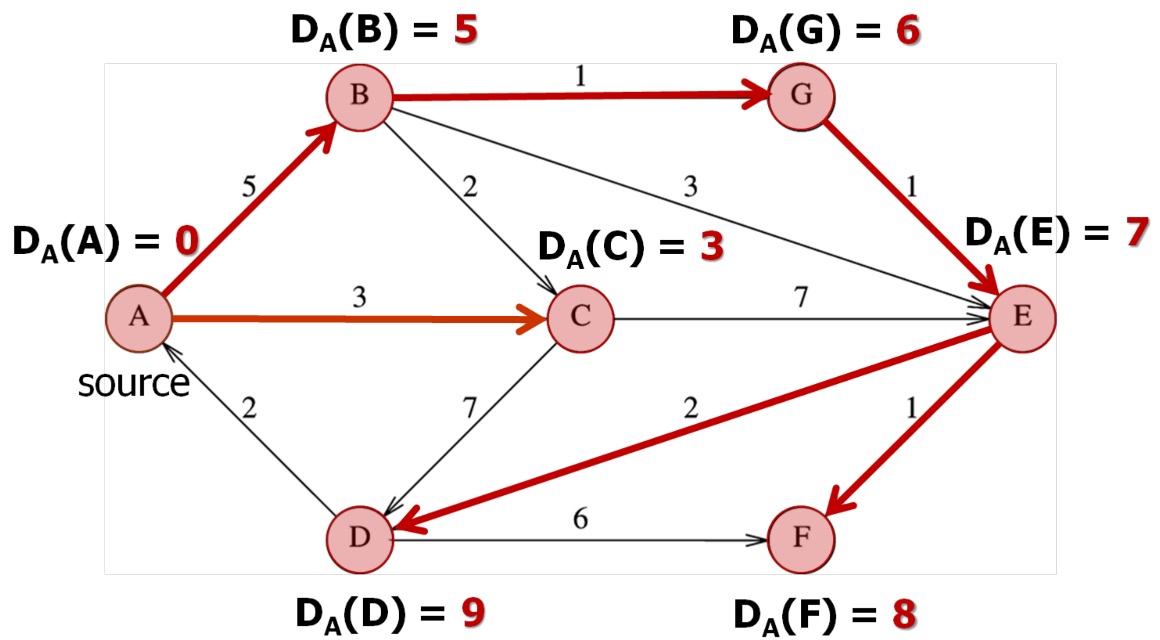
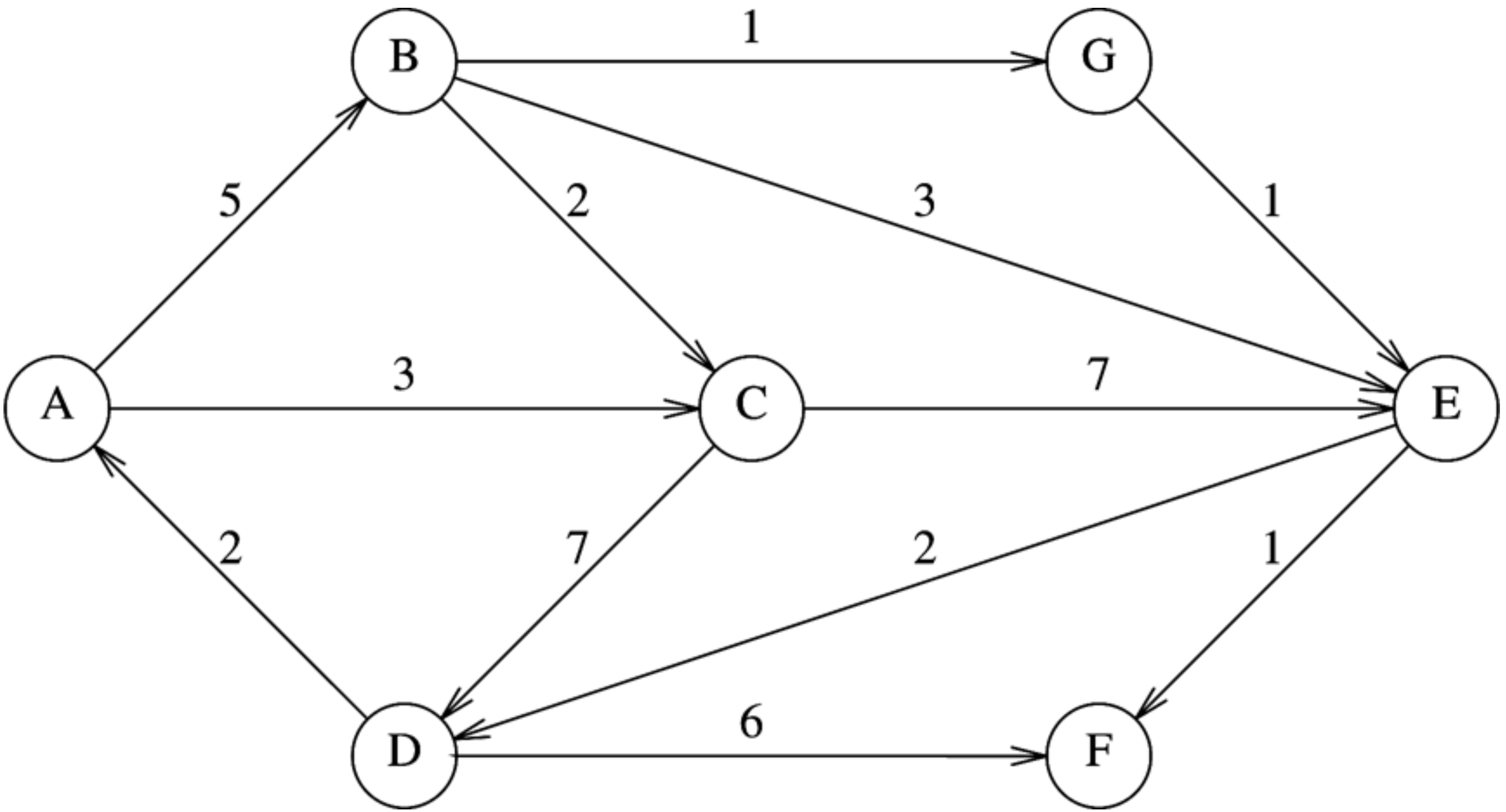
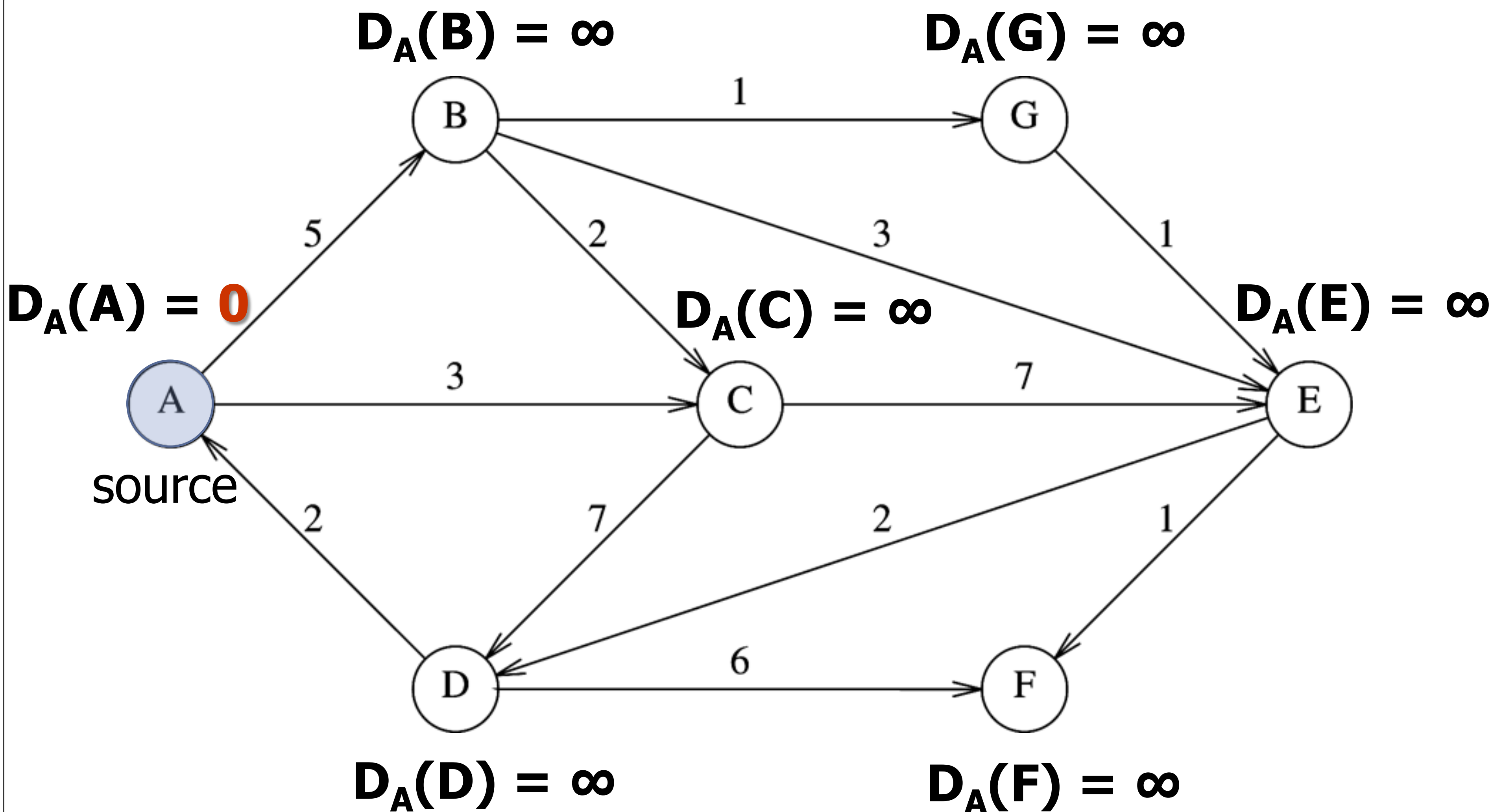
Dijkstra's Algorithm

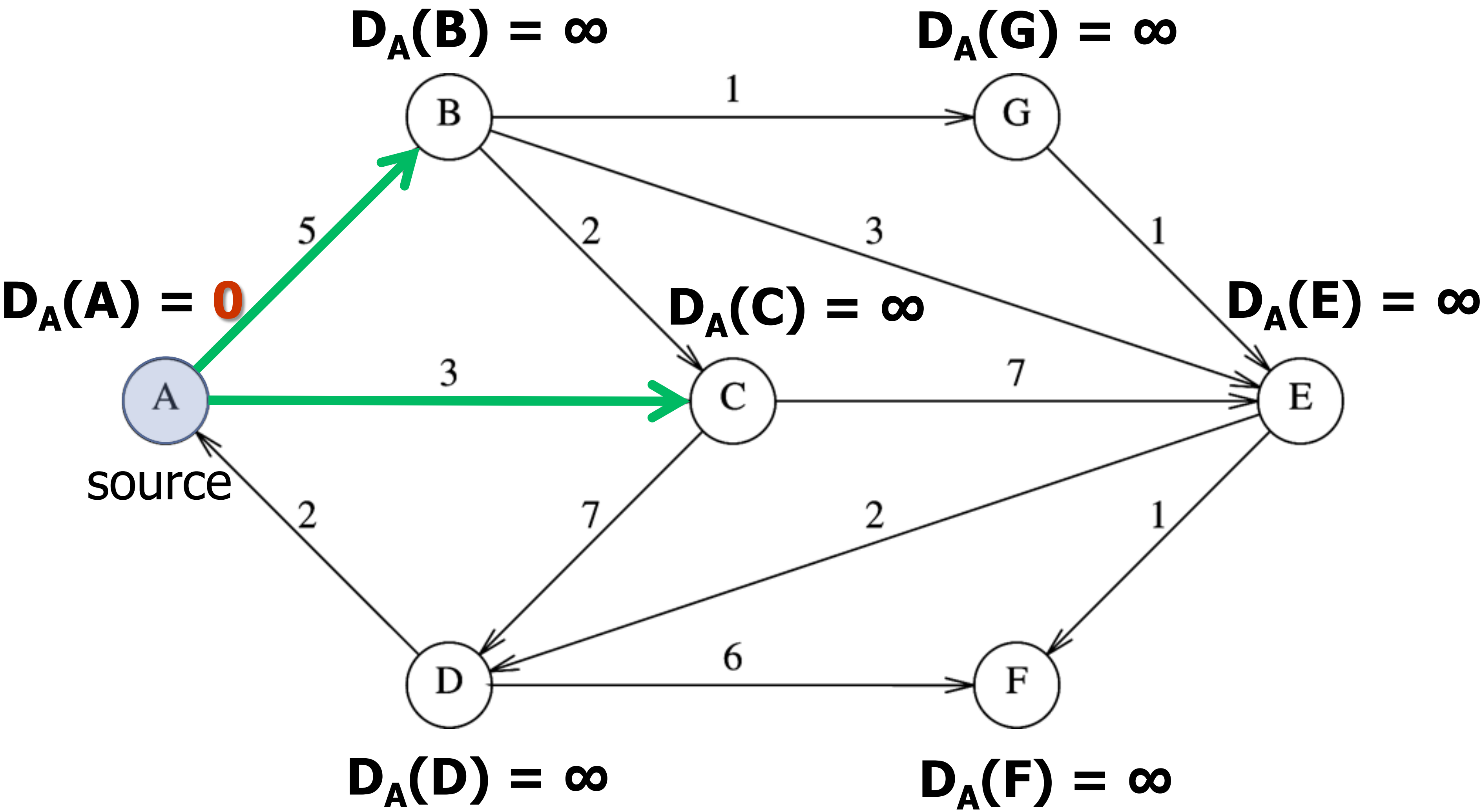


Directed Weighted Graph

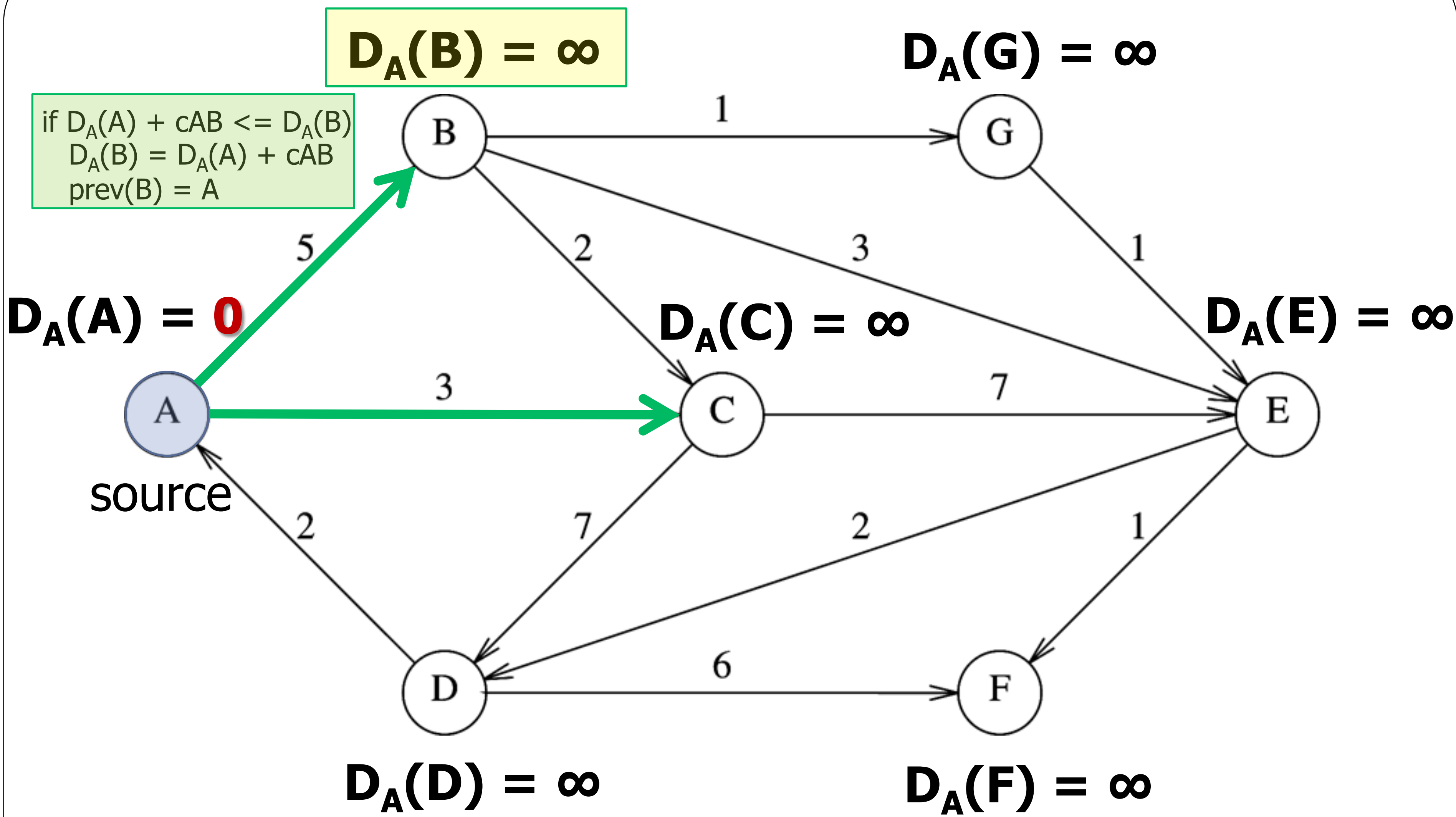




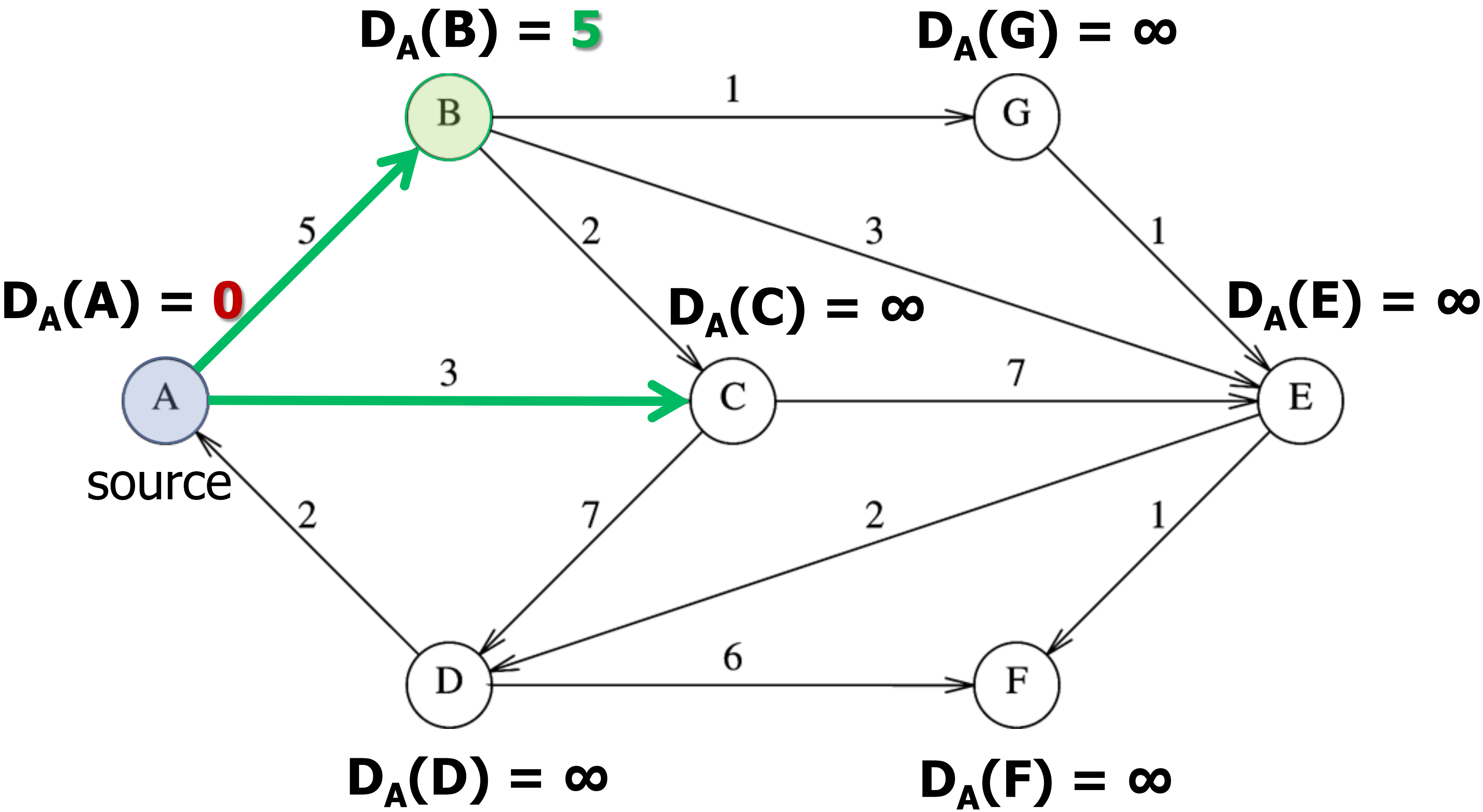
Weight f(n) = D_{source}(Destination)



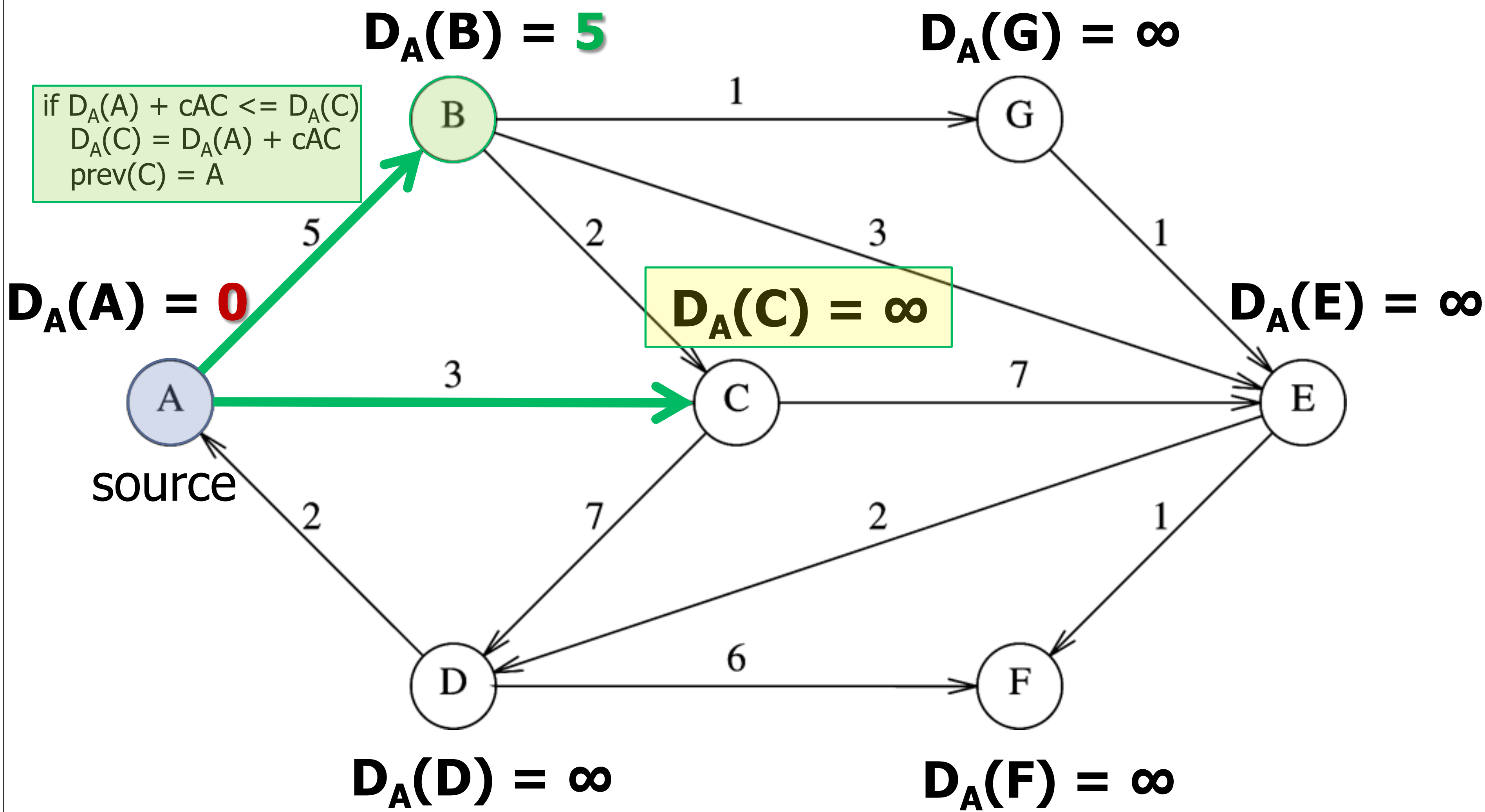
Weight $f(n) = D_{source}(Destination)$



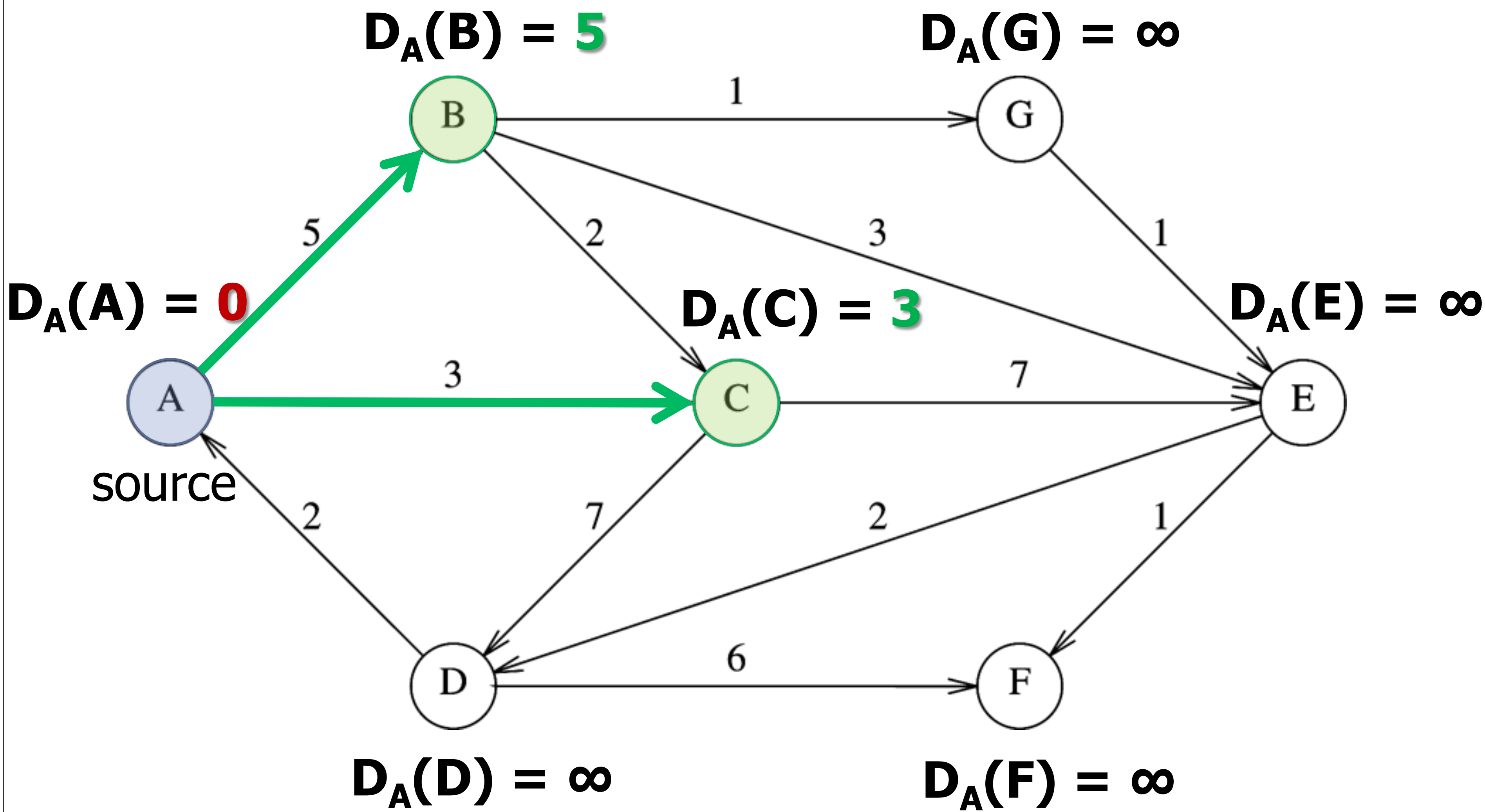
Weight f(n) = D_{source}(Destination)



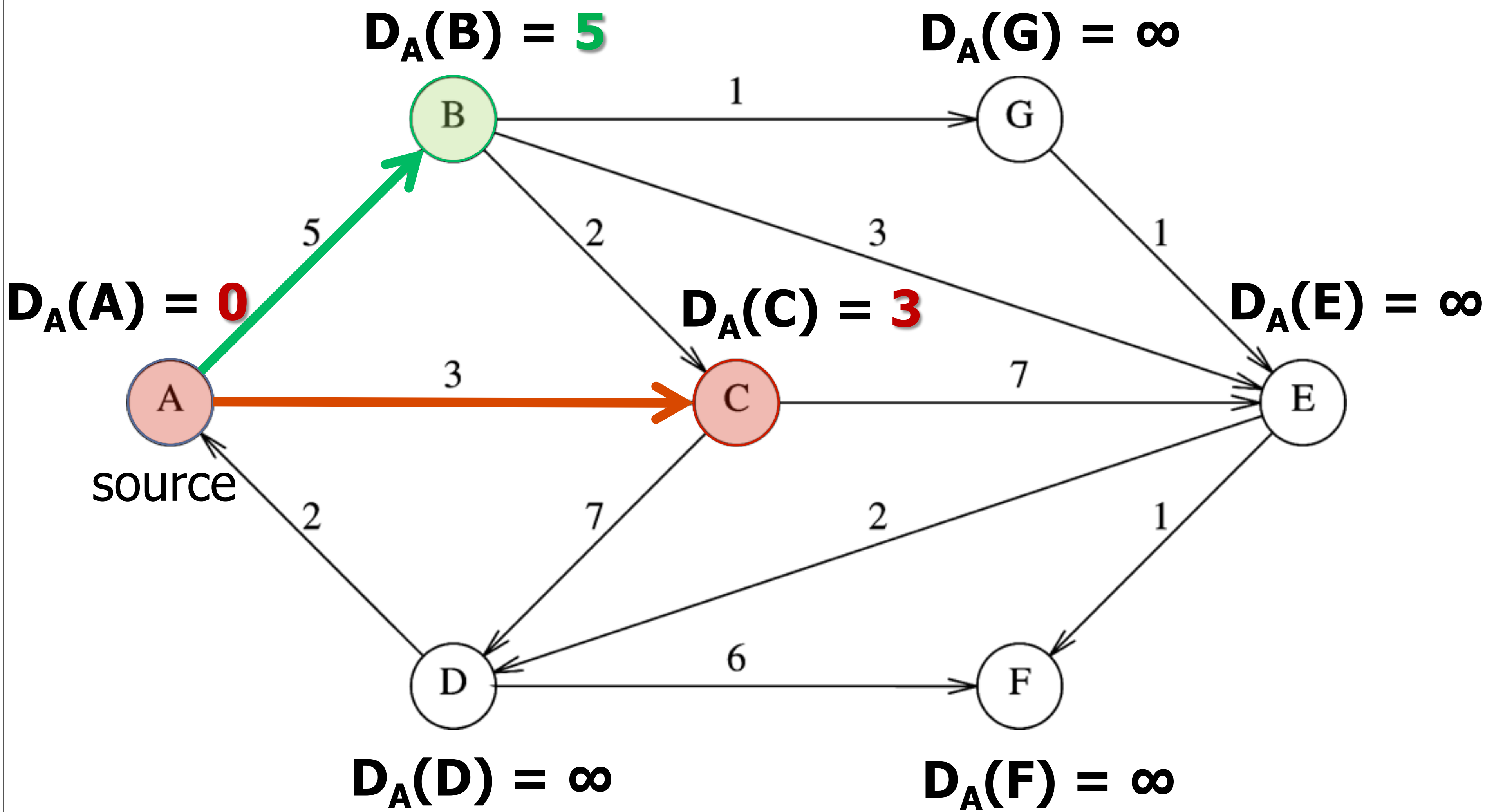
Weight $f(n) = D_{source}(Destination)$



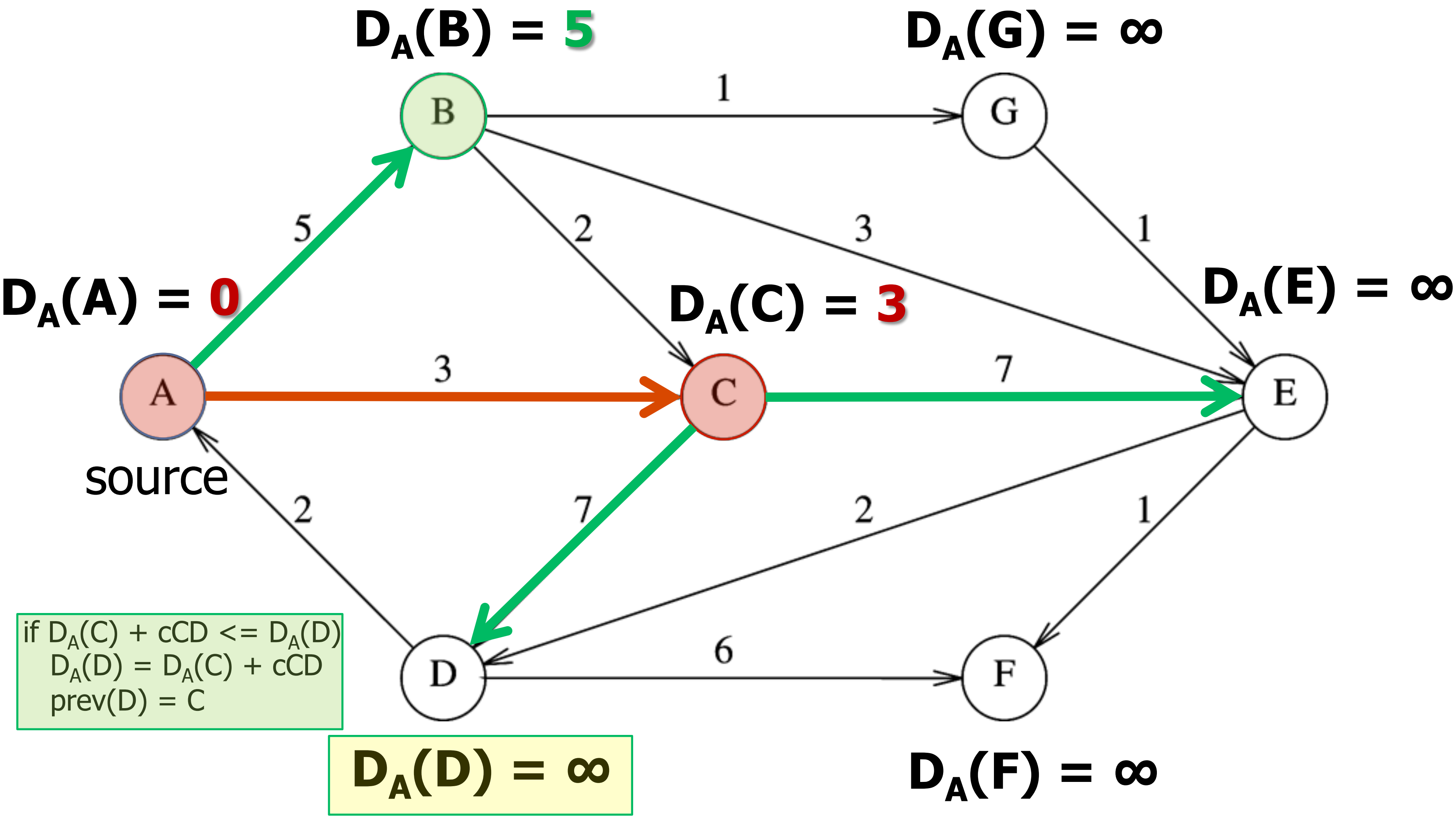
Weight $f(n) = D_{source}(Destination)$



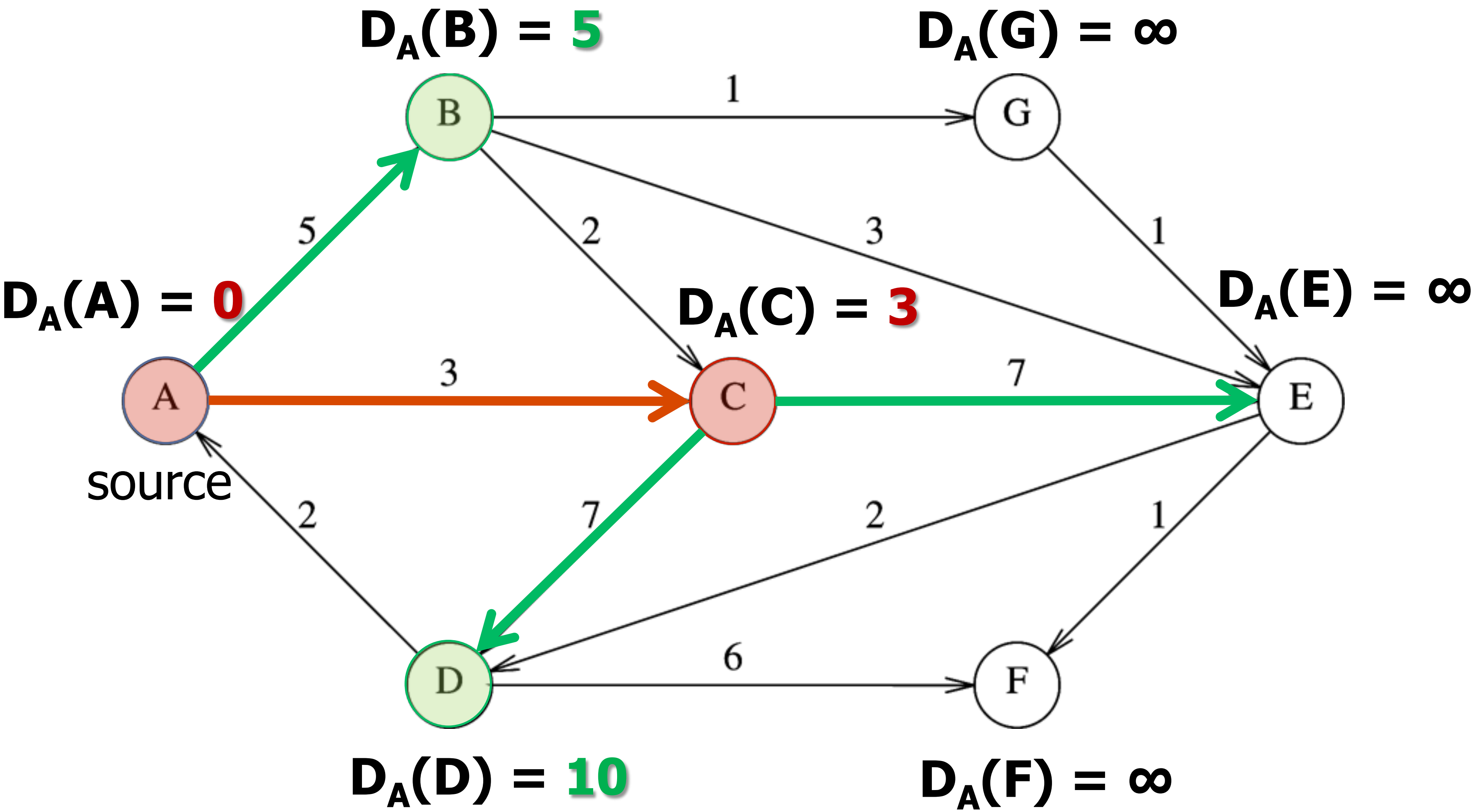
Weight f(n) = D_{source}(Destination)



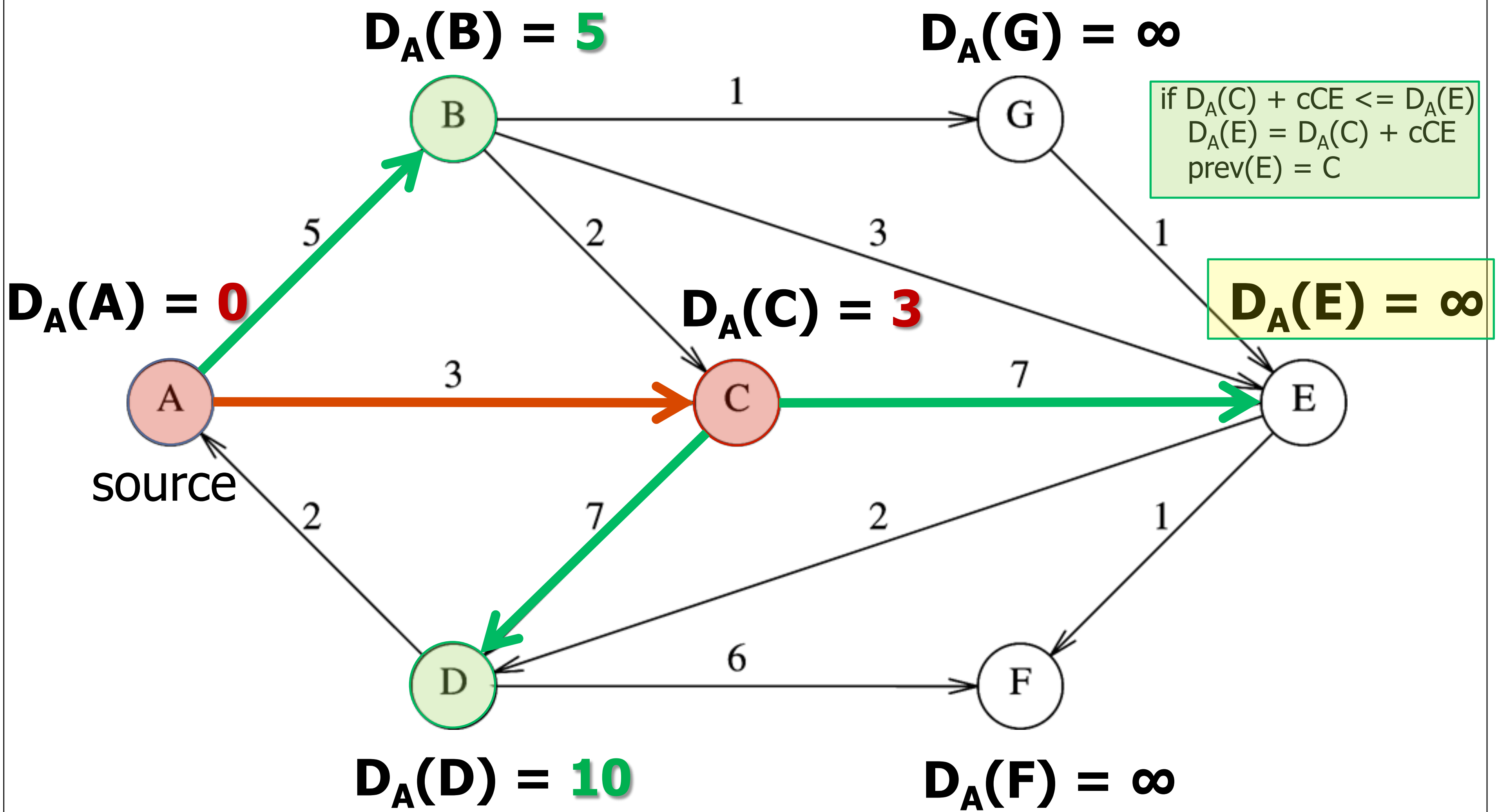
Weight $f(n) = D_{source}(Destination)$



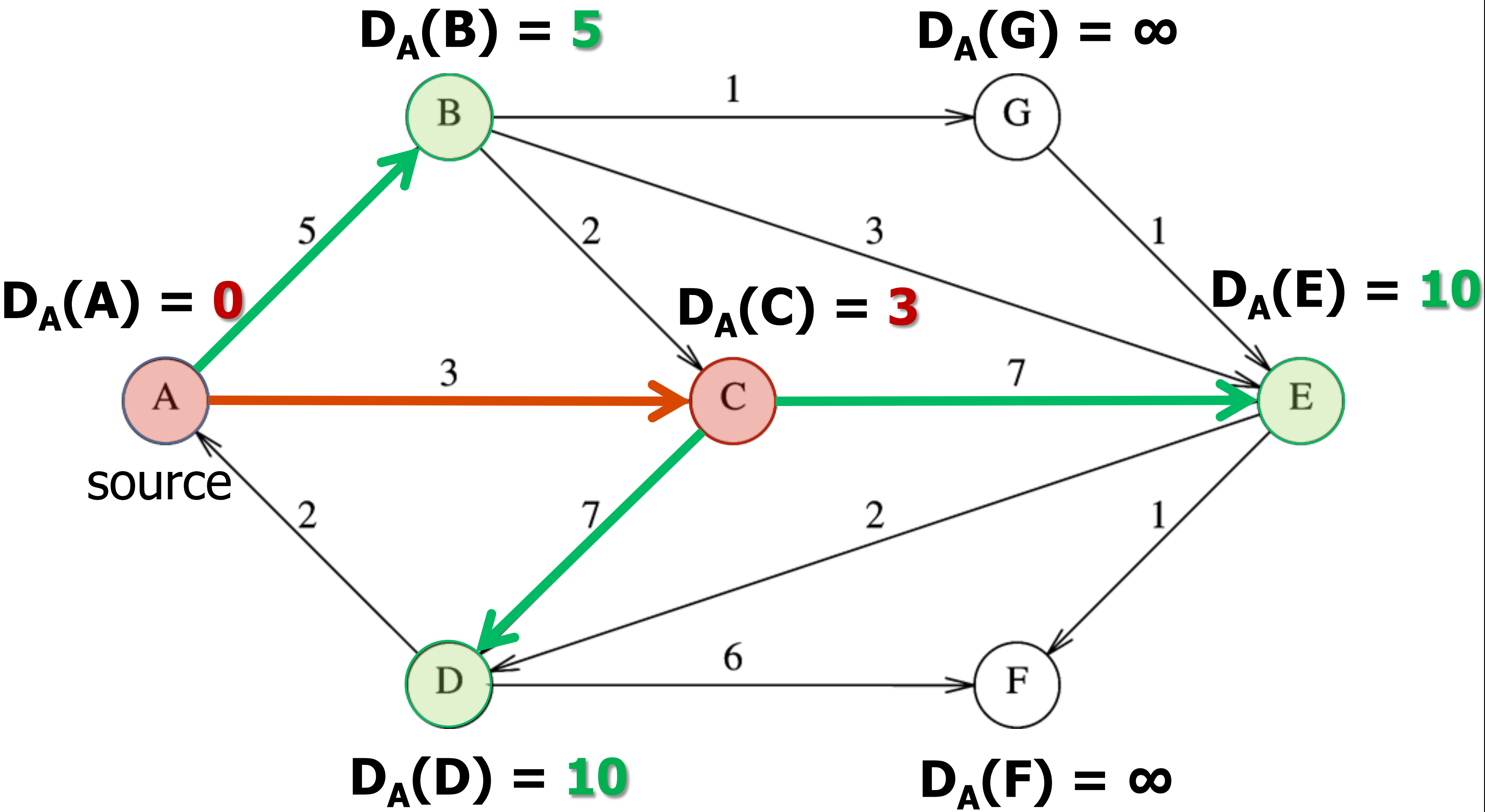
Weight $f(n) = D_{source}(Destination)$



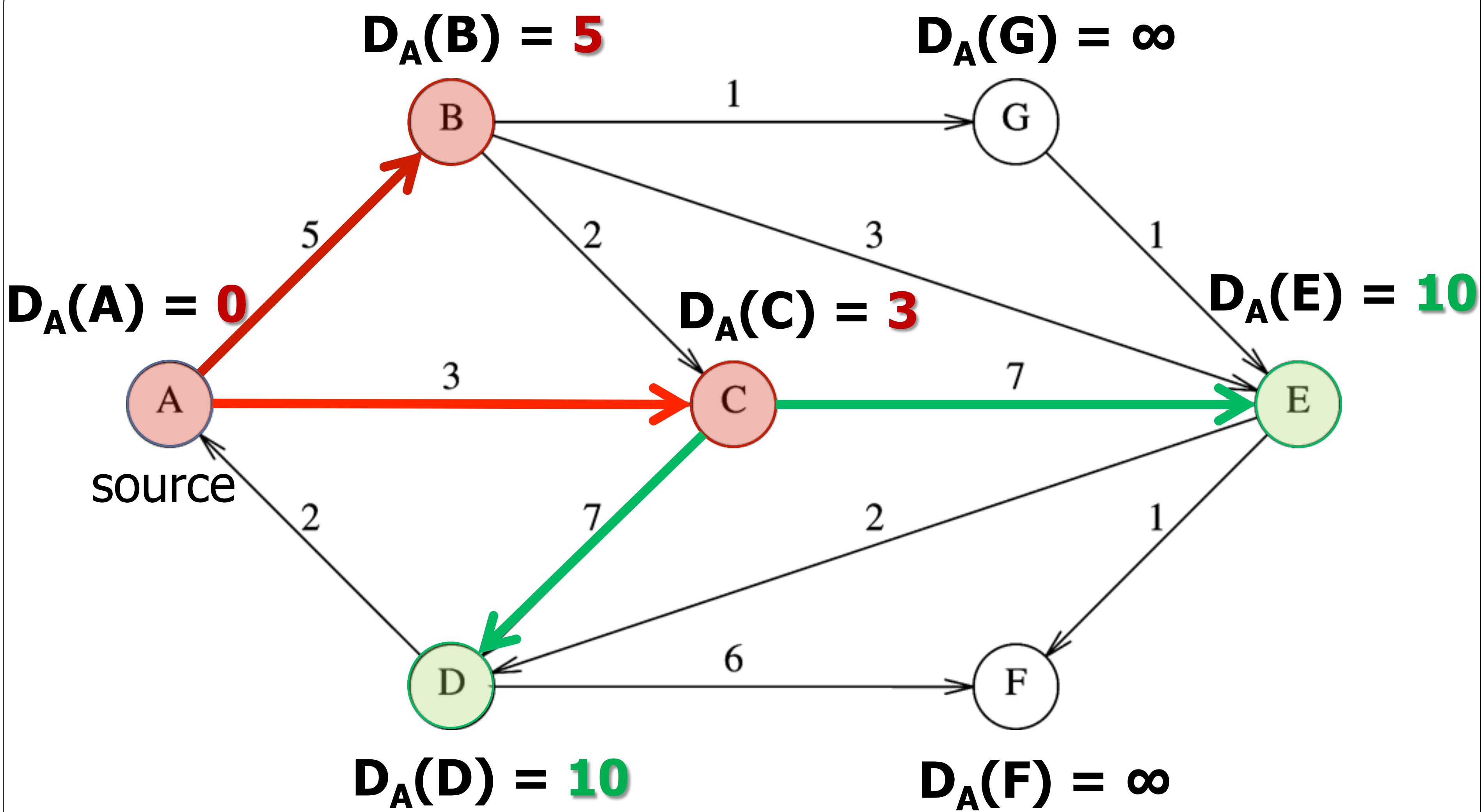
Weight $f(n) = D_{source}(Destination)$



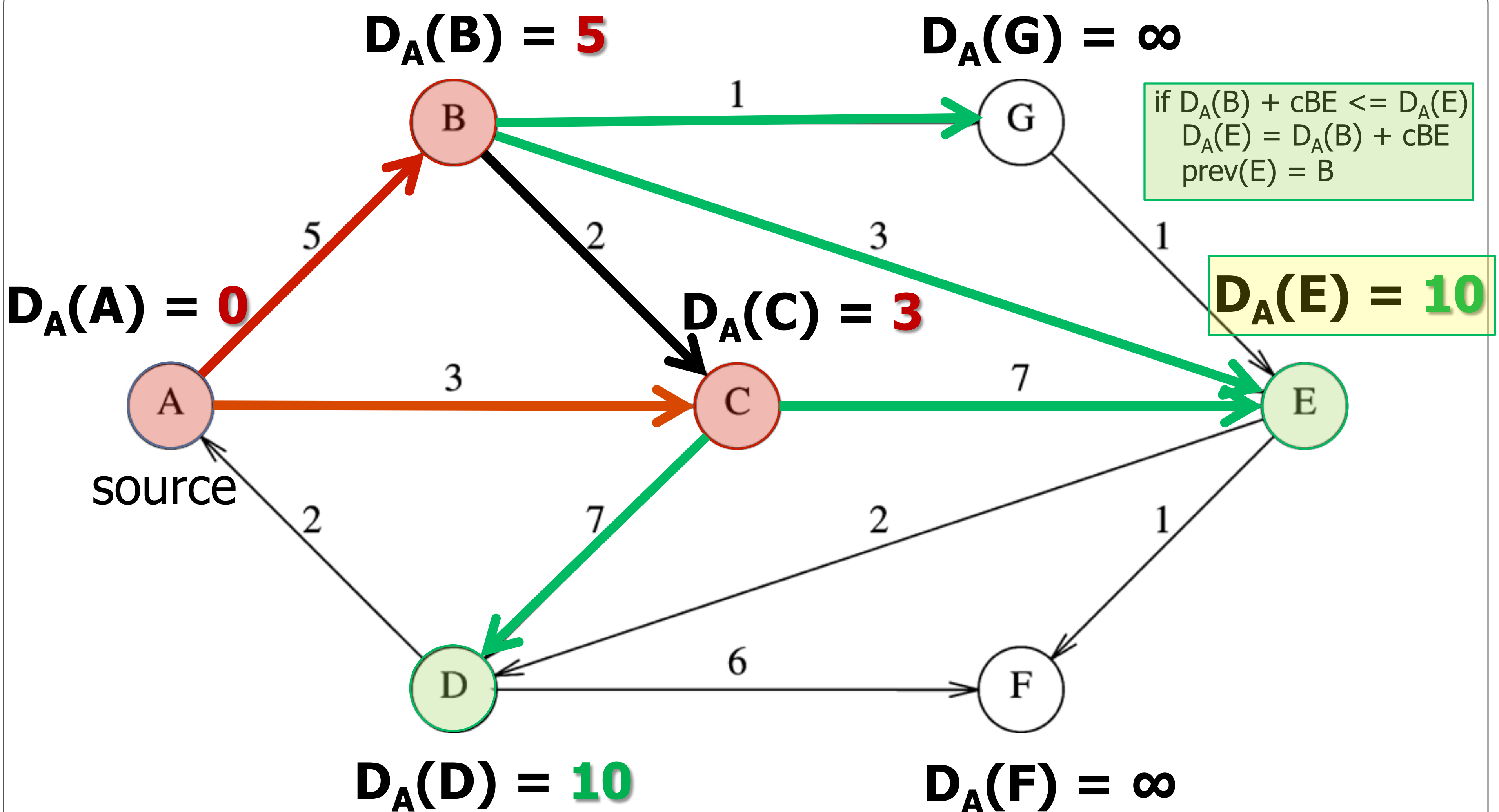
Weight f(n) = D_{source}(Destination)



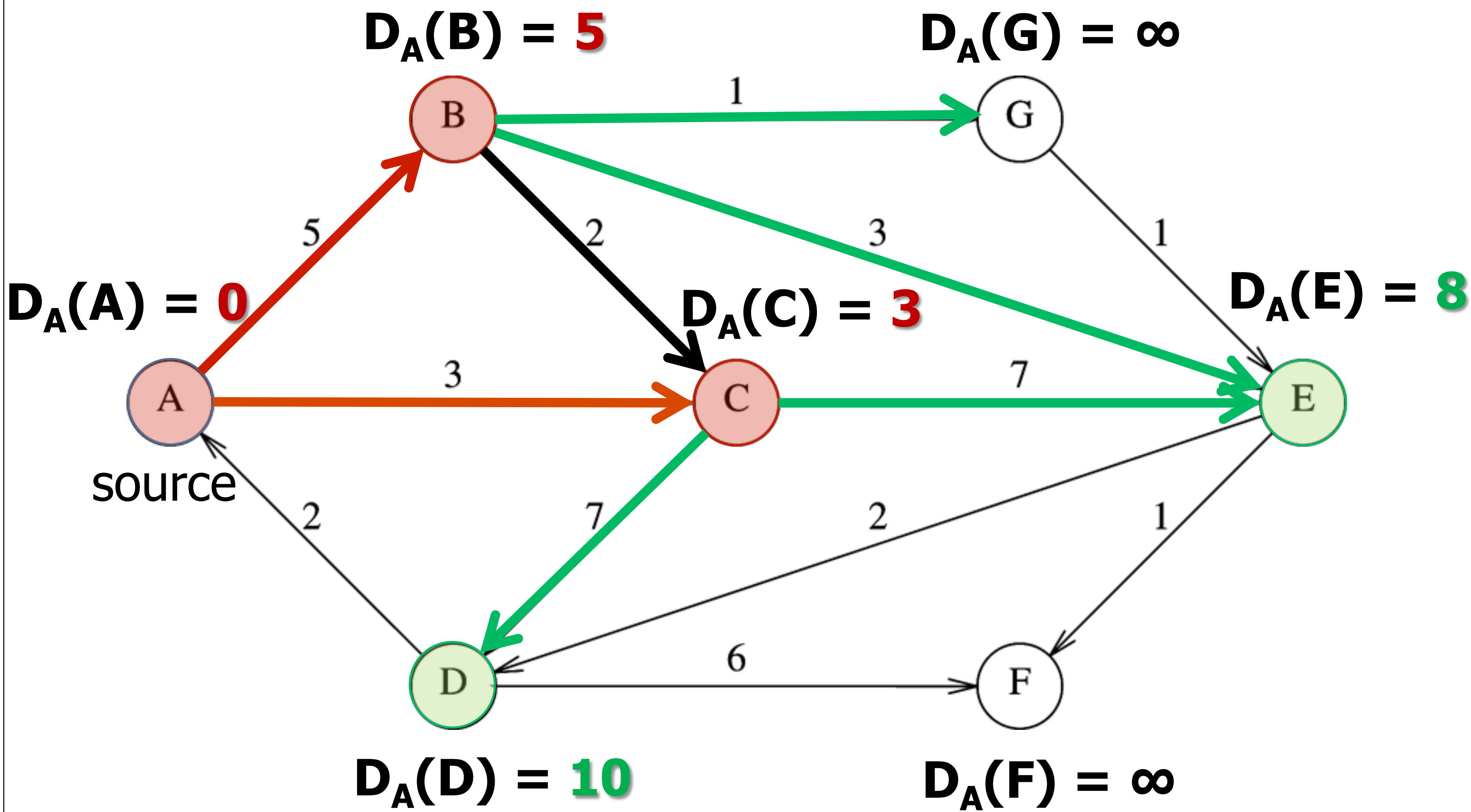
Weight f(n) = D_{source}(Destination)



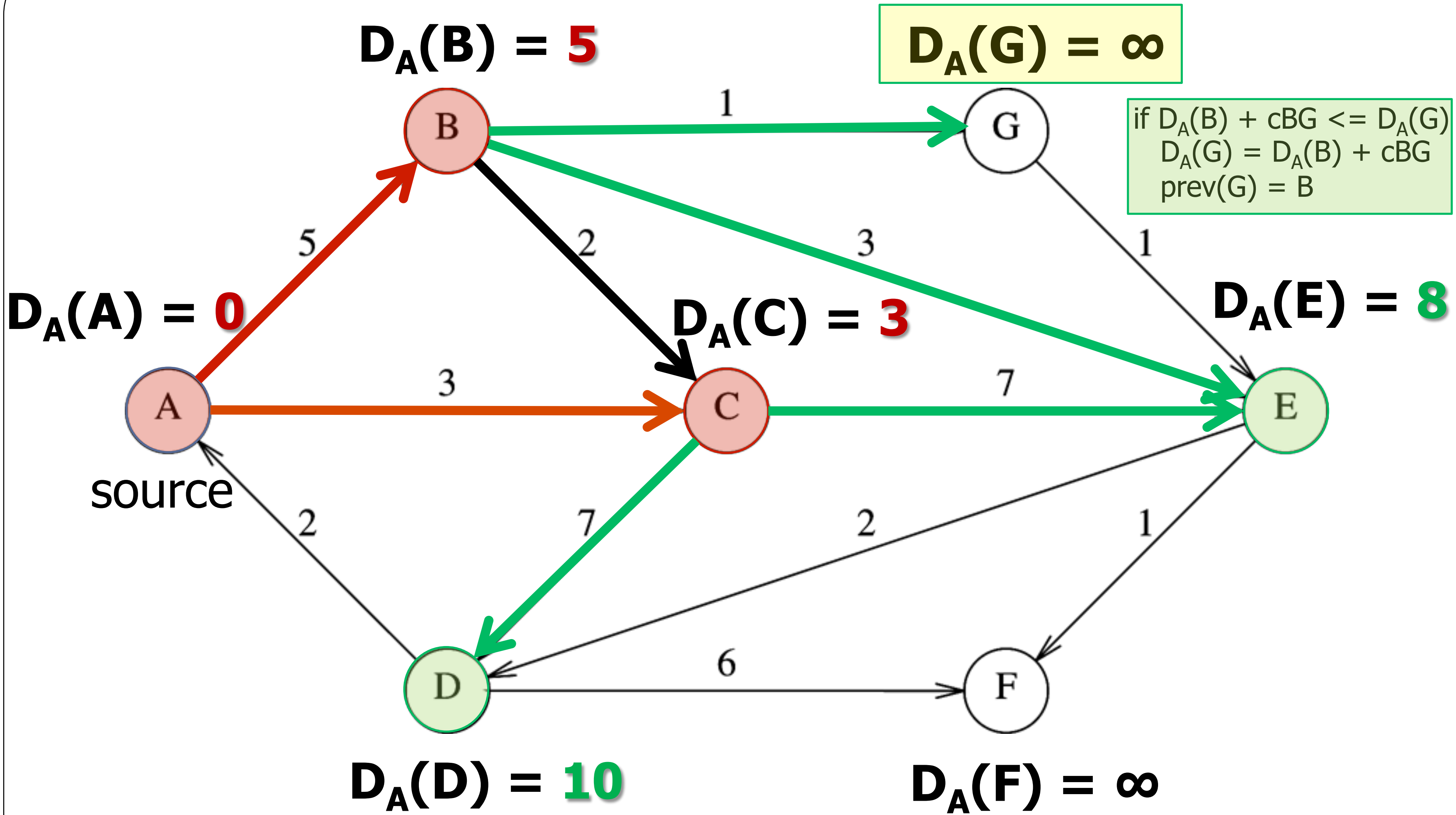
Weight $f(n) = D_{source}(Destination)$



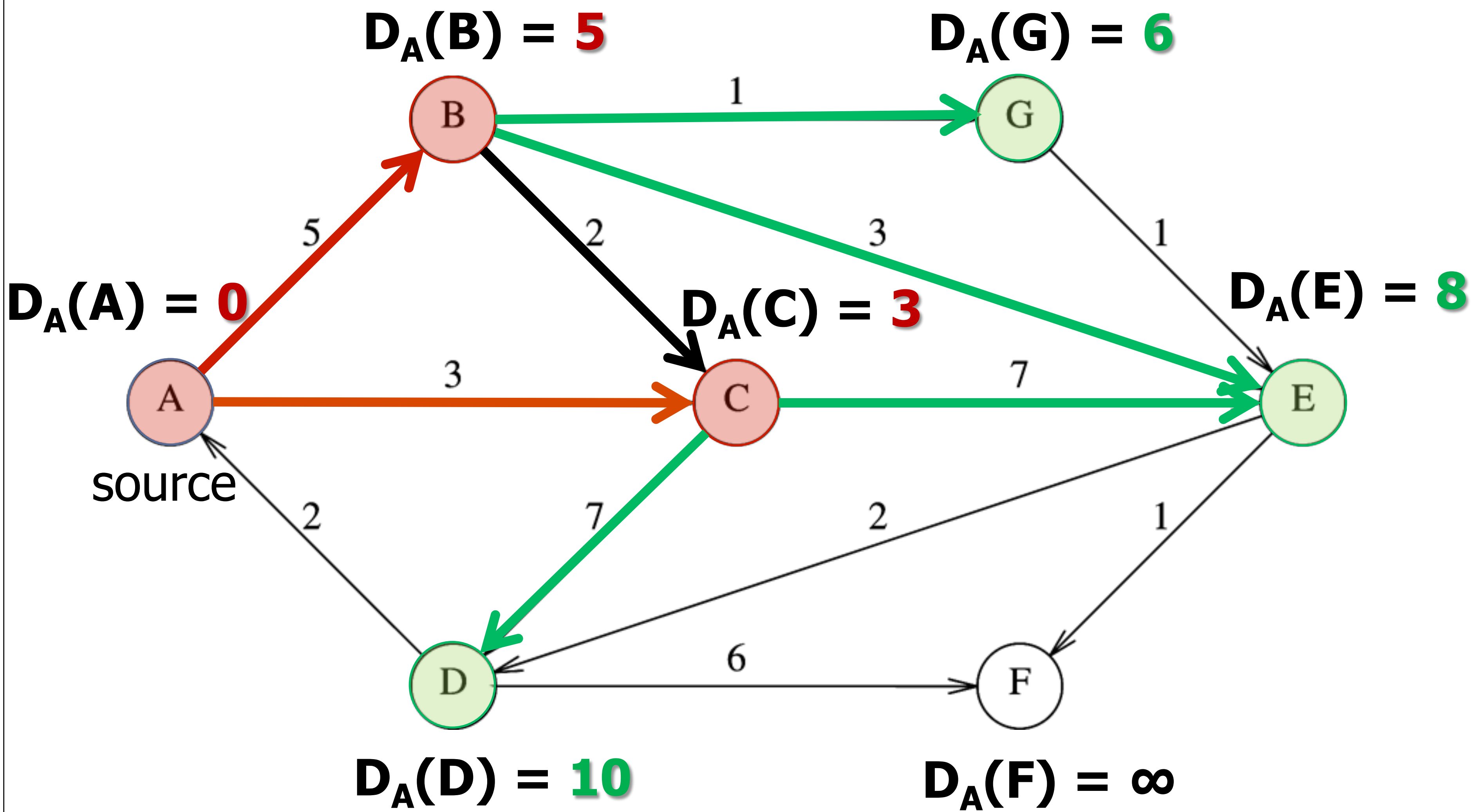
Weight $f(n) = D_{source}(Destination)$



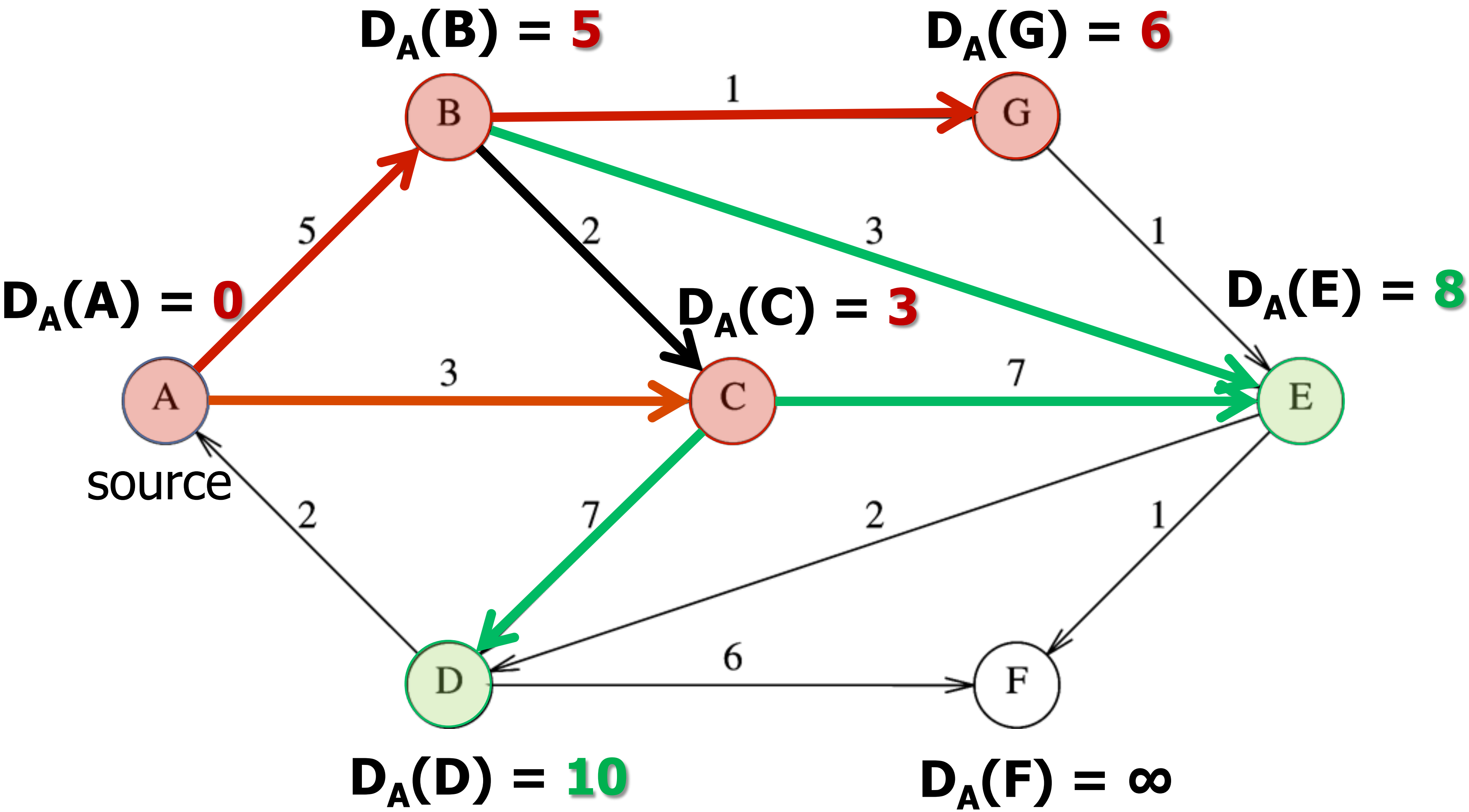
Weight f(n) = D_{source}(Destination)



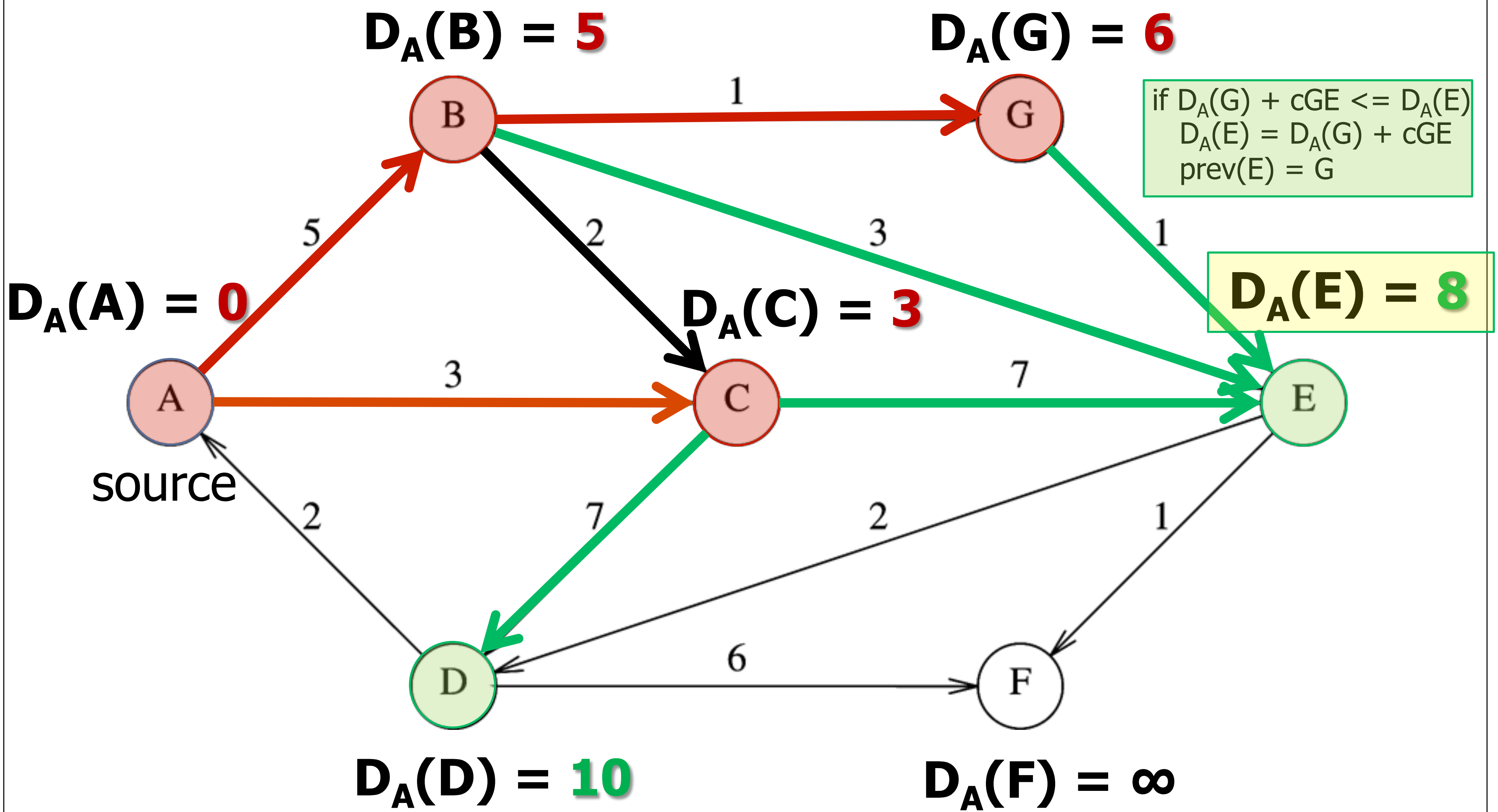
Weight $f(n) = D_{source}(Destination)$



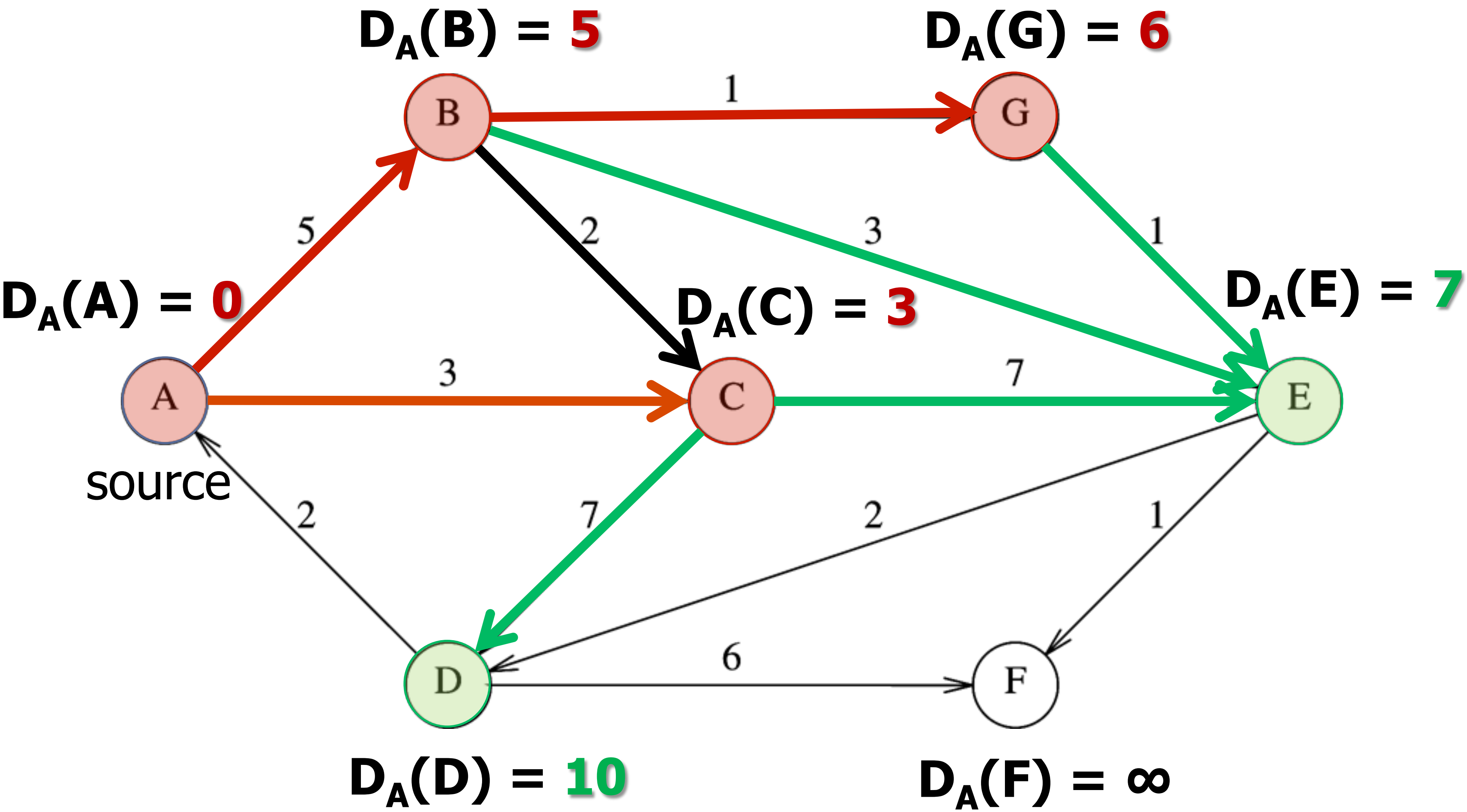
Weight $f(n) = D_{source}(Destination)$



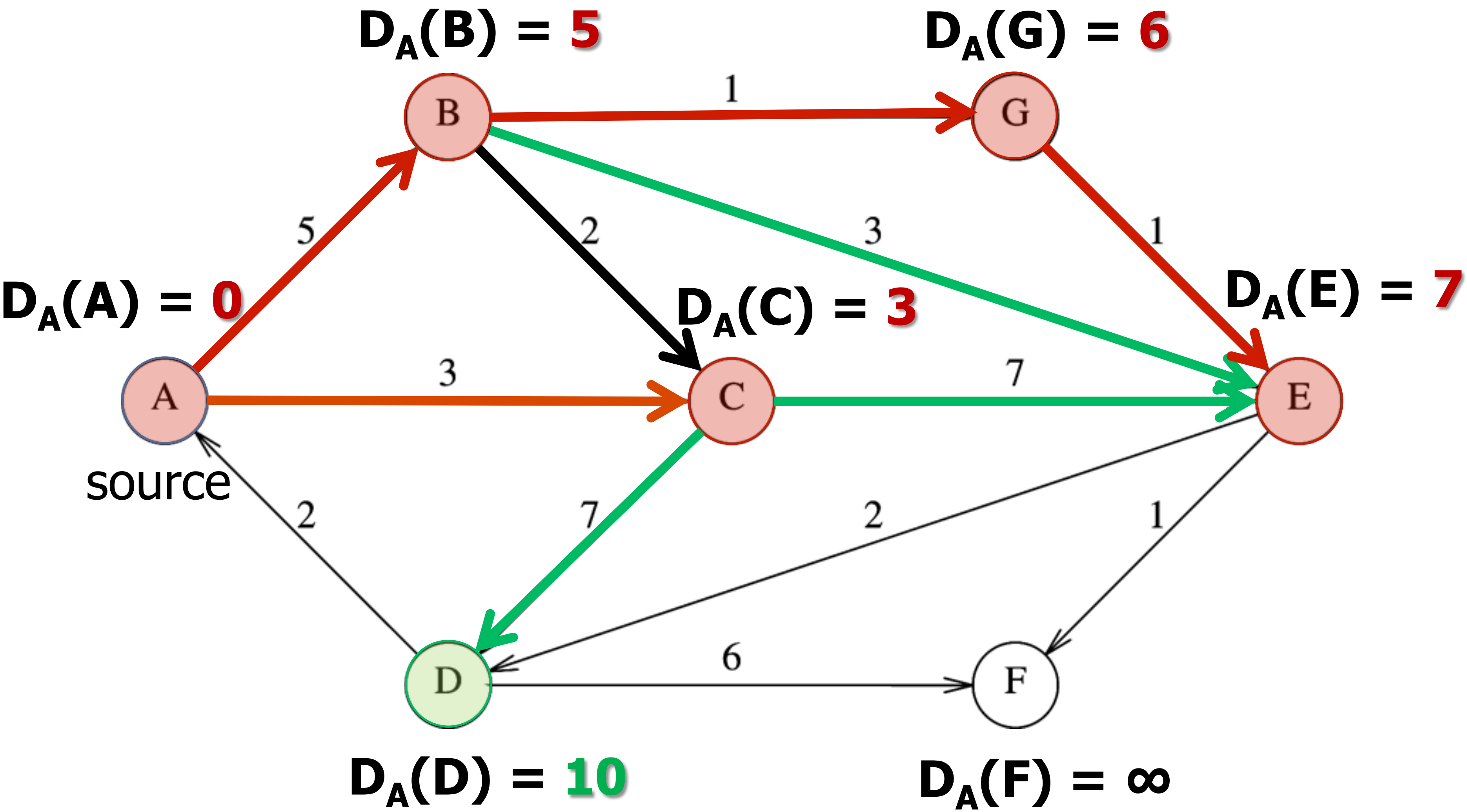
Weight $f(n) = D_{source}(Destination)$



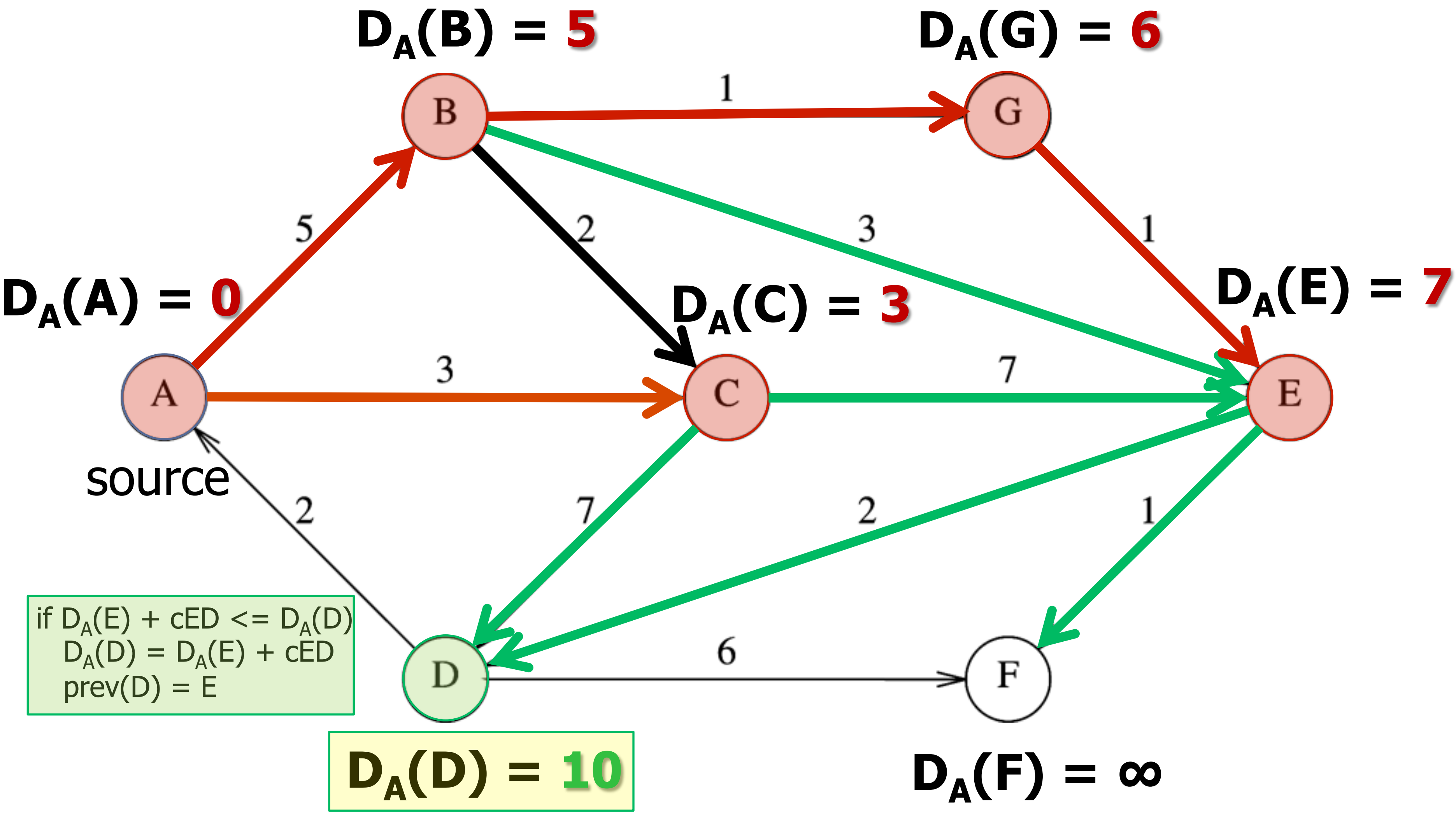
Weight f(n) = D_{source}(Destination)



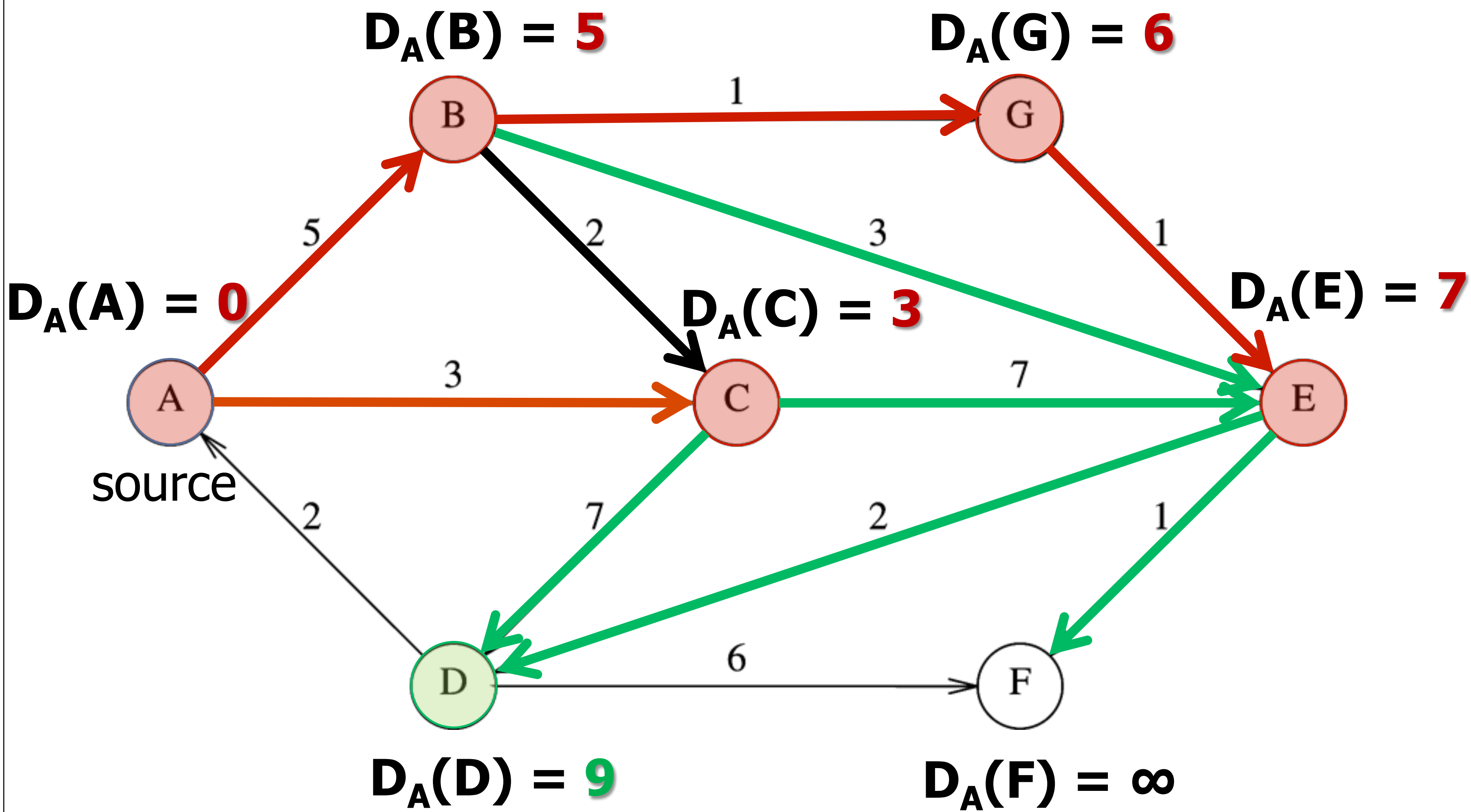
Weight $f(n) = D_{source}(Destination)$



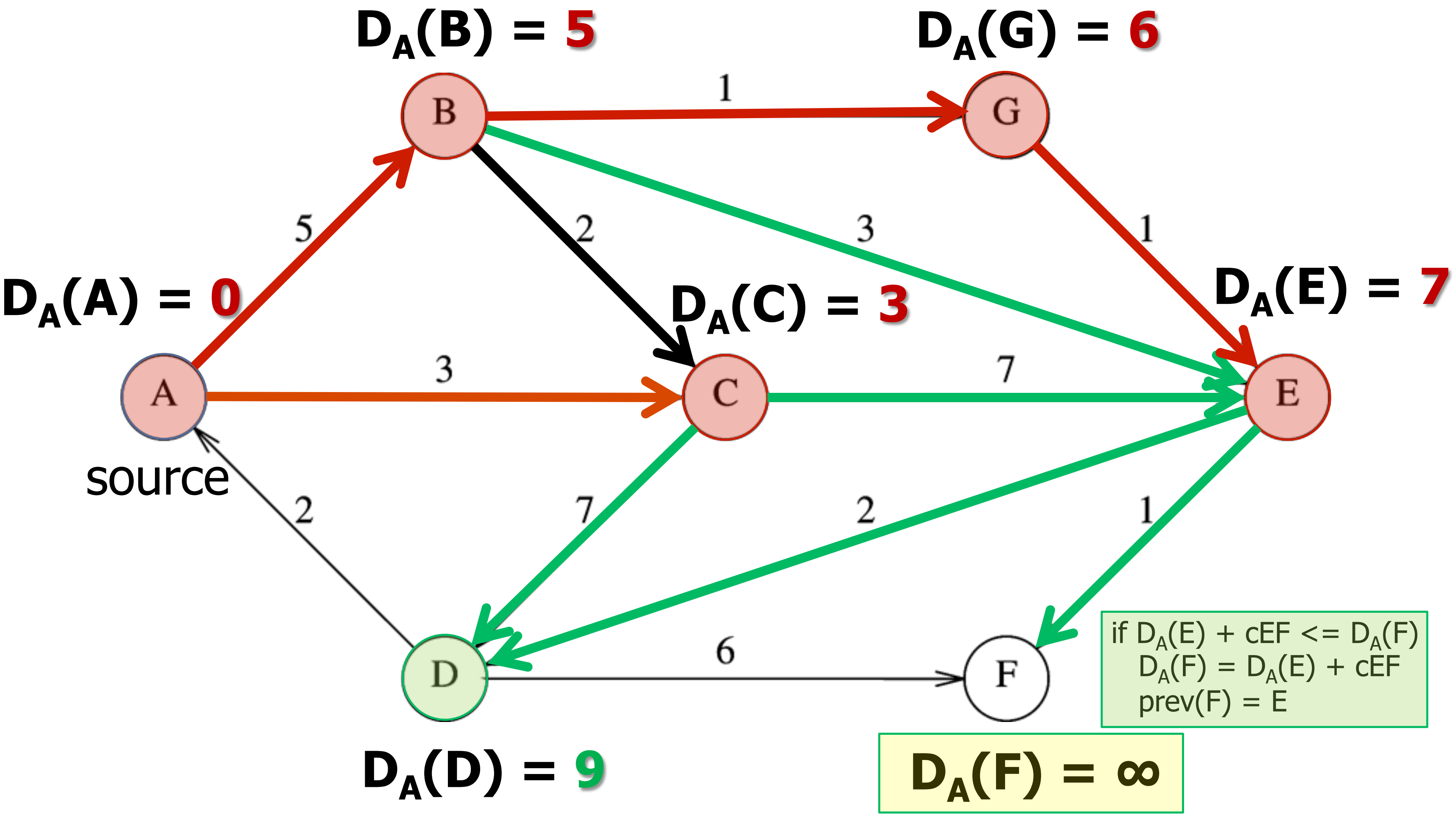
Weight $f(n) = D_{source}(Destination)$



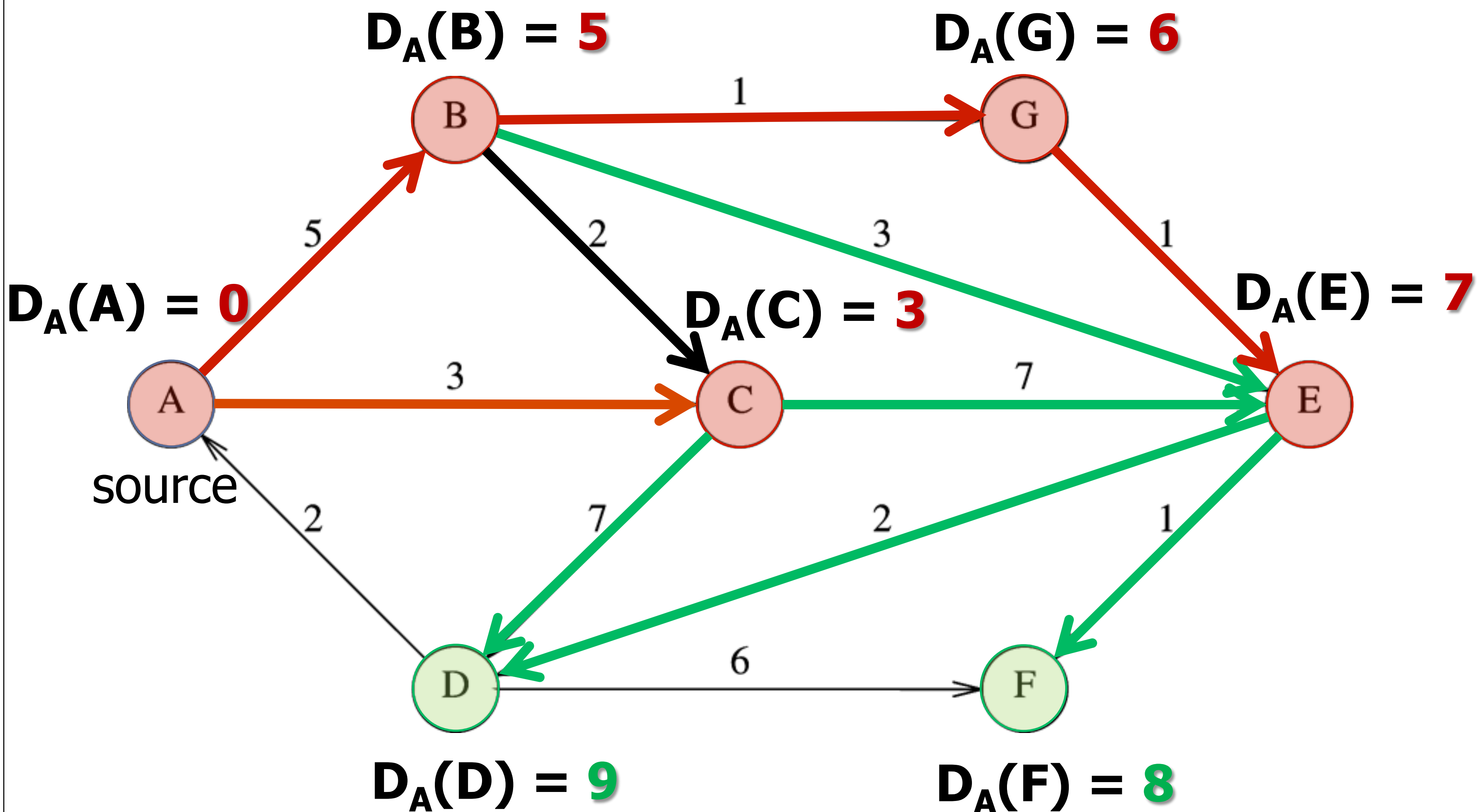
Weight $f(n) = D_{source}(Destination)$



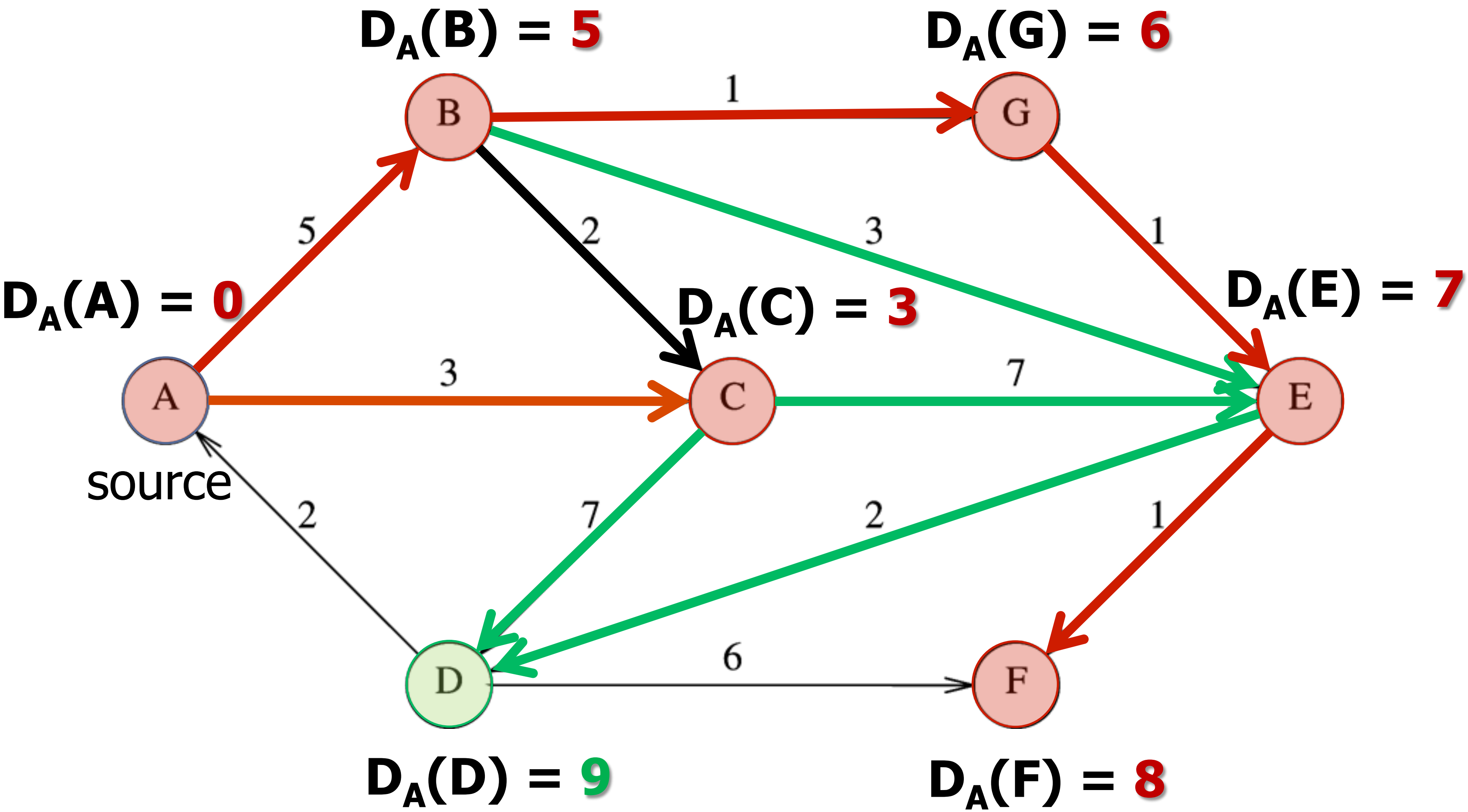
Weight $f(n) = D_{source}(\text{Destination})$



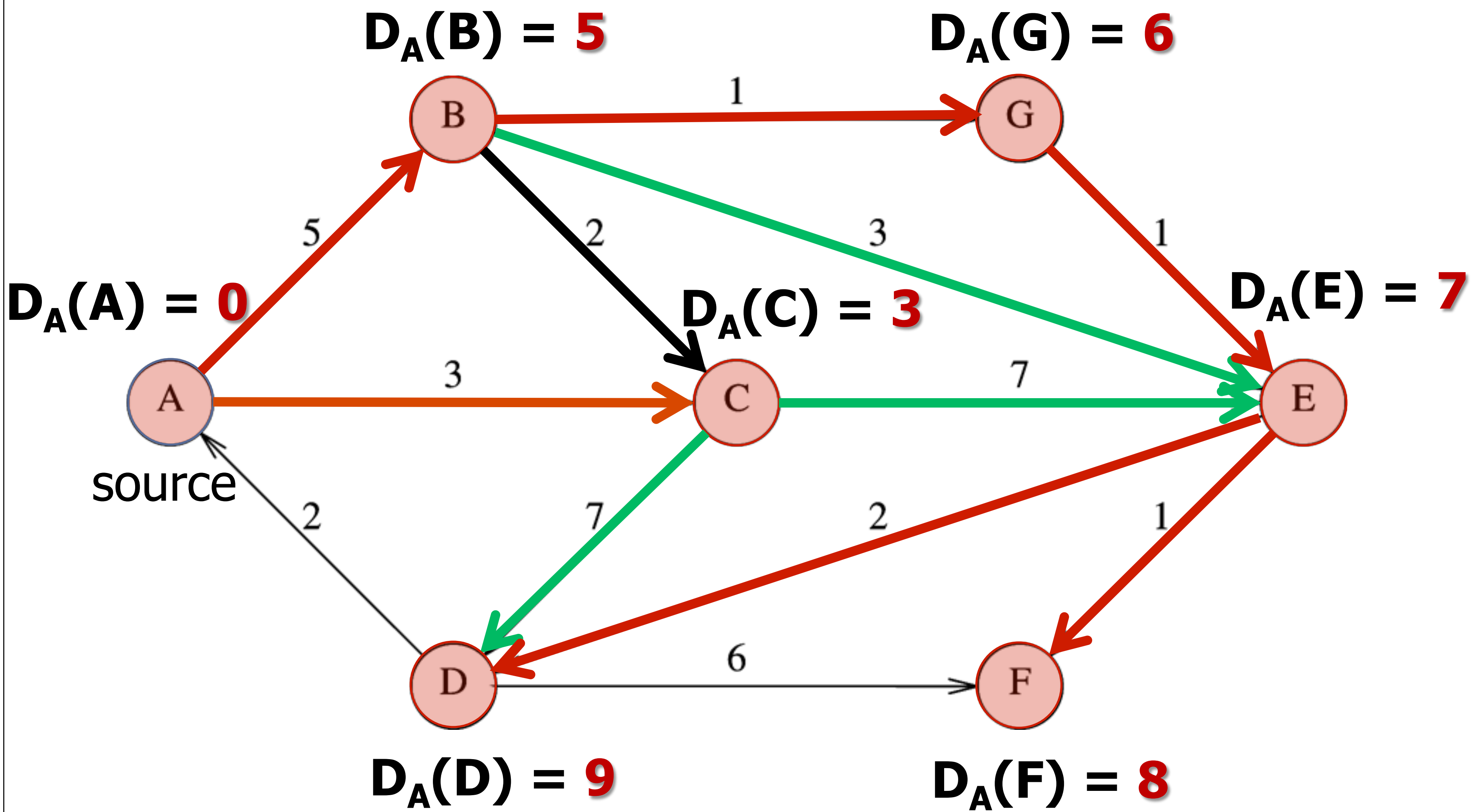
Weight f(n) = D_{source}(Destination)



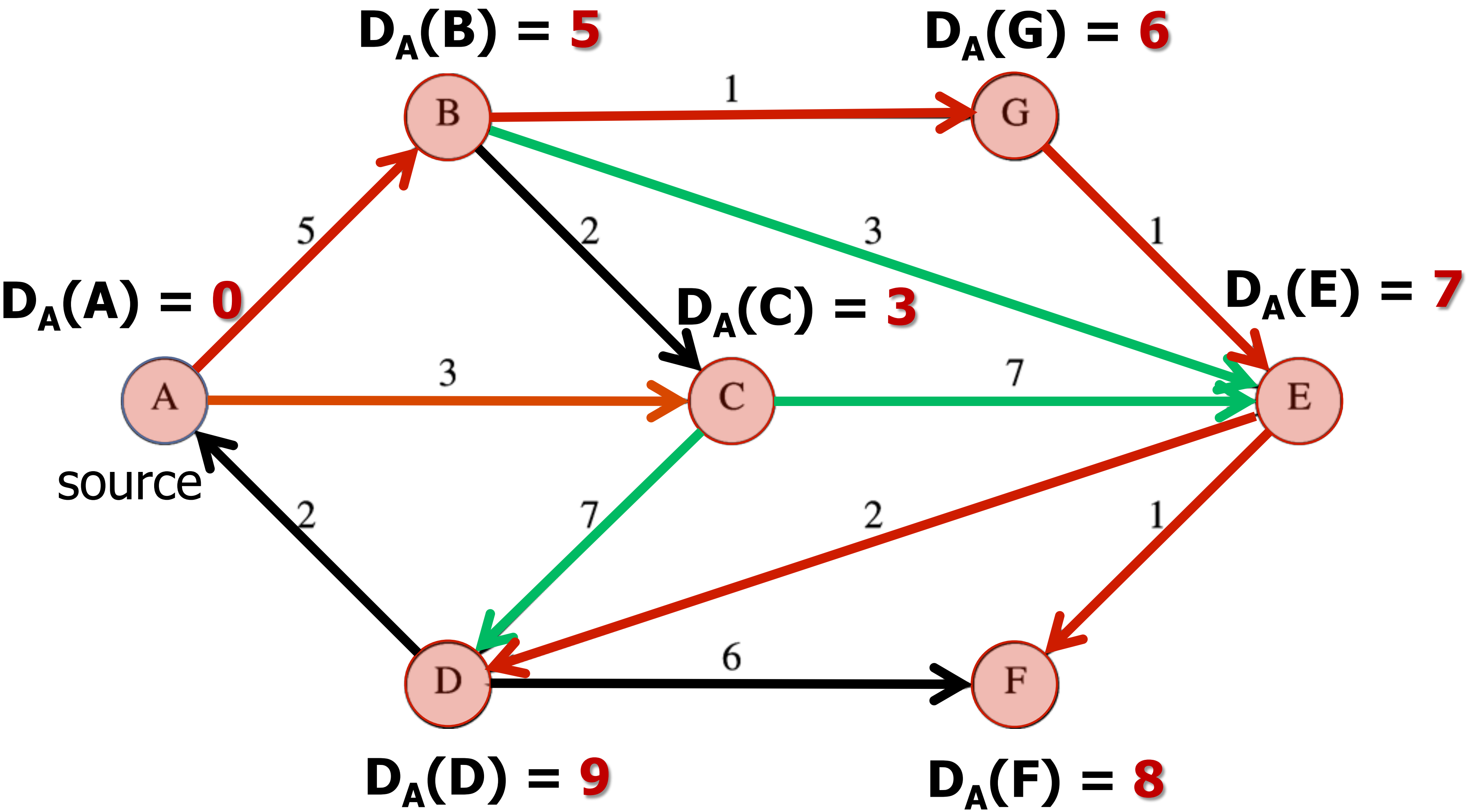
Weight f(n) = D_{source}(Destination)



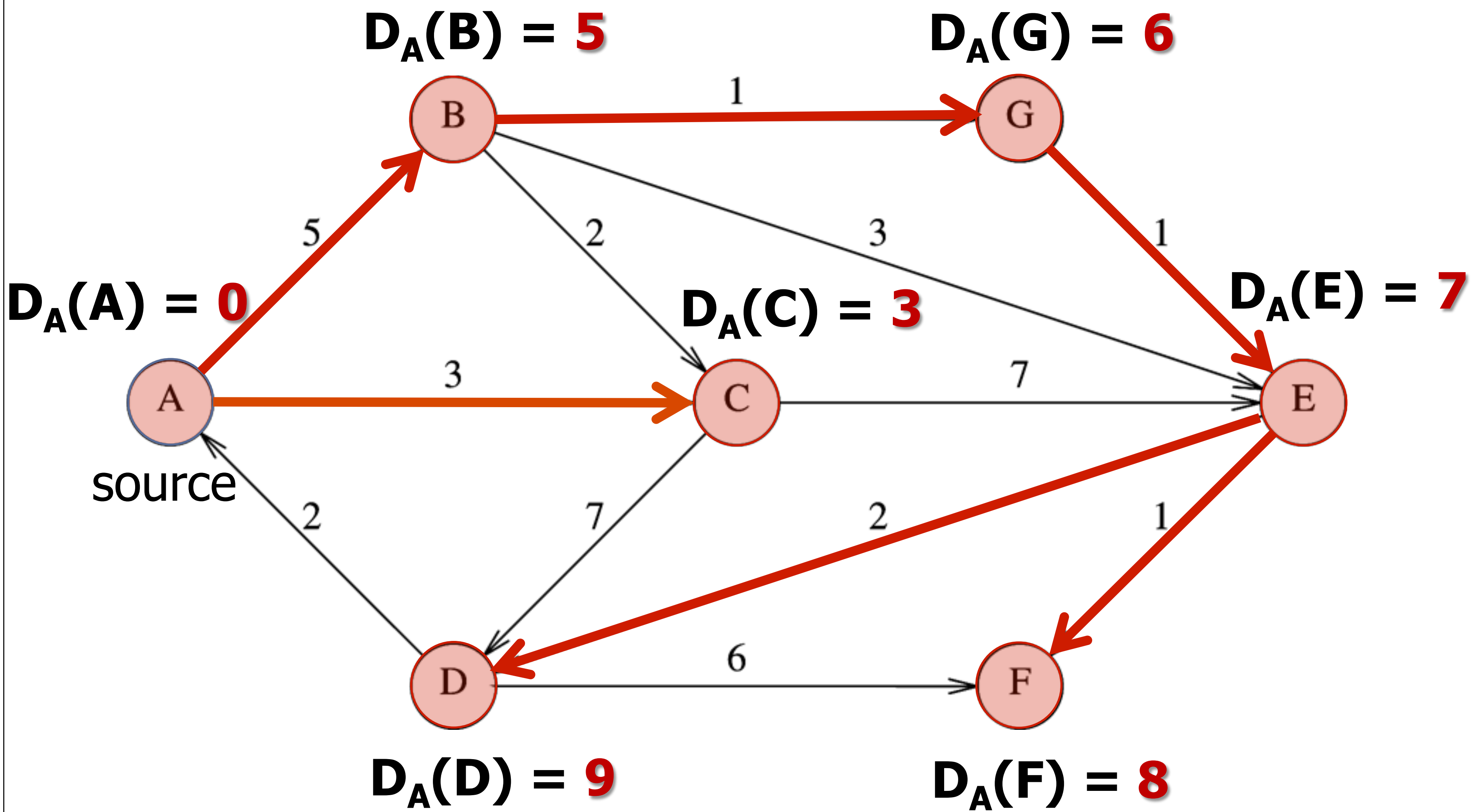
Weight $f(n) = D_{source}(Destination)$



Weight $f(n) = D_{source}(Destination)$



Weight $f(n) = D_{source}(Destination)$



Weight $f(n) = D_{source}(Destination)$

```
DIJKSTRA(G, w, s)
  INIT-SINGLE-SOURCE(G, s)
  S = ∅
  Q = G.V              // i.e., insert all vertices into Q
  while Q ≠ ∅
      u = EXTRACT-MIN(Q)
      S = S ∪ {u}
      for each vertex v ∈ G.Adj[u]
          RELAX(u, v, w)
```

INIT-SINGLE-SOURCE(G, s)

for each $v \in G.V$

$v.d = \infty$

$v.\pi = \text{NIL}$

$s.d = 0$

$\textsc{Relax}(u, v, w)$

if $v.d > u.d + w(u, v)$

$v.d = u.d + w(u, v)$

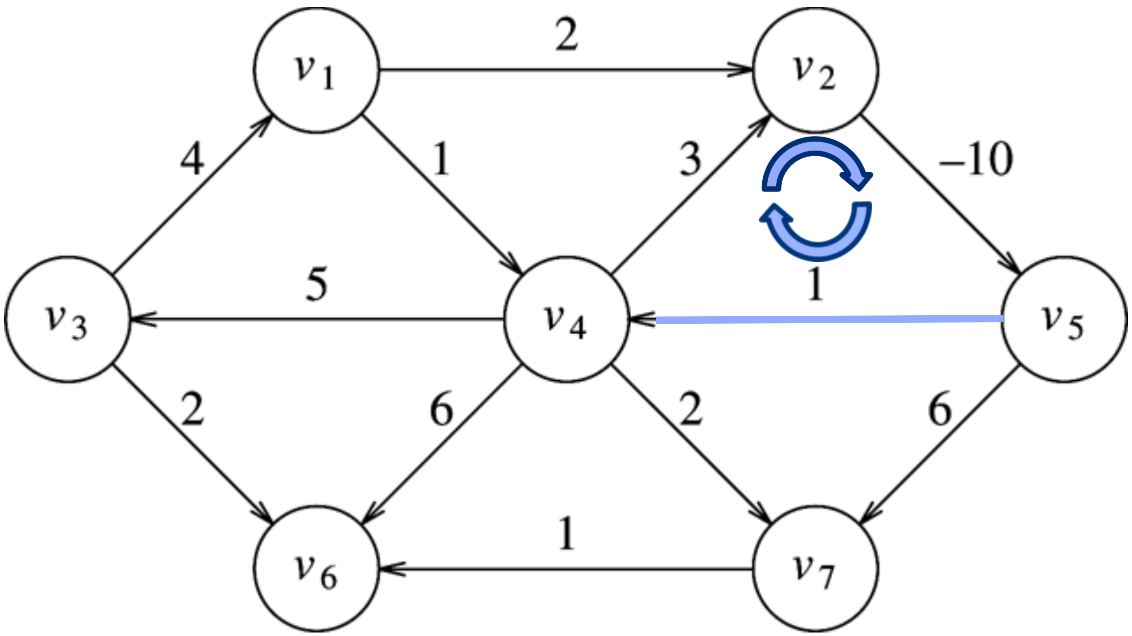
$v.\pi = u$

```
BFS(V, E, s)
  for each u ∈ V − {s}
      u.d = ∞
  s.d = 0
  Q = ∅
  ENQUEUE(Q, s)
  while Q ≠ ∅
      u = DEQUEUE(Q)
      for each v ∈ G.Adj[u]
          if v.d == ∞
              v.d = u.d + 1
              ENQUEUE(Q, v)
```

Algorithm Dijkstra

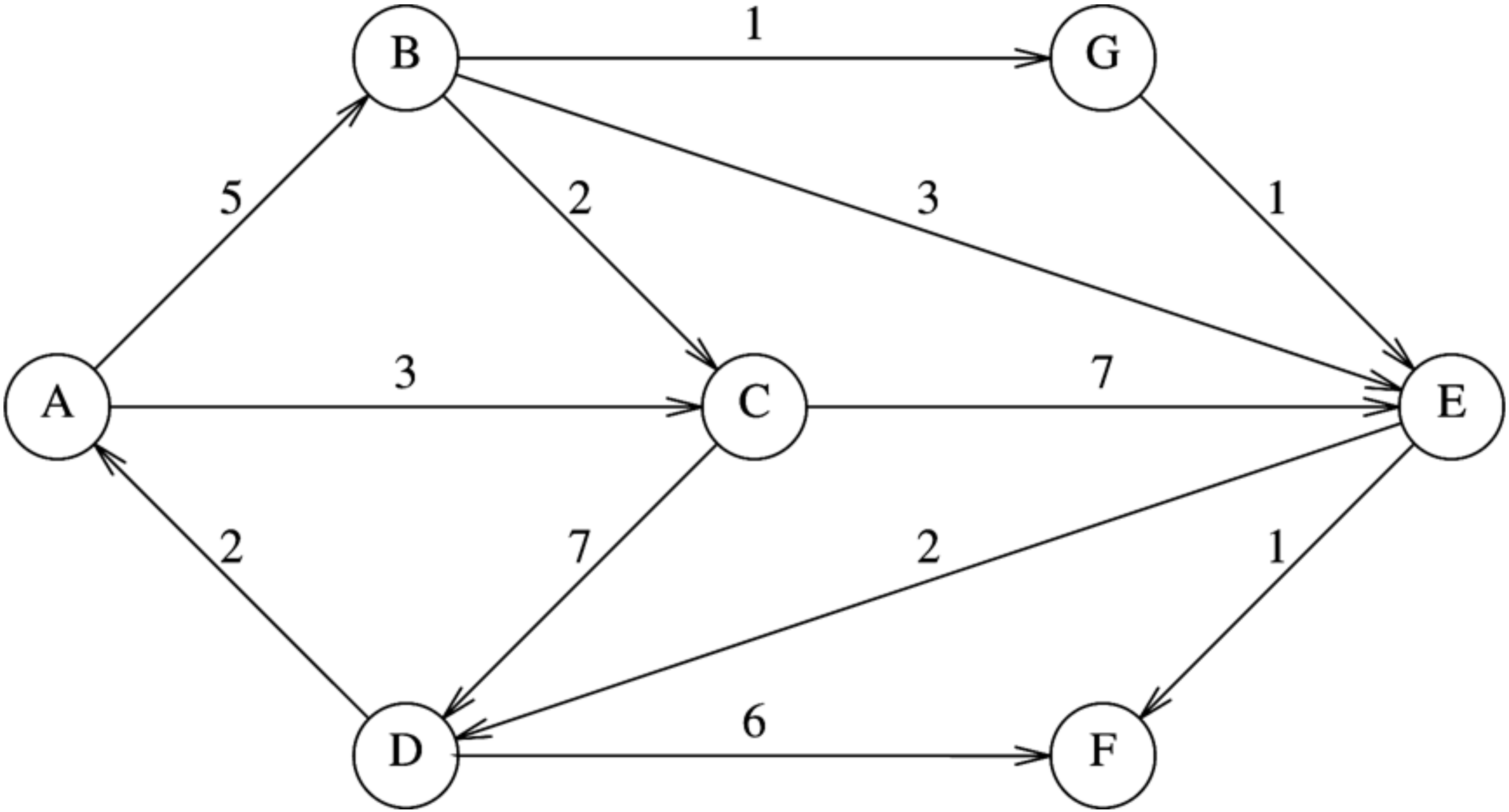
```
for each Vertex v do
   v.known <- false
   v.dist  <- infinity //distance from source vertex
   v.prev  <- NULL

   //s is the source vertex
    s.dist <- 0
    do n times
        v <- unknown vertex with minimum v.dist
        v.known <- true
        for each edge (v,w) do
             if v.dist + cvw <= w.dist then
                 w.dist <- v.dist + cvw
                 w.prev <- v
```

Negative Cost Cycle



Dijkstra's does not work! cost = **-∞**

Which source vertex has no shortest path?



Dijkstra's Analysis

- Analysis reflects a complete execution of algorithm
- n = number of vertices, m = number of edges
- Initialize vertices is O(n)
- Finding the vertex with the minimum cost:
  - Use a list/array is $O(n^2)$, linear scan
  - Use a binary min-heap is O(nlogn), deleteMin
- Update vertex's cost:
  - Use a list/array is O(1)
  - Use a binary min-heap is O(mlogn), (percolateUp)
- Overall running time:
  - list/array is $O(n^2)$
  - binary min-heap is O((n + m)logn)