```
def initial_graph()
 return {
 'A': {'B':1, 'C':4, 'D':2},
 'B': {'A':9, 'E':5},
 'C': {'A':4, 'F':15},
 'D': {'A':10, 'F':7},
 'E': {'B':3, 'J':7},
 'F': {'C':11, 'D':14, 'K':3, 'G':9},
 'G': {'F':12, 'I':4},
 'H': {'J':13},
 'I': {'G':6, 'J':7},
 'J': {'H':2, 'I':4},
 'K': {'F':6}
 }
print(initial_graph())
initial = 'A'
path = {}
adj_node = {}
queue = []
graph = initial_graph()
for node in graph:
 path[node] = float("inf")
 adj_node[node] = None
 queue.append(node)
path[initial] = 0
while queue:
 # find min distance which wasn't marked as current
 key_min = queue[0]
 min_val = path[key_min]
 for n in range(1, len(queue)):
 if path[queue[n]] < min_val:
 key_min = queue[n]
 min_val = path[key_min]
 cur = key_min
 queue.remove(cur)
 #print(cur)

 for i in graph[cur]:
 alternate = graph[cur][i] + path[cur]
 if path[i] > alternate:
 path[i] = alternate
 adj_node[i] = cur
```

```
x = str(input("Enter ending node: "))
print('The path between A to H')
print(x, end = '<-')
while True:
 x = adj_node[x]
 if x is None:
 print("")
 break
 print(x, end='<-')
```