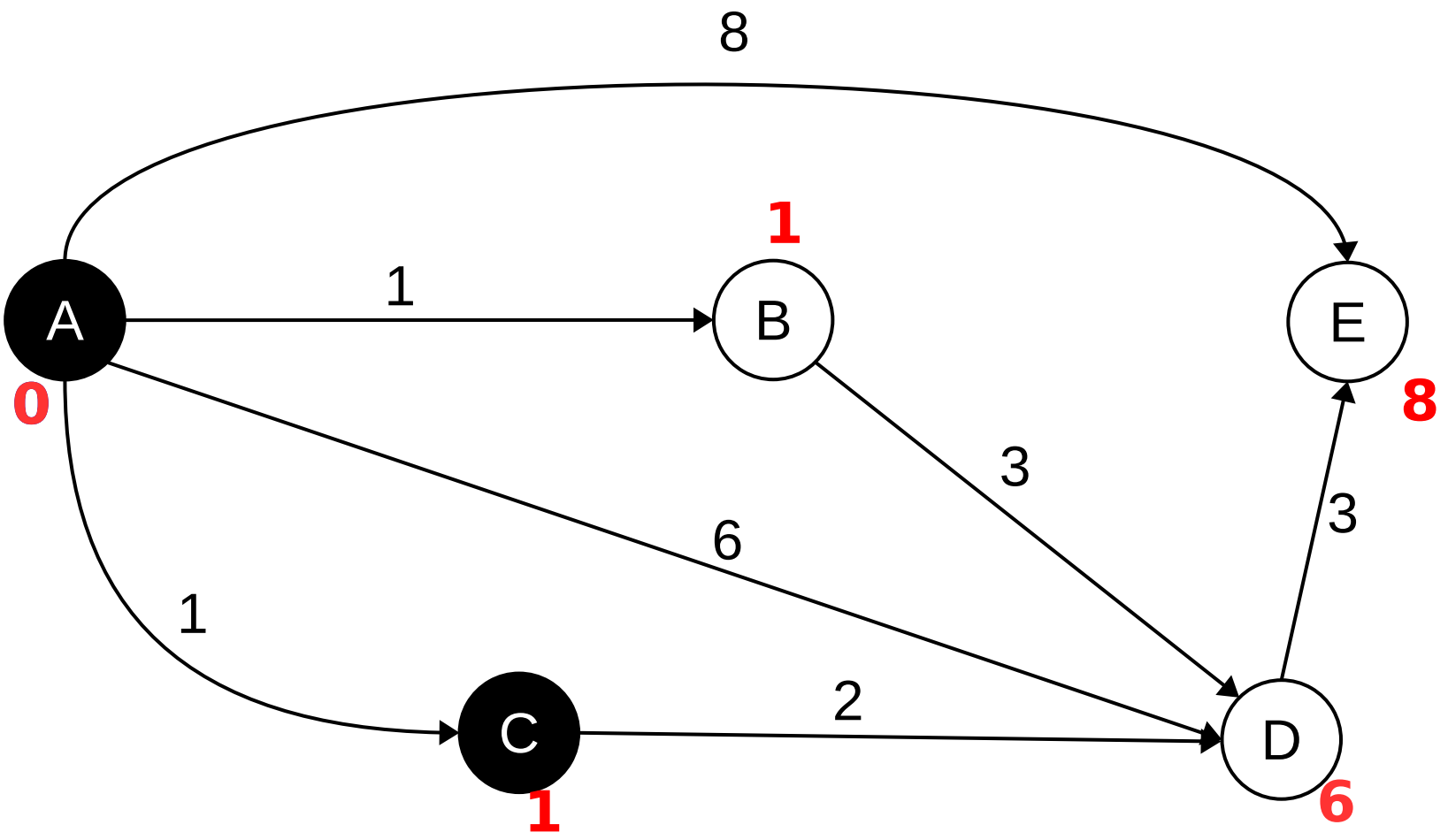

```
Get the next unvisited node u with min. d[u]
foreach( Node v  in  neighbors(u) ) do
        d[v] = min(  d[v], d[u] + edgeWeight(u,v) );
```

Dijkstra's Algorithm



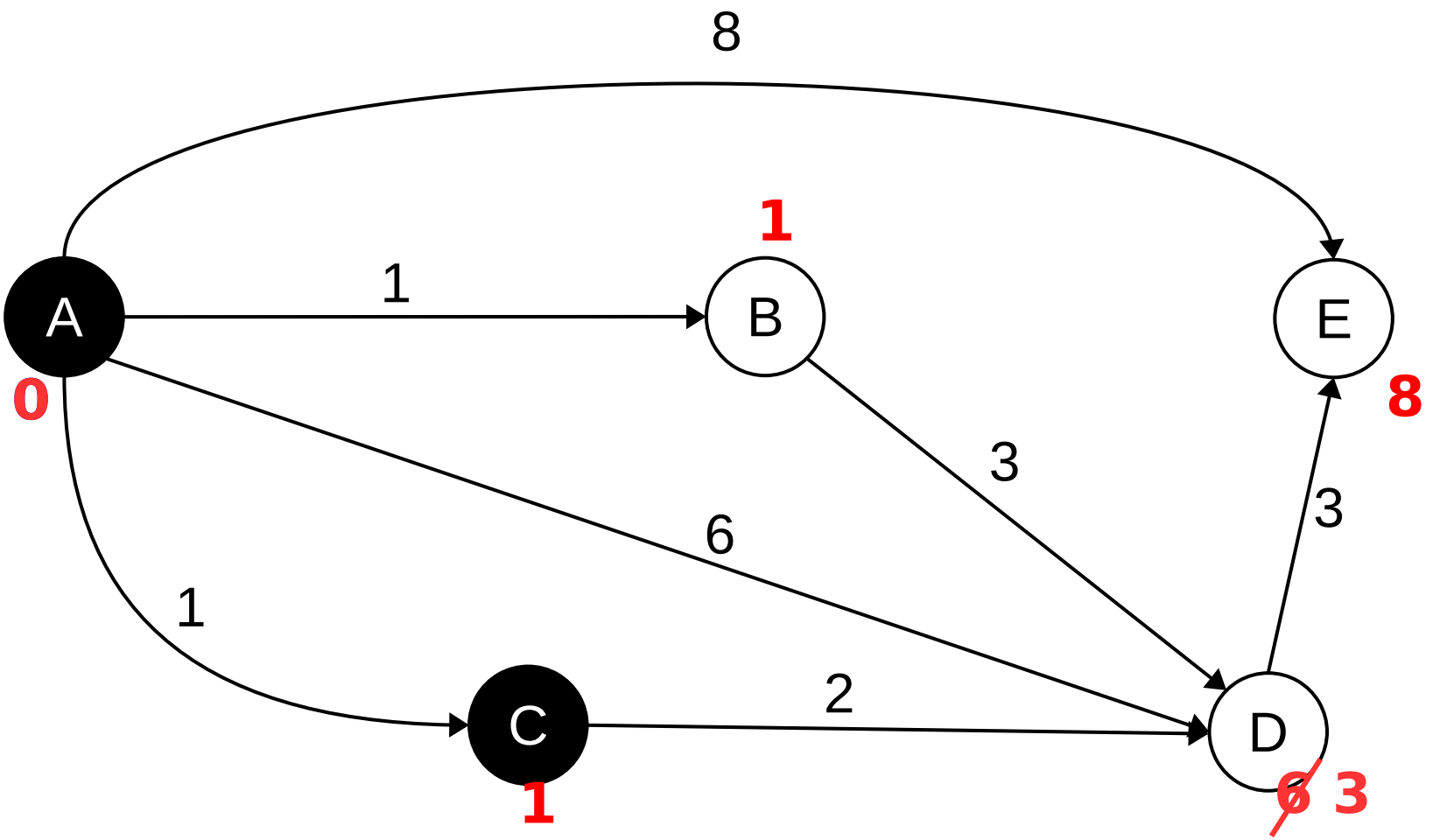
Get the next unvisited node u with min. d[u]
foreach(Node v **in** neighbors(u)) **do**
    d[v] = min(d[v], d[u] + edgeWeight(u,v));

Dijkstra's Algorithm



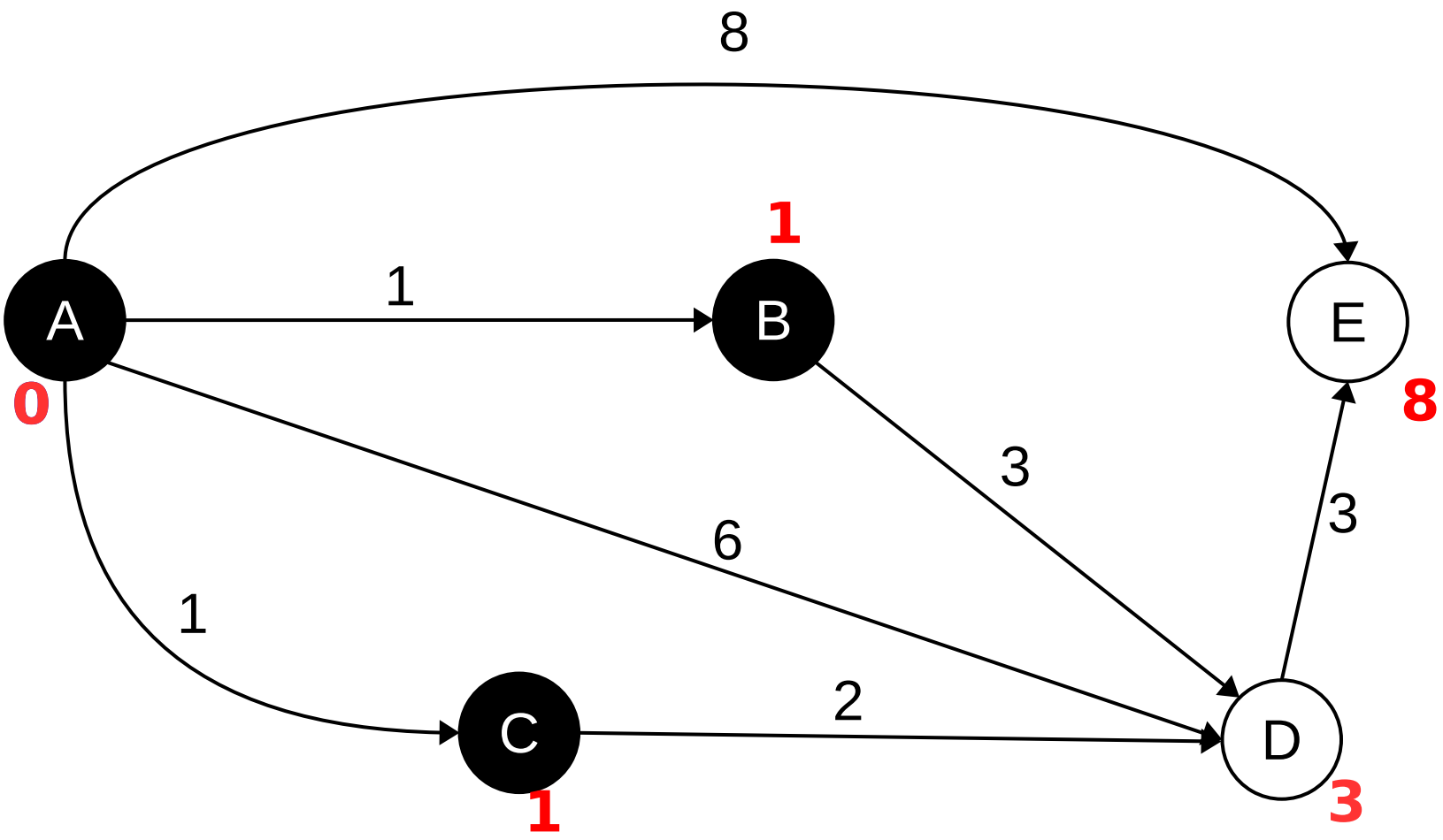
Get the next unvisited node u with min. d[u]
foreach(Node v **in** neighbors(u)) **do**
d[v] = min(d[v], d[u] + edgeWeight(u,v));

Dijkstra's Algorithm



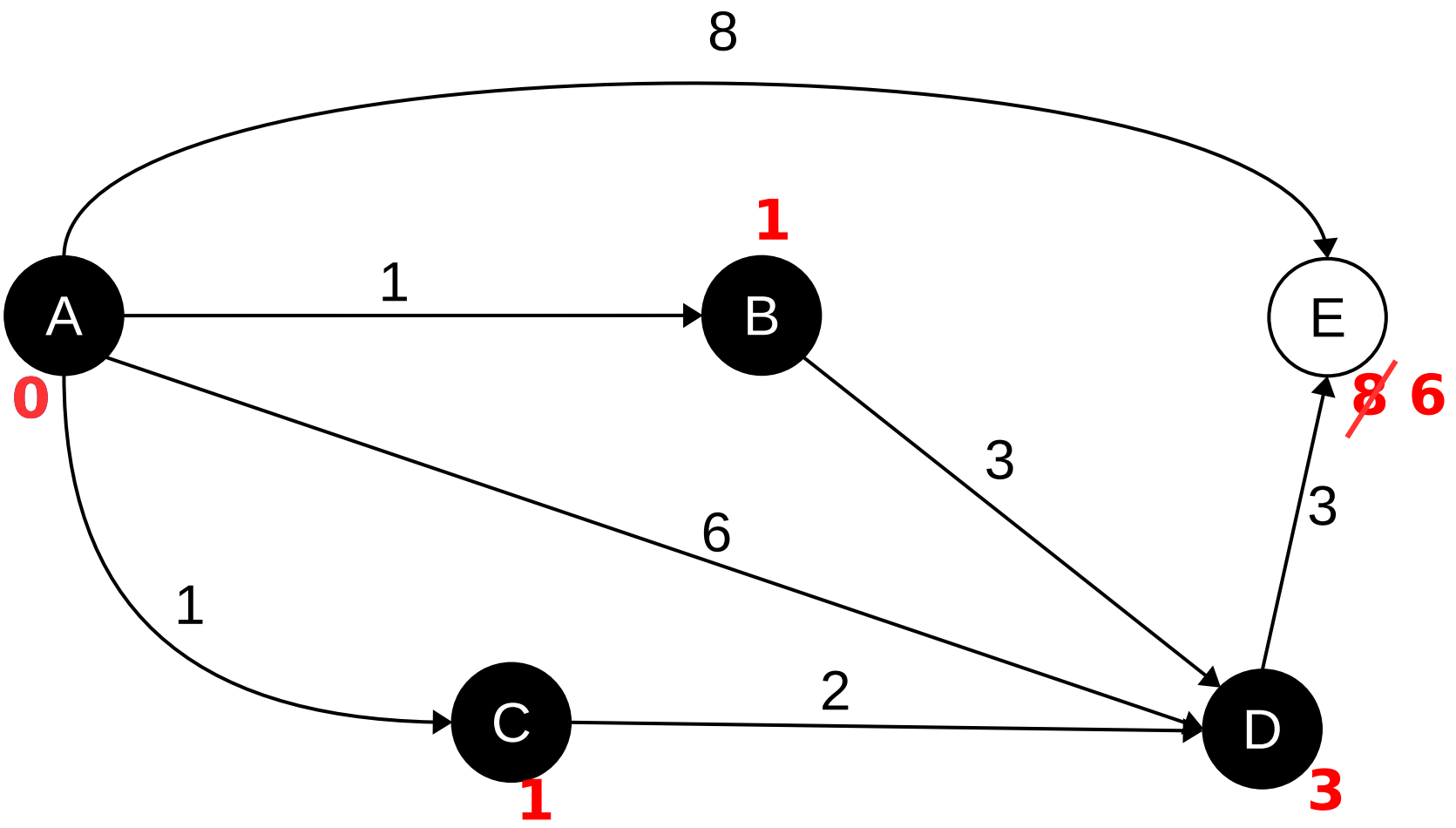
Get the next unvisited node u with min. d[u]
foreach(Node v **in** neighbors(u)) **do**
d[v] = min(d[v], d[u] + edgeWeight(u,v));

Dijkstra's Algorithm



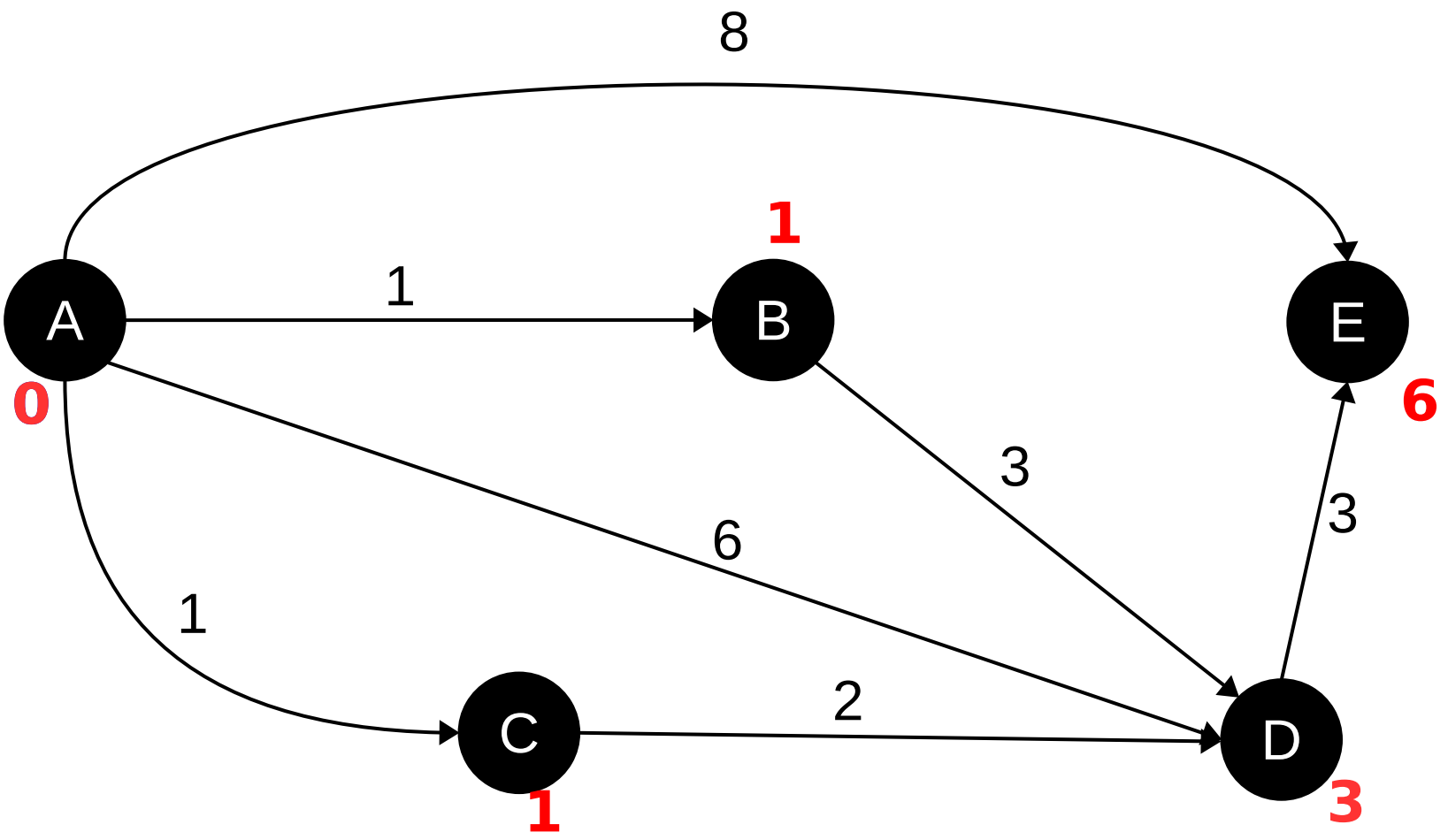

```
Get the next unvisited node u with min. d[u]
foreach( Node v  in  neighbors(u) ) do
        d[v] = min(  d[v], d[u] + edgeWeight(u,v) );
```

Dijkstra's Algorithm



Get the next unvisited node u with min. d[u]
foreach(Node v **in** neighbors(u)) **do**
d[v] = min(d[v], d[u] + edgeWeight(u,v));

Dijkstra's Algorithm




```
Get the next unvisited node u with min. d[u]
foreach( Node v  in  neighbors(u) ) do
        d[v] = min(  d[v], d[u] + edgeWeight(u,v) );
```

Dijkstra's Algorithm – Pseudo-code (version1)

Implementation with a PriorityQueue

O(M * Log(N))
M edges
N nodes

```
Dijkstra(G = (V,E), startNode):
    foreach Node v in V do
        distance[v] =  ∞;
        visited[v] = false;
        priorityQueue.insert(node, distance[v]);
    endfor

    distance[startNode] = 0;
    priorityQueue.update(startNode,0);

    while priorityQueue is not empty() do
        Node u = priorityQueue.getMin(); // O(1)
        foreach Node v  in  neighbors(u) such that visited[v] == false do
            distance[v] = min(  distance[v], distance[u] + edgeWeight(u,v) );
            if distance[v] is updated then
                priorityQueue.update( v , distance[v] );   // O(LogN)
            endif
        endfor
        visited[u] = true;
        priorityQueue.removeMin();  // O(LogN)
    endwhile
return distance;
```

Dijkstra's Algorithm – Pseudo-code

Implementation with a PriorityQueue

O(M * Log(N))
M edges
N nodes

```
Dijkstra(G = (V,E), startNode, endNode):
    foreach Node v in V do
        distance[v] =  ∞;
        visited[v] = false;
        priorityQueue.insert(node, distance[v]);
    endfor

    distance[startNode] = 0;
    priorityQueue.update(startNode, 0);

    while priorityQueue is not empty() do
        Node u = priorityQueue.getMin(); // O(1)
        foreach Node v  in  neighbors(u) such that visited[v] == false do
            distance[v] = min(  distance[v], distance[u] + edgeWeight(u,v) );
            if distance[v] is updated then
                priorityQueue.update( v , distance[v] );   // O(LogN)
            endif
        endfor
        visited[u] = true;
        priorityQueue.removeMin();  // O(LogN)
    endwhile
return distance;
```

How to get the actual shortest path between two nodes using Dijkstra?

Dijkstra's Algorithm – Pseudo-code (version2-with a parent map)

Implementation with a PriorityQueue

O(M * Log(N))
M edges
N nodes

```
Dijkstra(G = (V,E), startNode, endNode):
    foreach Node v in V do
        distance[v] =  ∞;
        visited[v] = false;
        priorityQueue.insert(node, distance[v]);
        parent[v] = nullptr;
    endfor
    distance[v] = 0;
    priorityQueue.update(startNode,0);

    while priorityQueue is not empty() do
        Node u = priorityQueue.getMin(); // O(1)
        foreach Node v  in  neighbors(u) such that visited[v] == false do
            distance[v] = min(  distance[v], distance[u] + edgeWeight(u,v) );
            if distance[v] is updated then
                priorityQueue.update( v , distance[v] );   // O(LogN)
            endif
        endfor
        visited[u] = true;
        priorityQueue.removeMin();  // O(LogN)
    endwhile
return distance, path ;
```

parent[v] = u;

How to get the actual shortest path between two nodes using Dijkstra?

1) Compute a parent map
2) Move thru' the parent map after emptying the priority queue.

```
std::vector<Node*> path;
Node* curNode = endNode;
while(curNode != nullptr)
{
        endNode.push_back(curNode);
        curNode = parent[curNode];
}
```

Dijkstra's Algorithm
- marking the nodes in the shortest path
From A to E