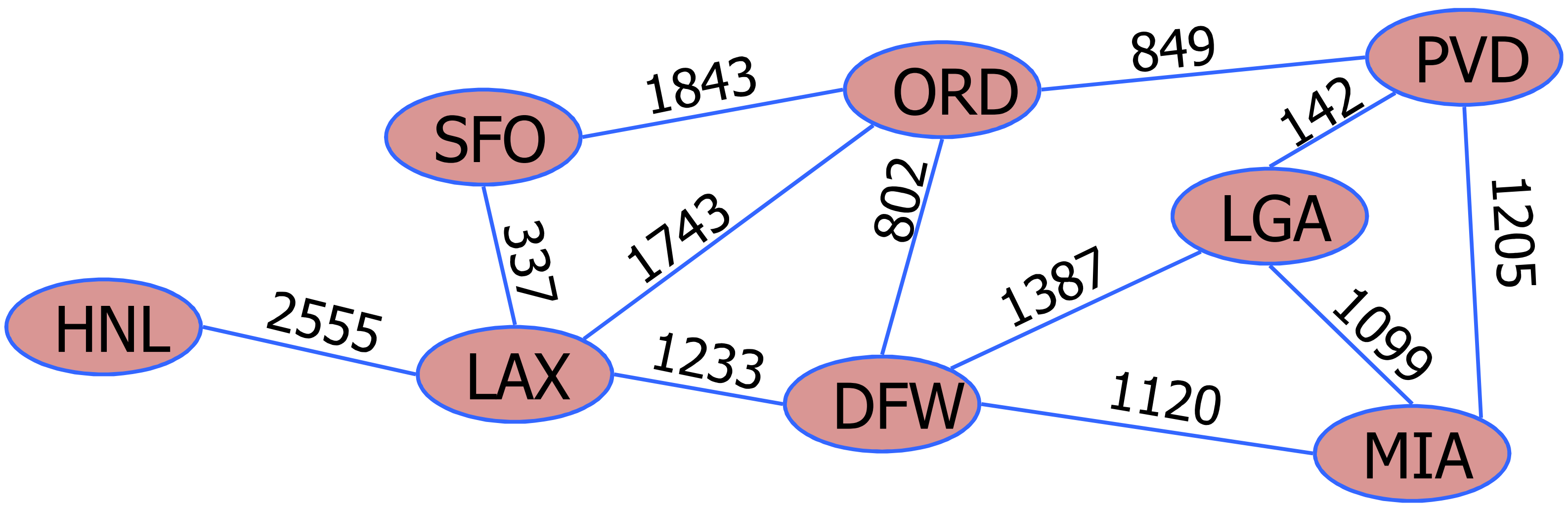
Outline

- Directed graphs
- Shortest path algorithms
  - Dijkstra
  - Bellman-ford

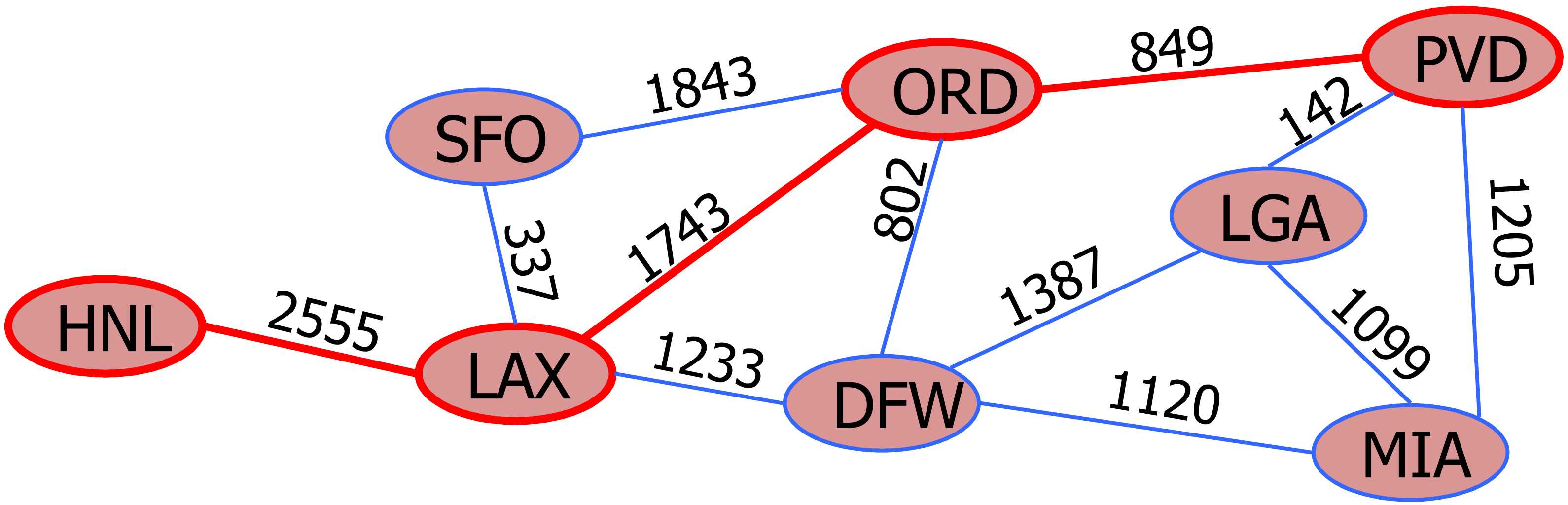
Weighted Graphs

- Each edge has an associated numerical value, called weight
- Edge weight may represent distances, costs, etc.
- Example:
  - Distance in miles between airports in a flight route graph



Shortest Paths

- A path of minimum total weight between two verticies
  - Length of a path is the sum of the weights of its edges
- Example:
  - Shortest path between PVD and HNL



Shortest Path Problems

- Single-source shortest path problems
  - Find shortest paths from a **source vertex** s (which is given) to all other vertices in a graph
  - Solution: a shortest-path tree rooted at s
    - which is also a spanning tree of G.
  - Algorithms: Dijkstra's algorithm and Bellman-Ford algorithm
  - If we just need to find a shortest path to one particular vertex, you can stop the algorithm earlier

Dijkstra Shortest Path Algorithm

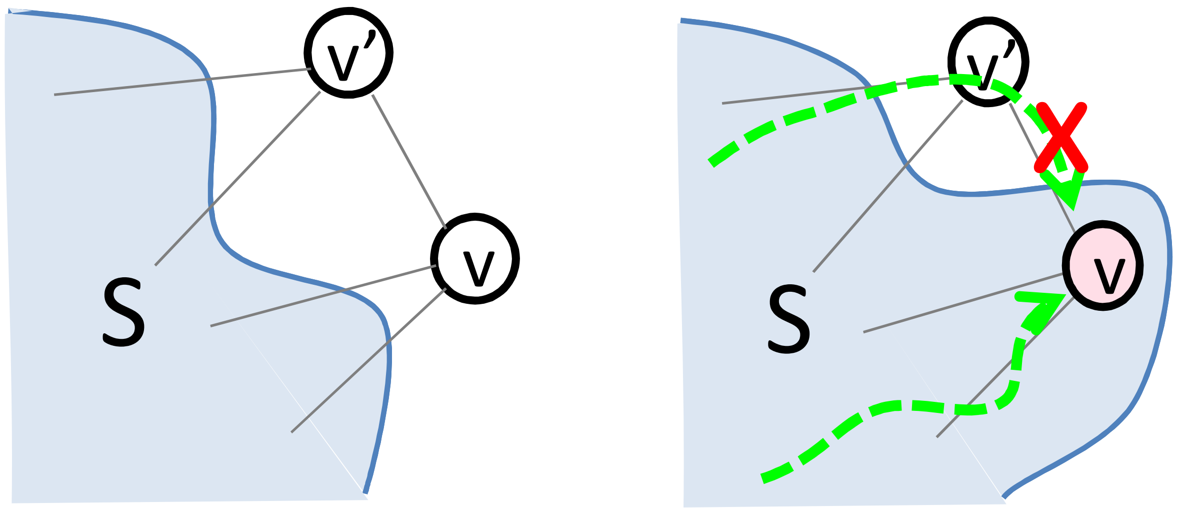
- For graphs with non-negative weights
- Algorithm
  - D[v]: length of a *currently best known path* from s to v
    - Initially, D[s] = 0 and D[v] = ∞ for all other vertices
  - Let S be a set of visited vertices
    - Initially, S is empty
  - Repeat the following until all vertices are added to S
    - Among vertices not in S, choose vertex v with the smallest D[v] and add v to S
    - For every vertex v' adjacent to v and v' is not in S, update D[v']:

$$D[v'] = \min (D[v'],\ D[v] + weight(v, v'))$$

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Optimality

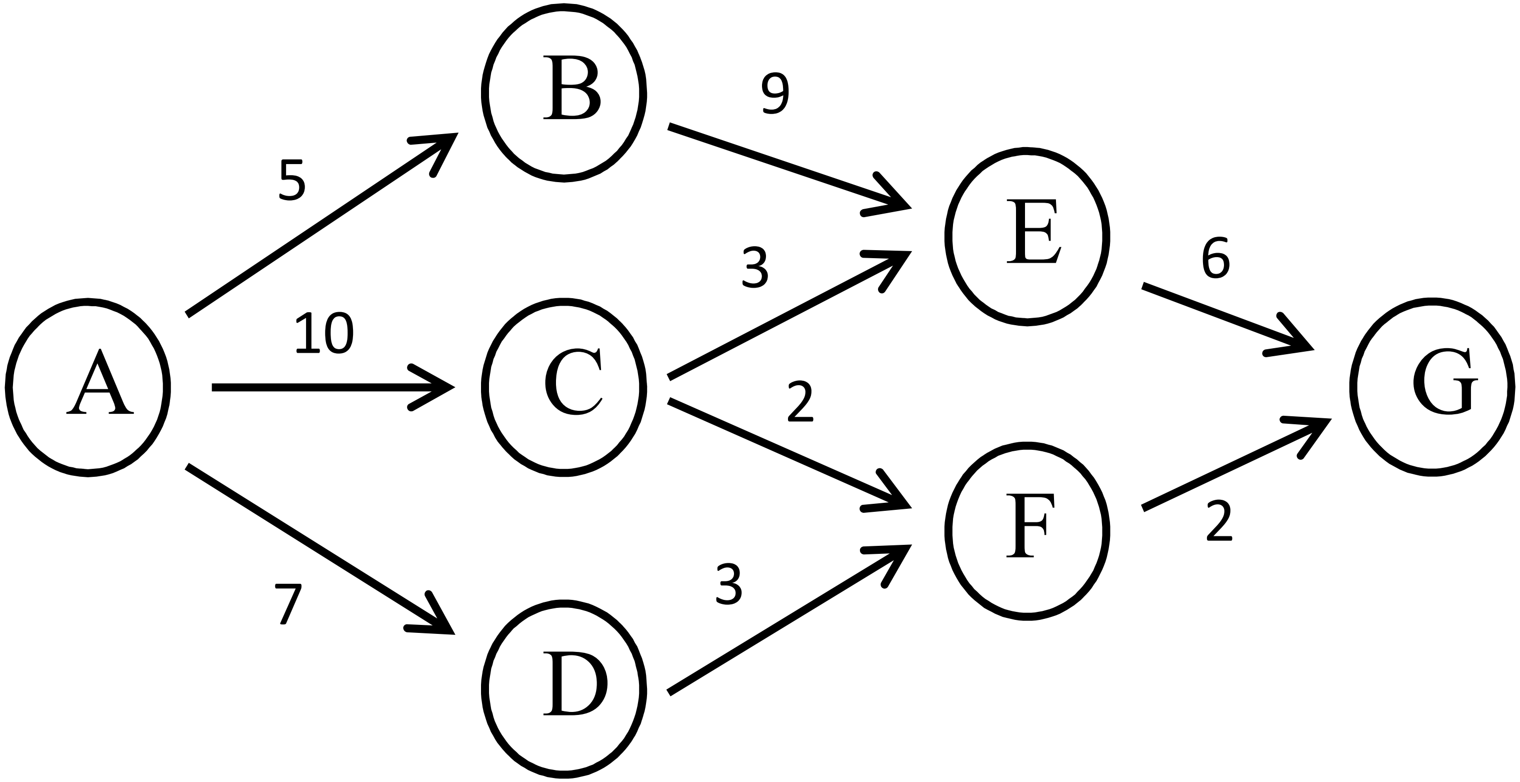
- Once v is added to S, it is the shortest path from s to v
- There is no v' ∉ S such that the path s → v' → v is shorter than the path s → v using vertices in S only
  - Proof by contradiction: If such v' exists, D[v] > D[v'], which violates the fact that D[v] ≤ D[v'] when v is added to S



Green arrow: shortest path to node

Dijkstra's Algorithm

Source : A

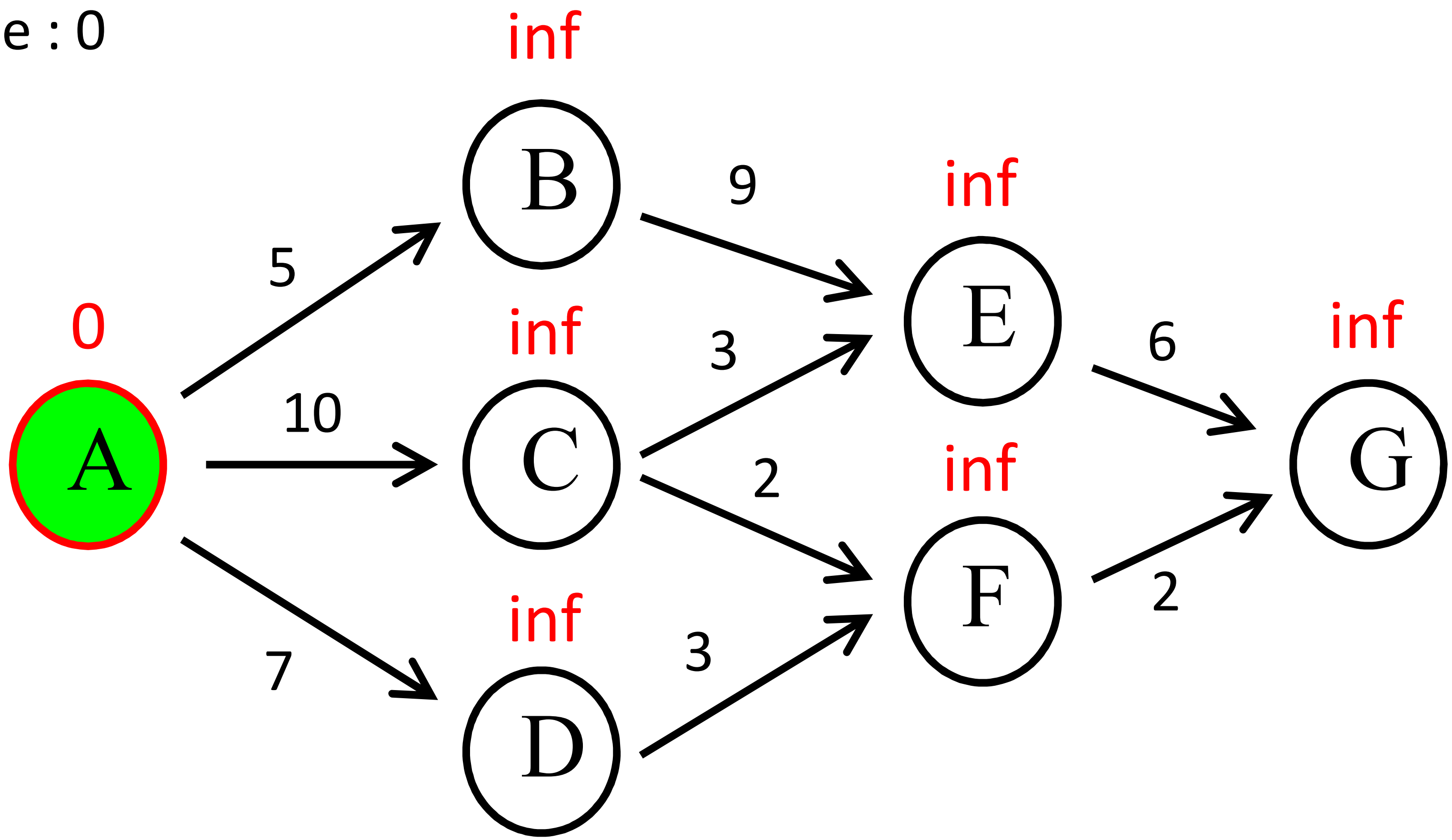


S : {}

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Dijkstra's Algorithm

Source : 0

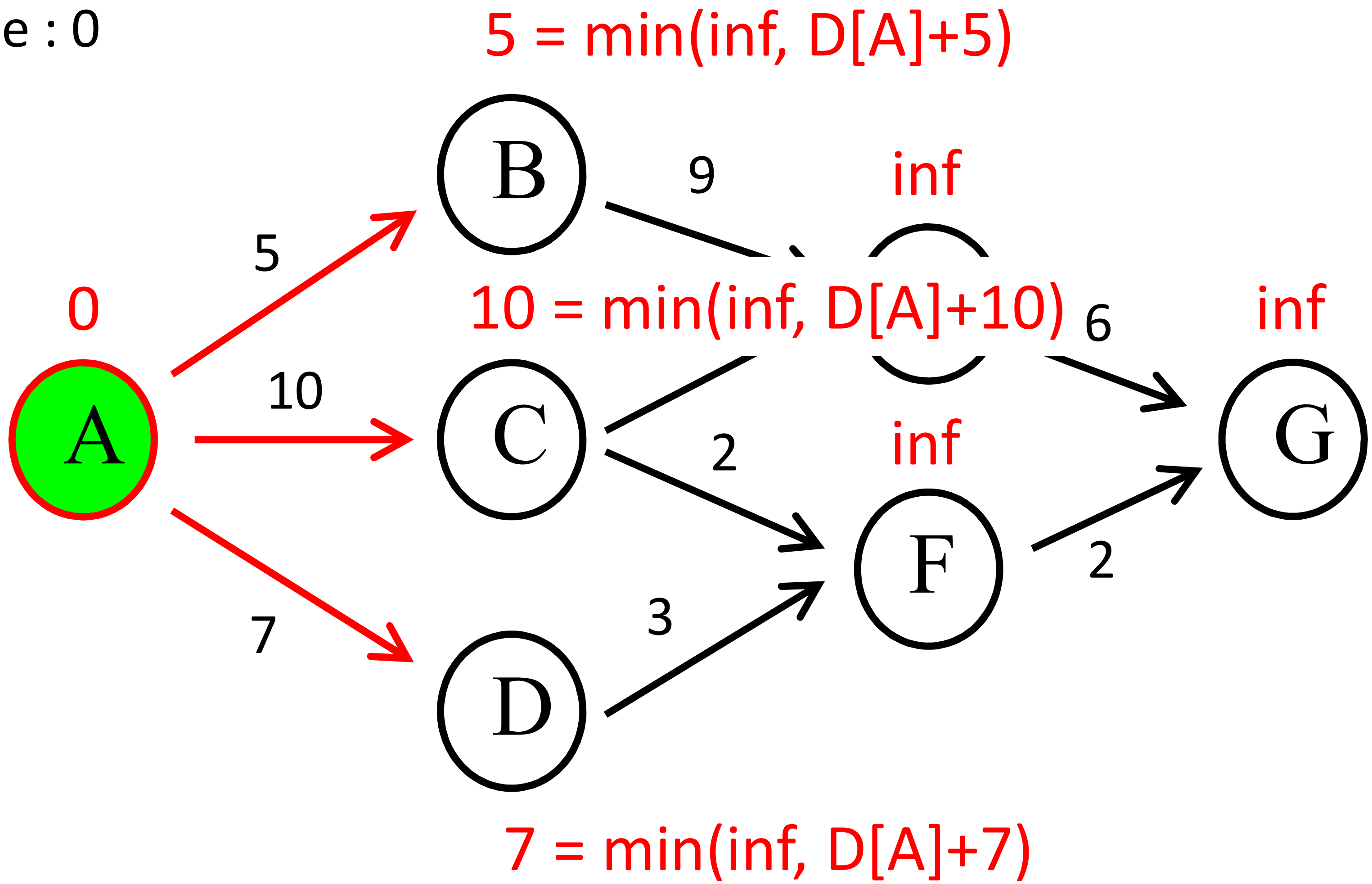


S : { A }

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Dijkstra's Algorithm

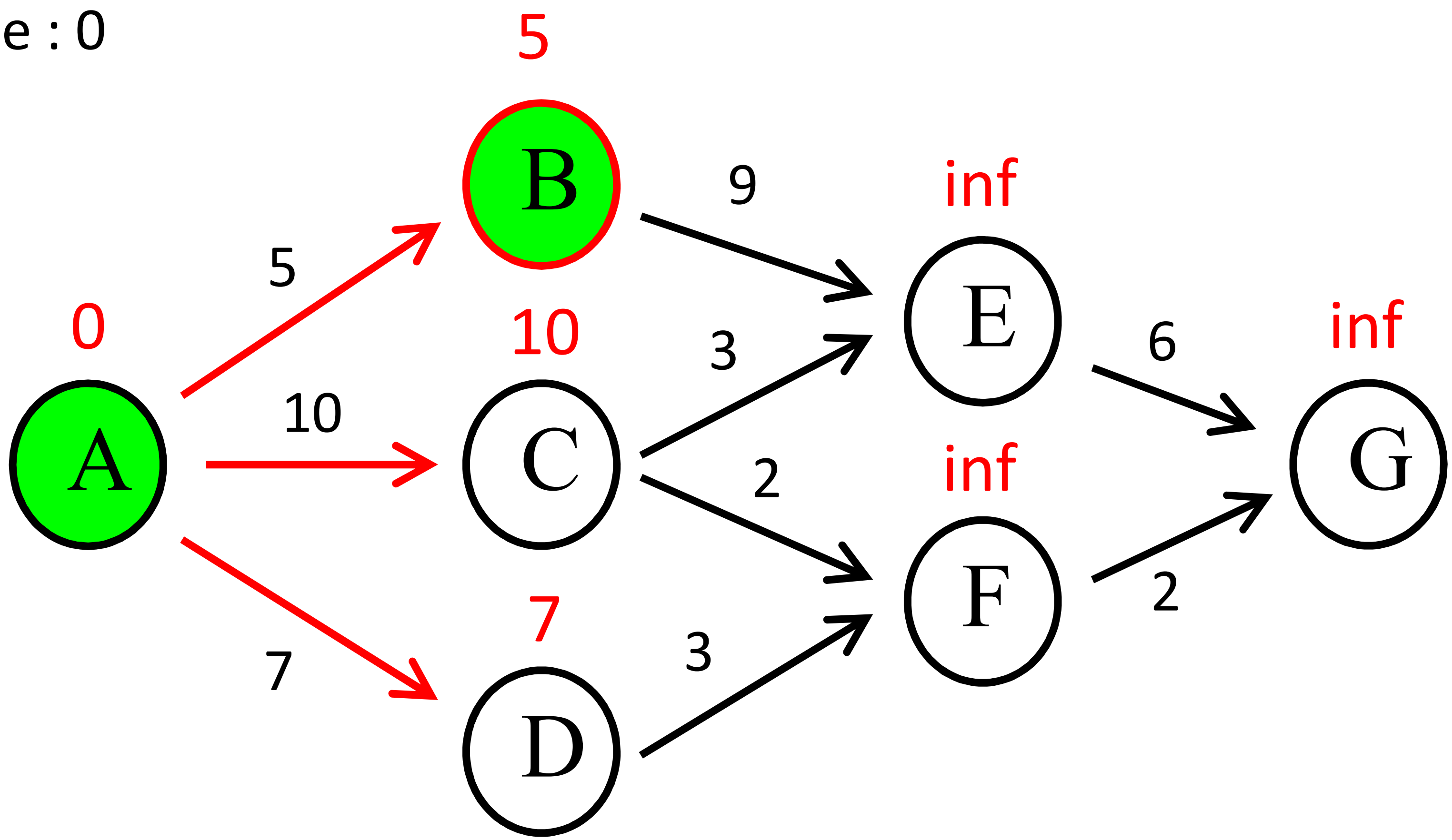
Source : 0



S : { A }

Dijkstra's Algorithm

Source : 0

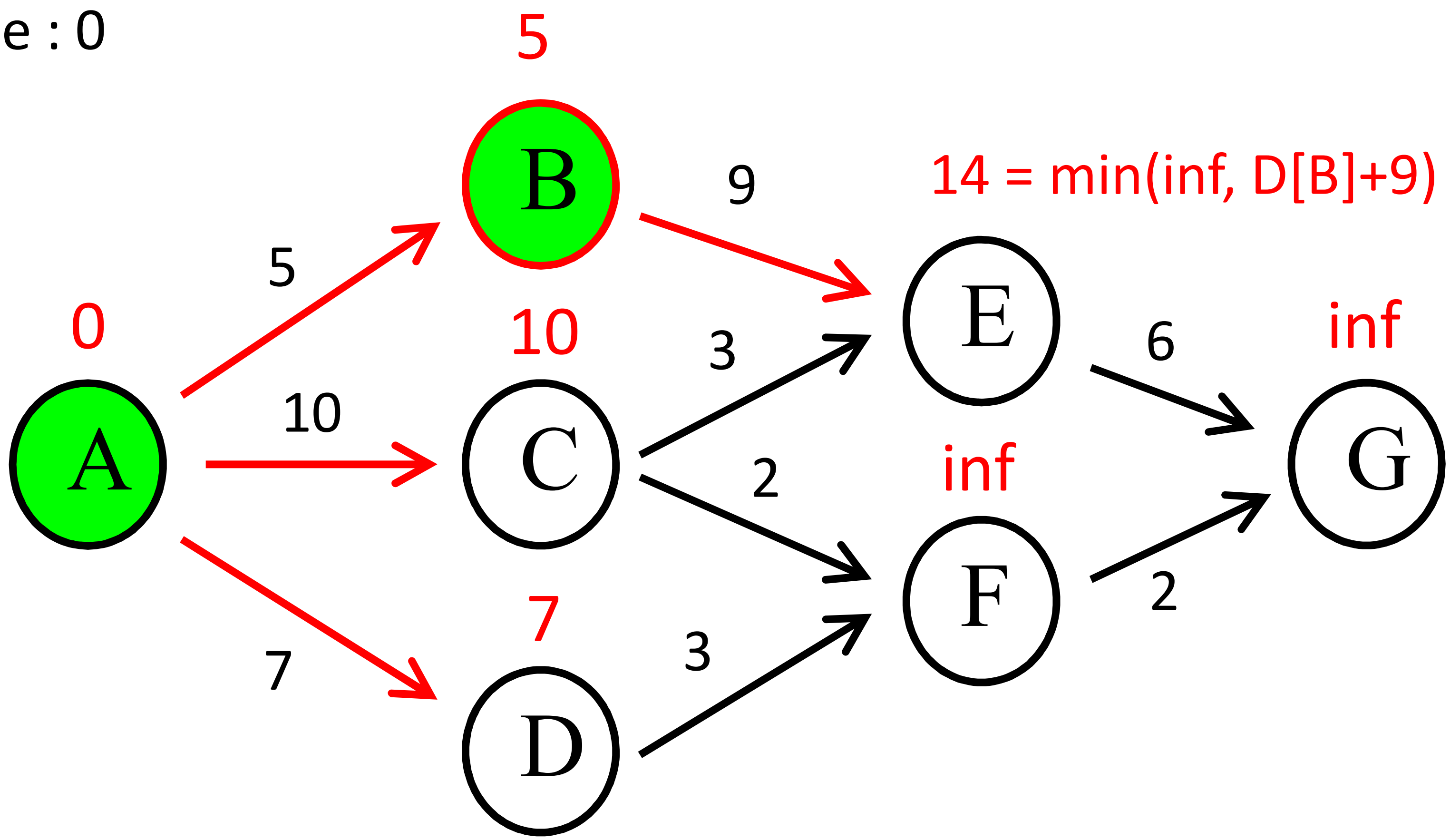


S : { A, B }

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Dijkstra's Algorithm

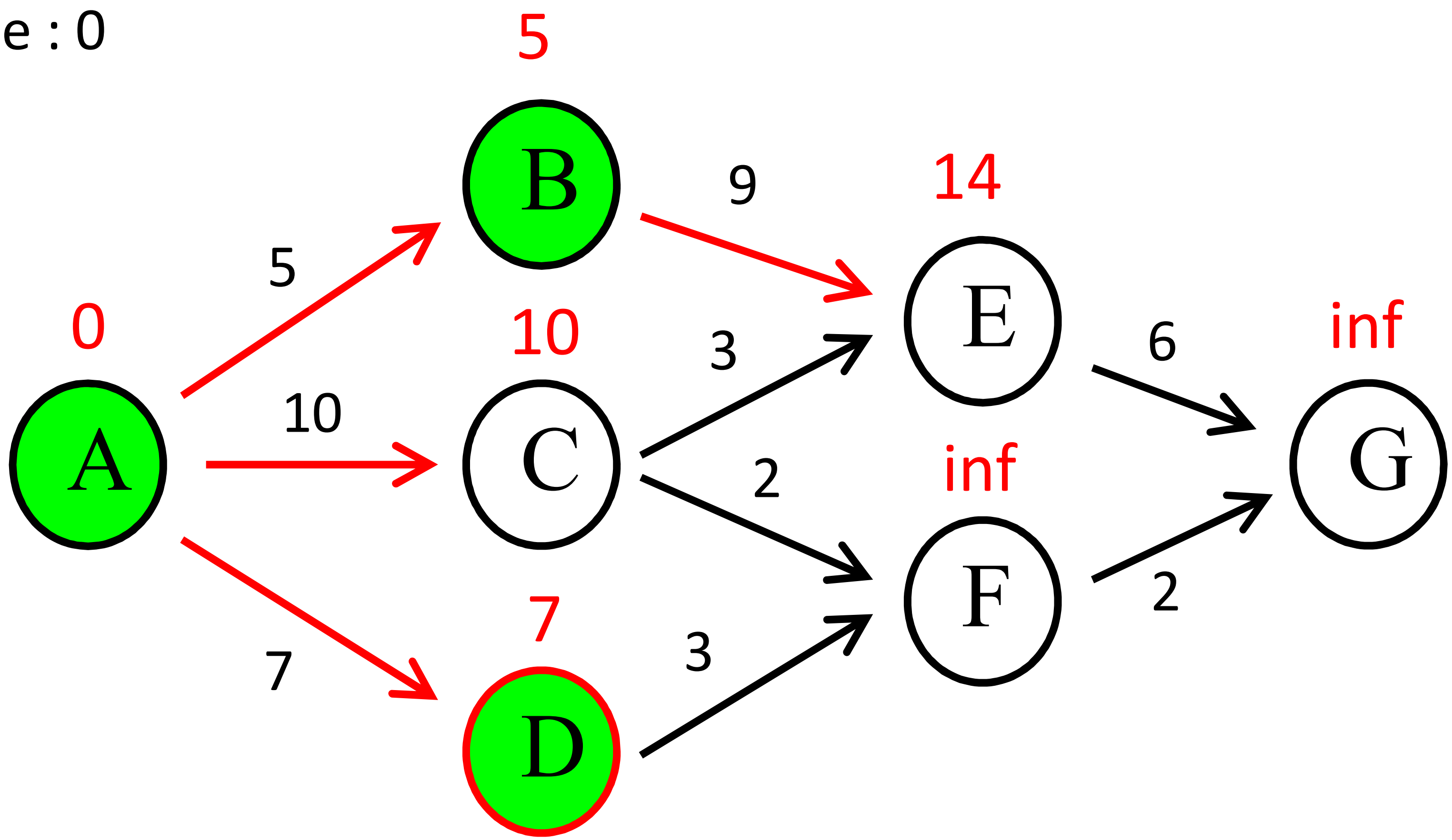
Source : 0



S : { A, B }

Dijkstra's Algorithm

Source : 0

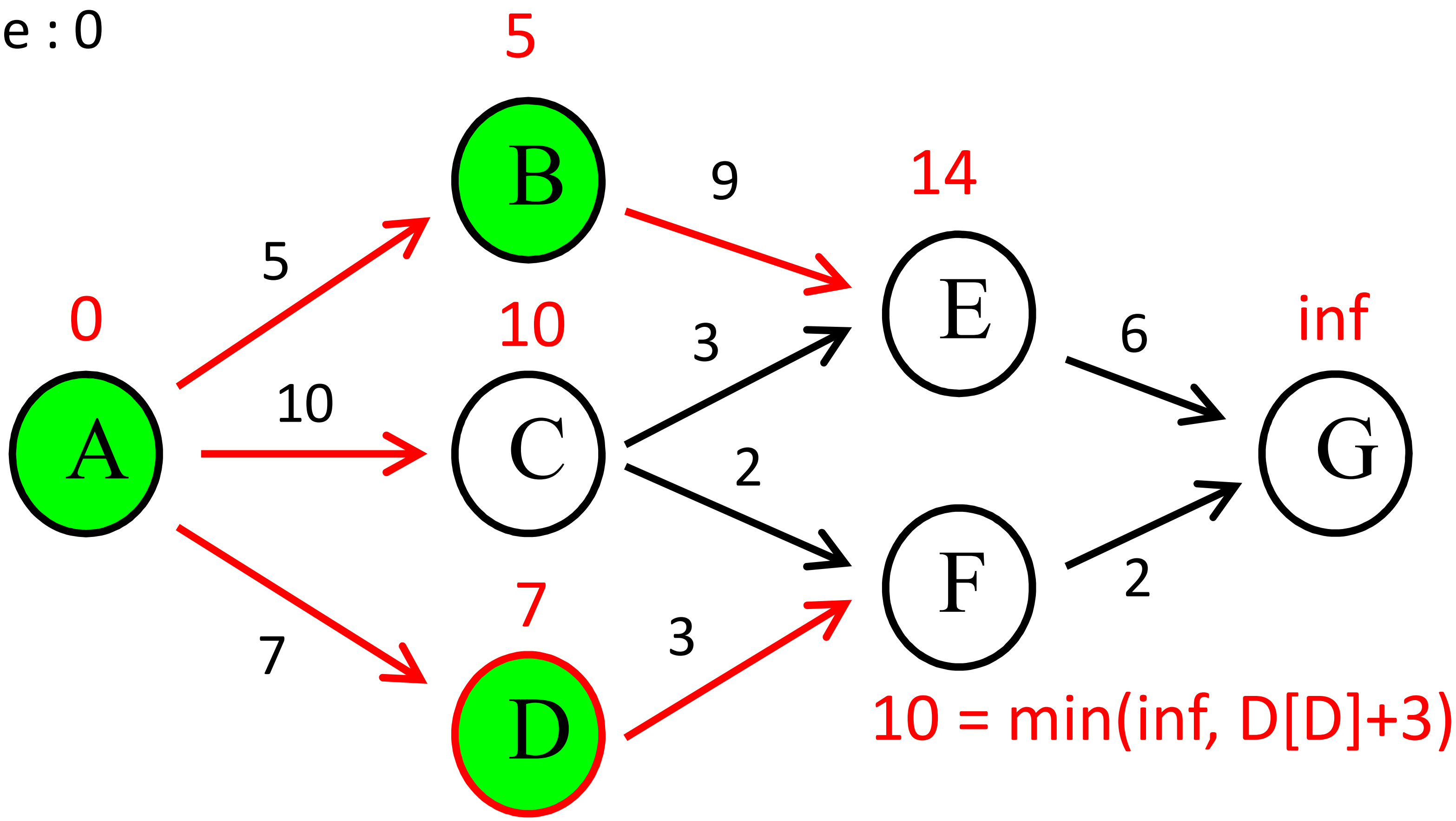


S : {A, B, D}

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Dijkstra's Algorithm

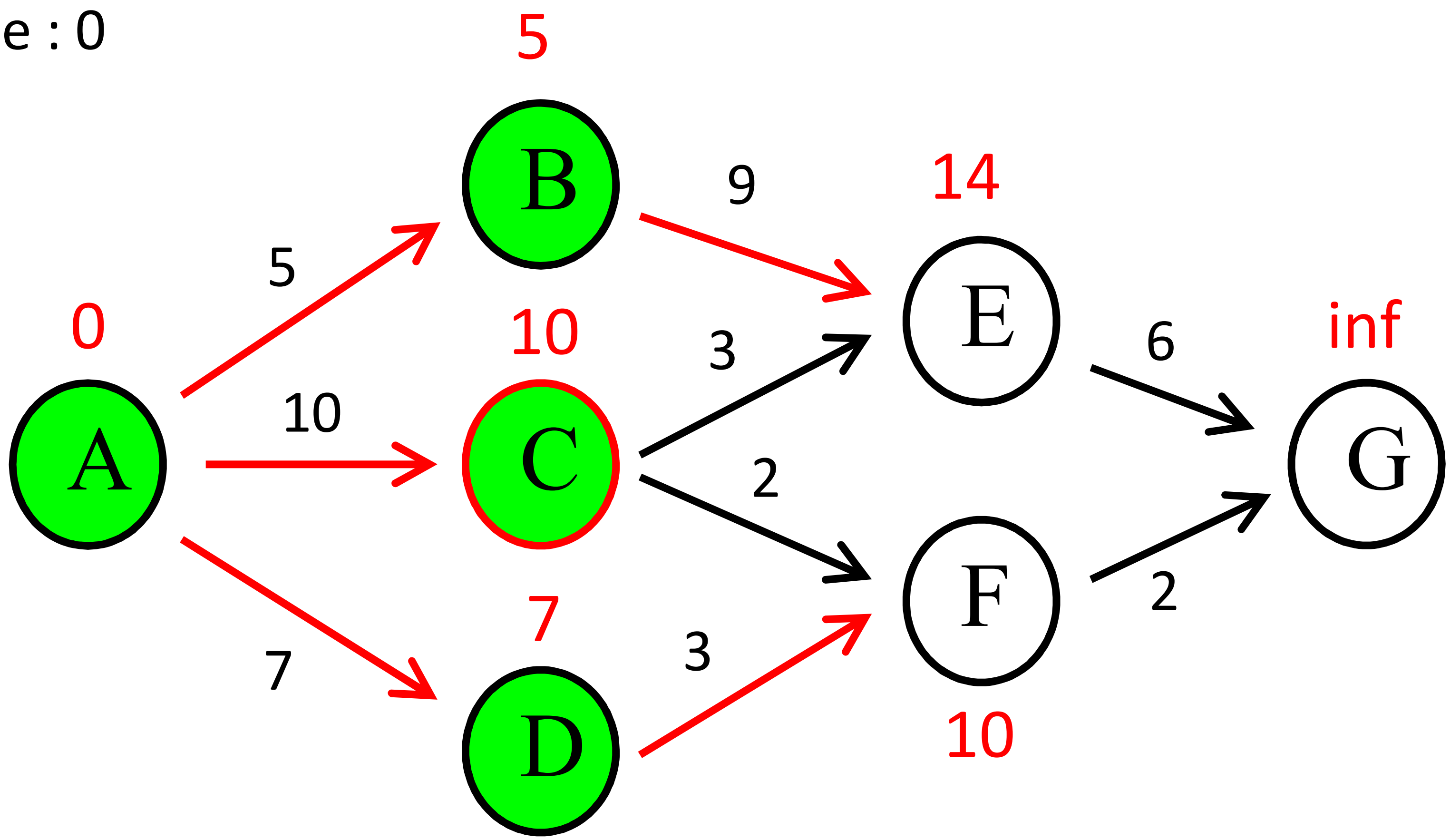
Source : 0



S : { A, B, D }

Dijkstra’s Algorithm

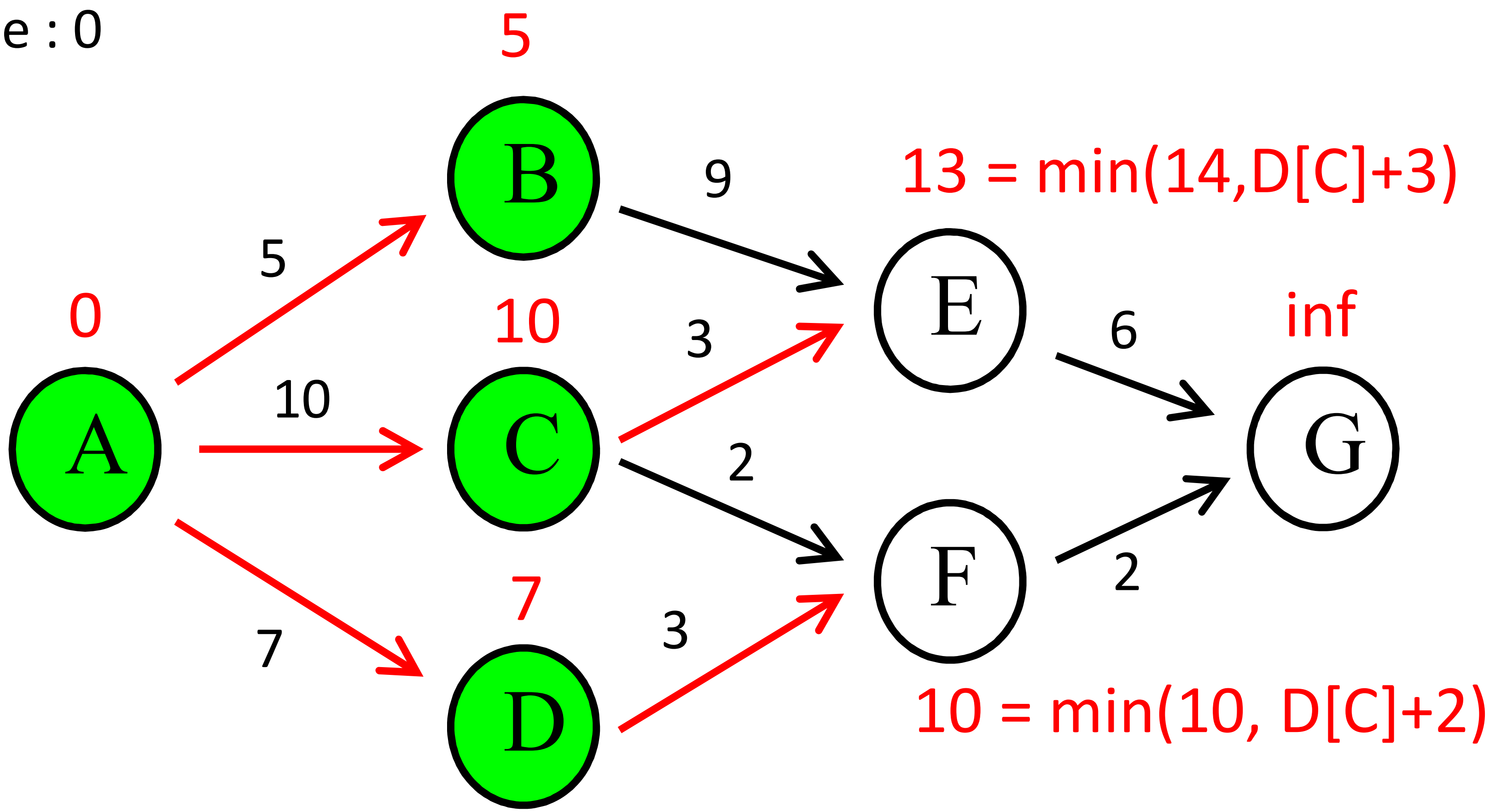
Source : 0



S : { A, B, C, D }

Dijkstra's Algorithm

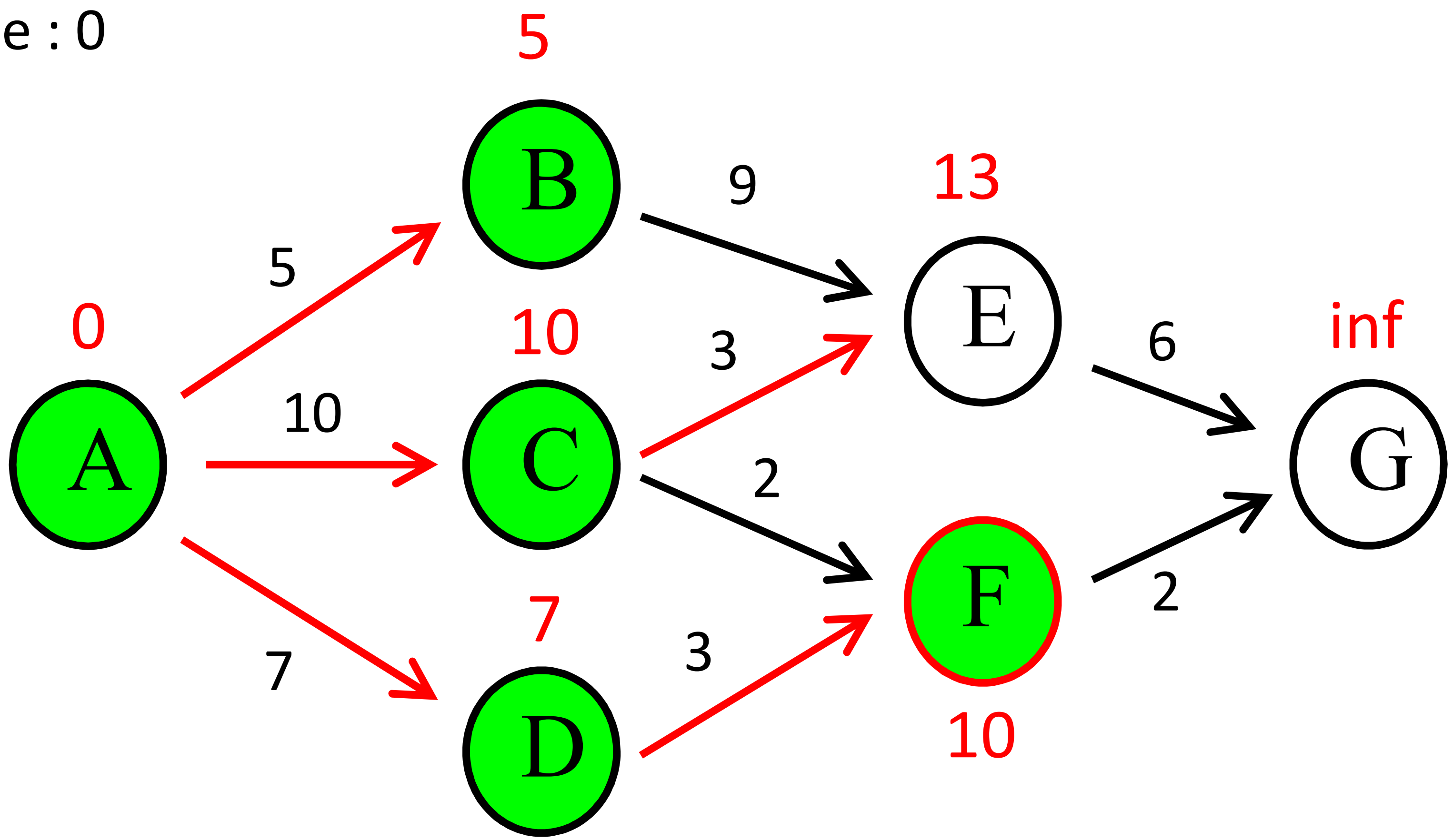
Source : 0



S : { A, B, C, D }

Dijkstra's Algorithm

Source : 0

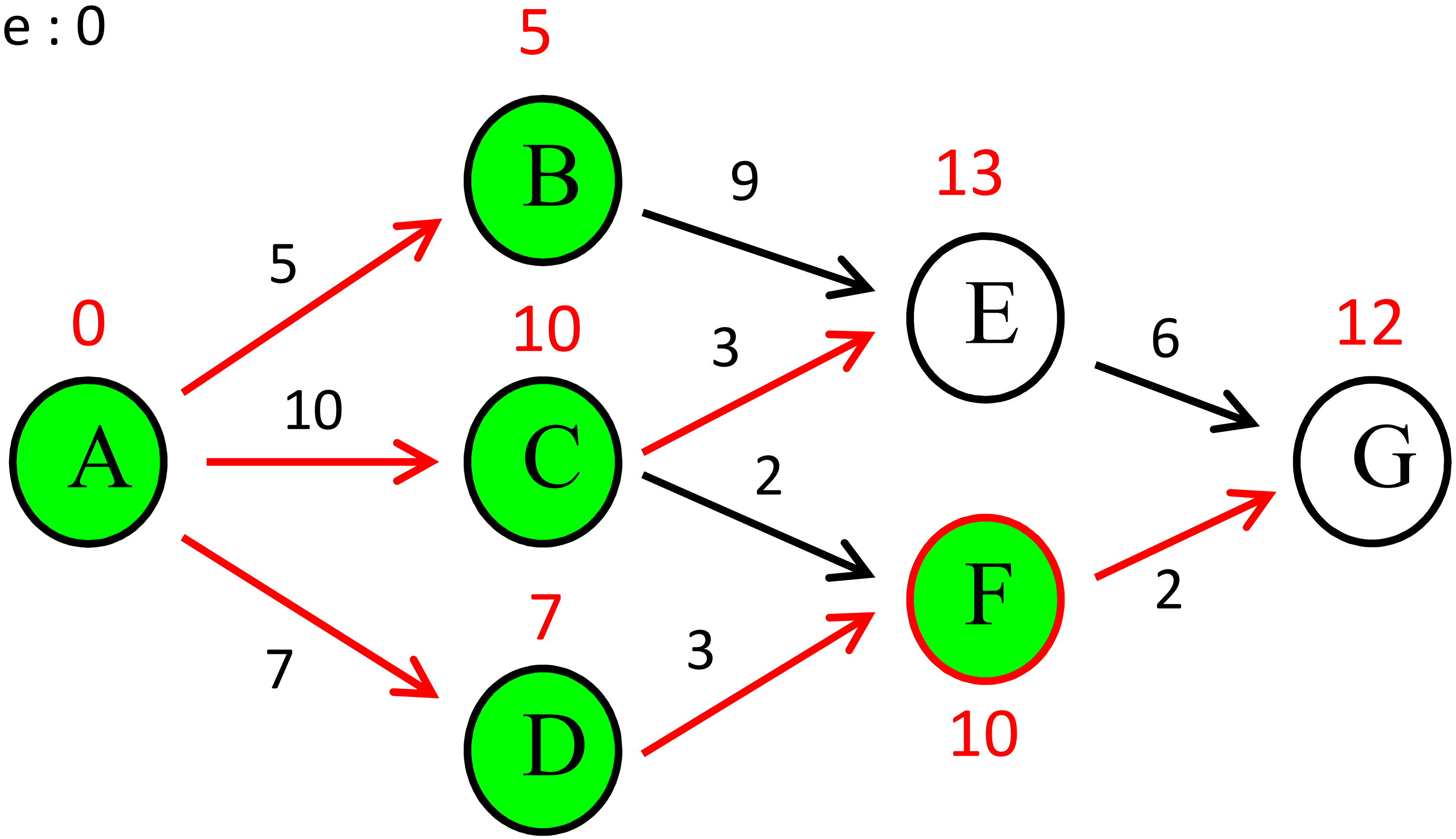


S : { A, B, C, D, F }

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Dijkstra's Algorithm

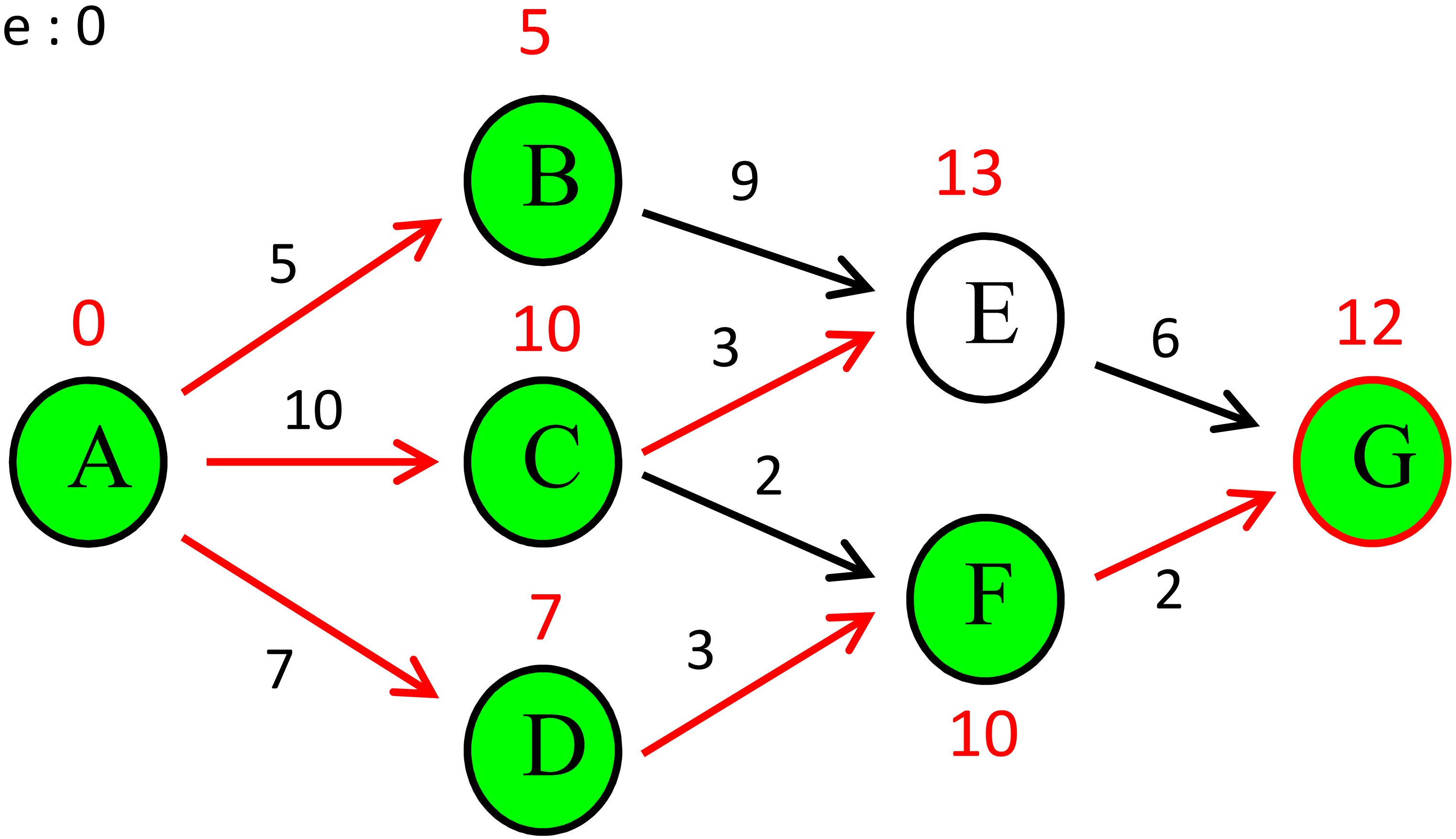
Source : 0



S : { A, B, C, D, F }

Dijkstra's Algorithm

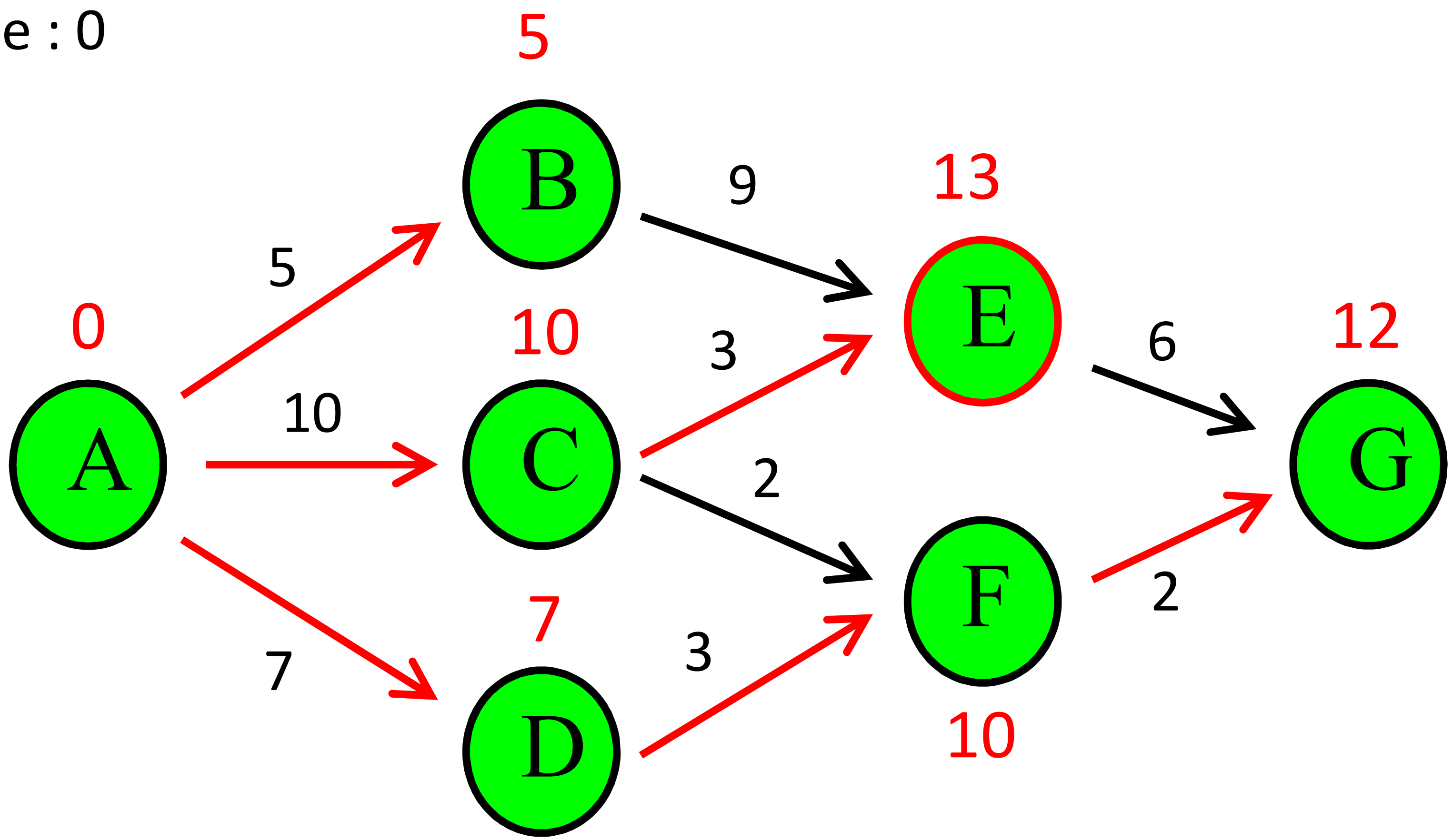
Source : 0



S : { A, B, C, D, F, G }

Dijkstra's Algorithm

Source : 0



S : { A, B, C, D, E, F, G }

Algorithm : Linear Search

```
temp = {}, S = {}
for all vertices v
    d(v) = inf
d(source) = 0
Put all vertices to temp
while temp is not empty    : n
    v ← d(v) is min in temp : n
    add v to S
    for all neighbor u of v    : # neighbor
        if d(u) > d(v) + length(v,u)
            d(u) = d(v) + length(v,u)
```

$O(m + n^2) = O(n^2)$ for linear search

Algorithm : Min Heap

```
temp = {}, S = {}
for all vertices v
    d(v) = inf
d(source) = 0
Put all vertices to temp
while temp is not empty   : n
    v ← d(v) is min in temp : log n
    add v to S
    for all neighbor u of v   : # neighbor
        if d(u) > d(v) + length(v,u)
            d(u) = d(v) + length(v,u) : log n
```

O(n log(n) + m log(n)) for min heap

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

Questions?