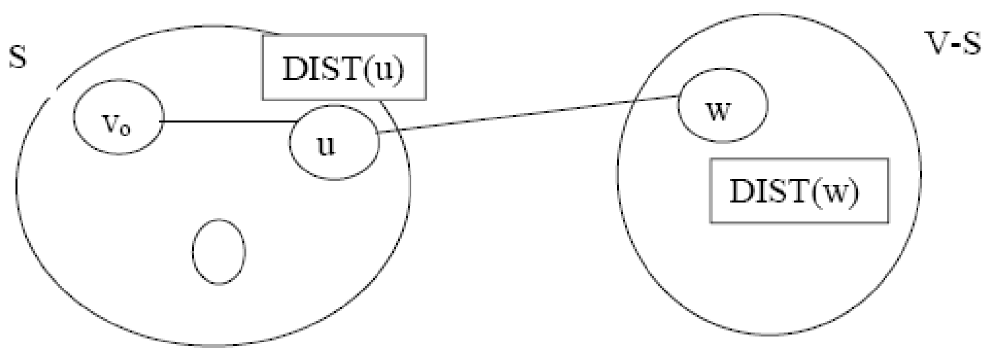
Single Source Shortest Paths

- Given a weighted graph G= (V,E) where the weights are >0.
- A source vertex, v_o belong to V.
- Find the shortest path from v_o to all other nodes in G.
- Shortest paths are generated in increasing order: 1,2,3,… ..

Dijkstra Algorithm

- S: Set of vertices (including v_o) whose final shortest paths from the source v_o have already been determined.
- For each node $w \in V-S$,
  Dist (w): the length of the shortest path starting from v_o going through only vertices which are in S and ending at w.
- The next path is generated as follows:
  It's the path of a vertex u which has Dist (u) minimum among all vertices in V-S
  Put u in S.
- Dist (w) for w in V-S may be decreased going though u.

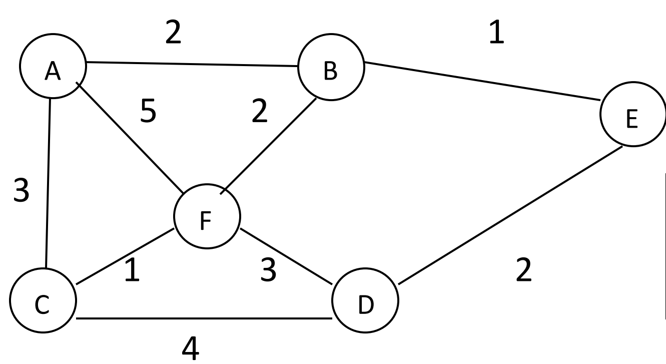


Compare Dist (u)+ cost (u,w) with Dist (w).

Algorithm:

```
Procedure SSSP (vo, cost, n)
Array S (1:n);
        Begin
        /* initialization*/
                For i=1 to n do
                S(i)=0, Dist (i)= cost (vo ,i)
                End for.
                S(vo)=1, Dist (vo)=o;
                For i=1 to n-1 do.
                        Choose u s.t. Dist (u)= min {Dist (w) } & S(w)=0
                        S(u)=1;
                        For all w with S(w)=0 do.
                                Dist (w)= min (Dist (w), Dist (u) +Cost (u,w))
                        End for.
                end for.
        end.
```

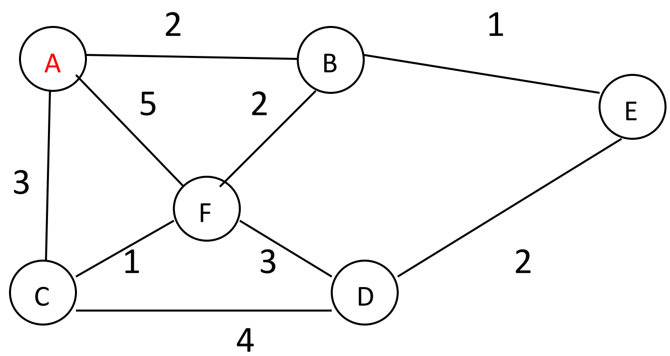
Example:

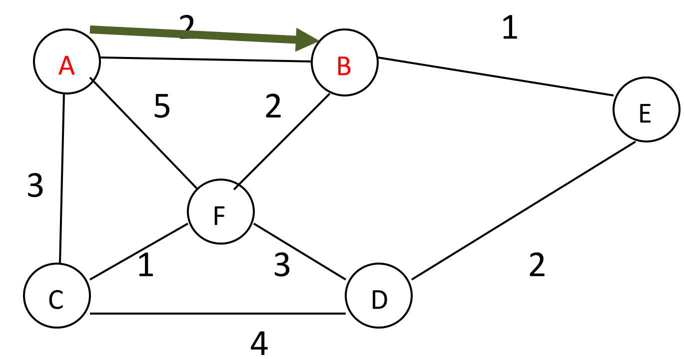


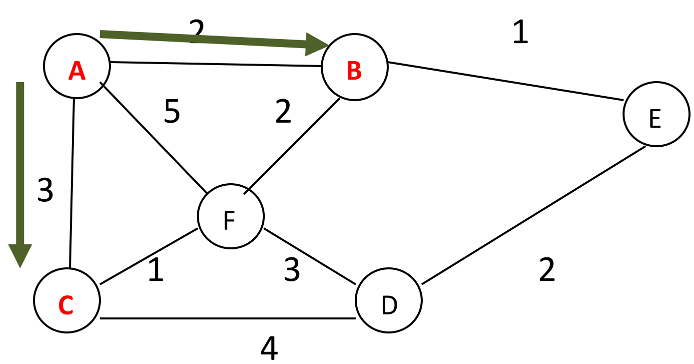
This will change to 5 to 4 because
Min{(A-F), (A-B-F)}
Min{5,4}

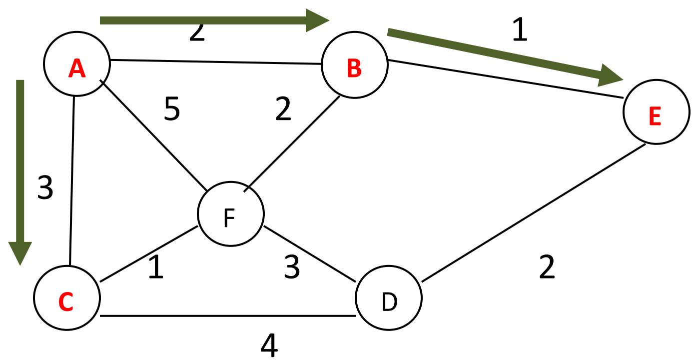
Iteration	N	D_B	D_C	D_D	D_E	D_F
Initial	{A}	2	3	∞	∞	5
1	{A,B}	2	3	∞	3	4
2	{A,B, C}	2	3	7(A-C-D)	3	4(no change because of same cost A-C-F)
3	{A,B, C,E}	2	3	5(min{(A-C-D,A-	3	4

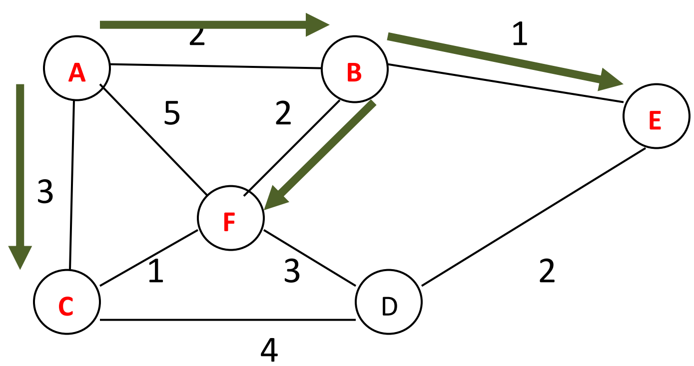
				B-E-D)}		
4	{ A,B, C,E,F}	2	3	5 no change	3	4
5	{ A,B, C,E,F,D}	2	3	5	3	4

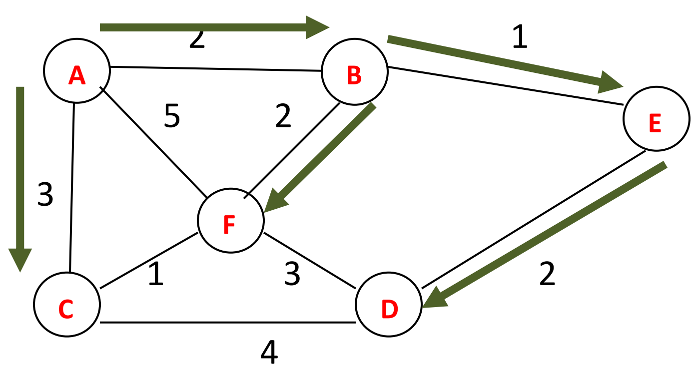












Implementation using min heap

- Build heap---------------- O(v)
- Extracting min element from min-heap & Adjusting min heap v times-- v $\log_2 v$
- Decrease key operation:
  - Delete min key from heap---- O(1).

- Adjust root ------$\log_2 v$
- We have to perform decrease key operation on rest of the vertices at max. When the value change from infinite, we have adjust min heap which takes $\log_2 v$ (v time) So $v \log_2 v$. At max we have perform this decrease key operation v-1 times so decrease key operation v-1 times take $v^2 \log_2 v$
- $v^2 \log_2 v$ we can write it as **$e\log_2 v$** because $e= v^2$ in dense graph worst case.

Time complexity: $\underbrace{O(v)}_{\text{Build heap}} + \underbrace{v \log_2 v}_{\text{Extracting min element from min-heap \& Adjusting min heap v times}} + e \log_2 v$

Time complexity: $O(v^2)$ when adjacency matrix if the input is represented using adjacency list it can be reduced to **$O((e+v) \log v)$** with the help of **binary heap.**

Drawback:

Dijkstra Algorithm will fail when there is negative weight cycle in the graph.