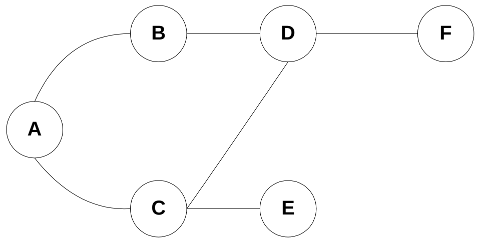
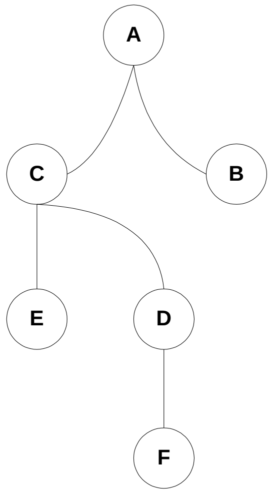
DFS & BFS

1. DFS (Deep First search)

Find the shortest path from A to F

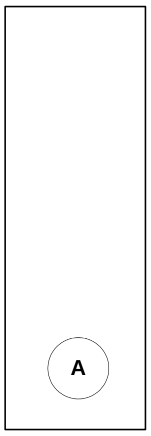


From the start point A, we get the tree diagram:

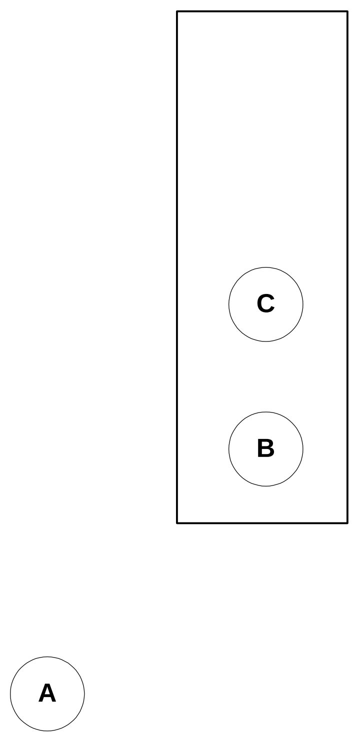


We can use heap (first in last out) to implement it. When you pop out an item from the heap, you need to put its close items into heap. The detail is listed below:

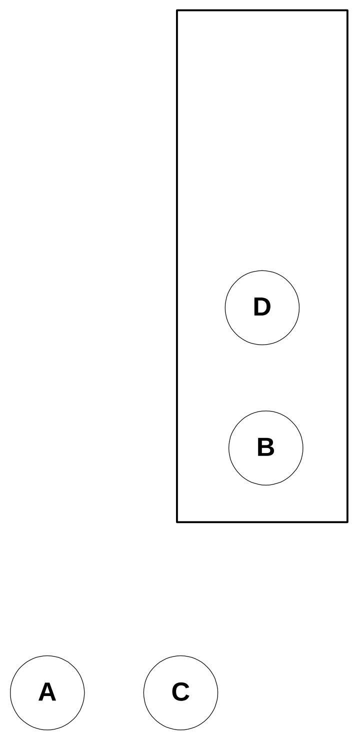
Step 1:



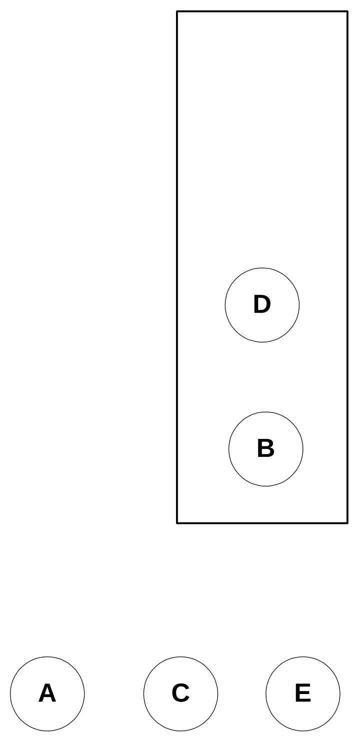
Step 2:



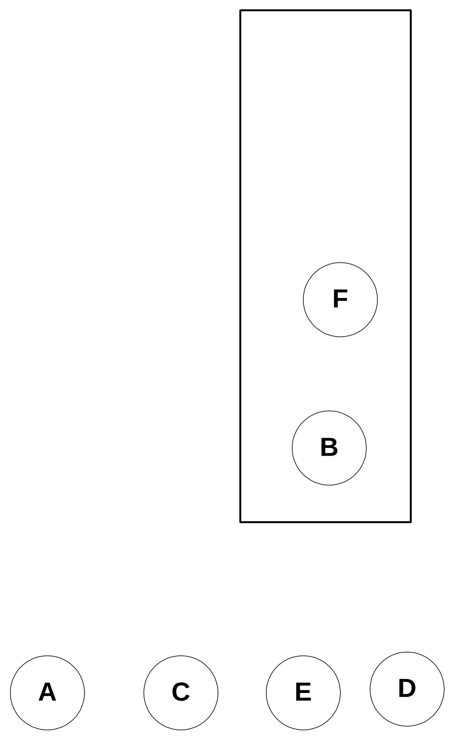
Step 3:



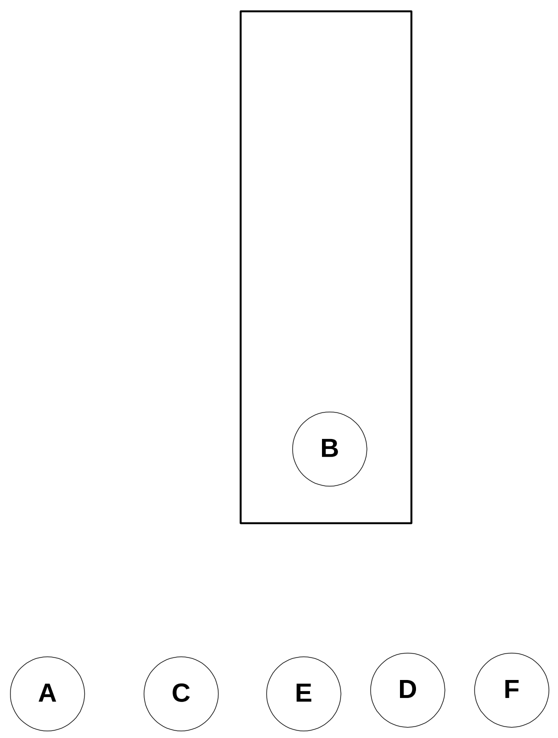
Step 4:



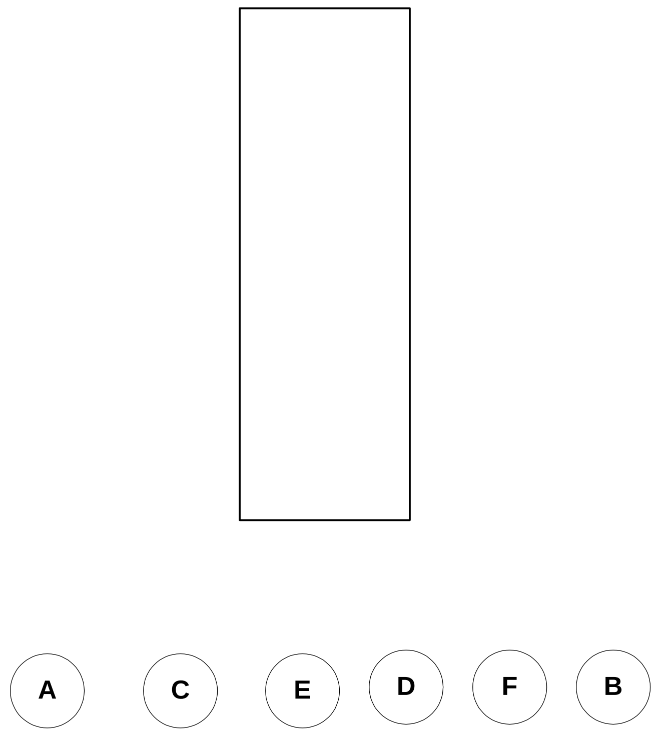
Step 4:



Step 5:

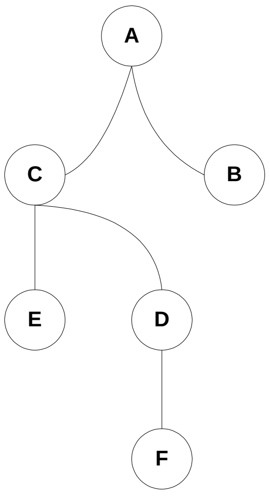


Step 6:



```
Stack.add(item[0])
i = 1
While (len(stack)!=0){
Item = Stack.pop(i)
i--
        While(neighbors of items){
                If(Neighbor is not in Stack or in Out_queue)
                {
                        Stack.add(Neighbor)
                        i++
                }
        }
Out_queue.add(item)
}
```

Actually if we can get a tree first, and then back trace the shortest path:



Define a structure: { node, parent node }.

{A, NONE}

{B, A}

{C, A}

{D, C}

{E, C}

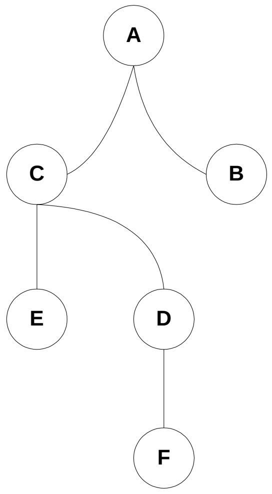
{F, D}

Then If we go from E to A,

E->C->A (parent node should be the next step)

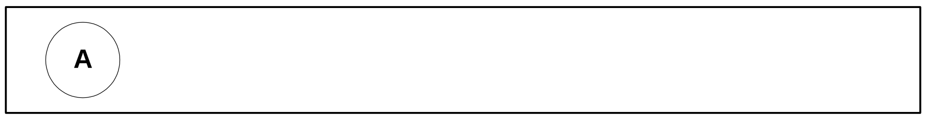
So we can use DFS to construct a tree first.

2 BFS (Breadth first search)

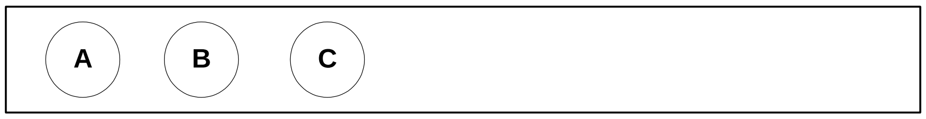


We can use queue (first in first out) to implement the BFS.

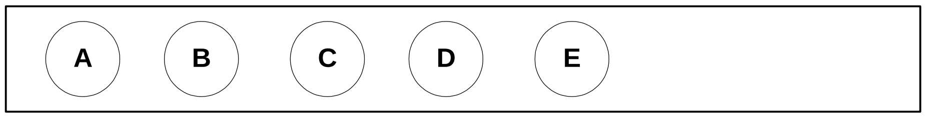
Step 1: first layer



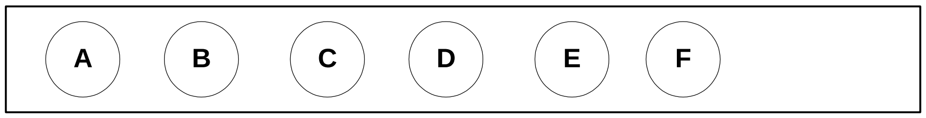
Step 2: second layer



Step 3: 3 layer



Step 4: forth layer



```
queue.add(item[0])
i = 1
While (len(queue)!=0){
Item = queue.pop(0)
i--
        While(neighbors of items){
                If(Neighbor is not in queue or in Out_queue)
                {
                        queue.add(Neighbor)
                        i++
                }
        }
Out_queue.add(item)
}
```