Warren Fernandes

TE COMPS B Batch B

8940

Informed Search Algorithm:-(Greedy)

```
 from queue import
PriorityQueue



graph3 = {
    'a': {'b':2,'c':2},
    'b': {'a':2,'d':1},
    'c': {'a':2,'d':8,'f':3},
    'd': {'b':1,'c':8,'e':2,'s':3},
    'e': {'d':2,'h':8,'r':2,'s':9},
    'f': {'c':3,'g':2,'r':2},
'g': {'f':2},
    'h': {'e':8,'p':4,'q':4},
    'p': {'h':4,'q':15,'s':1},
    'q': {'h':4,'p':15},
    'r': {'e':2,'f':2},
    's': {'d':3,'e':9,'p':1}
}


heuristic = {'s': 0, 'a': 5, 'b': 7, 'c': 4, 'd': 7, 'e': 5, 'f': 2, 'g': 0,
'h':11, 'p': 14, 'q': 12, 'r': 3}
```

```
def Greedy(graph, start, goal):
# set of visited nodes
visited = set()     expanded=[]
queue = PriorityQueue()
queue.put((0, start))


    while queue:
        cost, node = queue.get()
current = node[-1]         if
current not in visited:
visited.add(current)
expanded.append(current)


            if current == goal:
return node, expanded


            neighbours = graph[current]
for i in neighbours:                 if
i not in visited:
                    total_cost = heuristic[i]
queue.put((total_cost, node+i))



path, expanded = Greedy(graph3,'s','g')
output = [char for char in path] print("\nThe optimal path using a greedy
search is : " + "->".join(output)) print("The states expanded are:")
print(expanded)
```

OUTPUT:-

```
The optimal path using a greedy search is : s->e->r->f->g
The states expanded are:
['s', 'e', 'r', 'f', 'g']
```