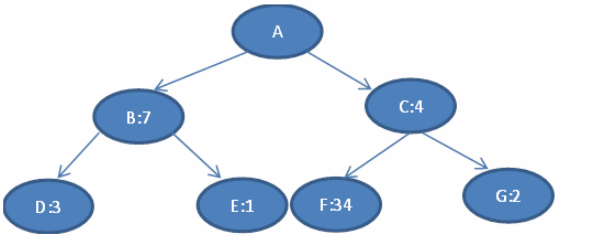
Implementation of Best First Search: The first heuristic/informed search algorithm using python

In BFS and DFS, when we are at a node, we can consider any of the adjacent as the next node. So both BFS and DFS blindly explore paths without considering any cost function.

Best First Search falls under the category of Heuristic Search or Informed Search.

Best First Search (BFS) is a heuristic search algorithm that explores a graph by expanding the most promising node (adjacent node with lowest cost, this cost can be estimated using any heuristic function) first.
We are considering the following graph where the estimated heuristic costs are given.



Python code:

```
from queue import PriorityQueue
v = 14
graph = [[] for i in range(v)]

# Function For Implementing Best First Search
# Gives output path having lowest cost

def best_first_search(actual_Src, target, n):
    visited = [False] * n
    pq = PriorityQueue()
    pq.put((0, actual_Src))
    visited[actual_Src] = True

    while pq.empty() == False:
        u = pq.get()[1]
        # Displaying the path having lowest cost
        print(u, end=" ")
        if u == target:
            break

        for v, c in graph[u]:
```

```
            if visited[v] == False:
                visited[v] = True
                pq.put((c, v))
    print()

# Function for adding edges to graph

def addedge(x, y, cost):
    graph[x].append((y, cost))
    graph[y].append((x, cost))

 # The nodes shown in above example(by alphabets) are
# implemented using integers addedge(x,y,cost);
addedge(0, 1, 7)
addedge(0, 2, 4)
addedge(1, 3, 3)
addedge(1, 4, 1)
addedge(2, 5, 34)
addedge(2, 6, 2)
source = 0
target = 5

best_first_search(source, target, v)
```