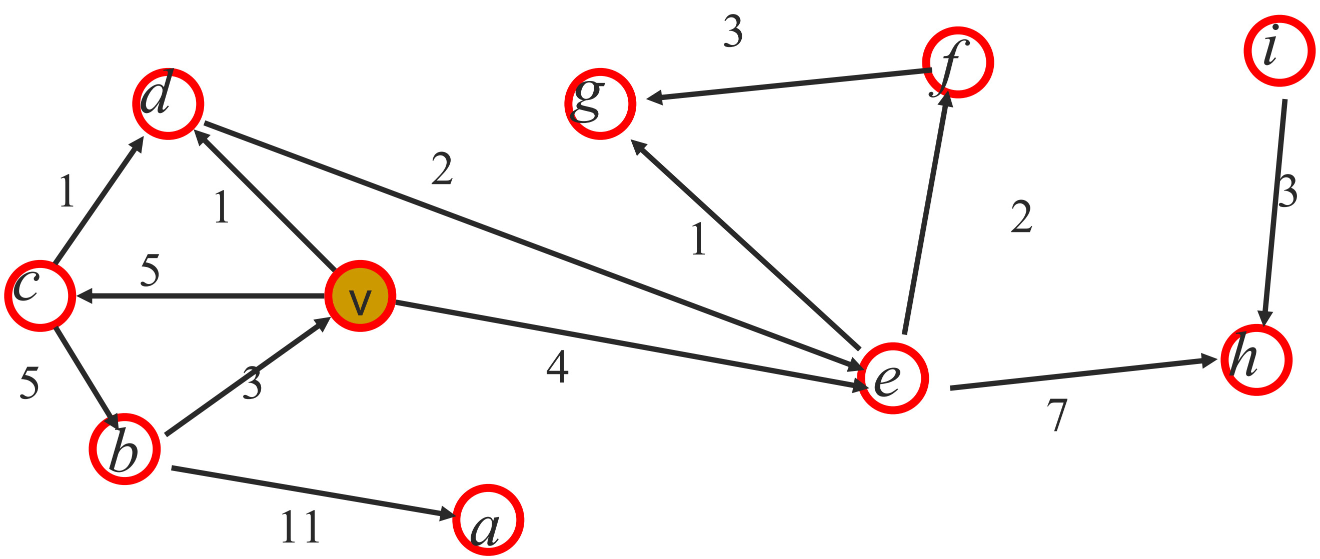
Bellman-Ford algorithm: Shorest paths with negative weights

Directed graphs with positive edge weights



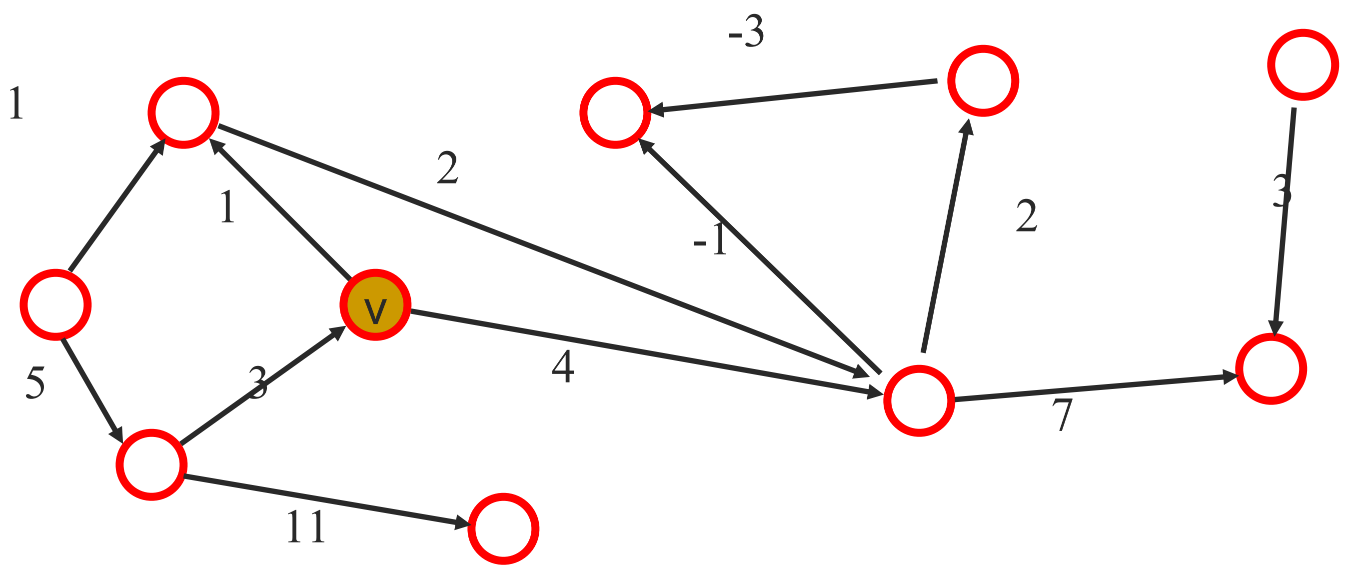
Does Dijkstra’s algorithm work?

Single source shortest paths for *directed* graphs with positive edge weights

- Dijkstra's algorithm works without changes except here edges are directed, that is $(a,b) \neq (b,a)$
- The big-oh worst case running time remains the same

Shortest paths in graphs containing *negative* edges

- Not possible for undirected graphs
- What about directed graphs?



Negative edges and negative-weight cycles

- If G is directed, compute single-source shortest path problem using Bellman-Ford shortest path algorithm
- Negative-weight cycles are discovered

Algorithm `Bellman-Ford(G,v)`

Input: A simple directed graph G with possible negative edge-weights, a distinguished vertex v in G

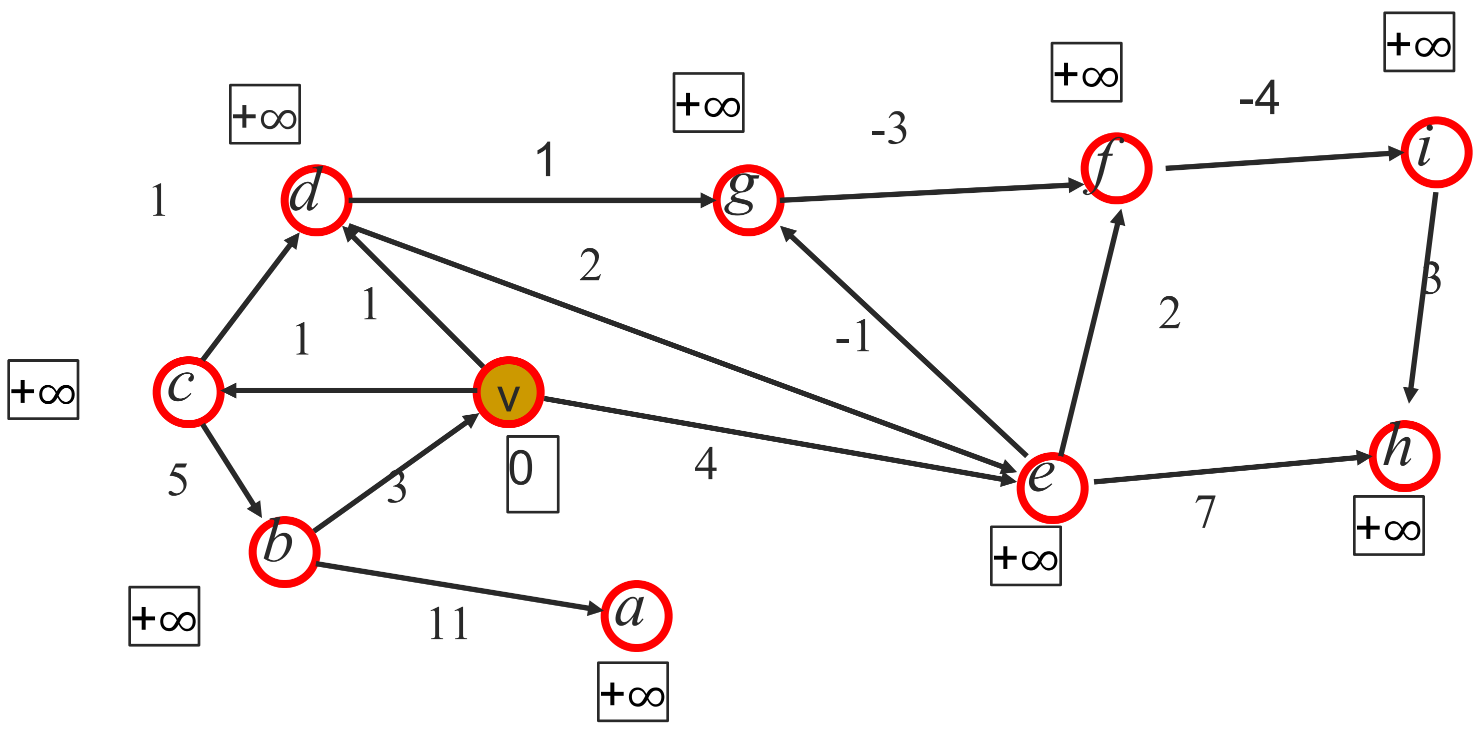
Output: A label D[u] for each vertex u in G such that D[u] is the distance from v to u in G.

Algorithm Bellman-Ford(G,v)

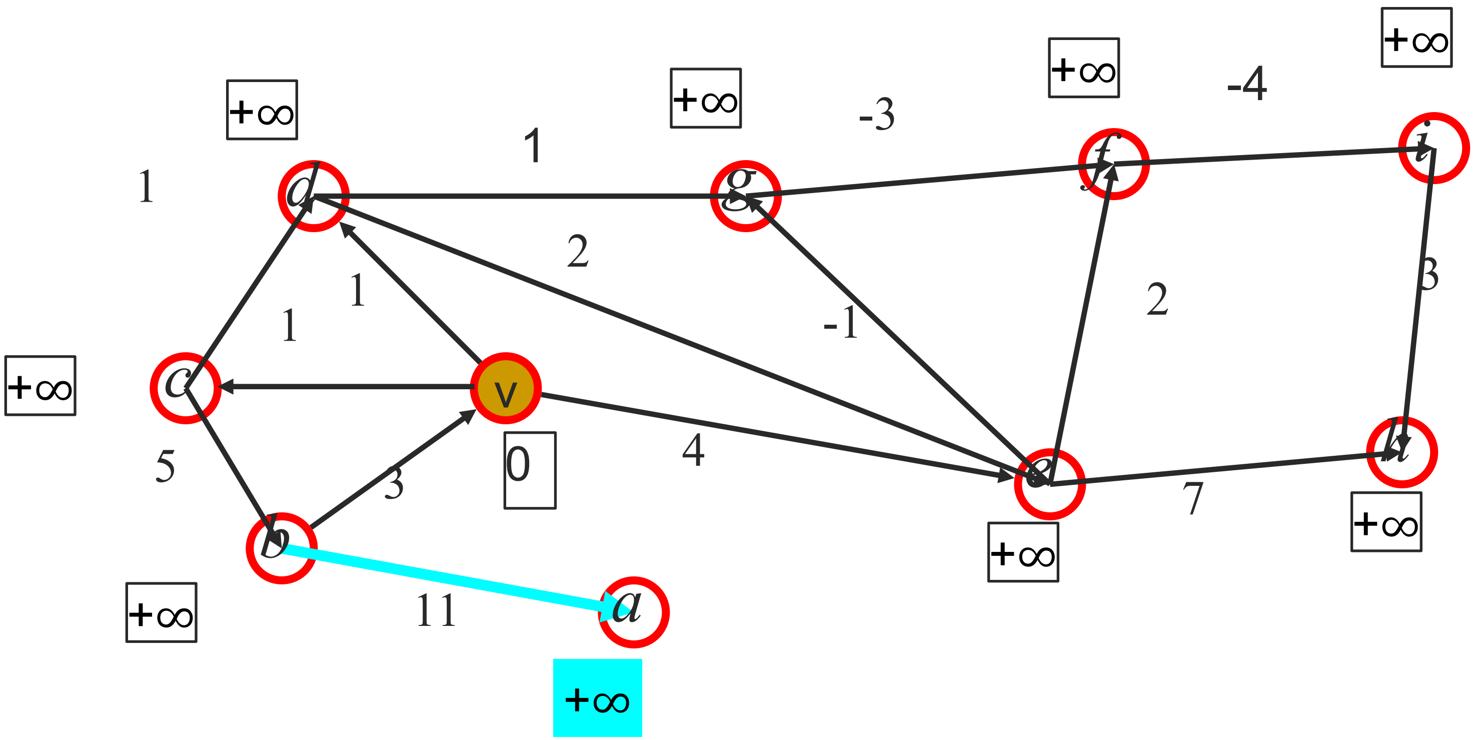
performs n-1 times a relaxation of every edge in the graph

```
D[v]←0
for each vertex u ≠ v of G do
   D[u]← +∞
for i ←1 to n-1 do
   for each edge (u,z) in G do
      if D[u]+w((u,z)) < D[z] then
            D[z]←D[u]+w((u,z))
if there are no edges left with potential
   relaxation operations then
   return D
else
   return "G contains a negative cycle"
```

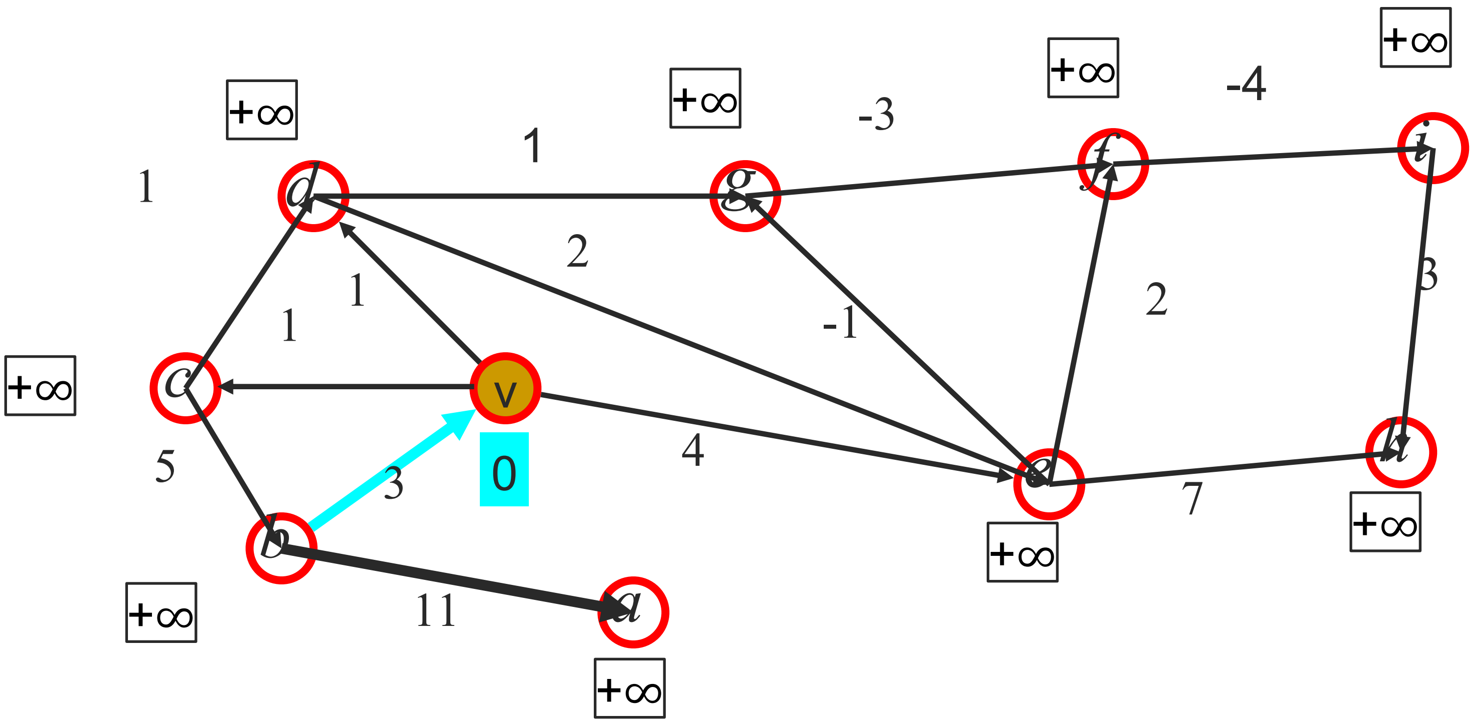
Initialize …



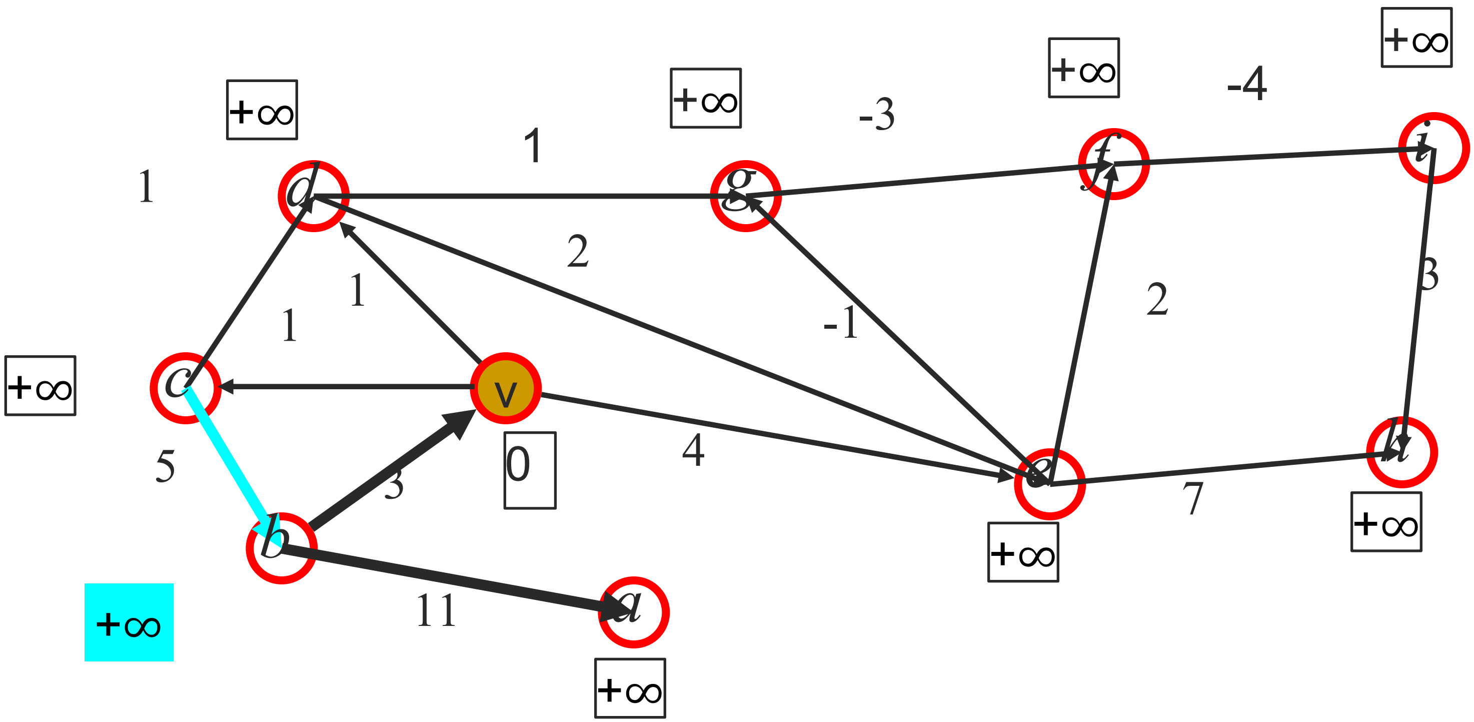
i=1



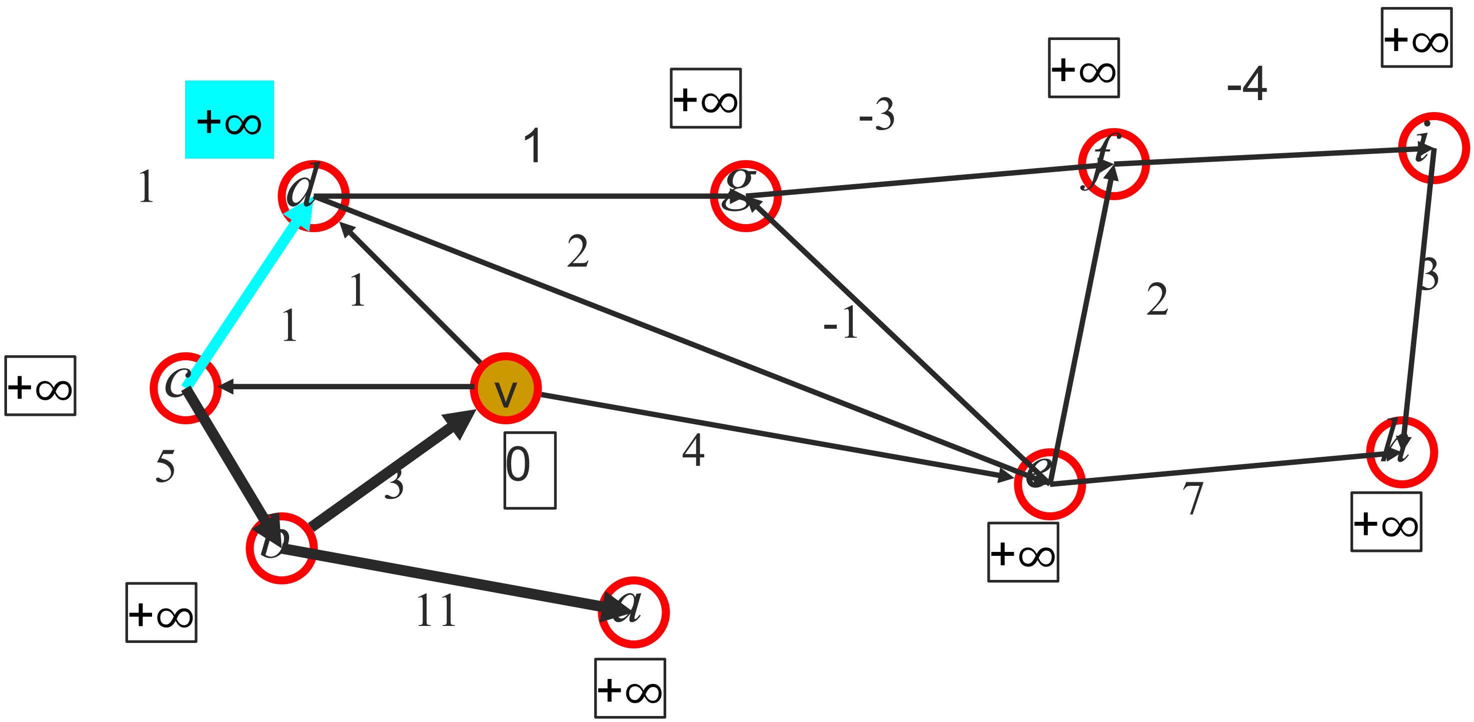
i=1



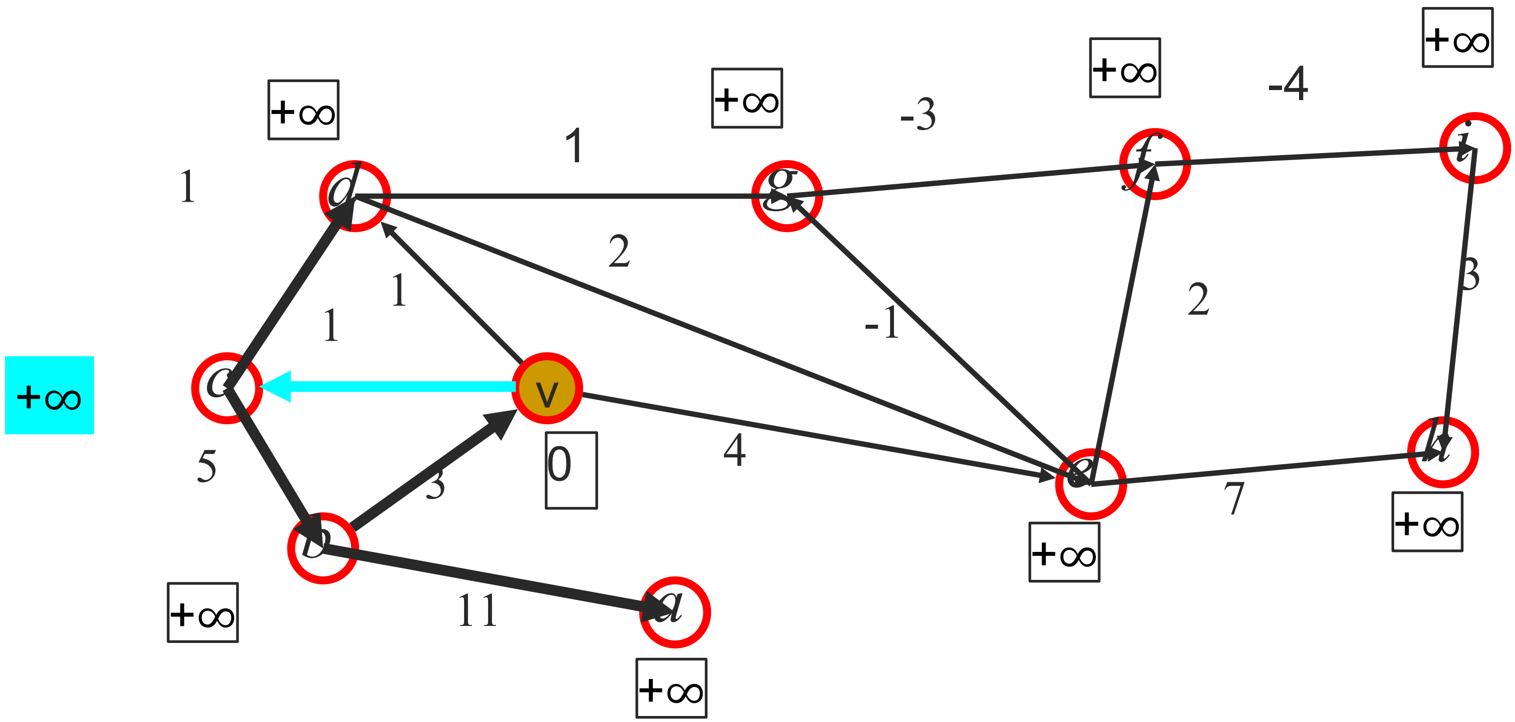
i=1



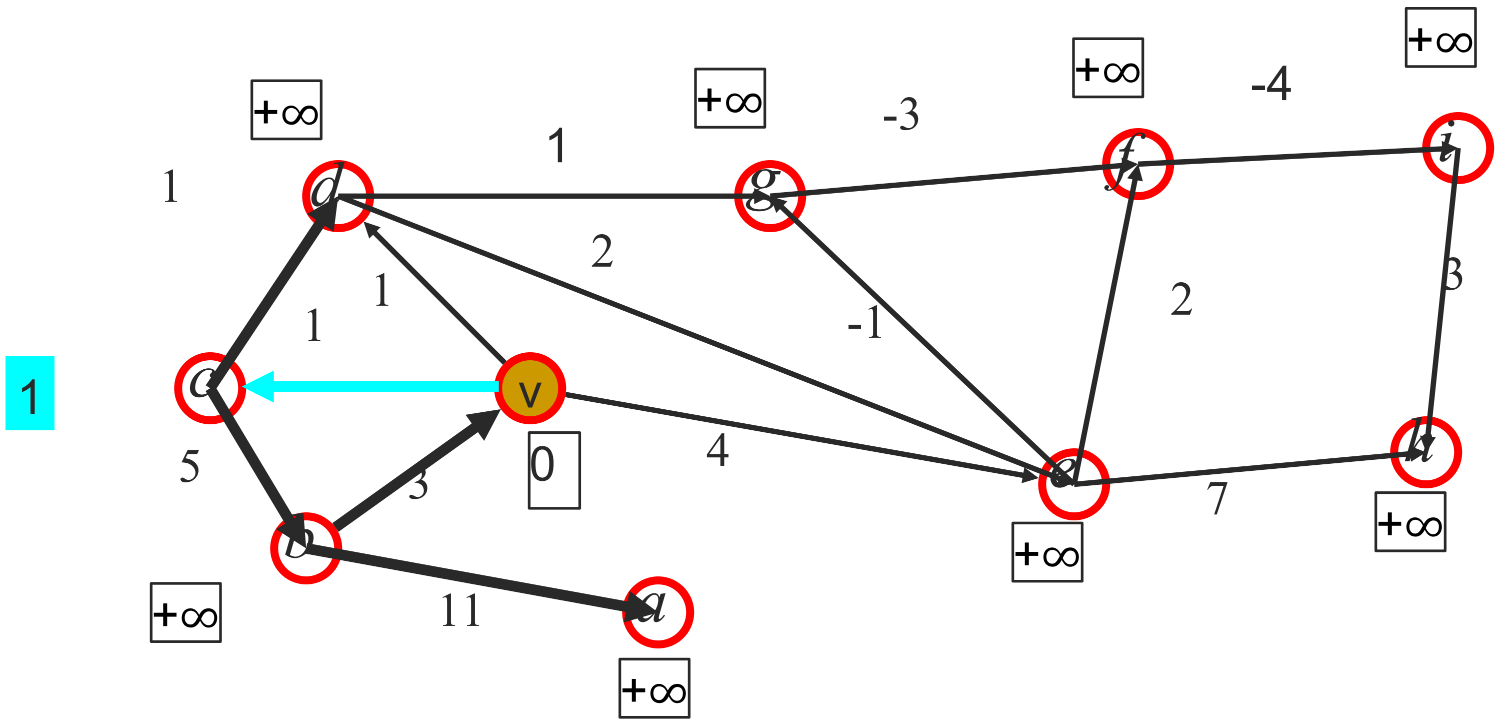
i=1



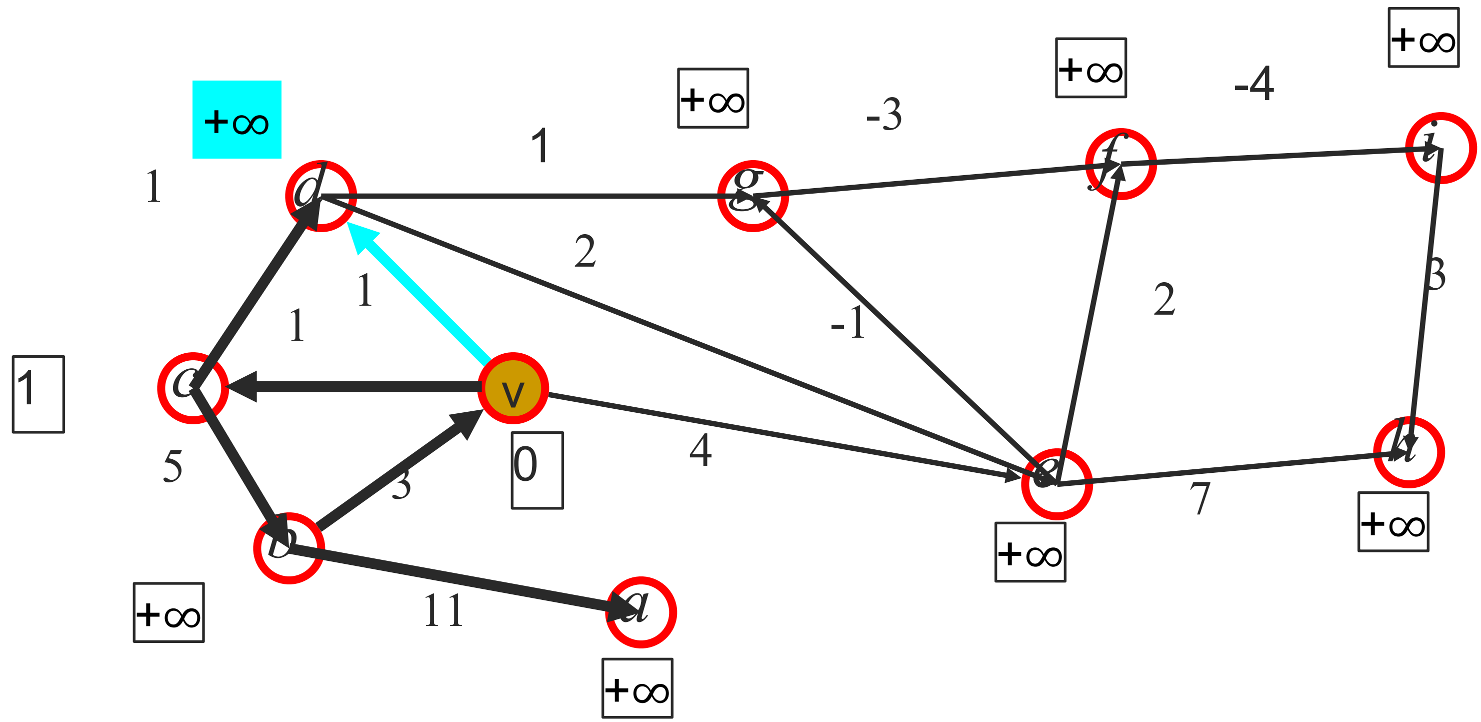
i=1



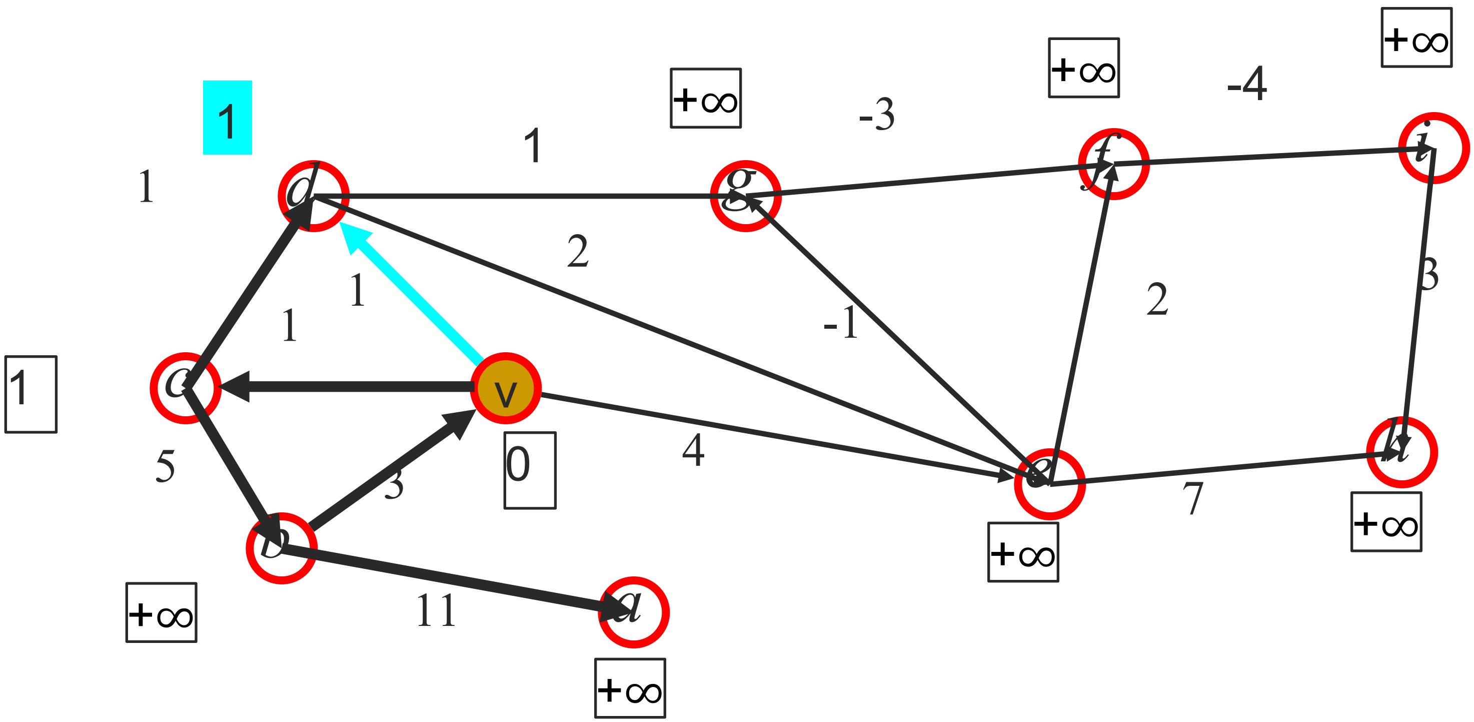
i=1



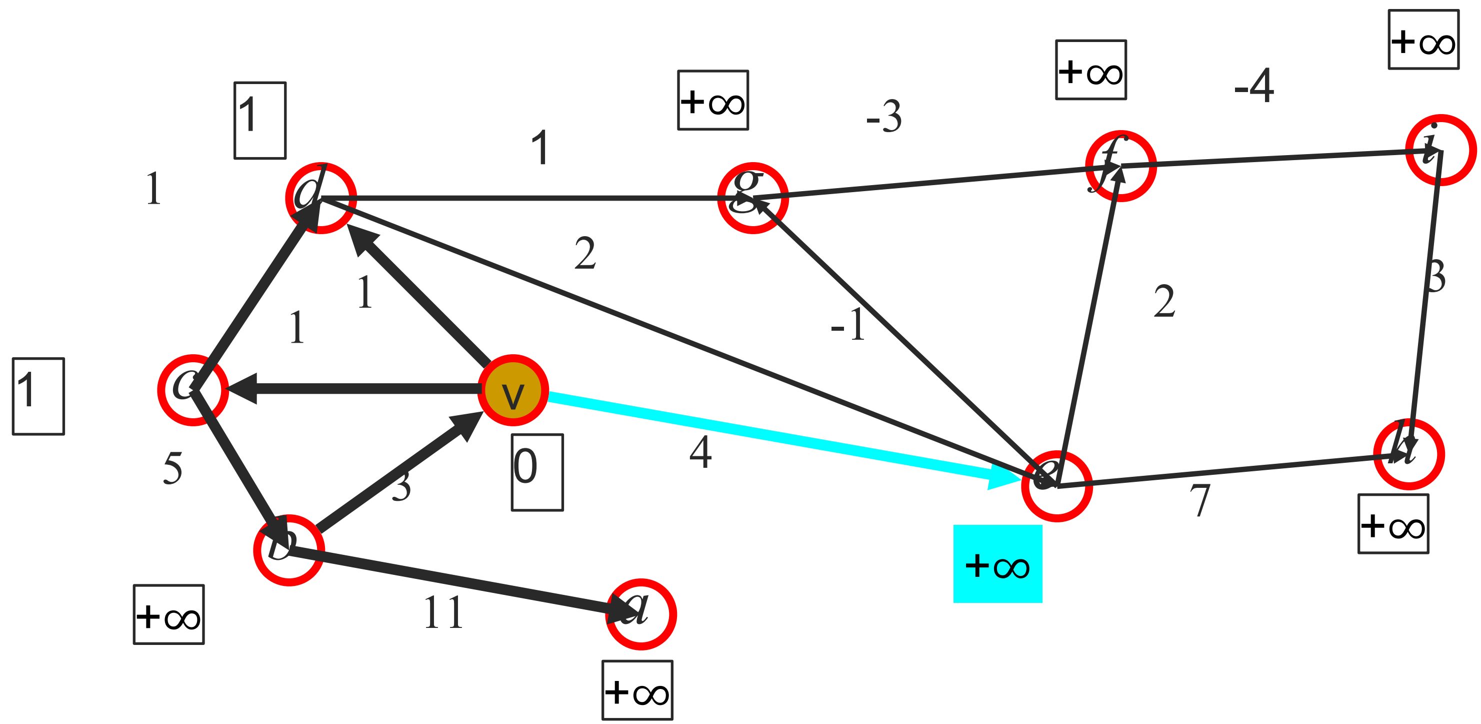
i=1



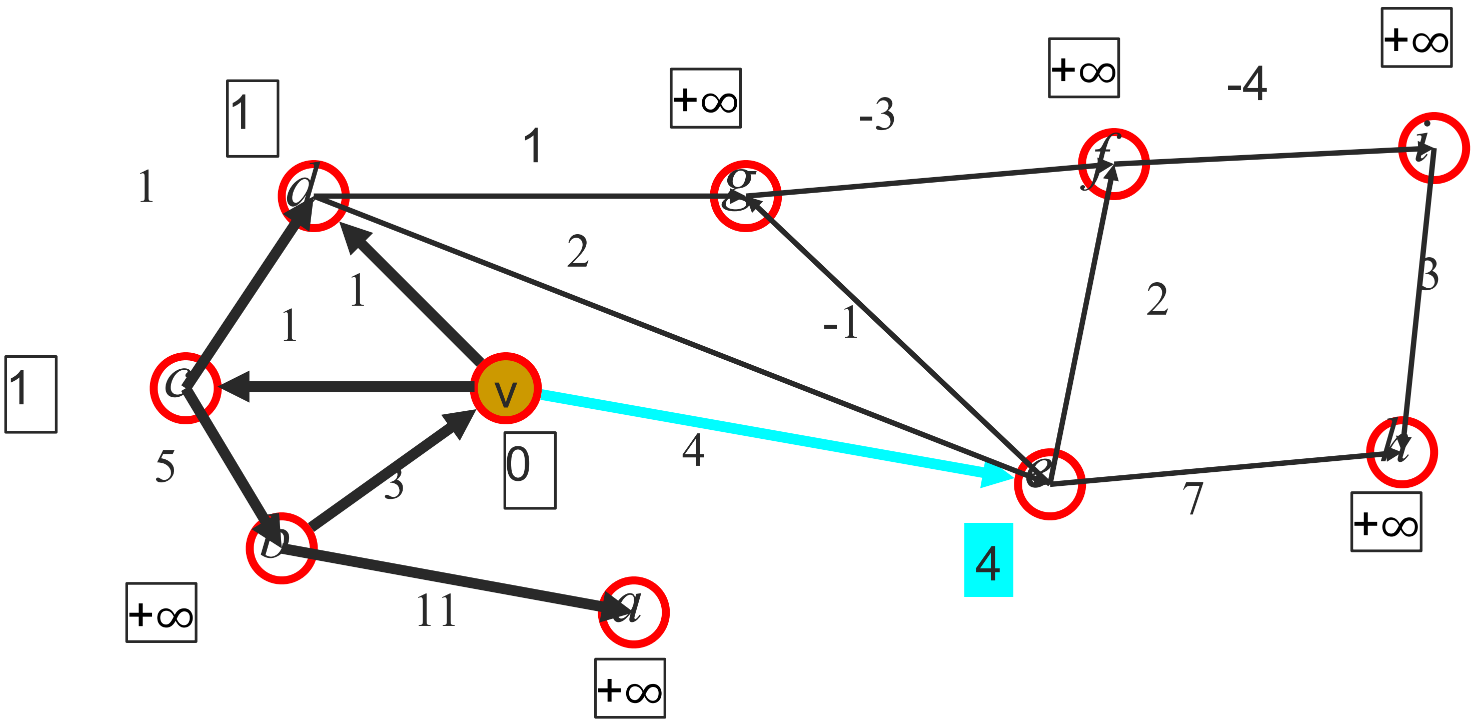
i=1



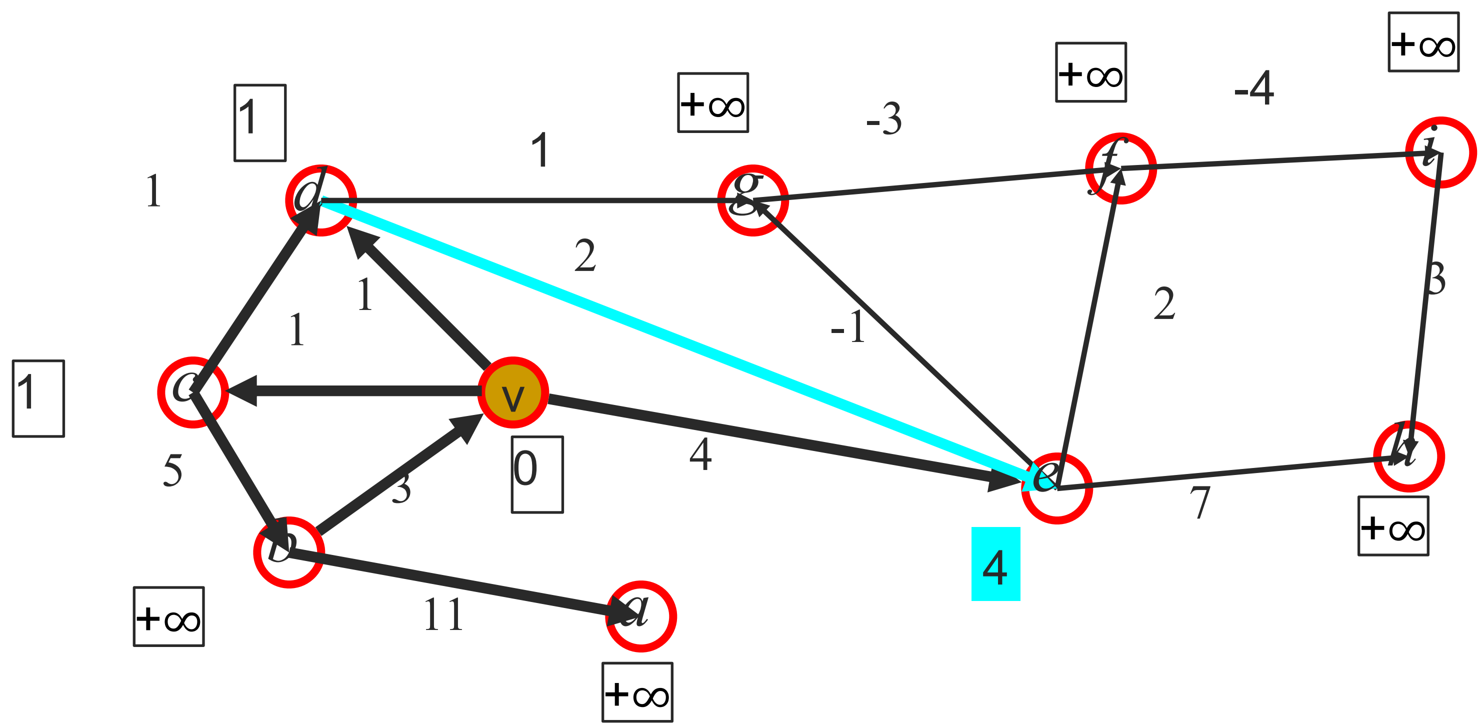
i=1



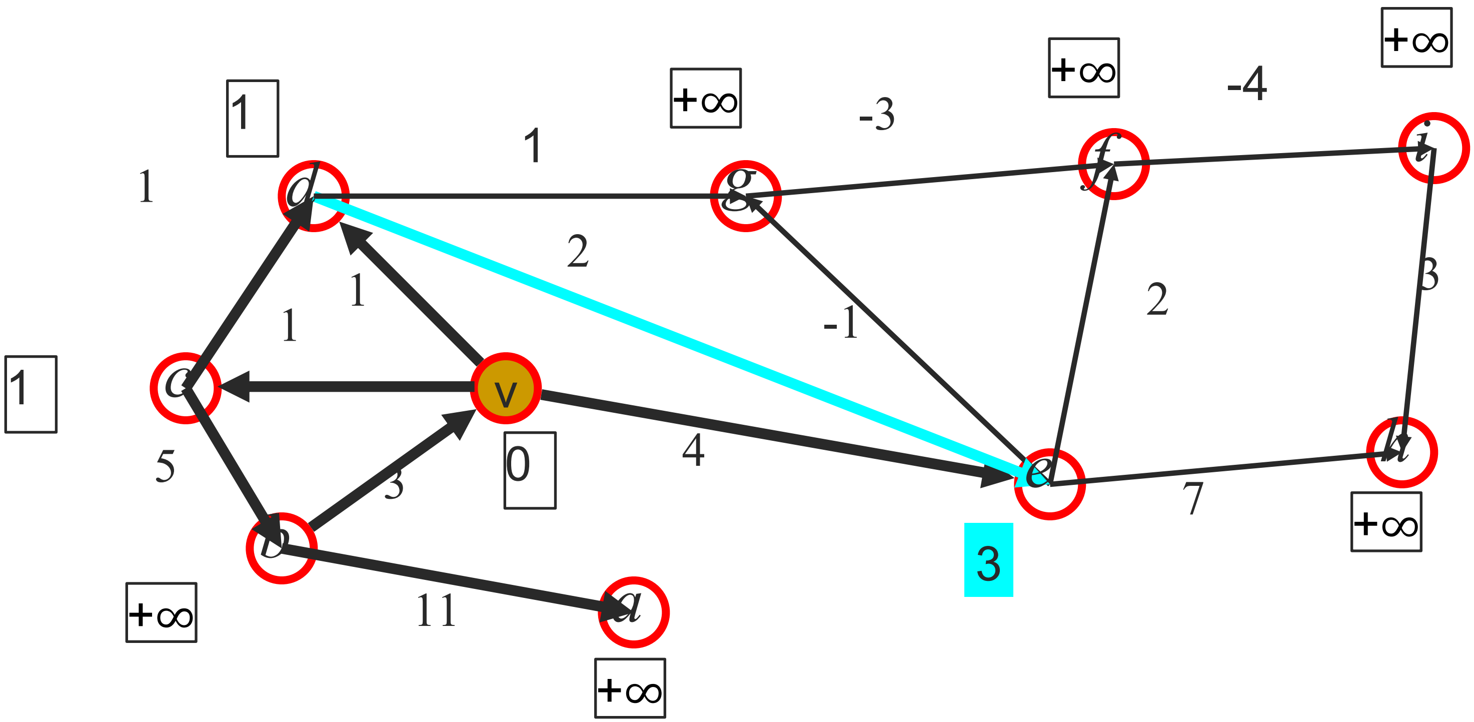
i=1



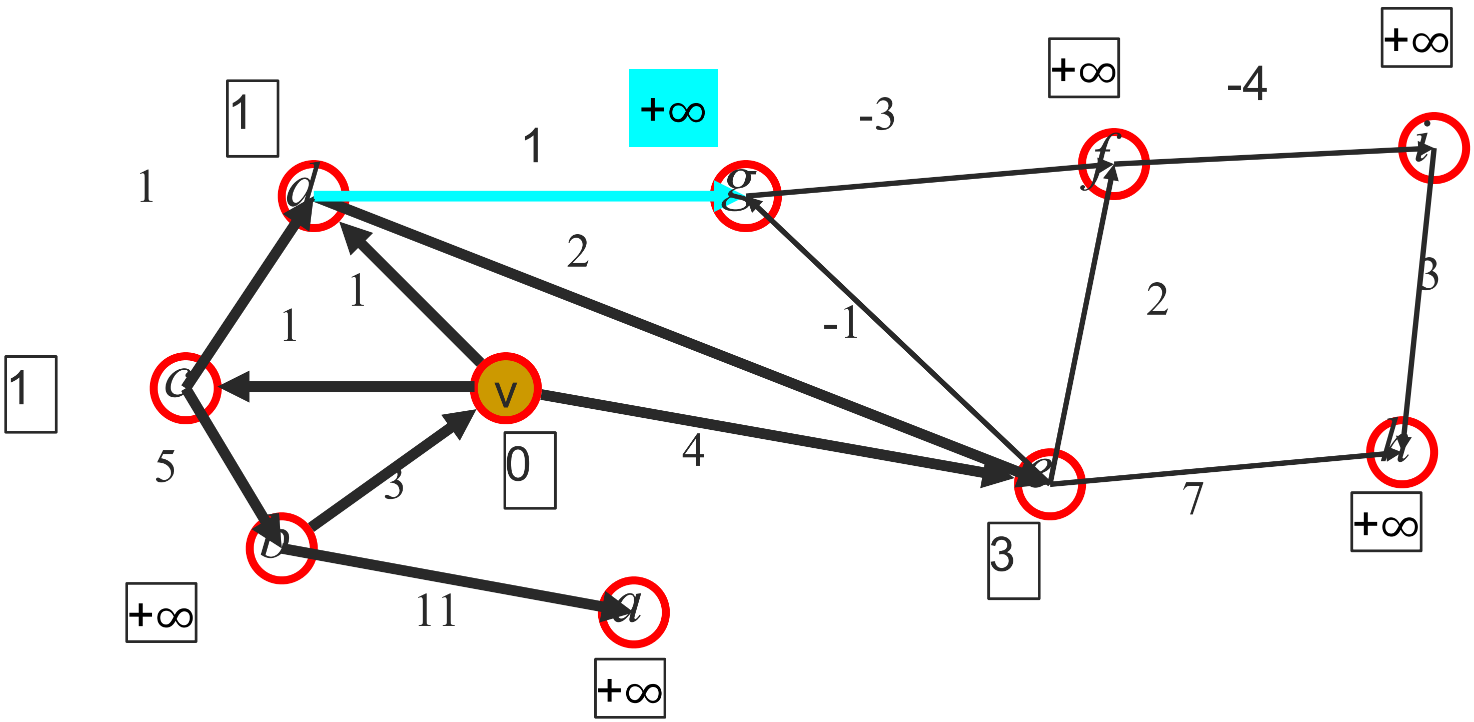
i=1



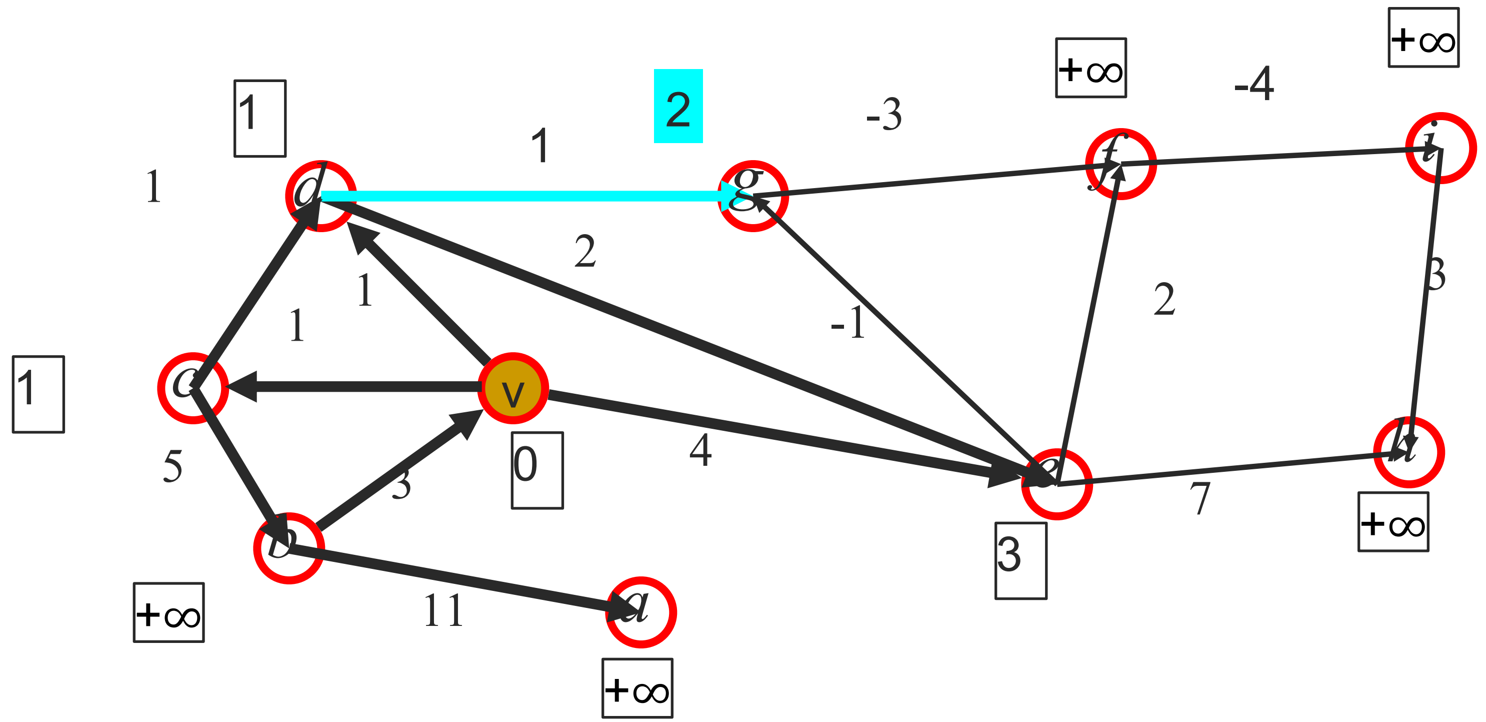
i=1



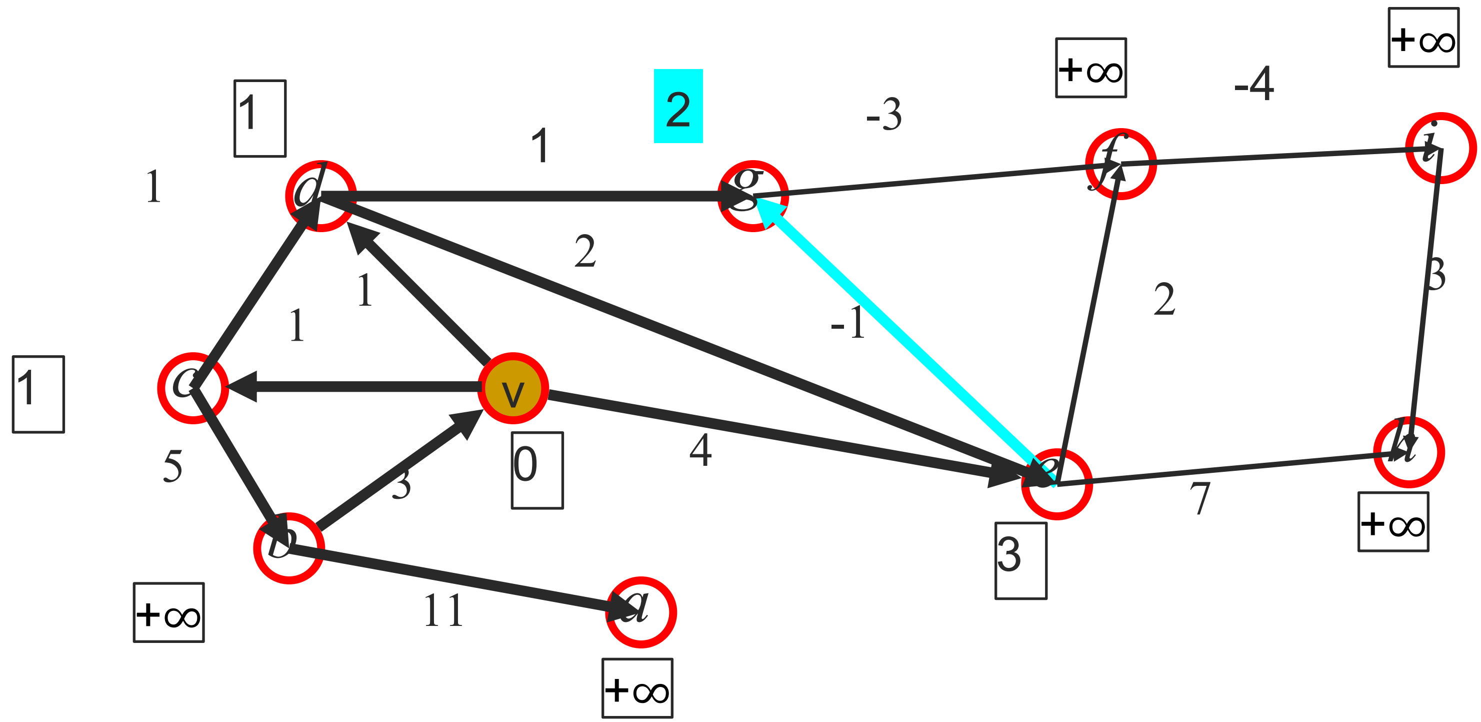
i=1



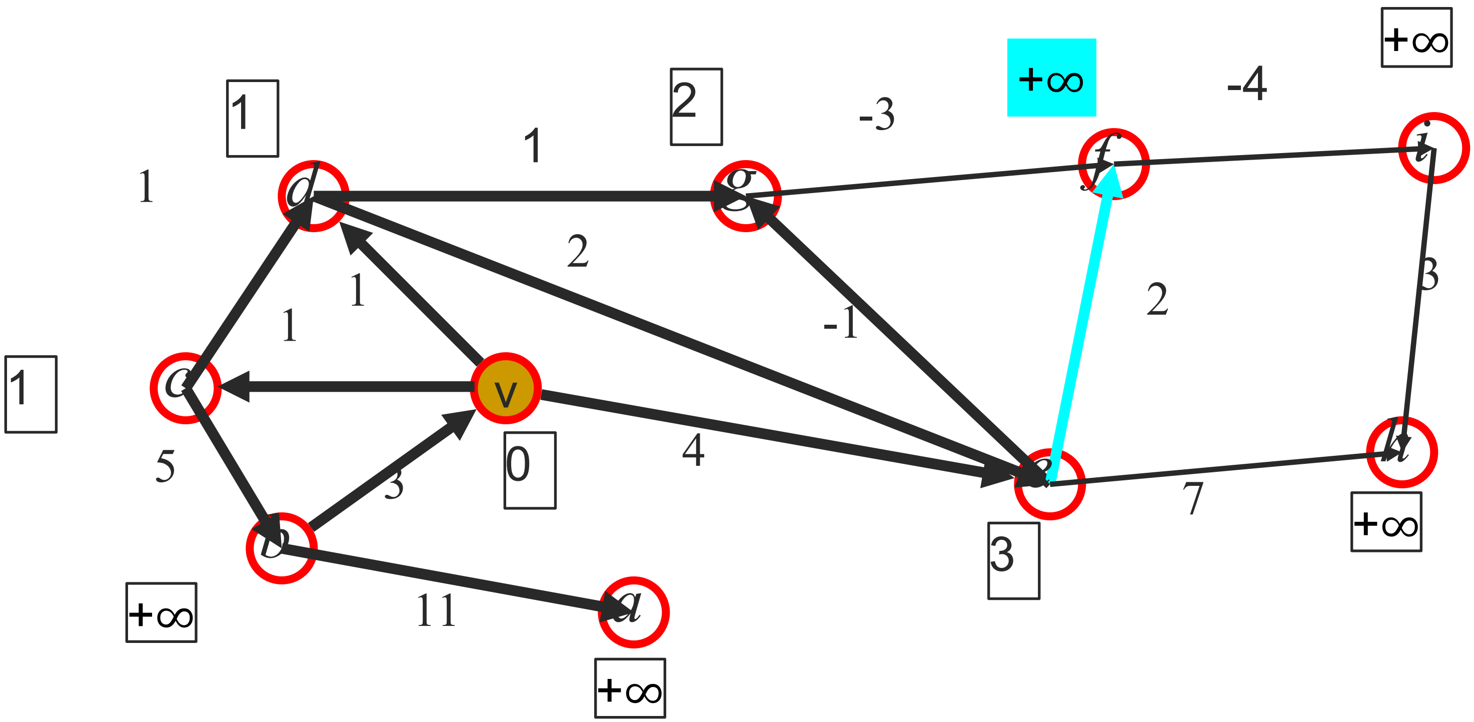
i=1



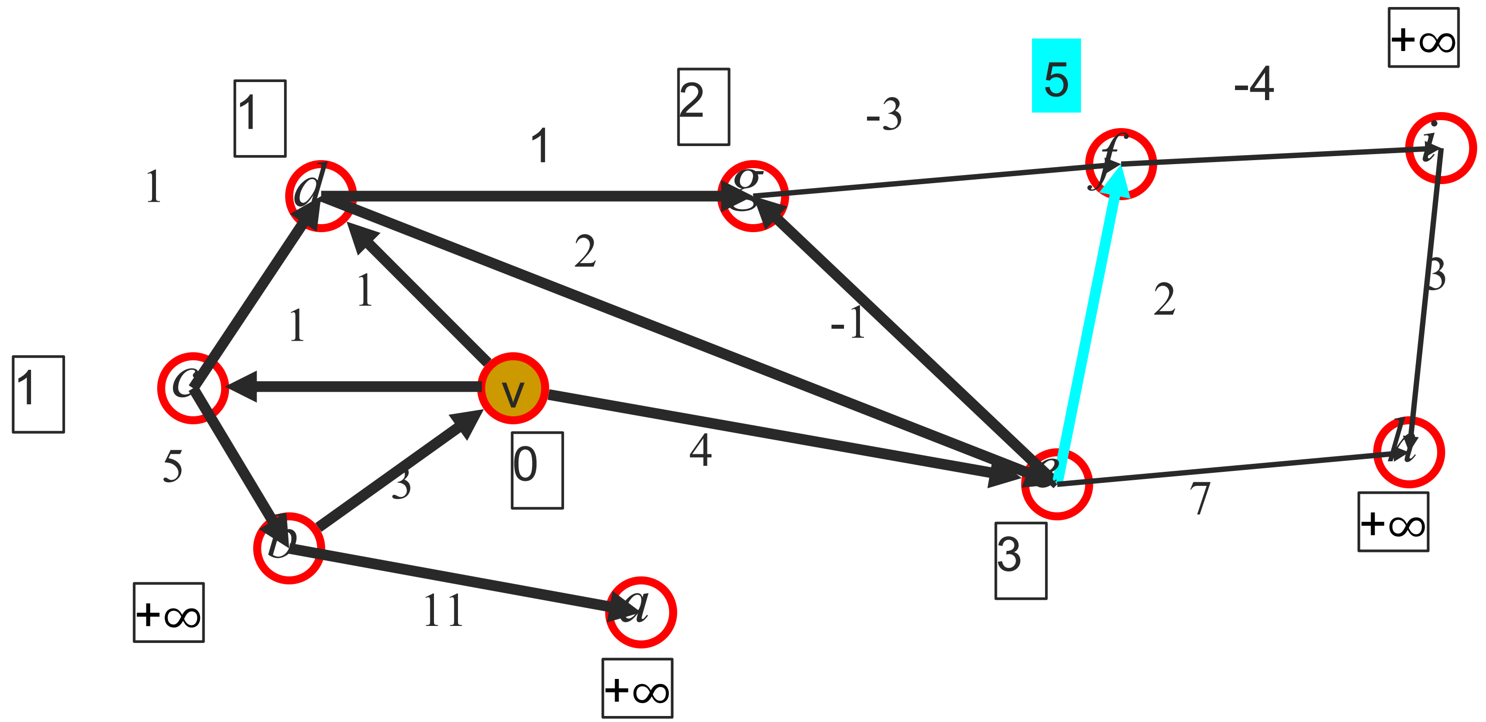
i=1



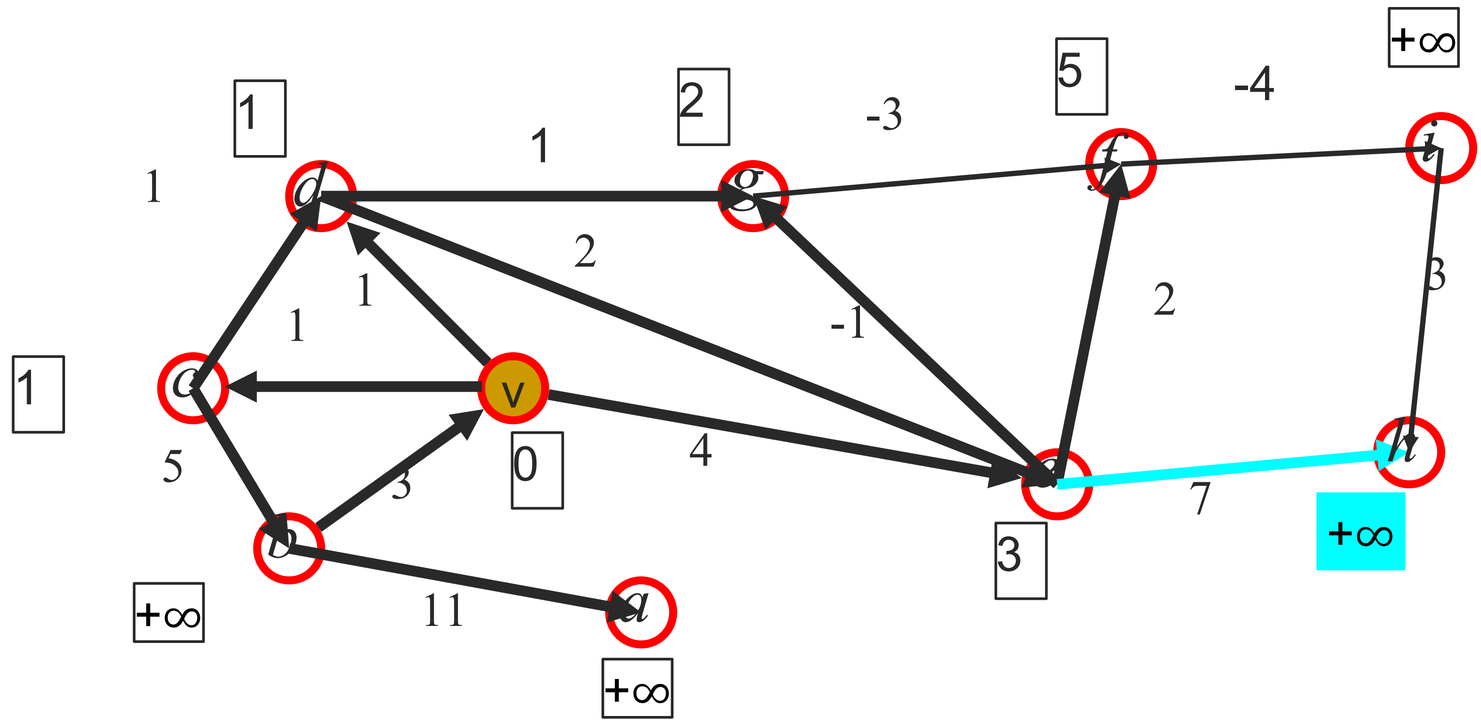
i=1



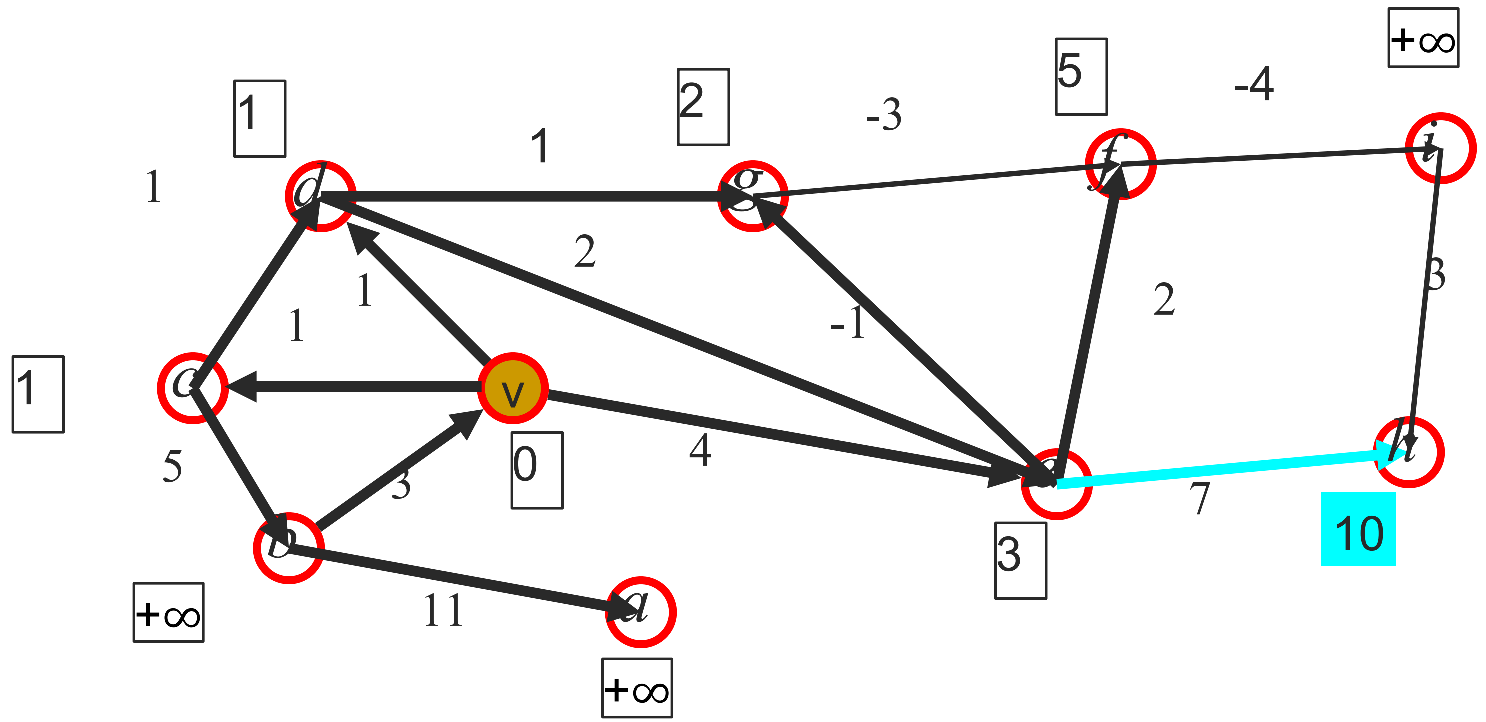
i=1



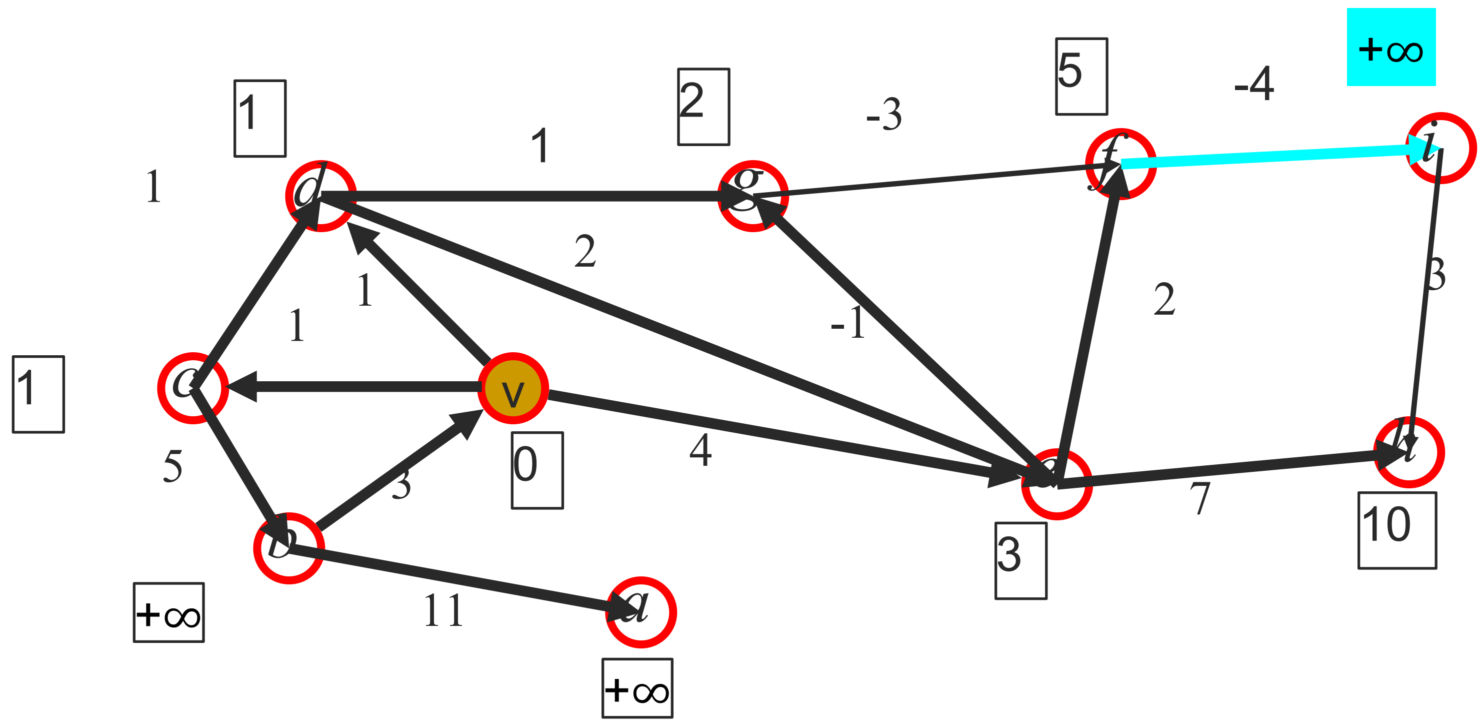
i=1



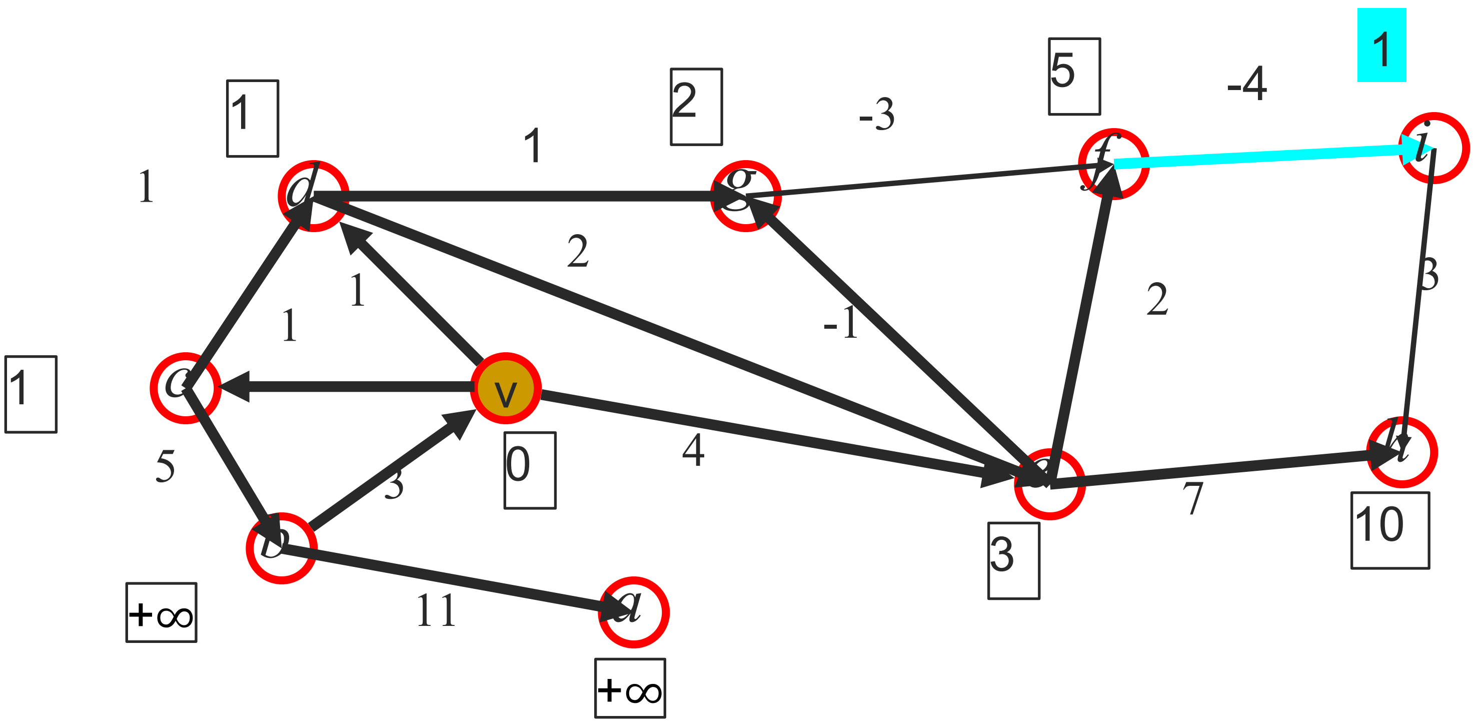
i=1



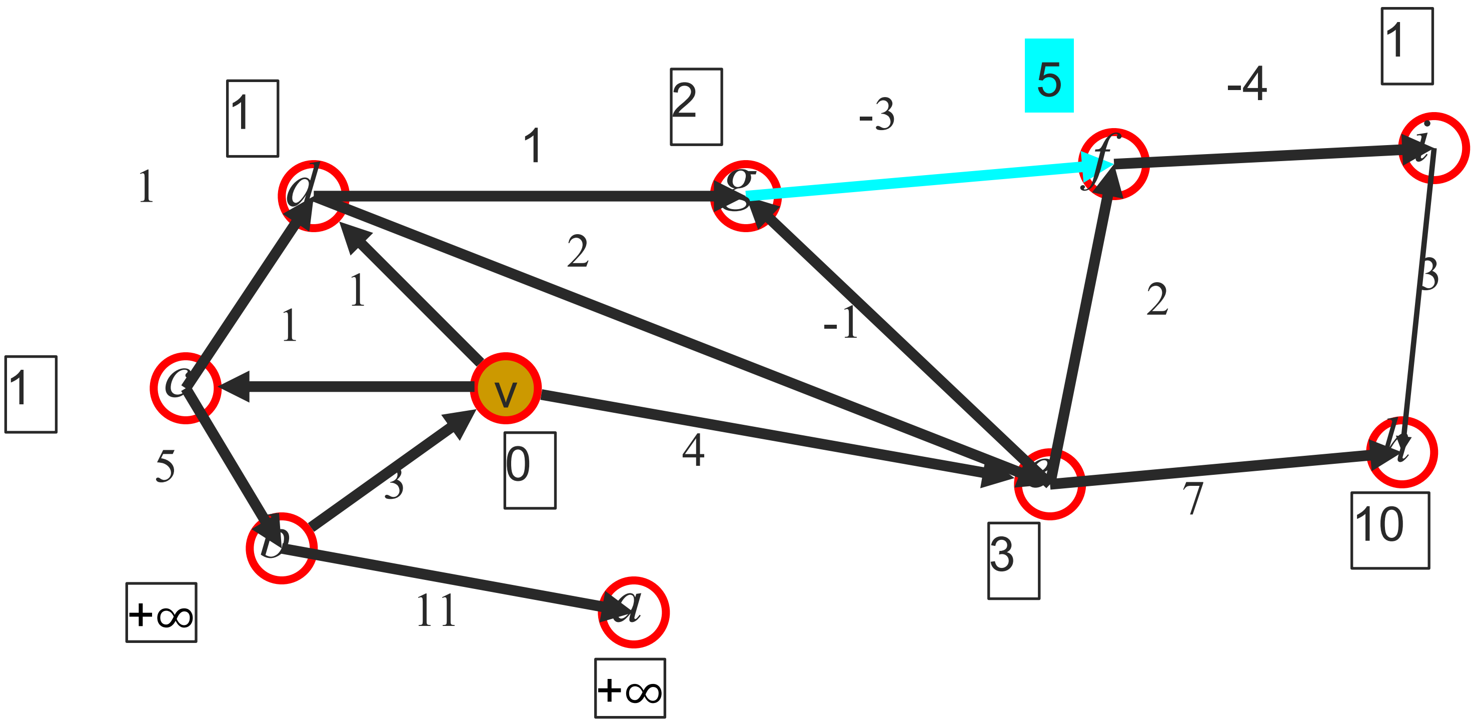
i=1



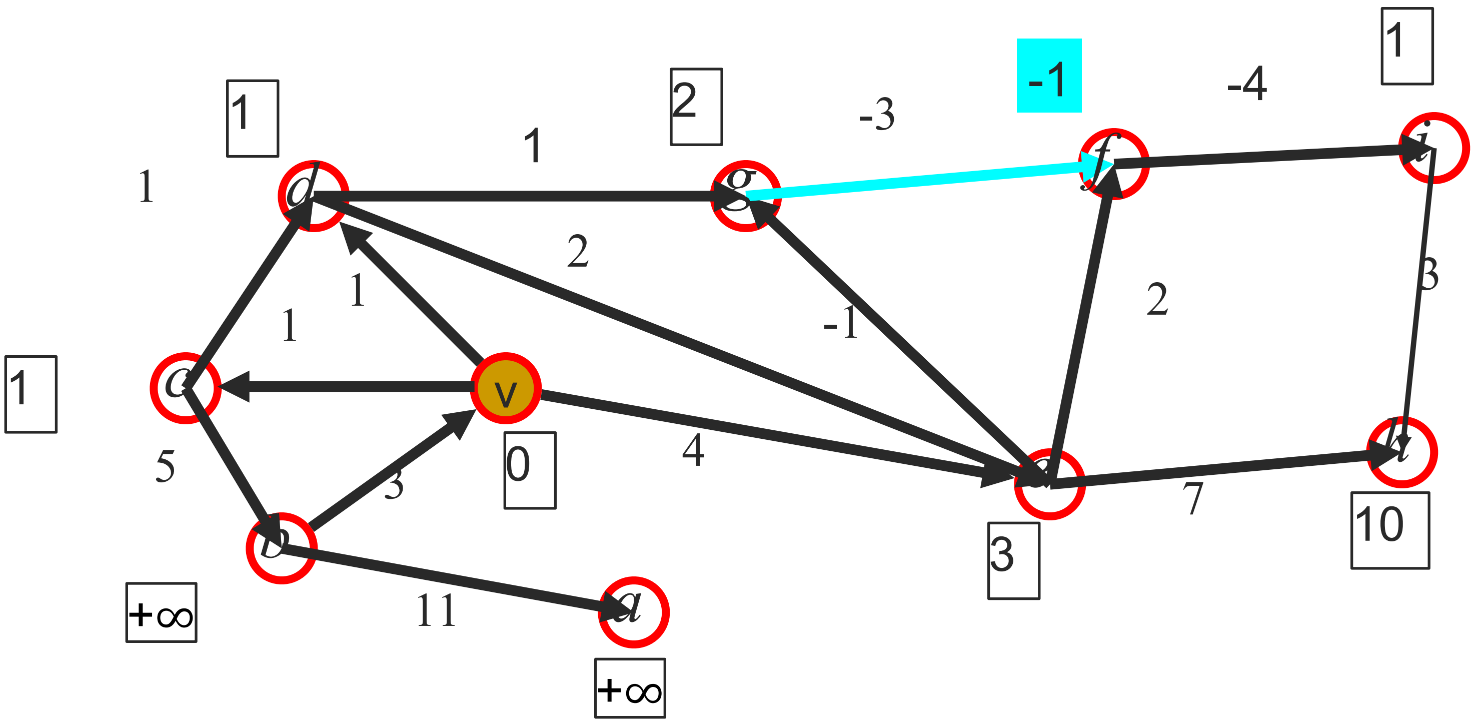
i=1



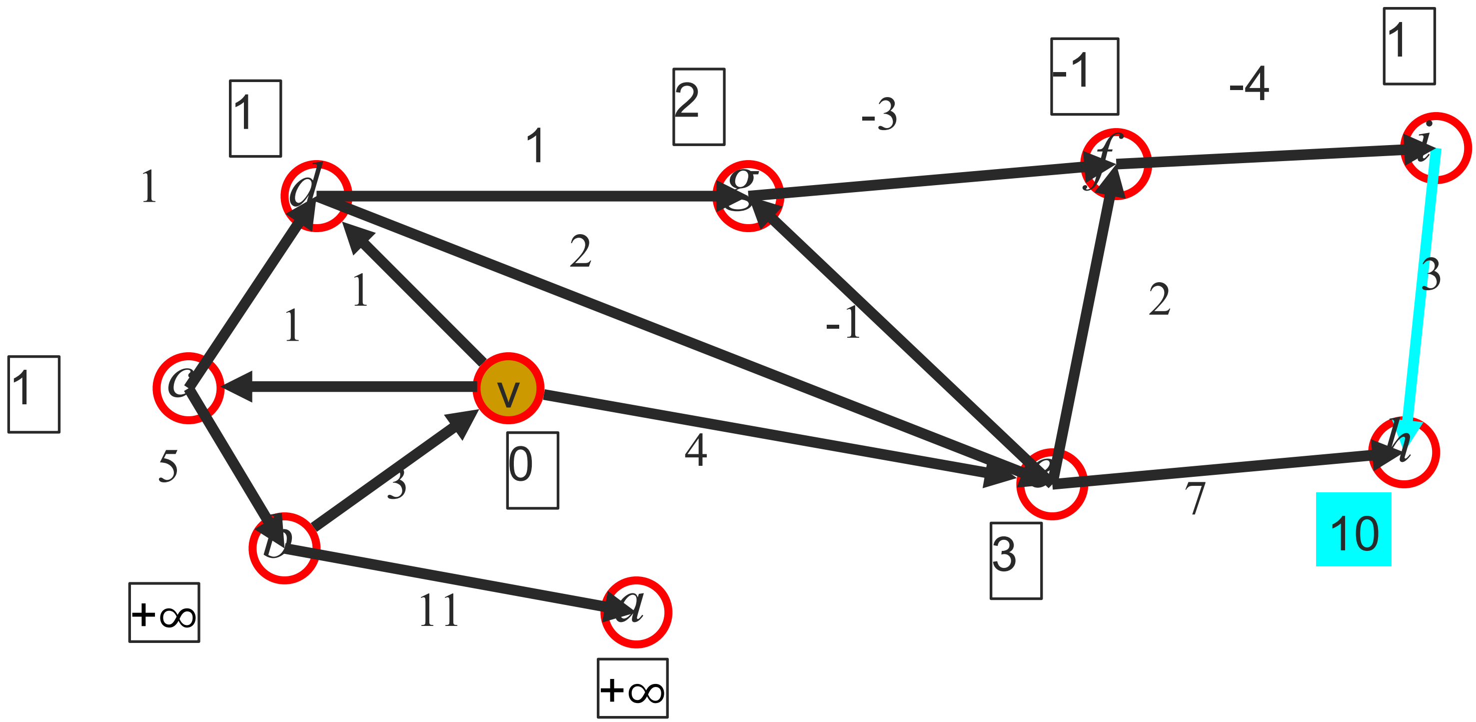
i=1



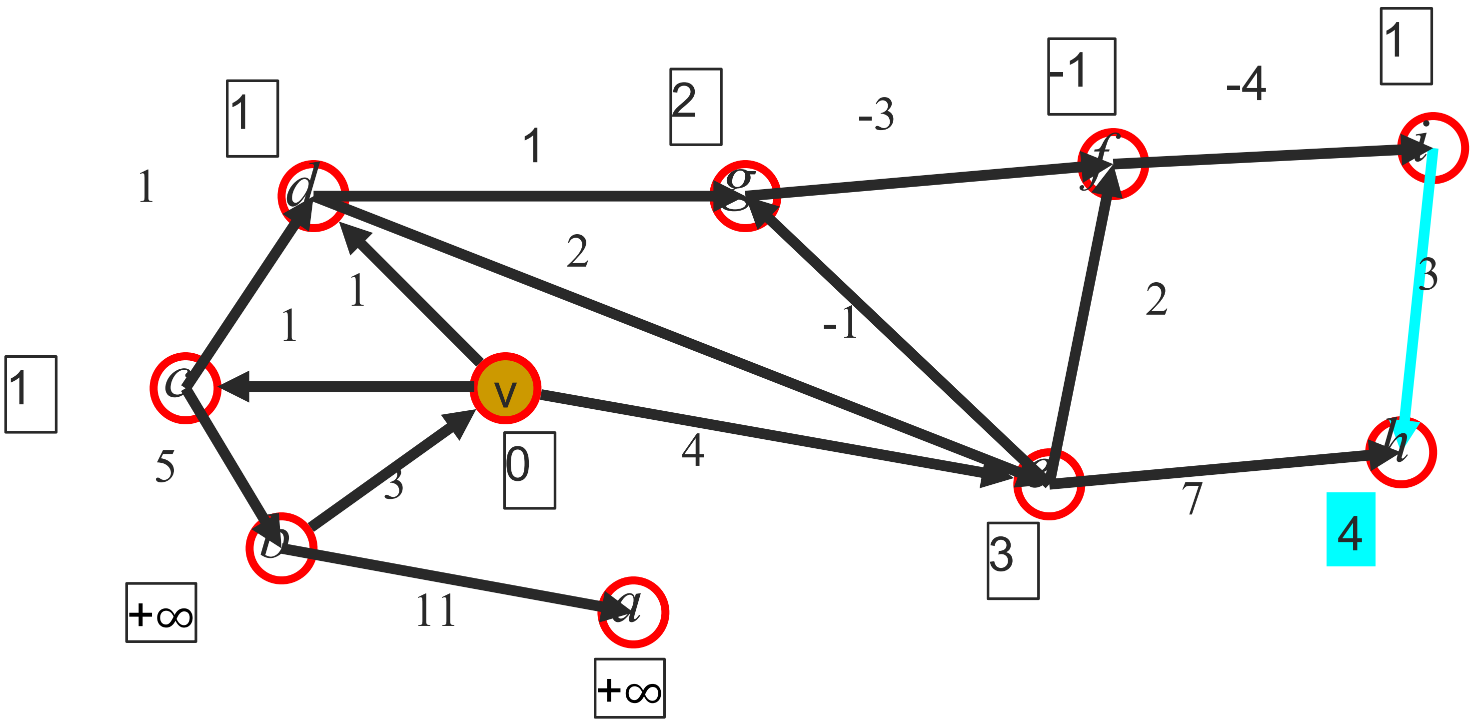
i=1



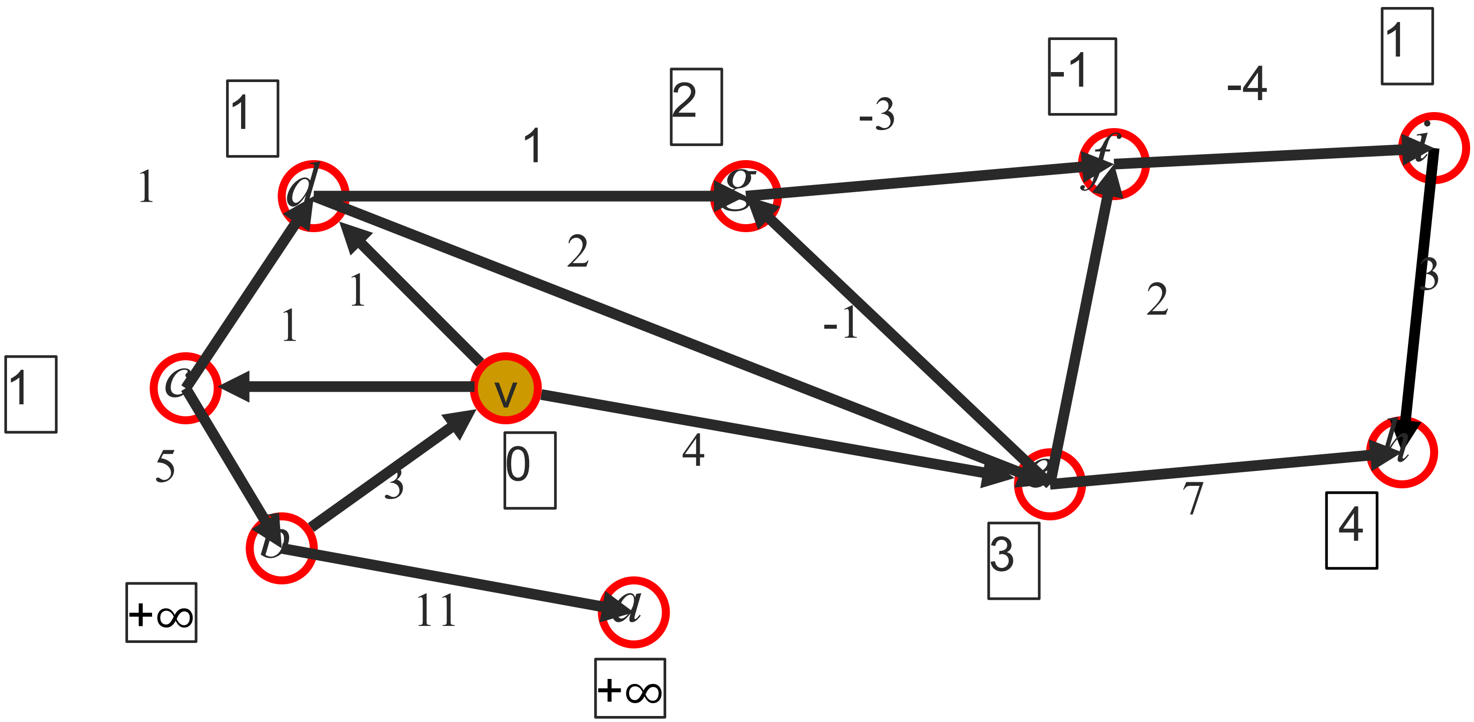
i=1



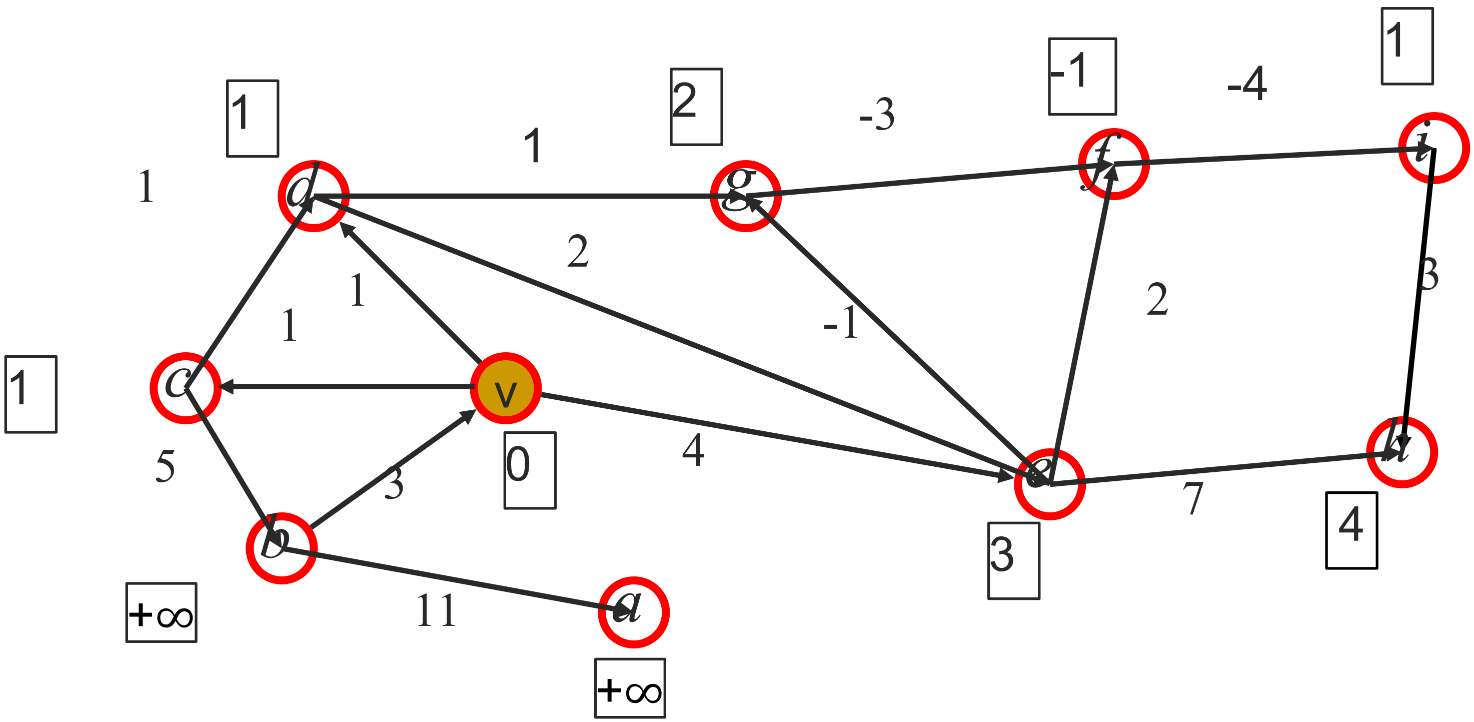
i=1



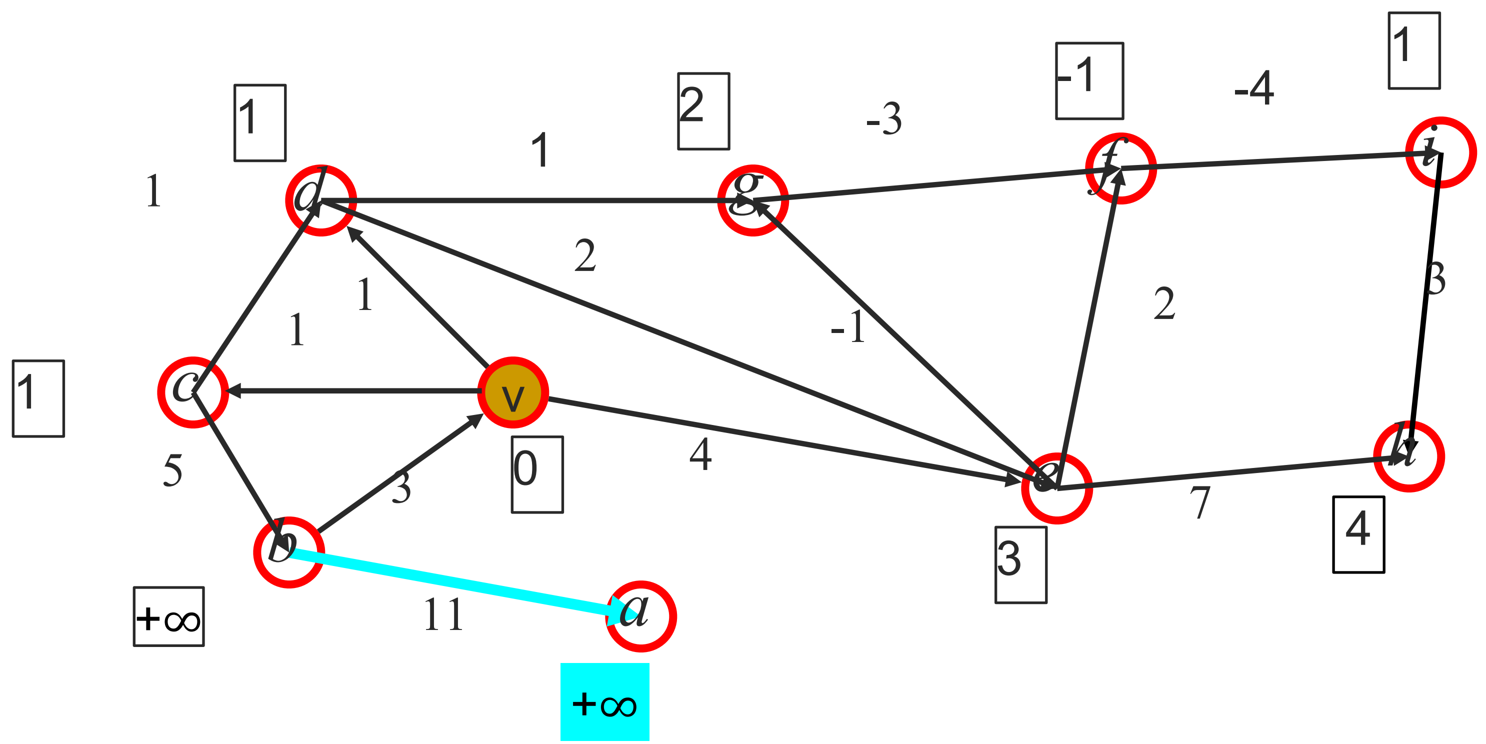
i=1



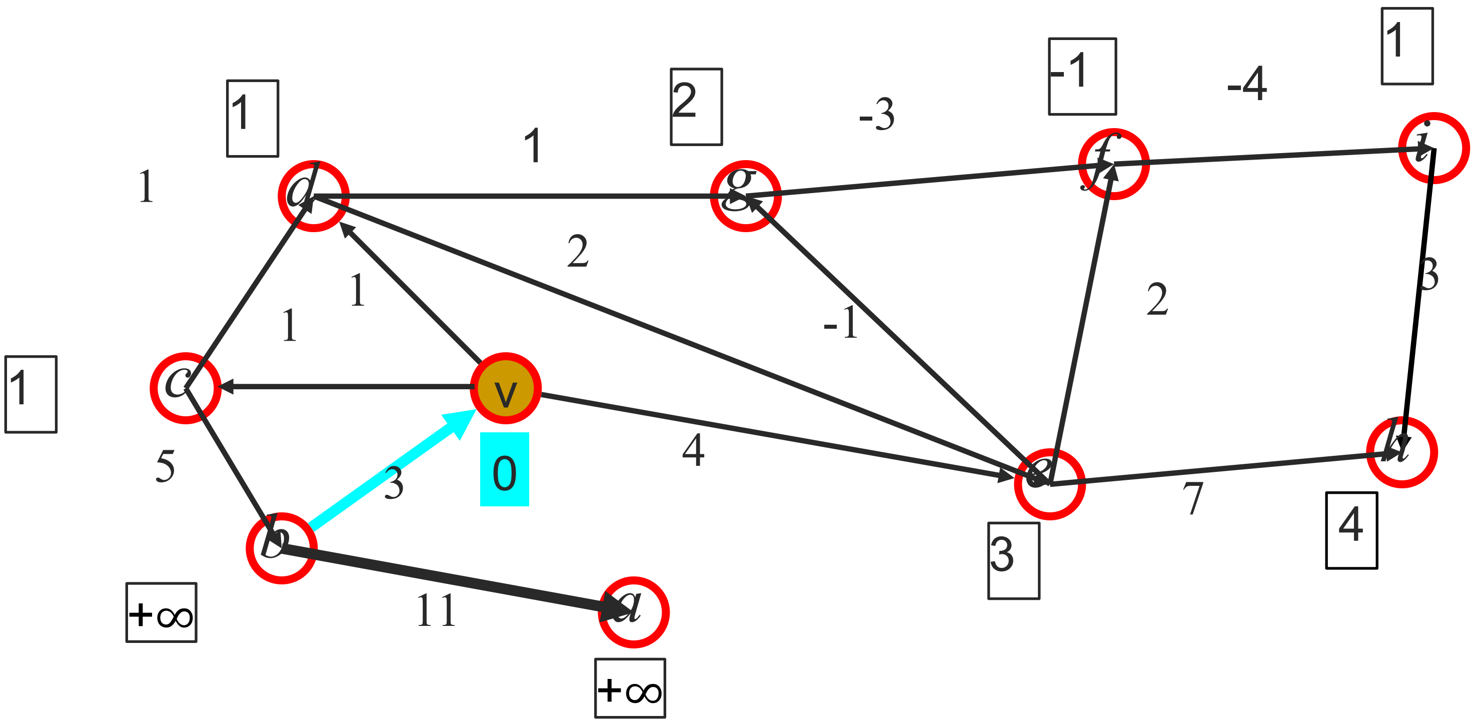
i=1



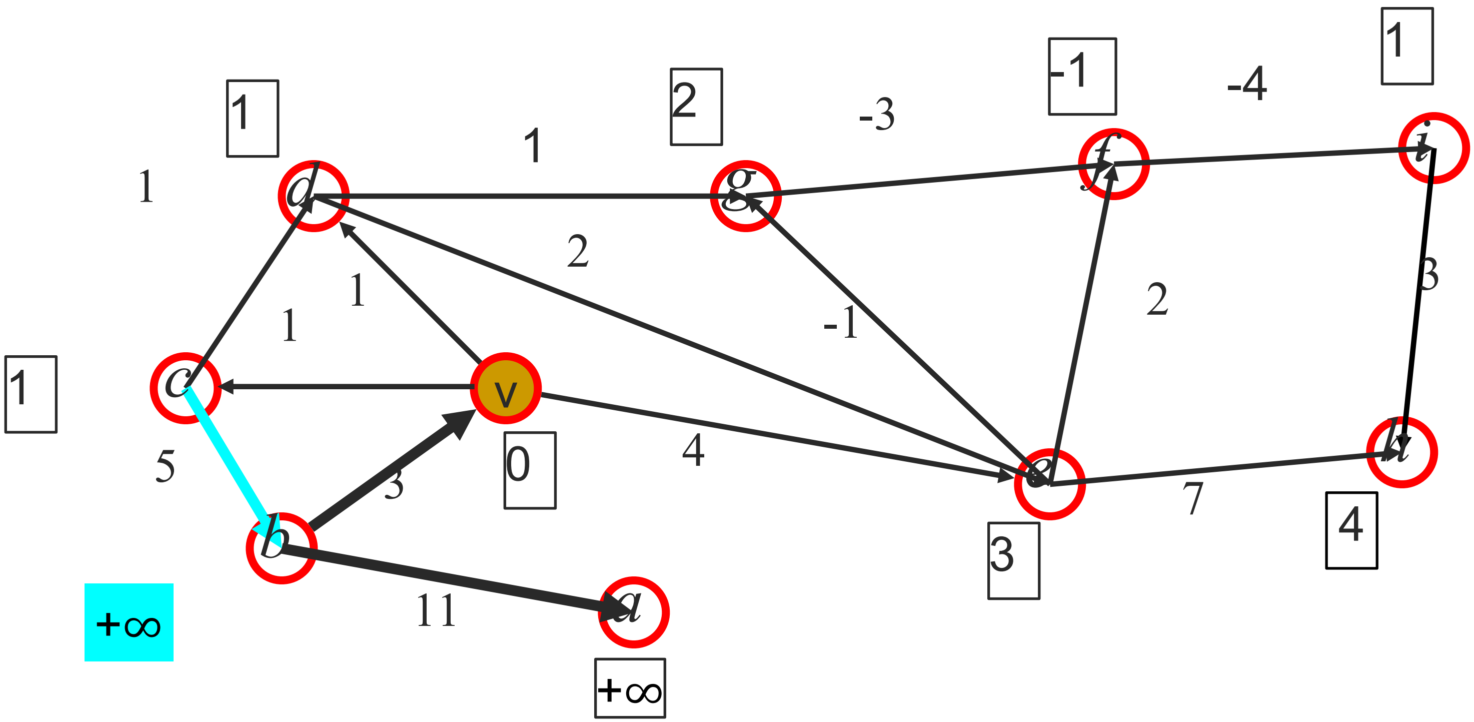
i=2



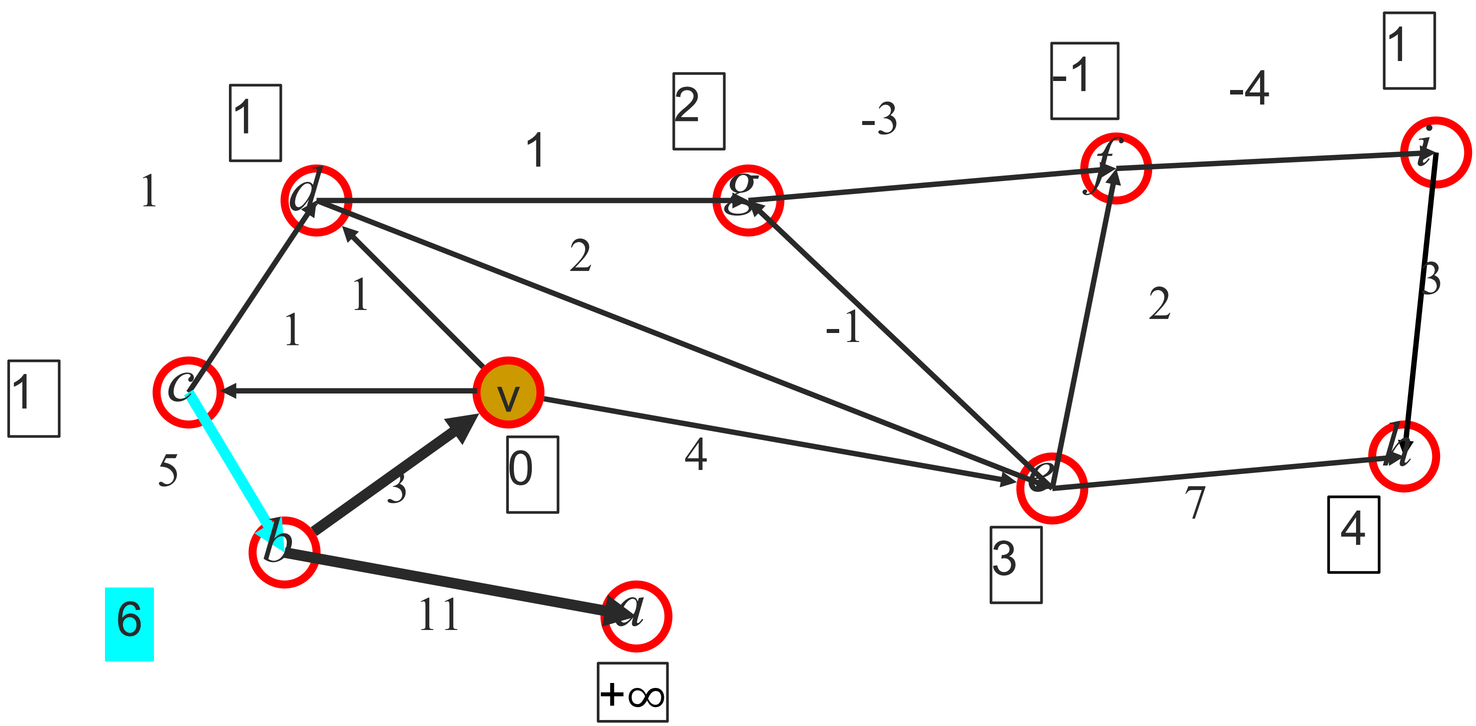
i=2



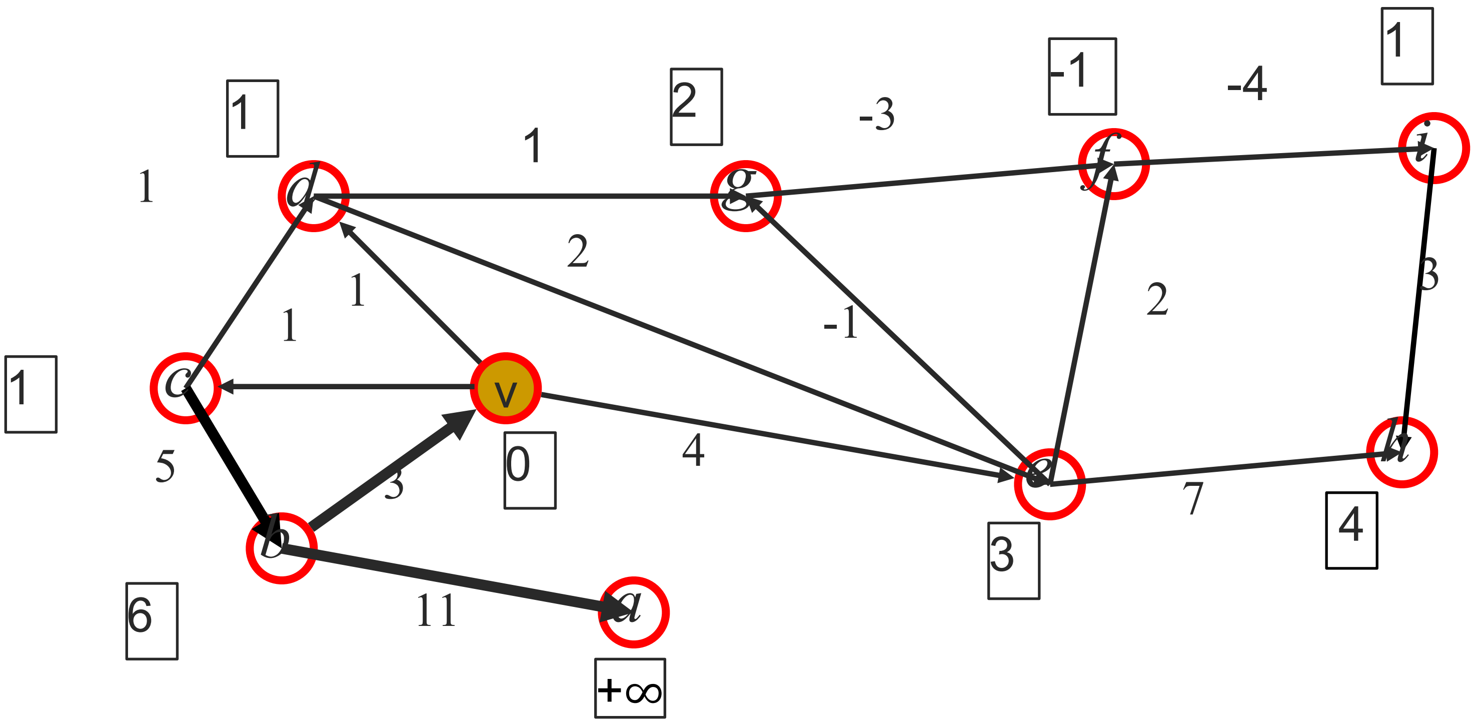
i=2



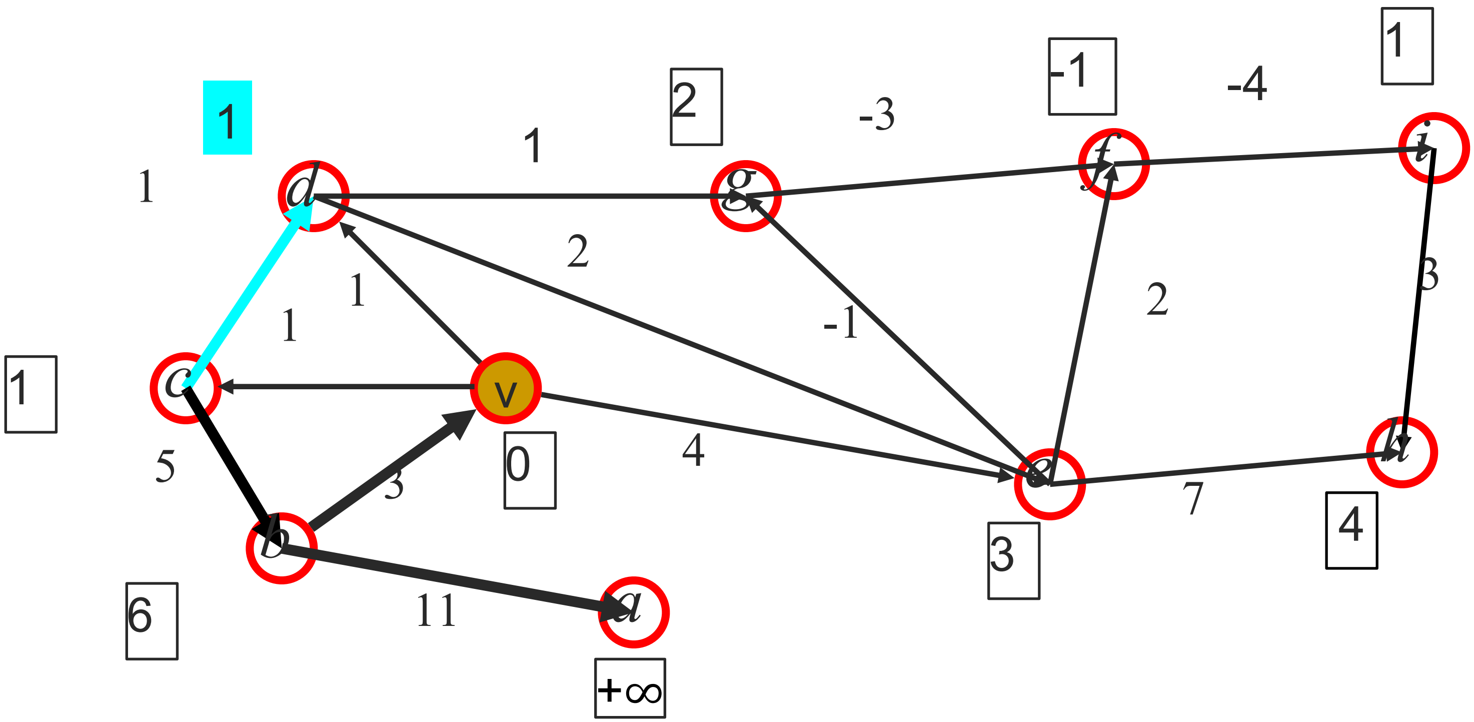
i=2



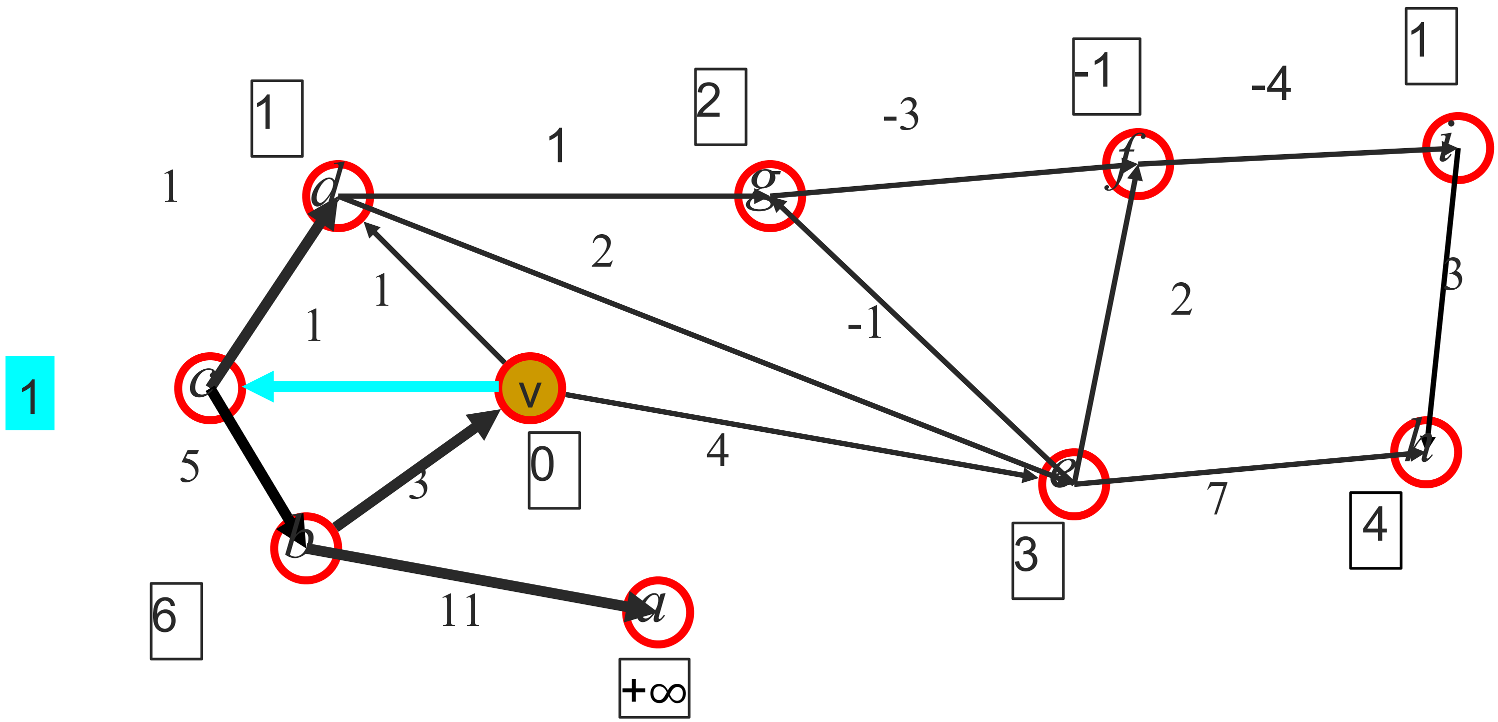
i=2



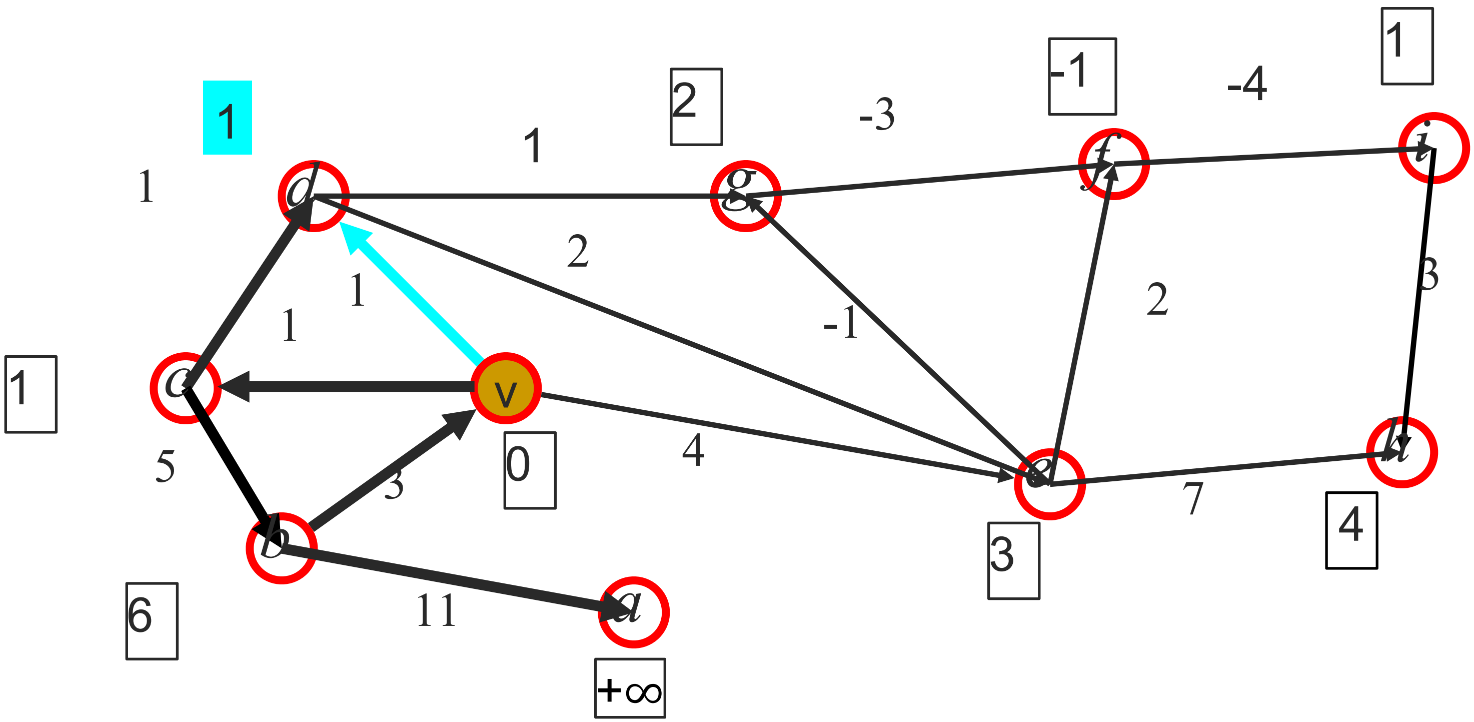
i=2



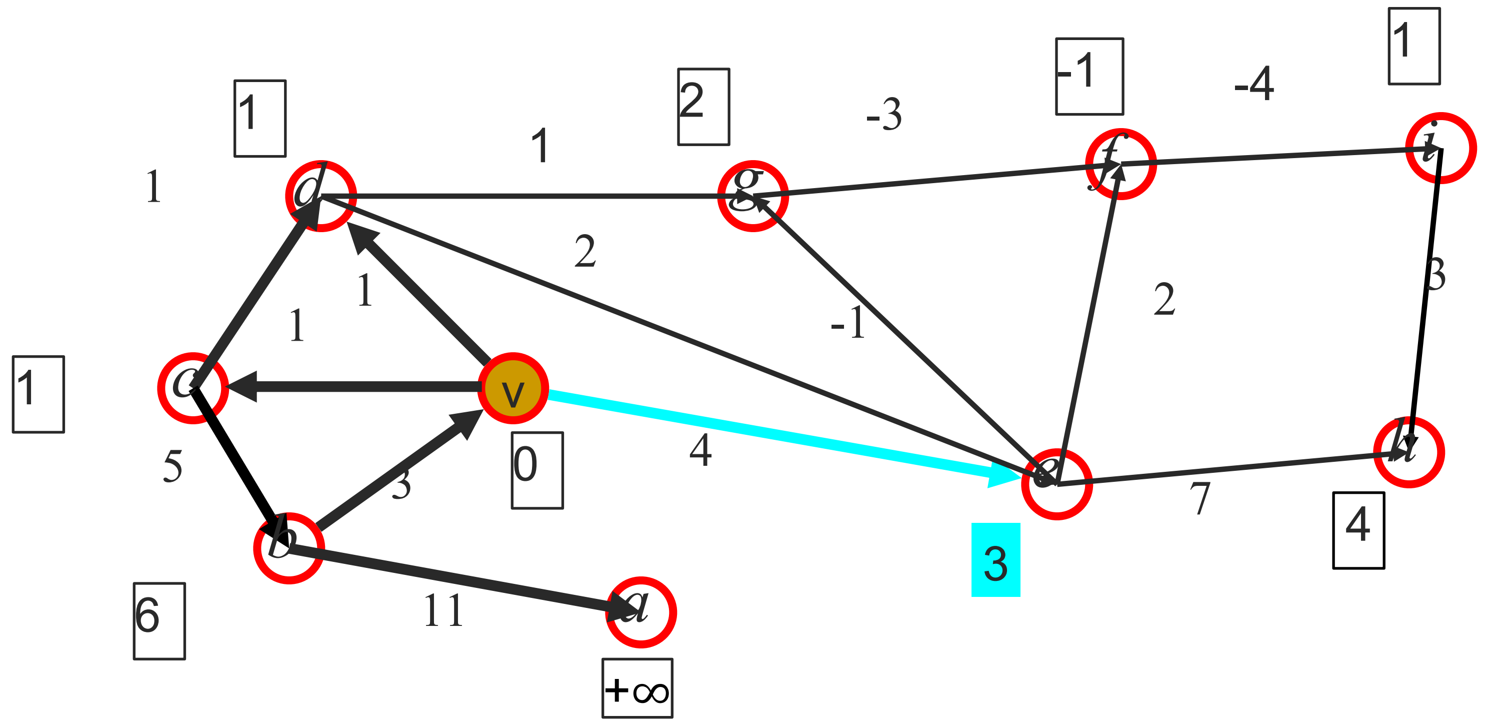
i=2



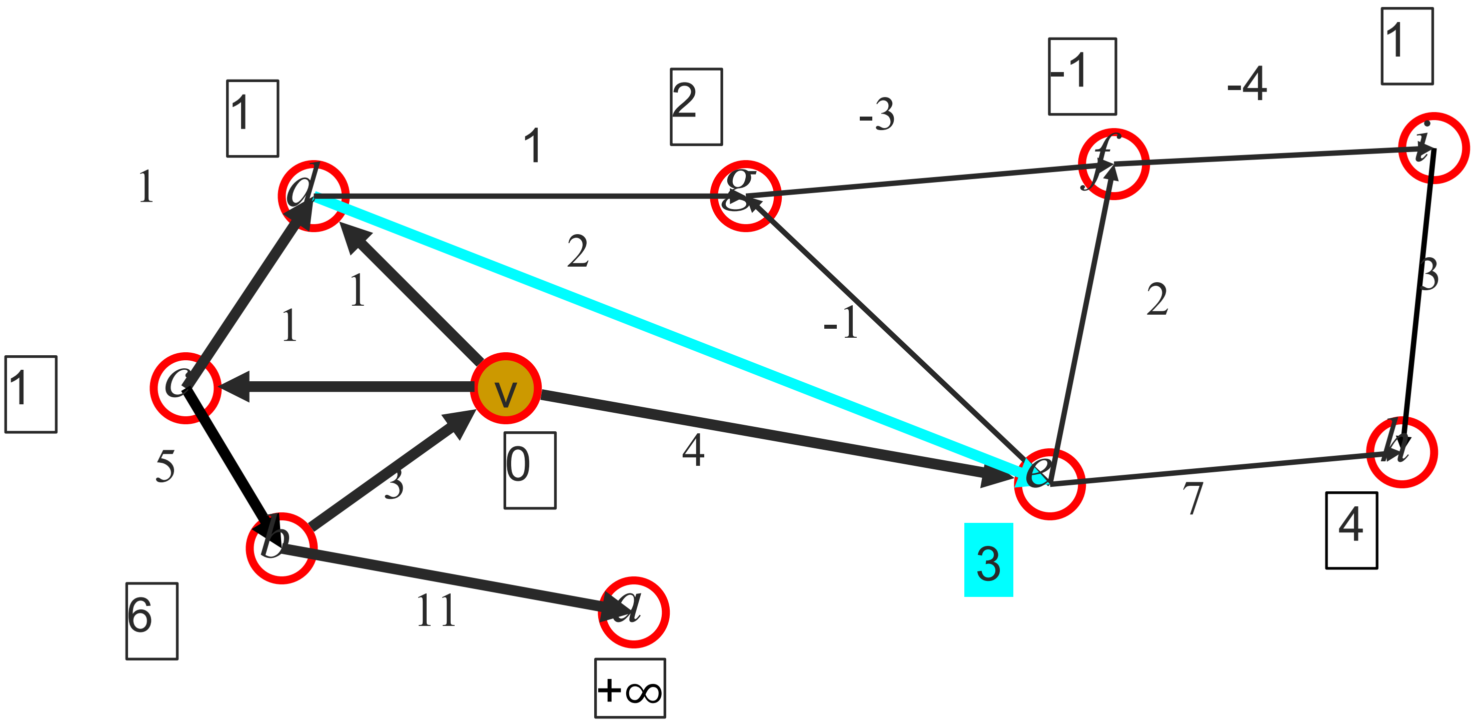
i=2



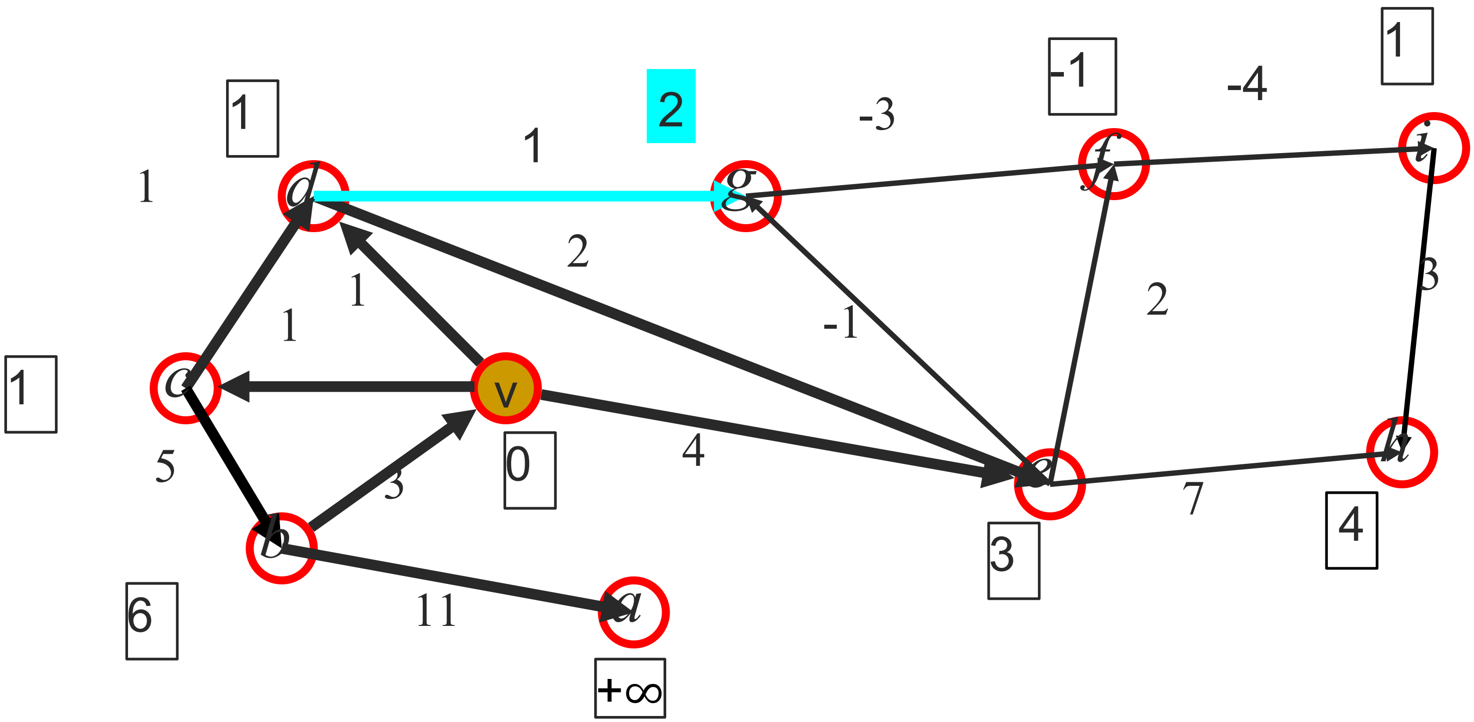
i=2



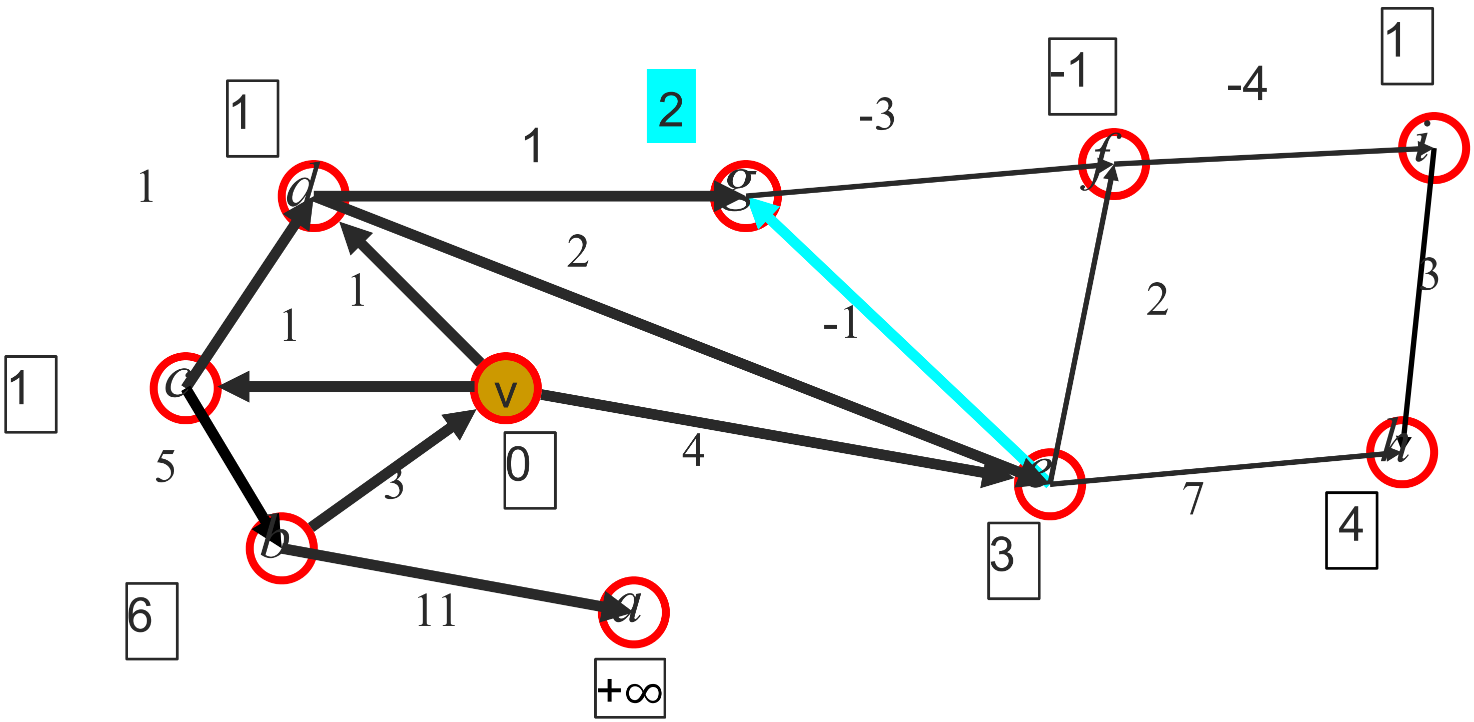
i=2



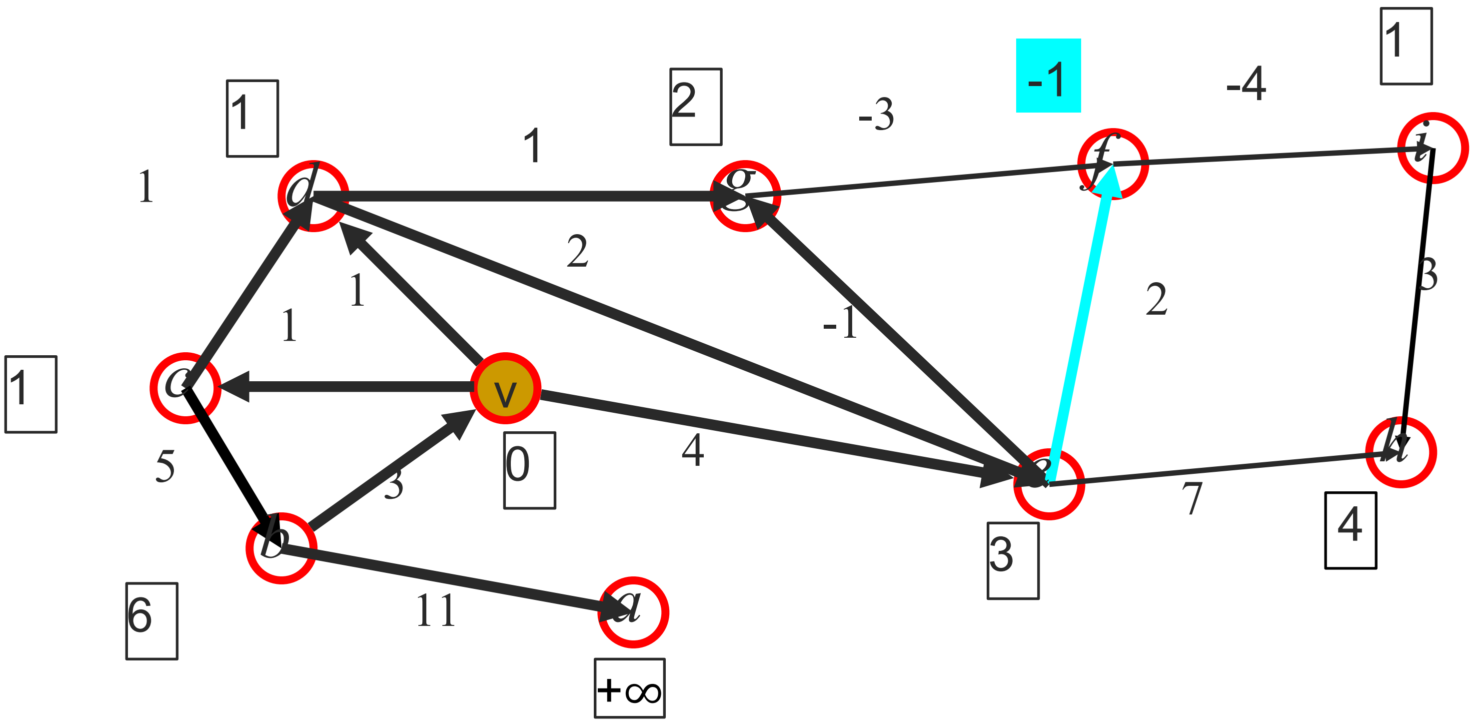
i=2



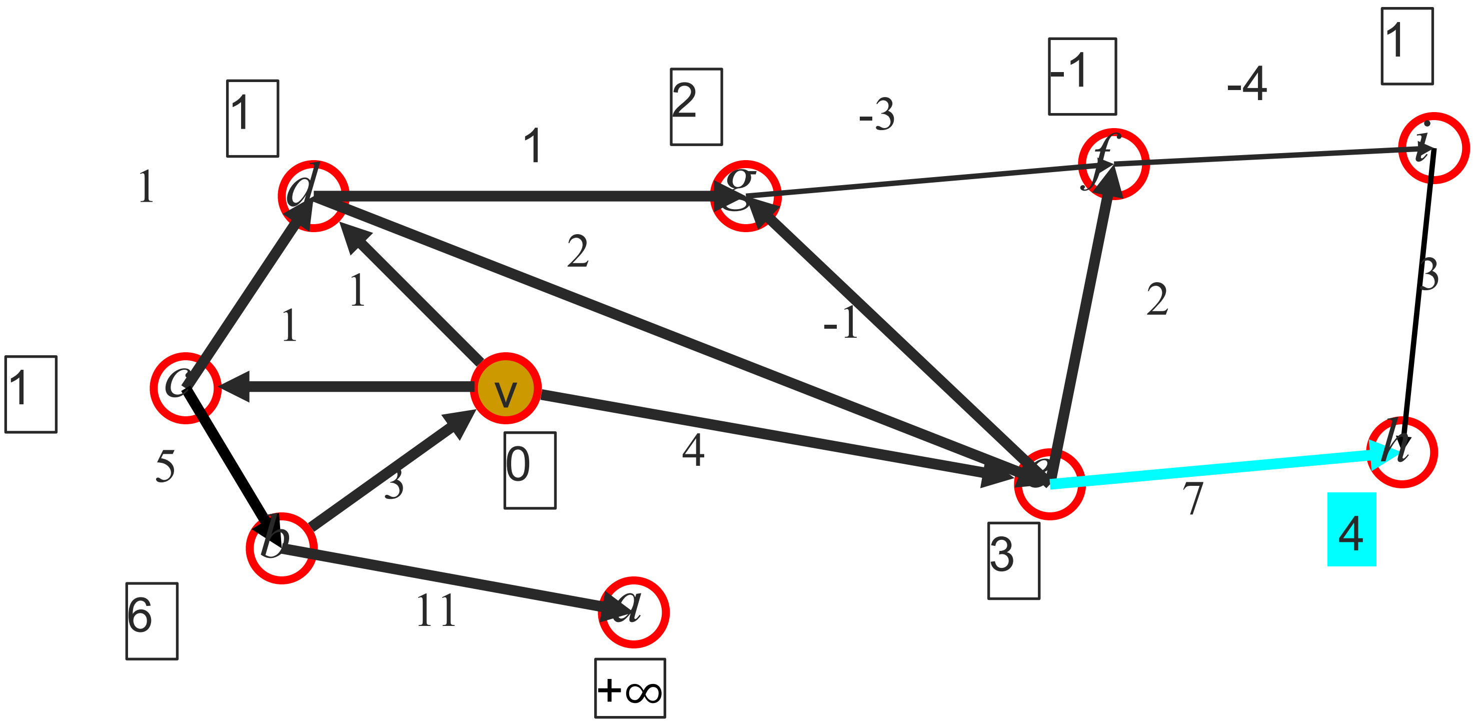
i=2



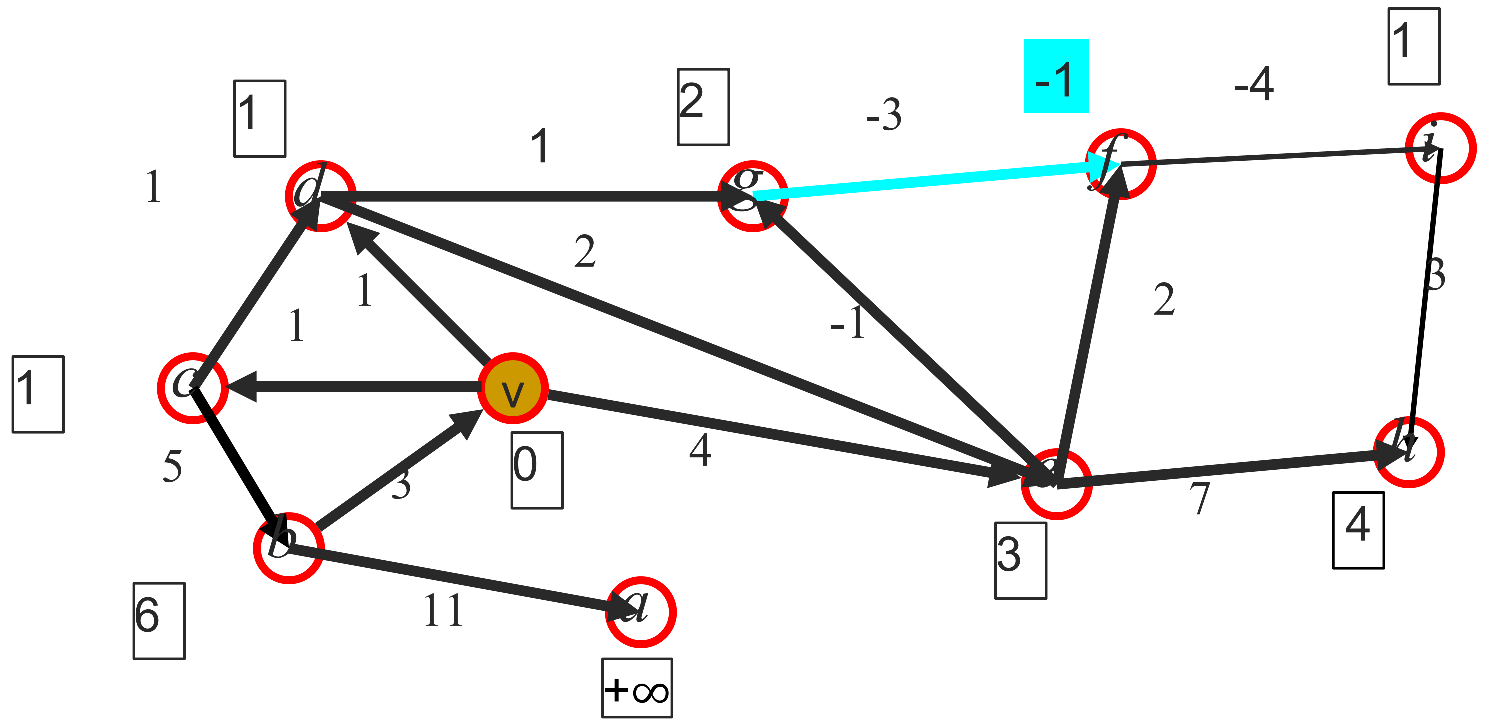
i=2



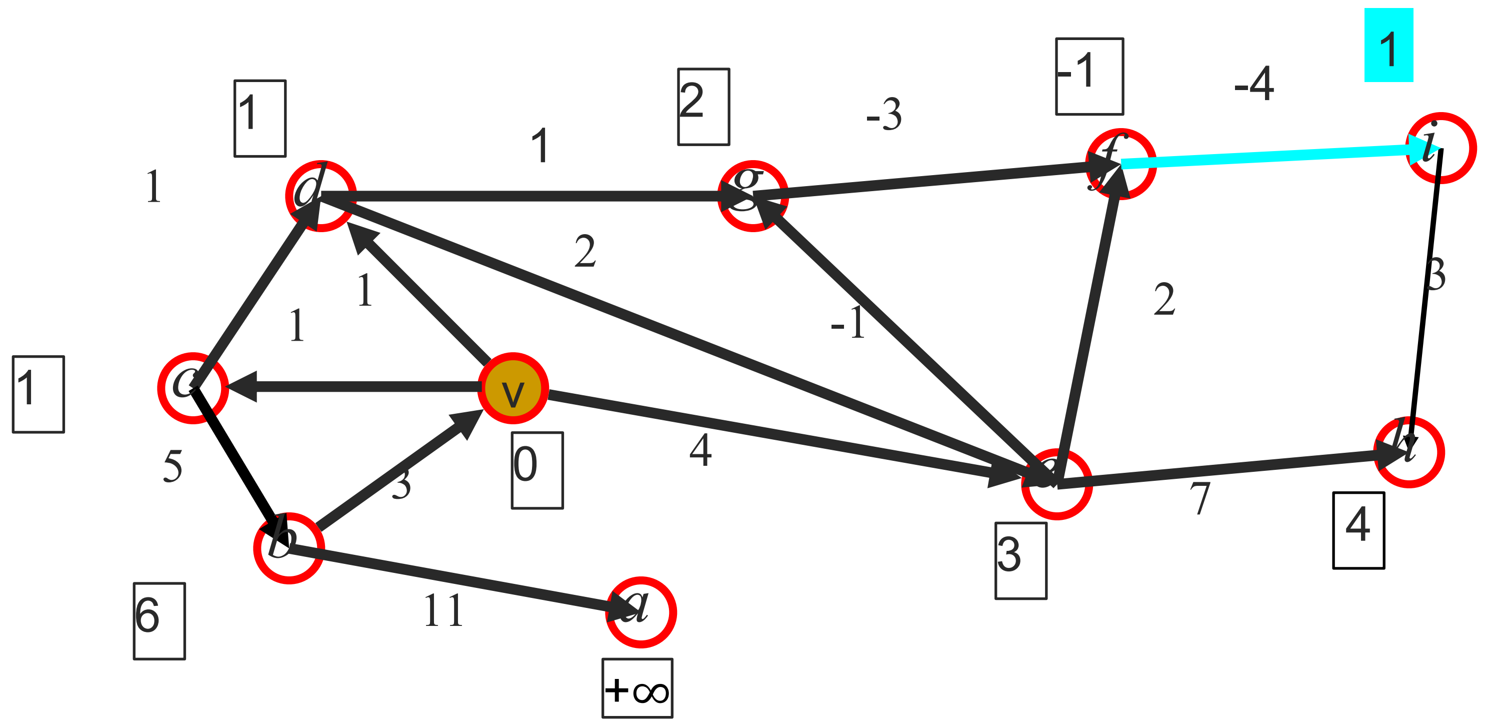
i=2



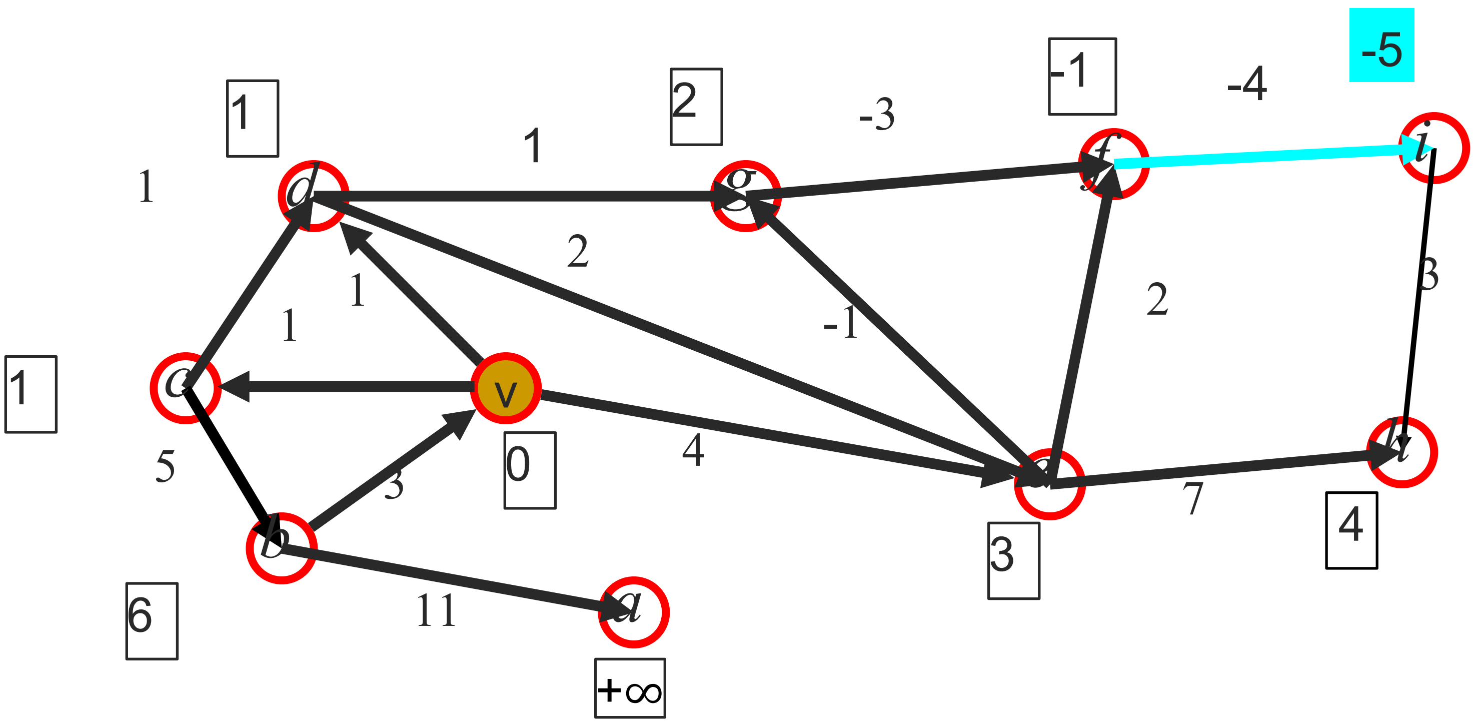
i=2



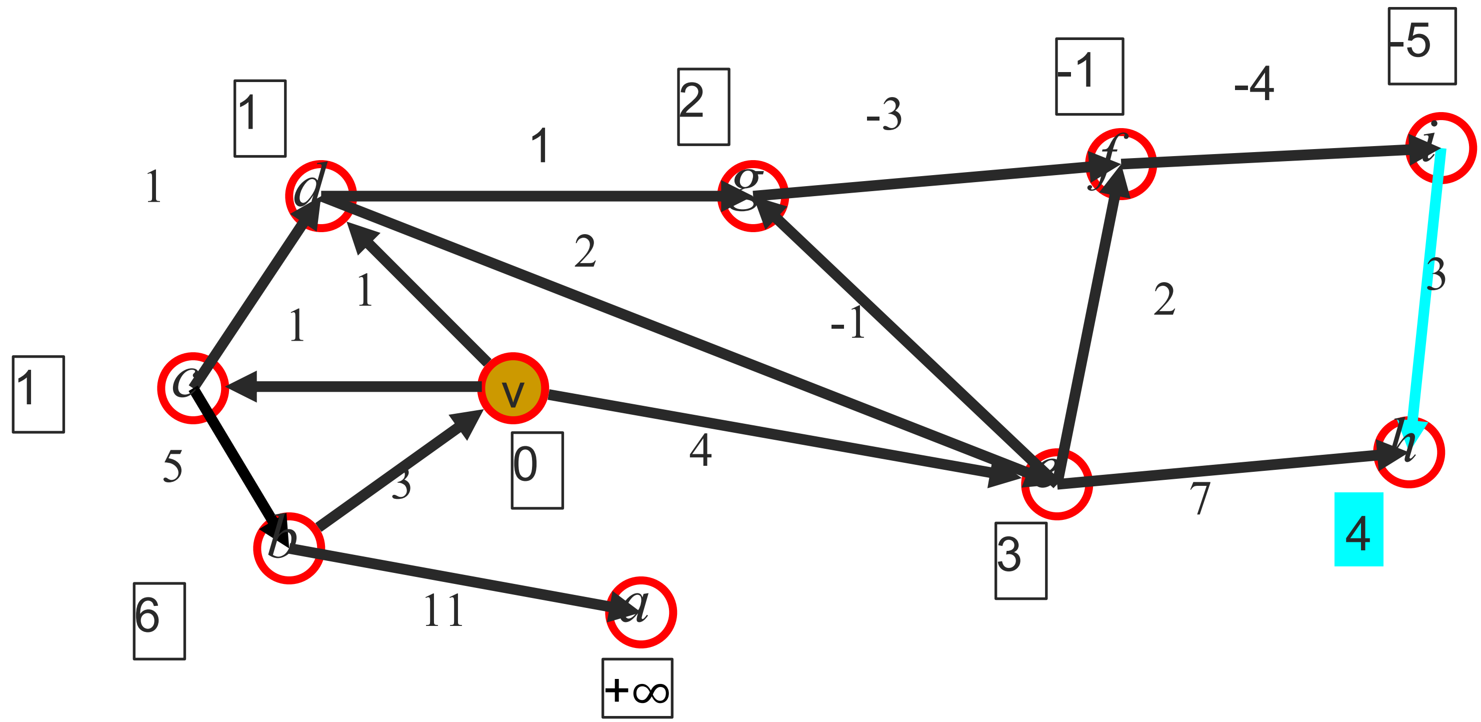
i=2



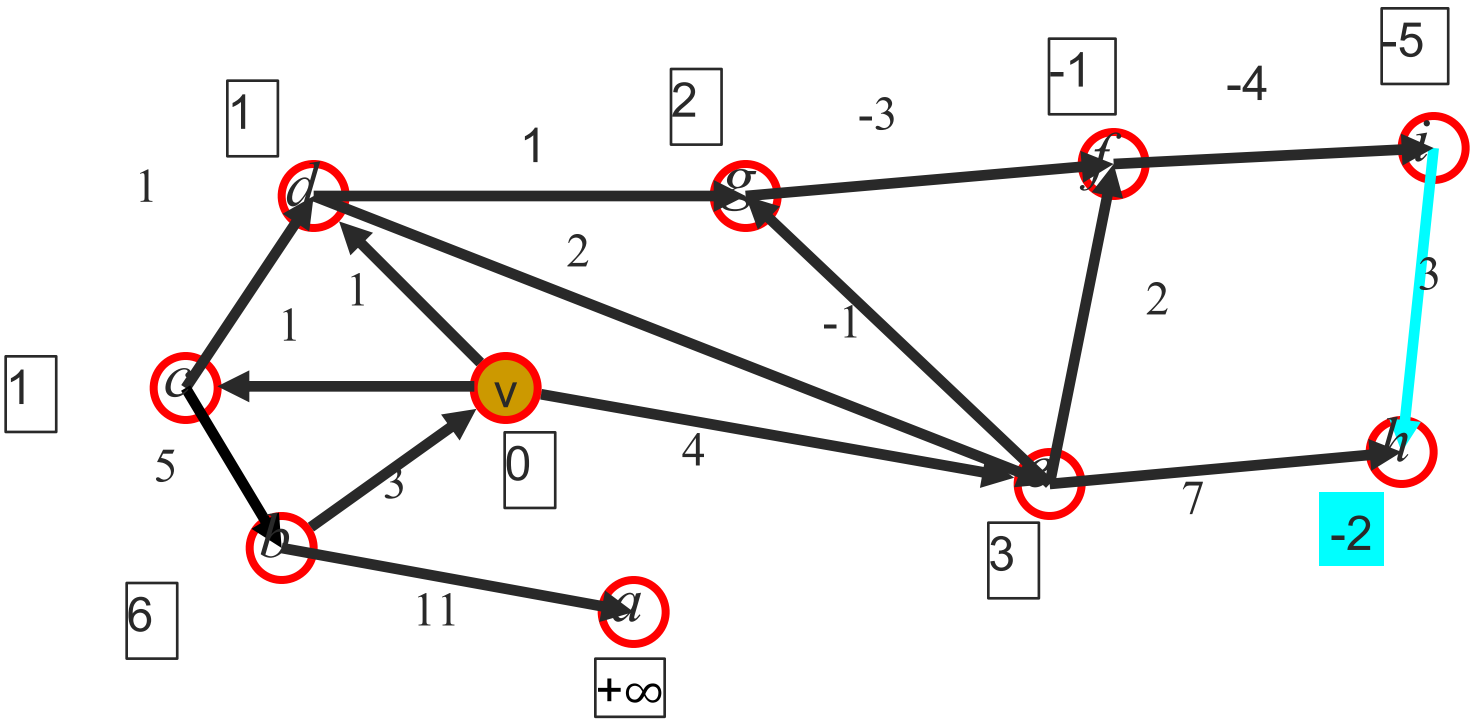
i=2



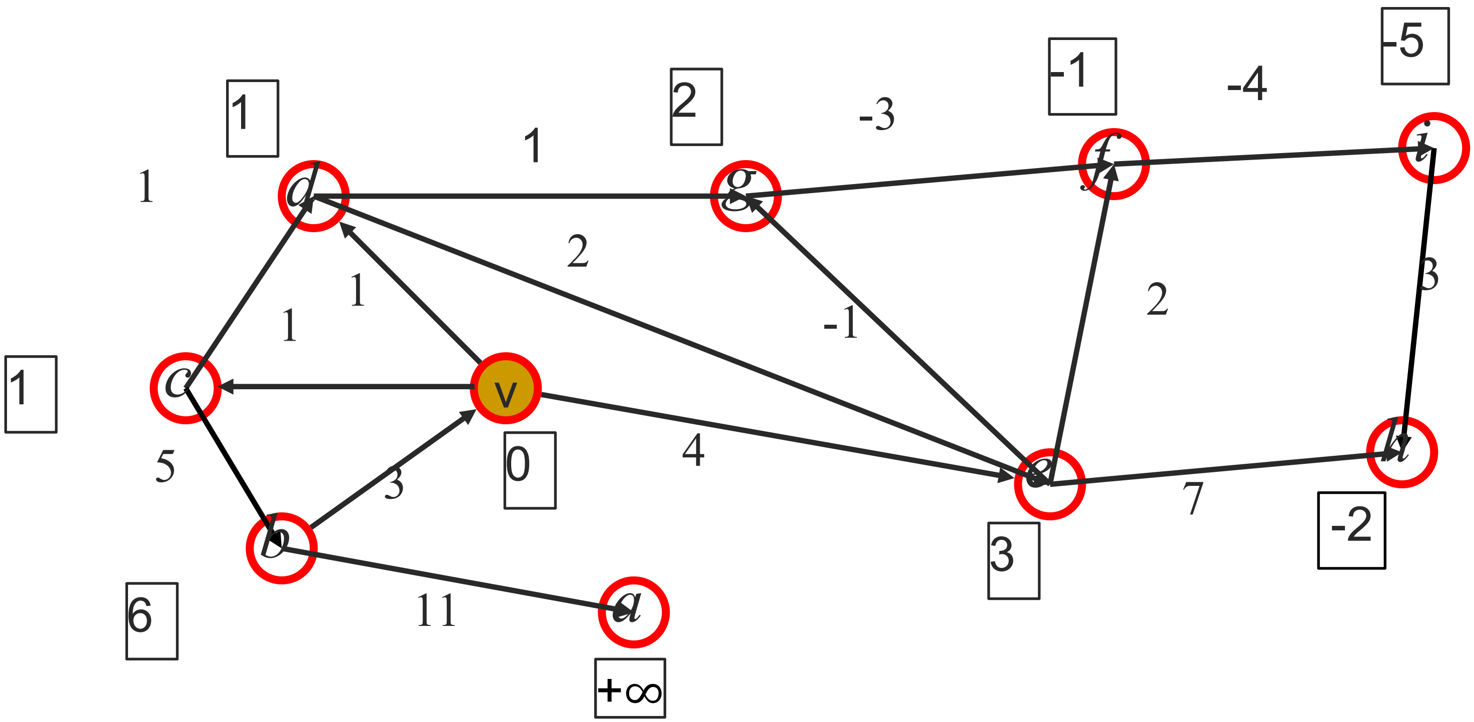
i=2



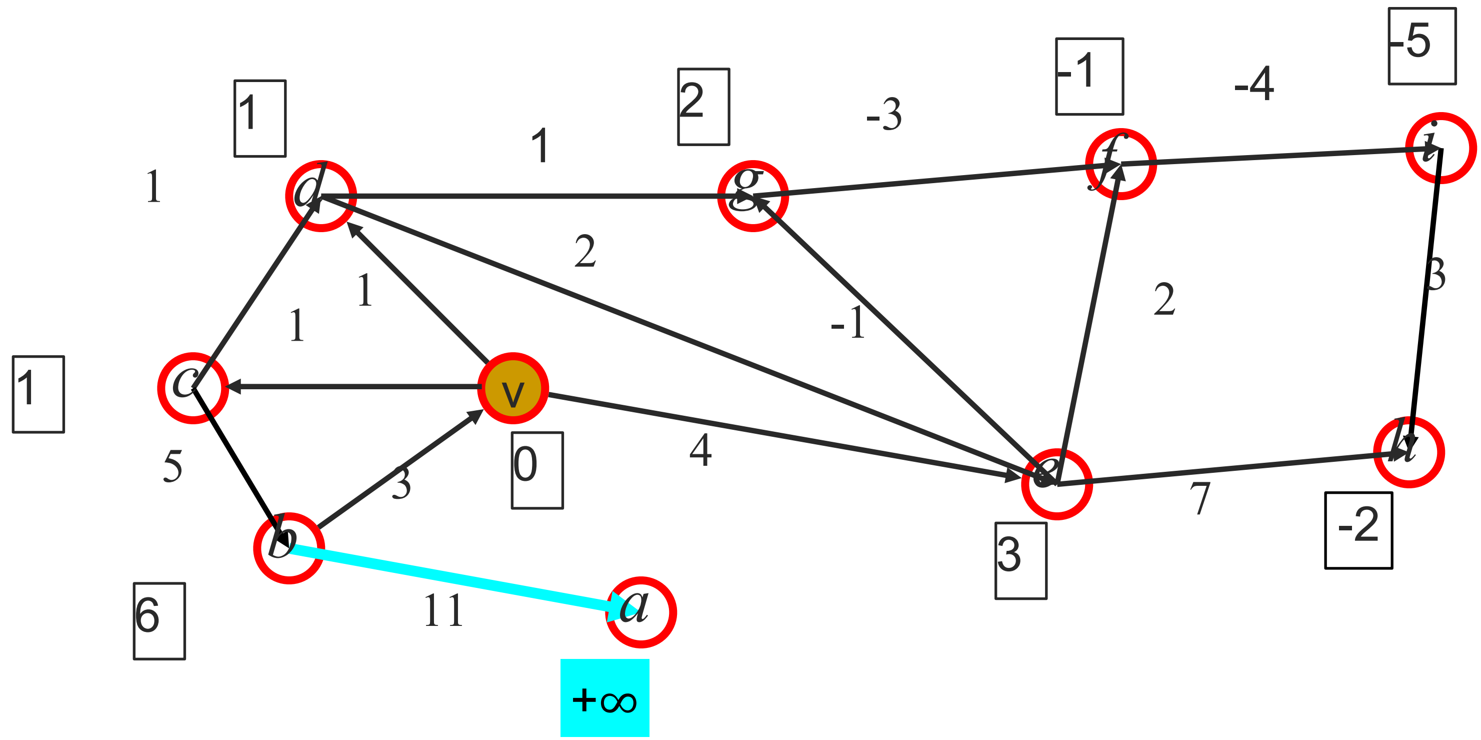
i=2



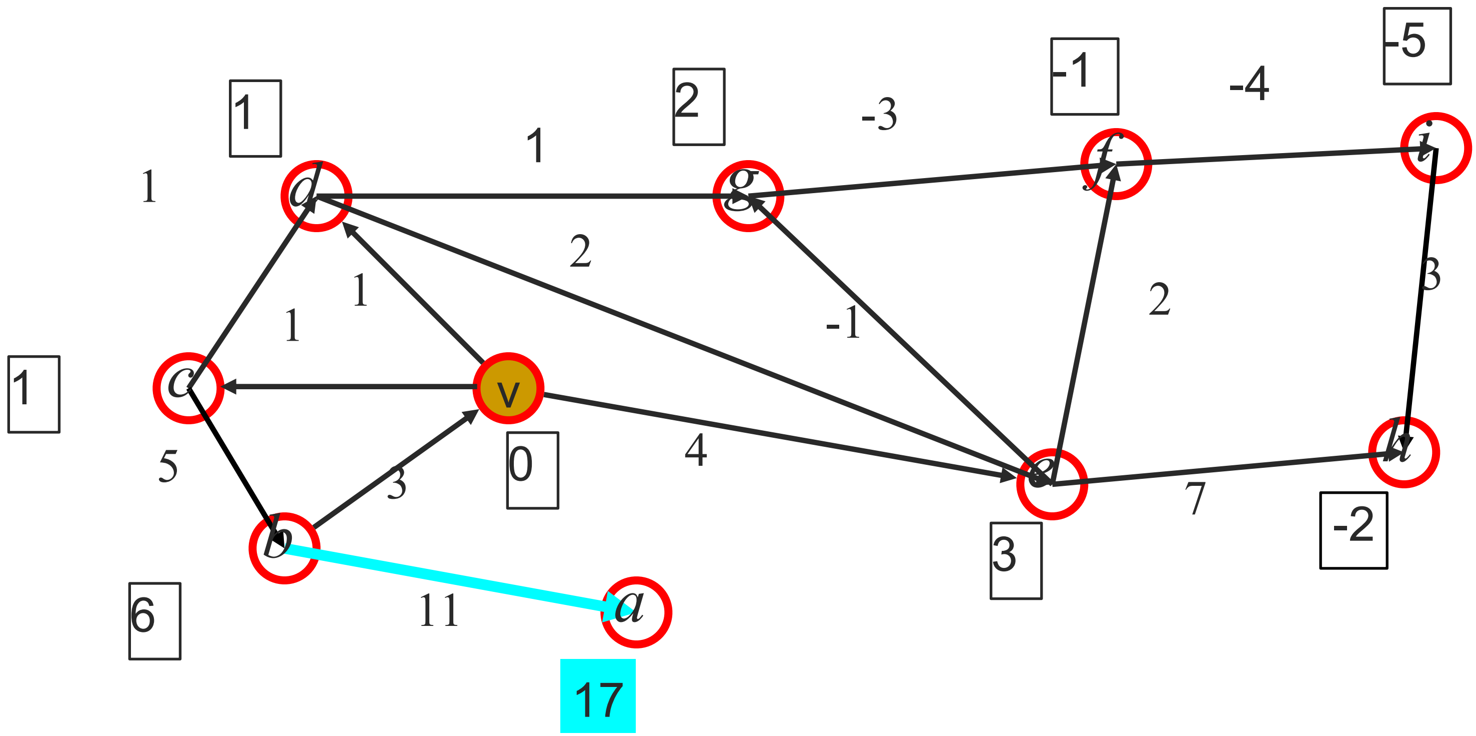
i=2



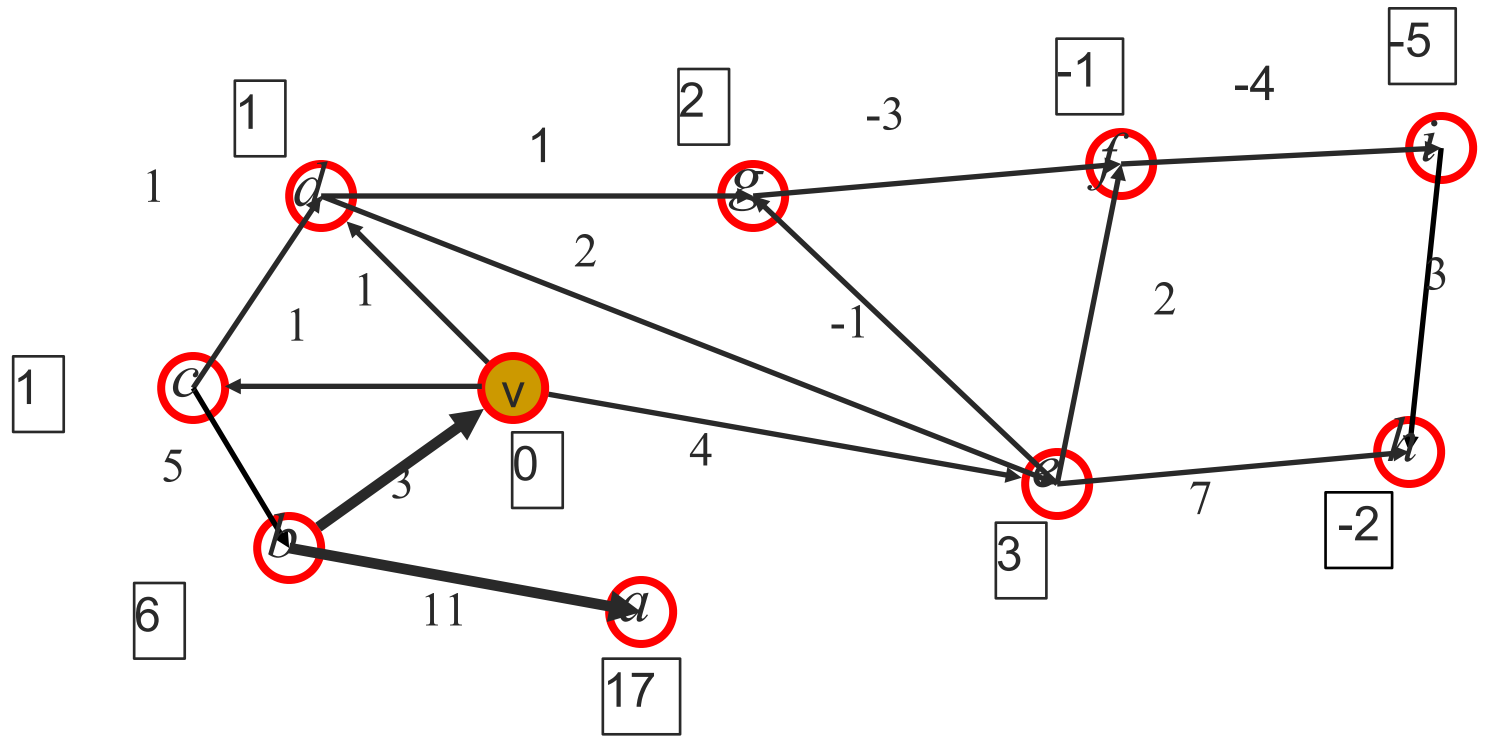
i=3



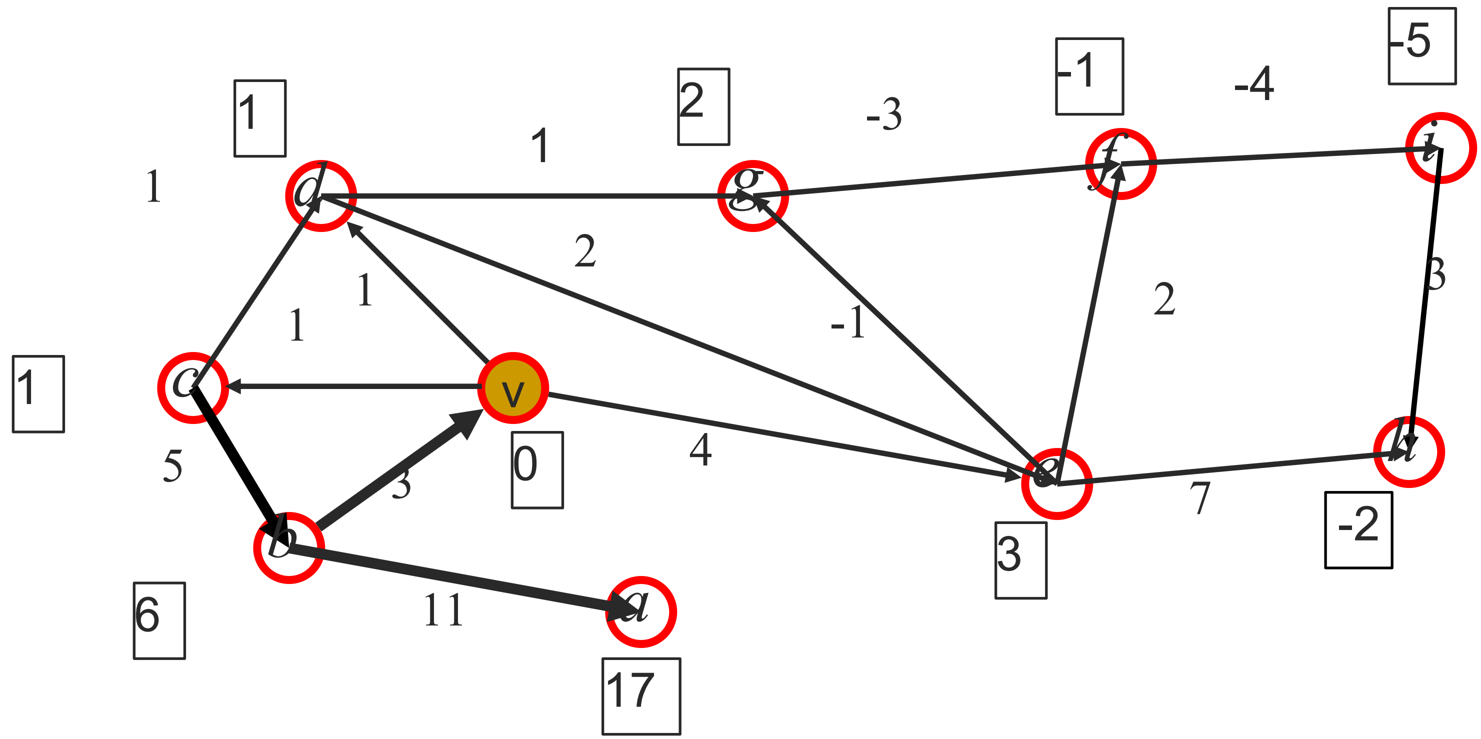
i=3



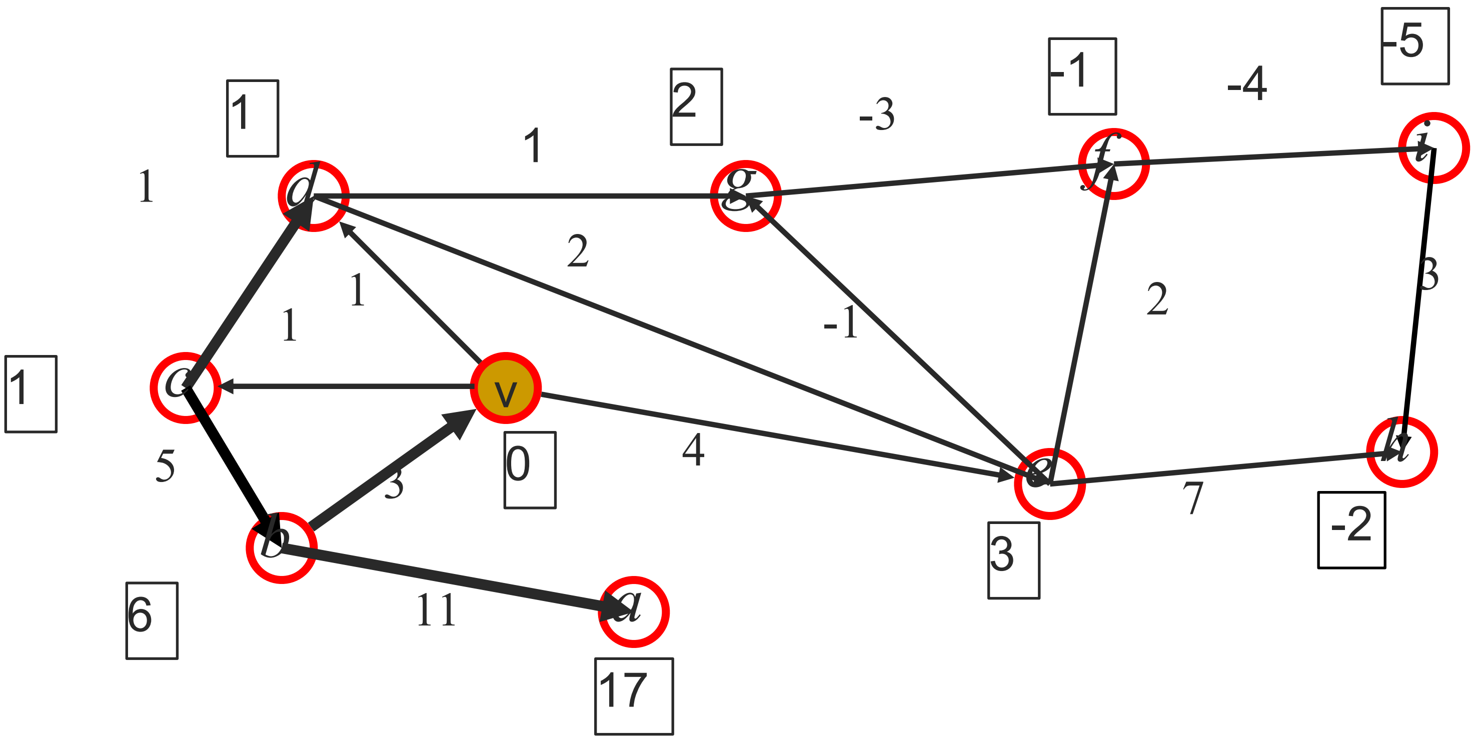
i=3



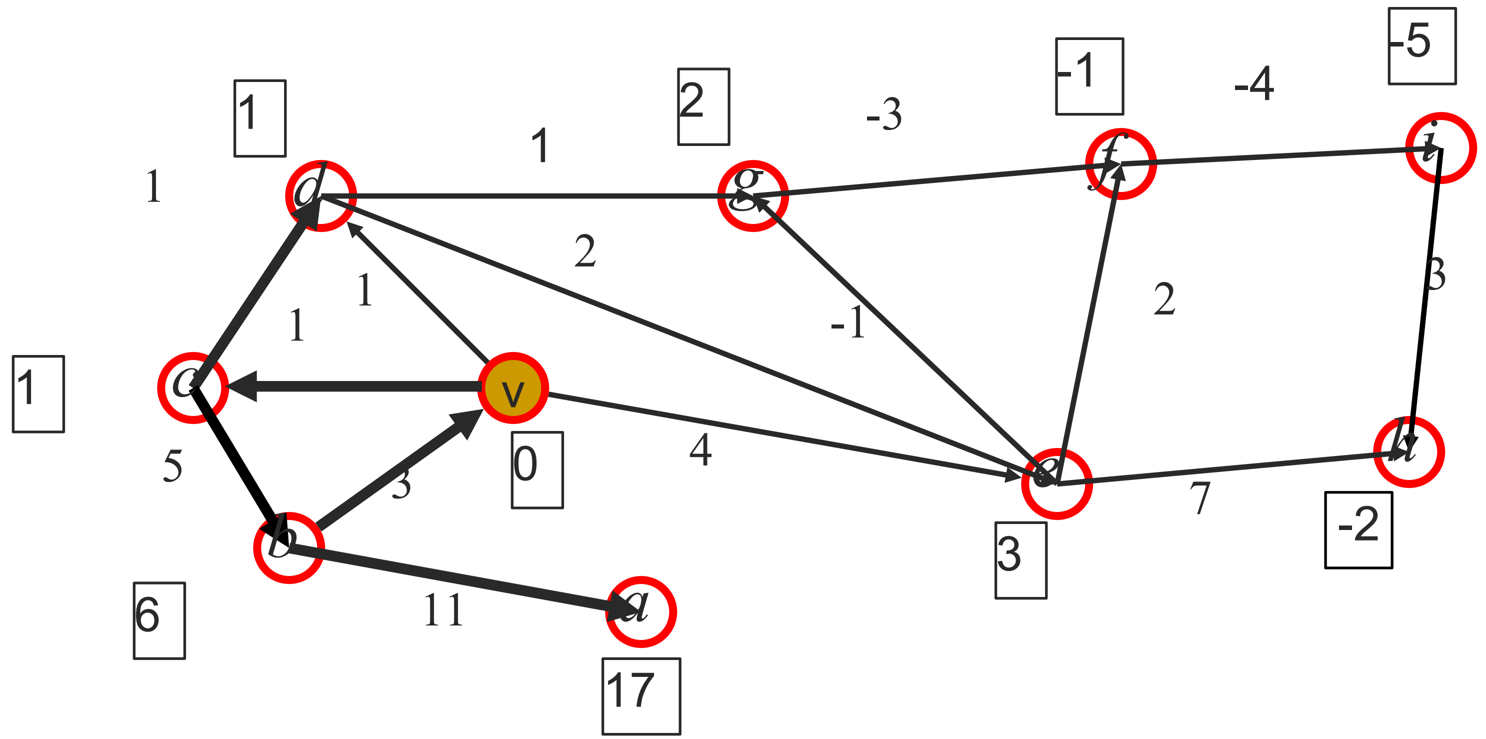
i=3



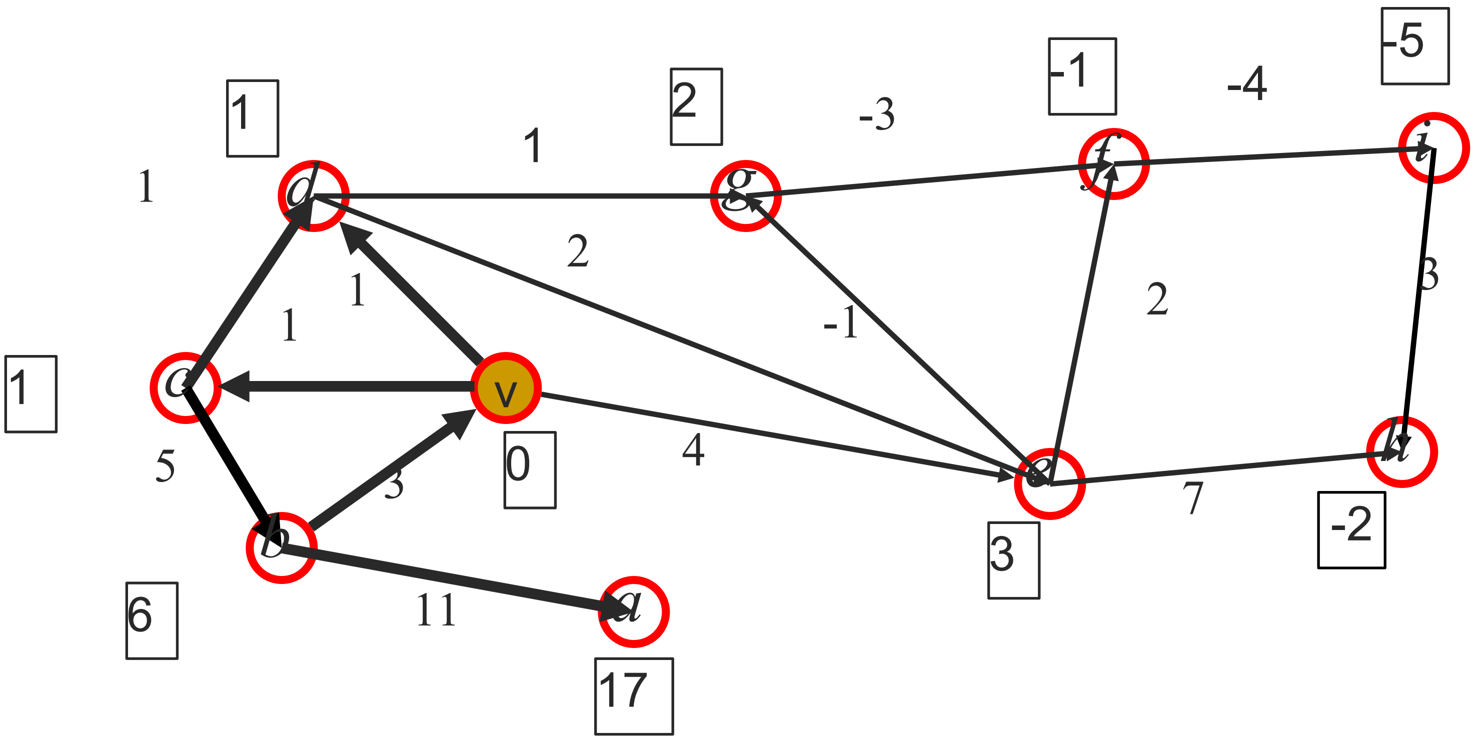
i=3



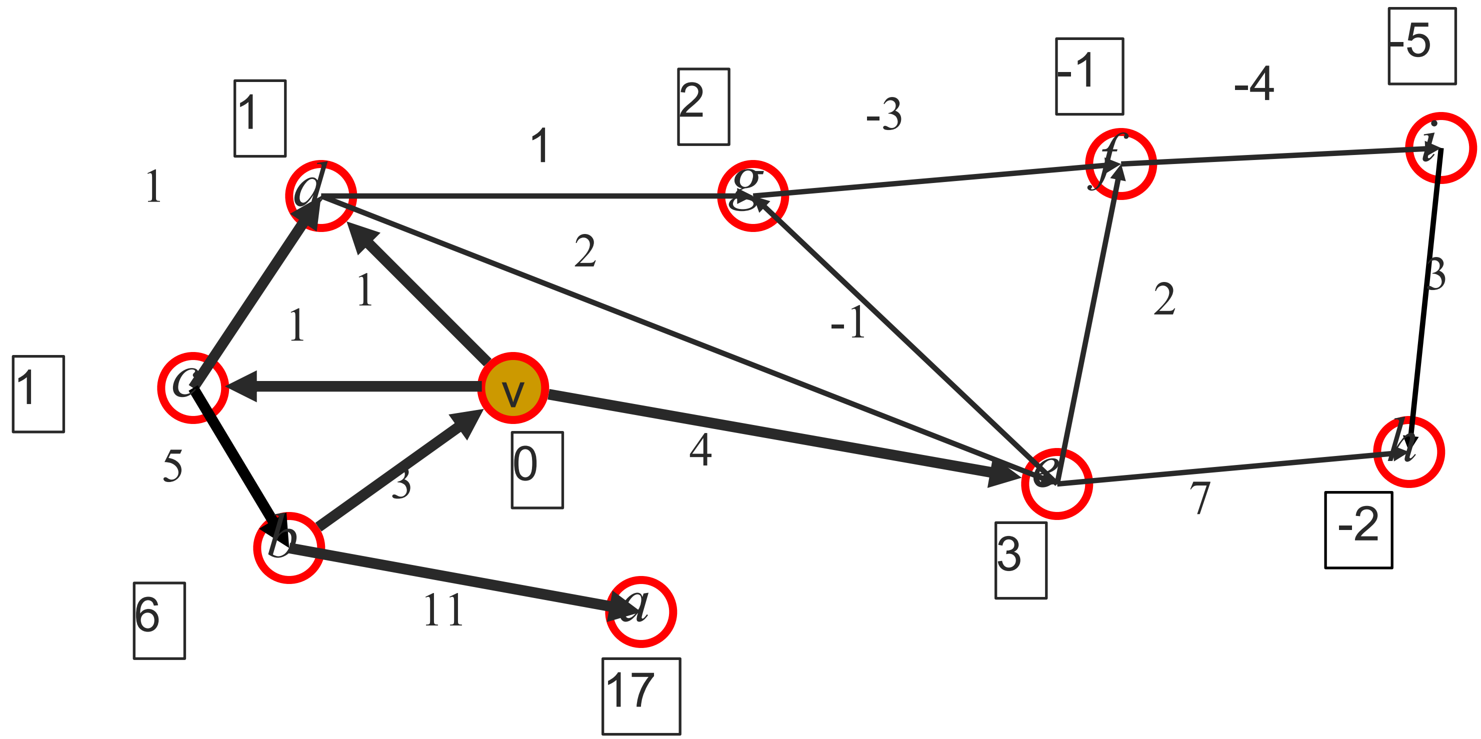
i=3



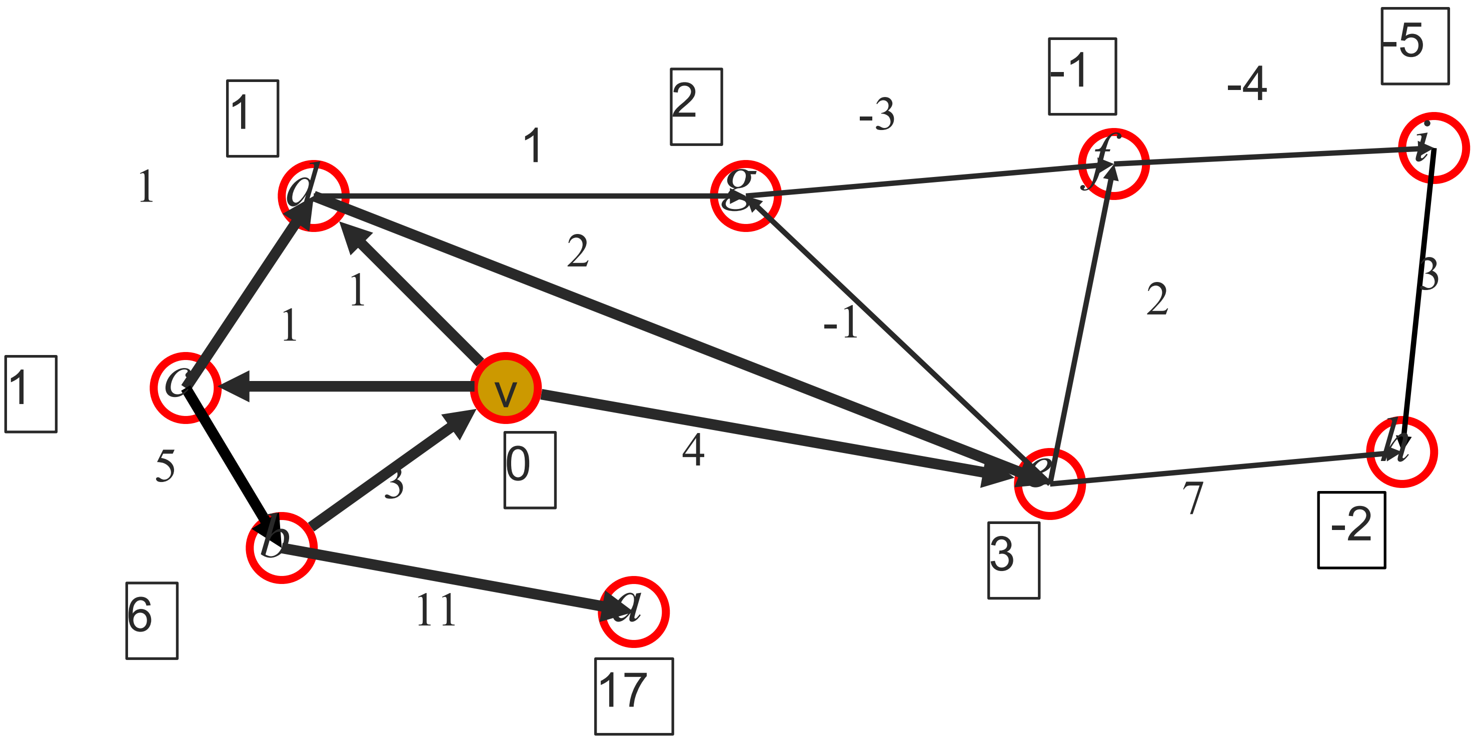
i=3



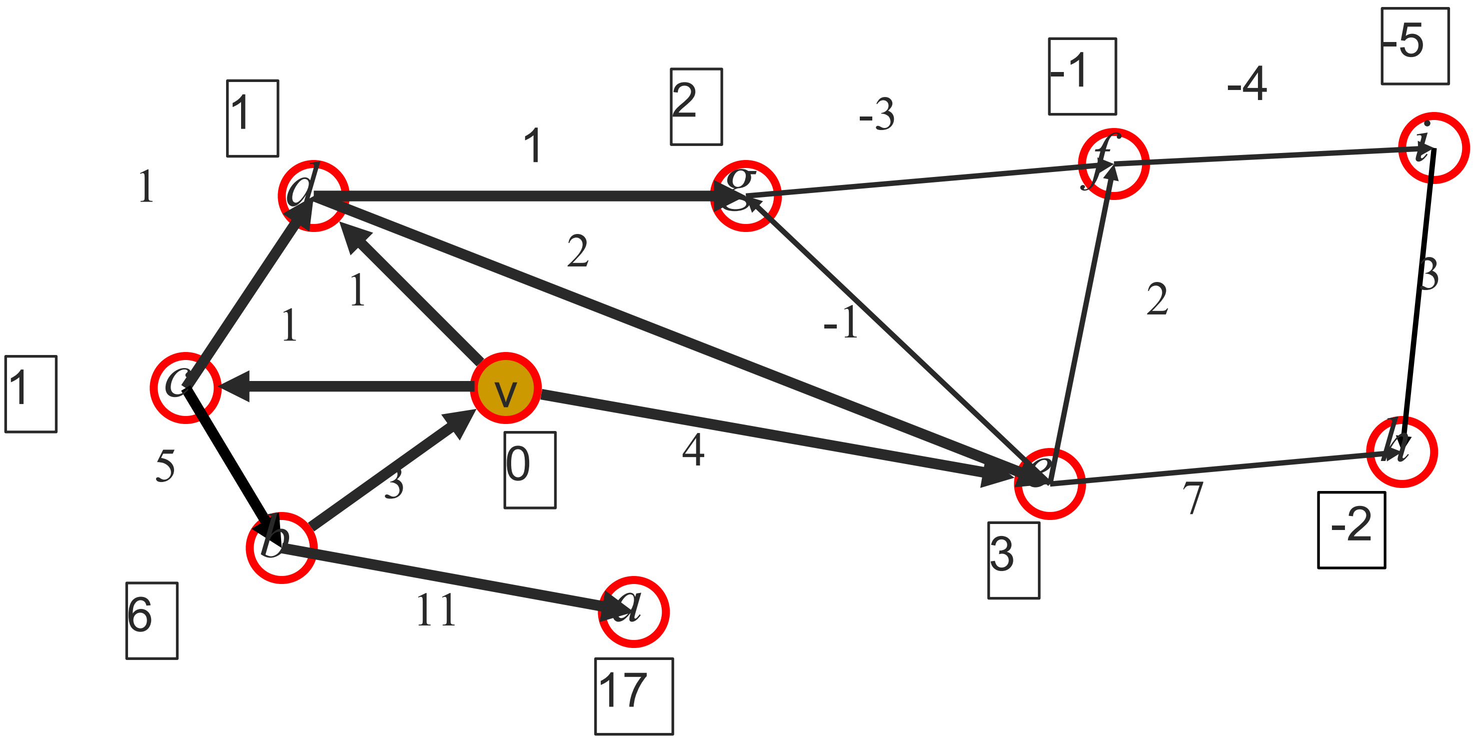
i=3



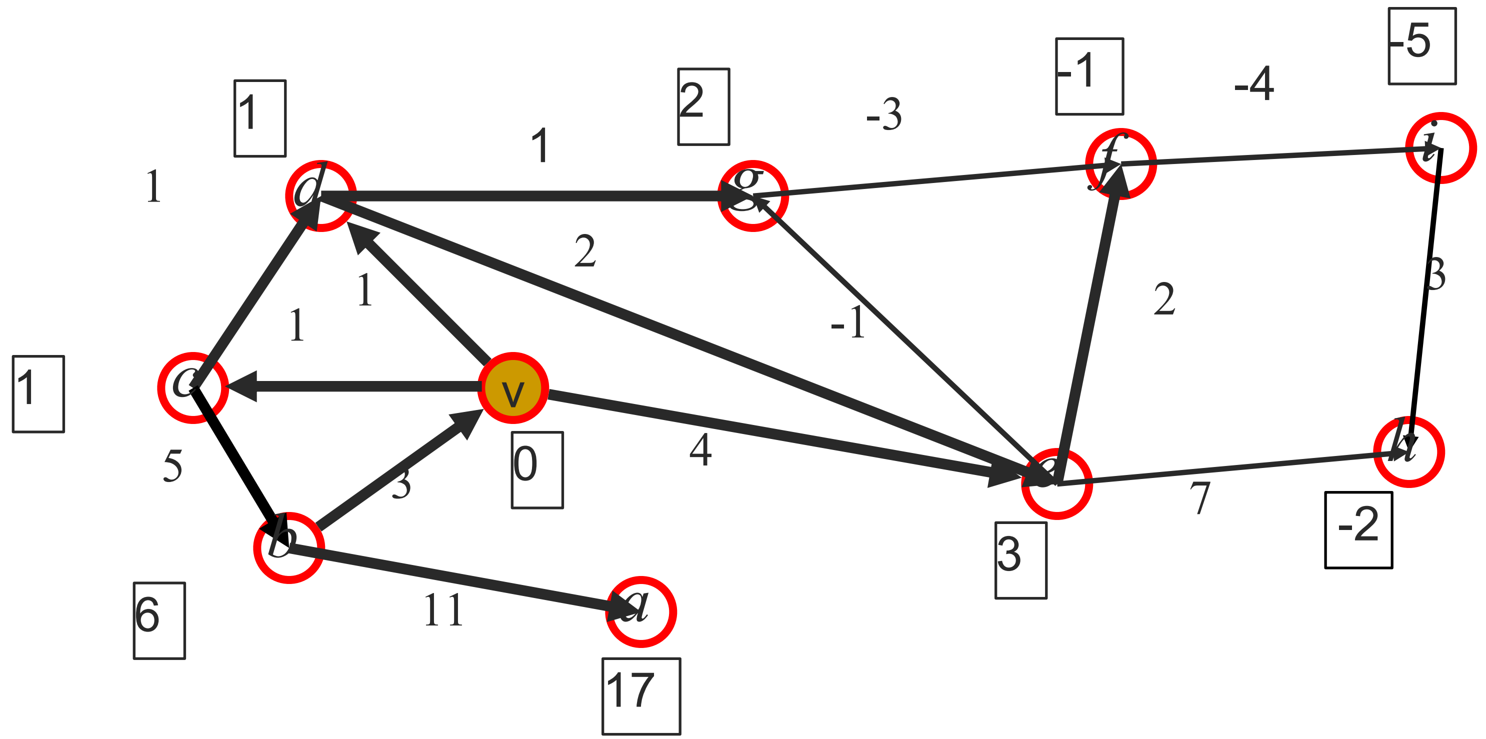
i=3



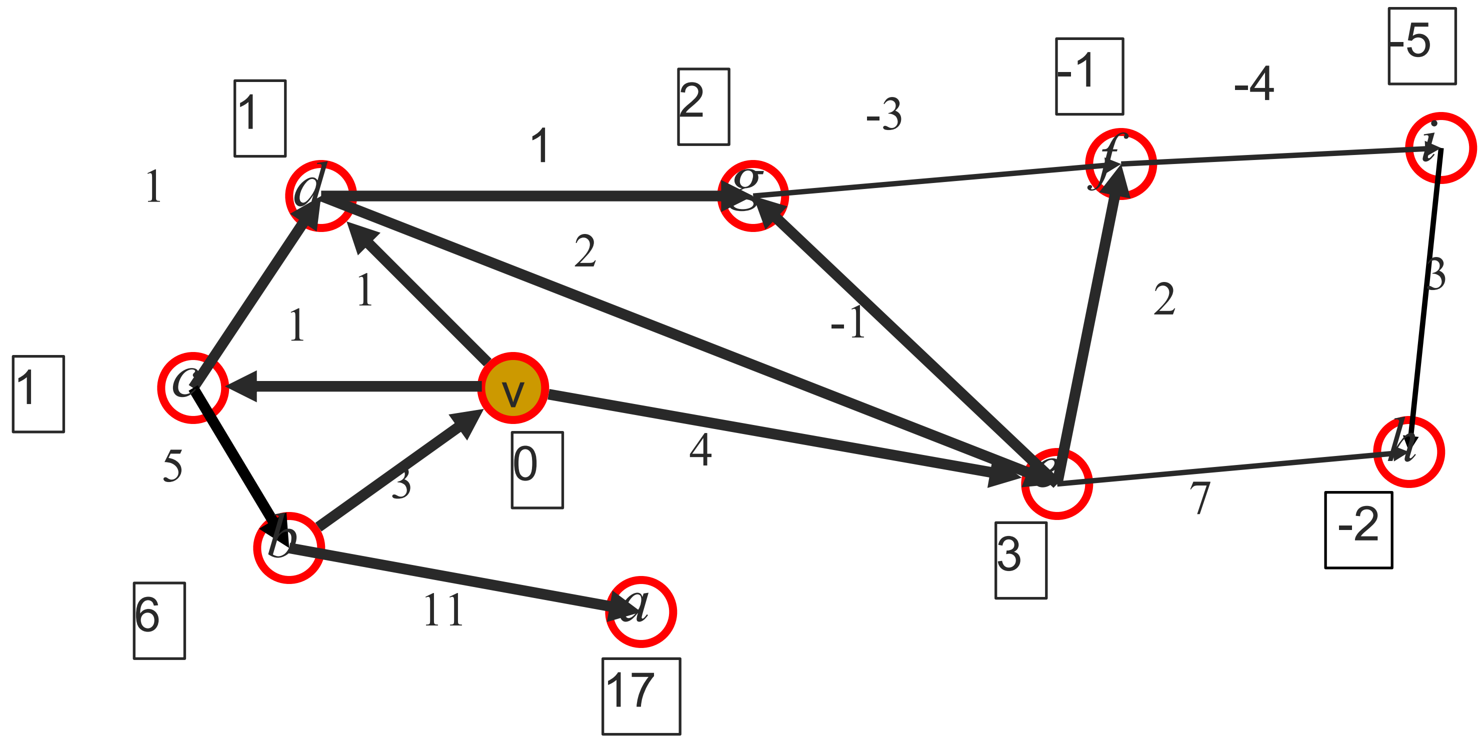
i=3



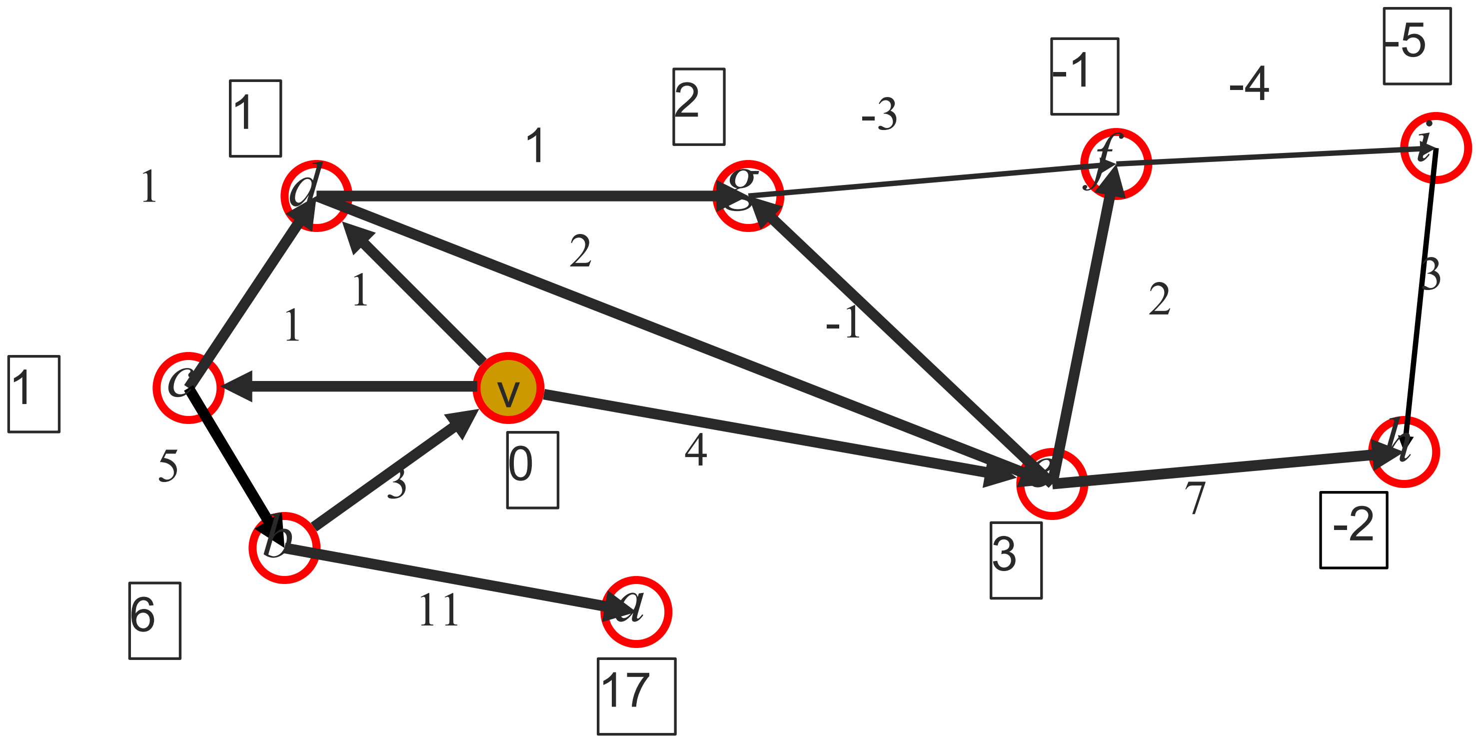
i=3



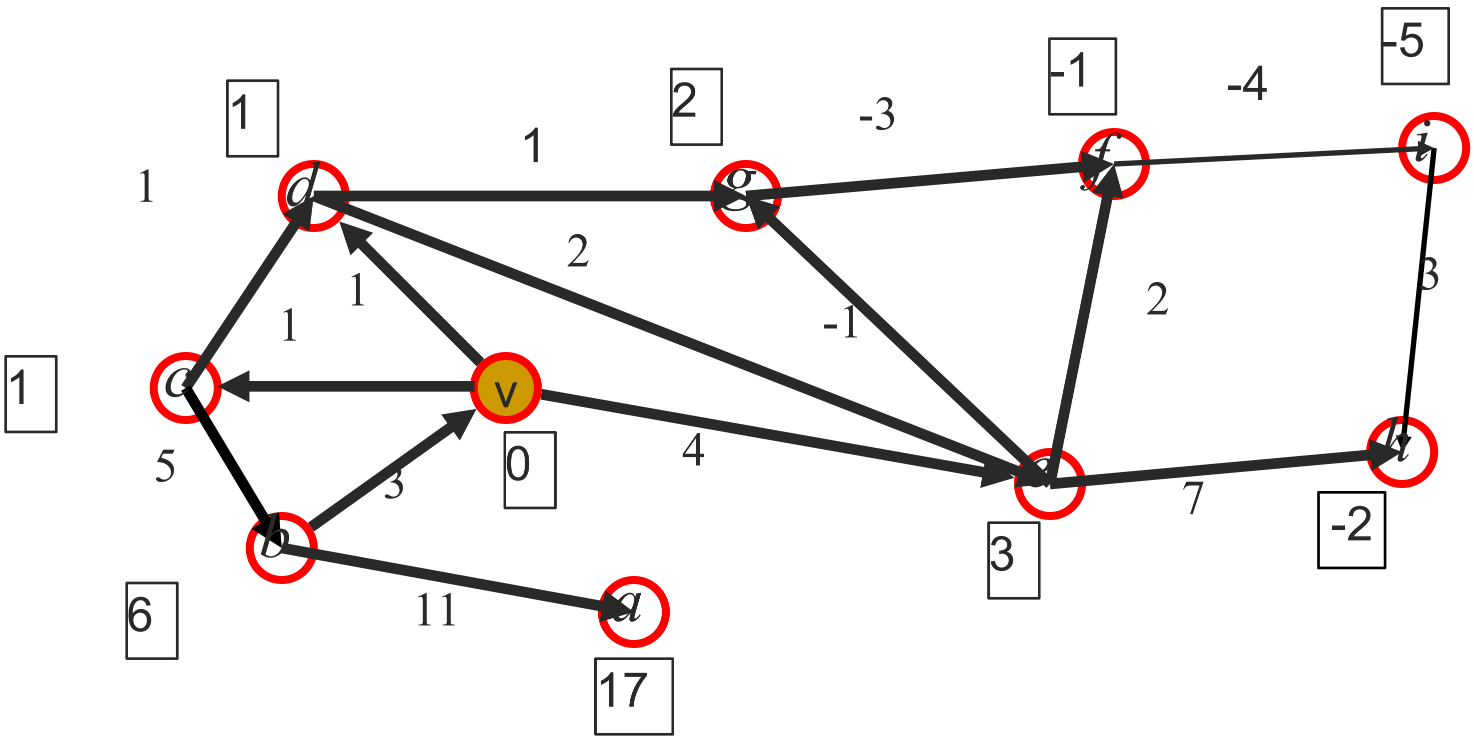
i=3



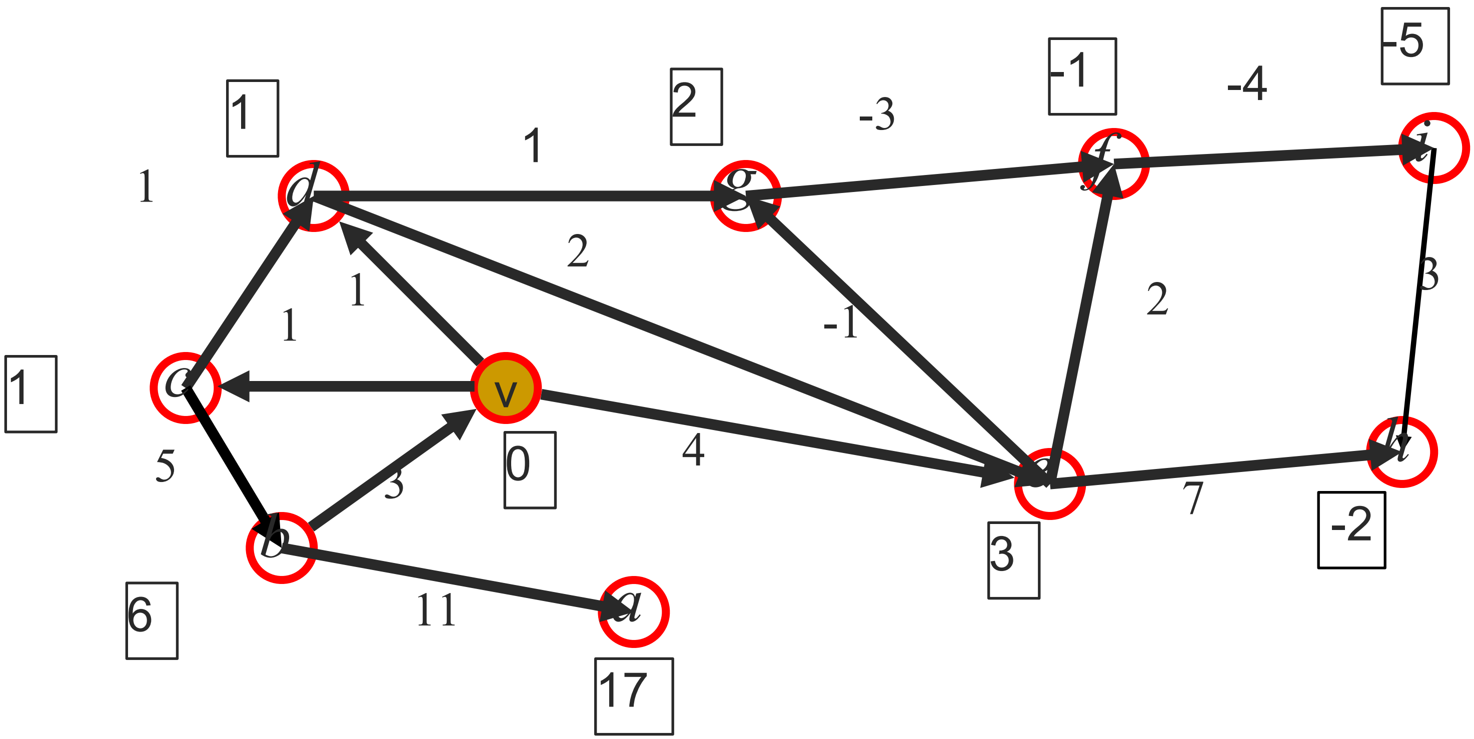
i=3



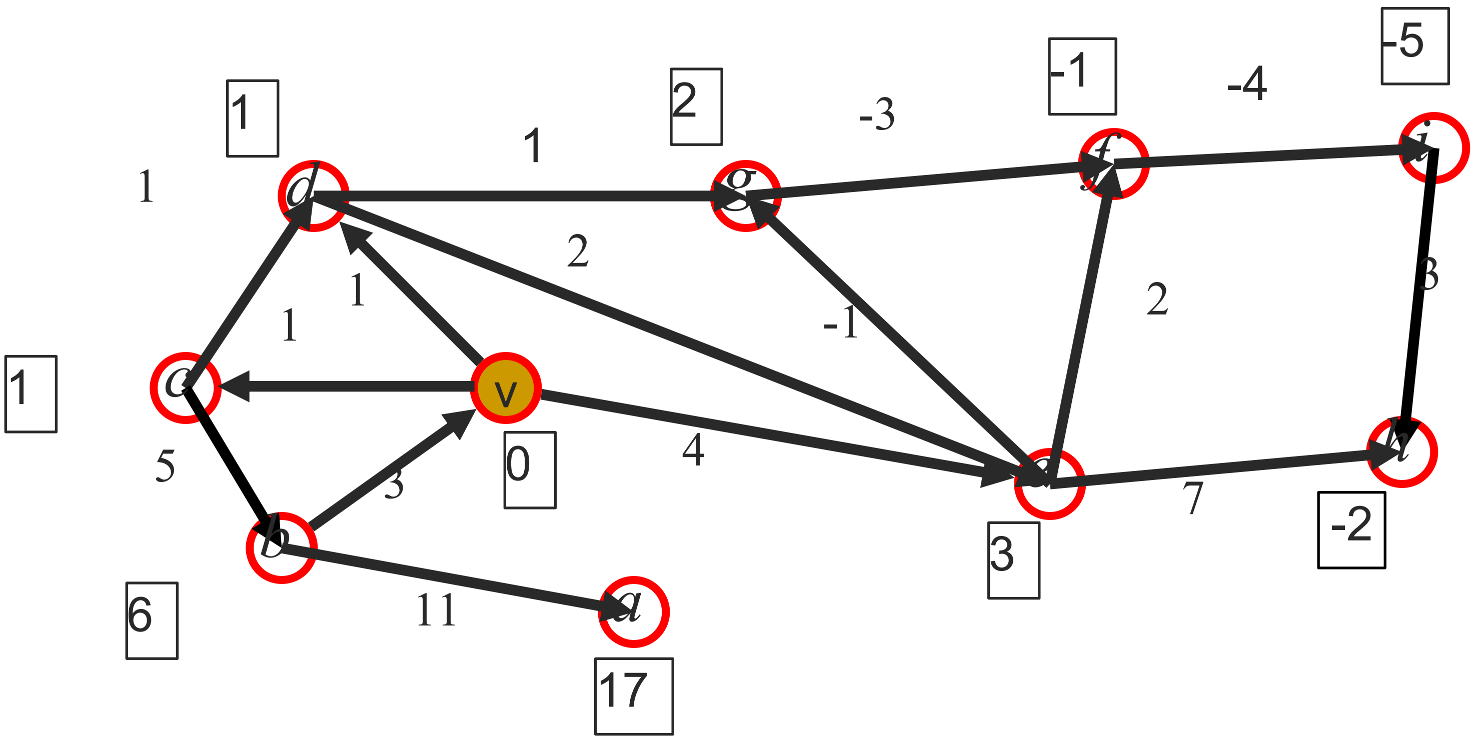
i=3



i=3



i=3



Algorithm continues until i=n-1

- In this example: $i = 9$, no more changes starting at $i = 4$

Running Time

Algorithm Bellman-Ford(G,v)

Relaxtion phase performs n-1 times a relaxation of every edge in the graph

```
D[v]←0
for each vertex u ≠ v of G do
   D[u]← +∞
for i ←1 to n-1 do
   for each edge (u,z) in G do
      if D[u]+w((u,z)) < D[z] then
            D[z]←D[u]+w((u,z))
if there are no edges left with potential
   relaxation operations then
   return D
else
   return "G contains a negative cycle"
```

Running time of Bellman-Ford algorithm

- O(nm)

Proof of correctness:

Observe: there is always a path of length $D(u)$ from v to u.

But how do we know $n-1$ relaxation phases suffice to compute $D(u)=d(u)$ for all nodes, if there is no negative cycle?

Proof by induction that invariant holds.

Invariant: After j relaxation phases, D(u) is the length of the shortest path with ≤ j edges from v to u for all nodes u.

<u>Claim</u>: Invariant holds throughout the algorithm.
Proof by induction:
<u>Base case</u>: True at start when j=0
<u>Induction Step:</u> Assume Invariant holds after phase j.
Let T be the tree of nodes whose shortest paths from v contain no more than j edges. Suppose the shortest path from v to z contains j+1 edges. Then there is some edge from v to u with j edges followed by an edge {u,z} such that d(z)=d(u)+weight({u,z}).
By induction, d(u)=D(u), so D(z) will be relaxed to d(z) when edge {u,z} is considered.
→ Invariant holds after phase j+1.
Since every shortest path in a graph with no neg cycles has length no more than n-1, after n-1 phases, we're done.

If D's continue to drop after n-1 phases
→ there is a negative cycle

How is this proof different from the proof of correctness for Dijkstra’s alg?

Shortest Paths in directed <u>acyclic</u> graphs

- Can we do faster than Bellman-Ford?

All-pairs shortest paths

- For graphs with nonnegative edges
  - Run Dijkstra for each vertex (as a source).
  - n times $O(m \log n)$ is: $O(n\, m \log n)$
- For digraphs with negative edges
  - Run Bellman-Ford for each vertex (as a source).
  - n times $O(n\, m)$ is: $O(n^2\, m)$
- Or use [Dynamic Programming]