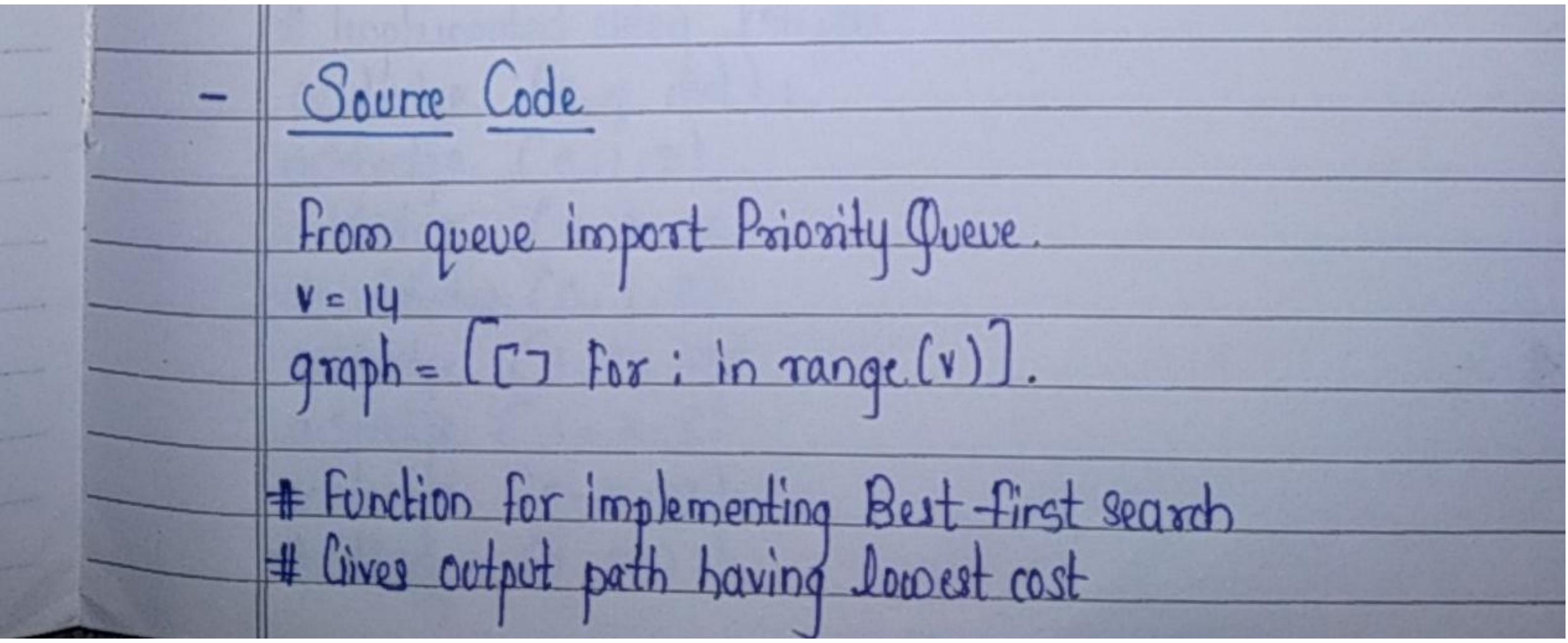

- Source Code

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

# Function for implementing Best-first search
# Gives output path having lowest cost
```

```
def best_first_search (source, target, n):
    visited = [0] * n
    visited = [0] = True
    pq = PriorityQueue()
    pq.put ((0, source))
    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:
        if 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 (0,1,8)
addedge (0, 2, 20)
#addedge (0, 3, 5)
addedge (1, 2, 20)
addedge (1, 3, 8)
addedge (2, 4, 12)
#addedge (2, 4, 12)
```

DATE :   /   /

```
addedge (3,4,12)
addedge (3,5,6)
#addedge (8,10,6)
#addedge (9,11,1)
addedge (4,5,12)

source = 0
target = 5
best_first_search (source, target, v)
```