Single Source Shortest Path

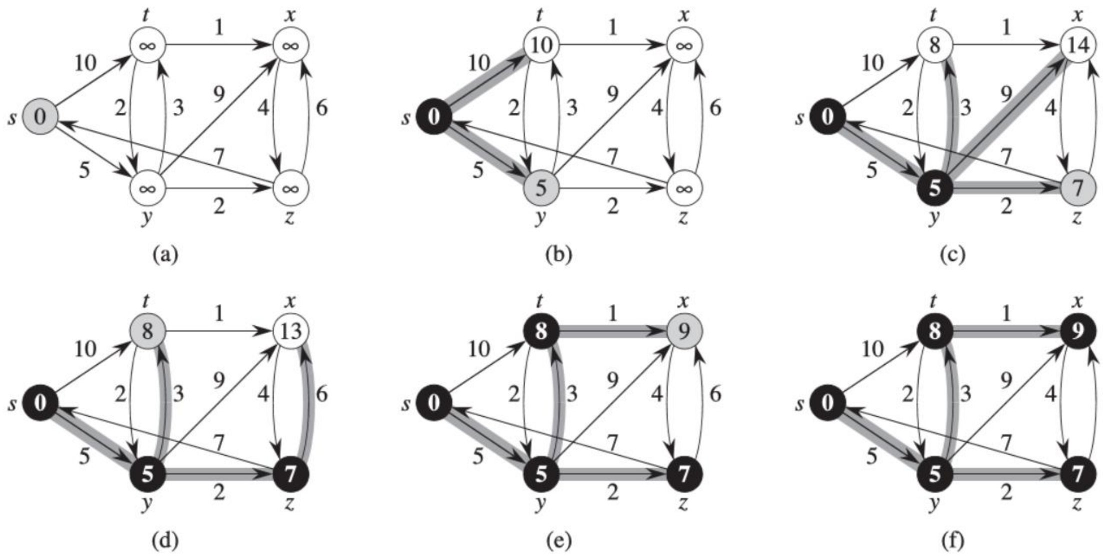
Dijkstra: A greedy approach to find the single source shortest path using min heap data structure. Dijkstra may fail to provide an optimal solution in the presence of negative edges in certain scenarios.

```
Dijkstra (G, src)
{
  A = {} //Shortest path tree (optional)
  for each vertex "v" belongs to G.V      ➔ (V)
    v. cost = ∞
    v. PI = NULL
  src. key = 0
  create a Min-heap "Q"
  for each vertex "v" belongs to G.V    ➔ (V)
  {
    Q. insert(v)      ➔ (V* log V)
  }
  while (Q != {})
  {
    v = Q. ExtMin()   ➔ (V*logV)
    A. insert(v) //insert the vertex in shortest path tree (optional)
    for each vertex "u" belongs to G. Adj[v]
    {
//v. cost: cost to reach predecessor, w (u, v): weight of edge between vertices "u" and "v".
      if ("u" belongs to Q and u. cost > v. cost + w (u, v))
      {
        u. cost = w (u, v)
        u. PI = v
// Since we have a heap in which the key of one of the vertices has been updated. So, we need
// to modify the heap to ensure that we can obtain the correct minimum value by using ExtMin
// function in the subsequent iteration.
        Q.heapify()        ➔ (E*logV)
//heapify function call to update the heap contents.
      }
    }
  }
  Return A    // return the shortest path tree
}
```

Time complexity: T(V,E) = 2* (V* log V) + E* log V

Since E ~ V^2 (In worst case scenario) so **T(V,E) = O($V^2 * log\ V$)**

Dry Run:



Initialization:

s	t	x	y	z
0	∞	∞	∞	∞
NULL	NULL	NULL	NULL	NULL

s	t	x	y	z
0	~~∞~~ 10	∞	~~∞~~ 5	∞
NULL	~~NULL~~ s	NULL	~~NULL~~ s	NULL

Y	t	x	Z
5	~~10~~ 8	~~∞~~ 14	~~∞~~ 7
s	~~s~~ y	~~NULL~~ y	~~NULL~~ y

z	t	x
7	8	~~14~~ 13
y	y	~~y~~ z

t	x
8	~~13~~ 9
y	~~z~~ t

X
9
t

Shortest Path Tree (SPT)