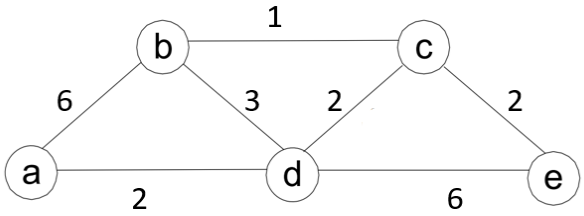
Assignment 2: Optimal Search

1. Go through Dijkstra's algorithm by hand for this graph:

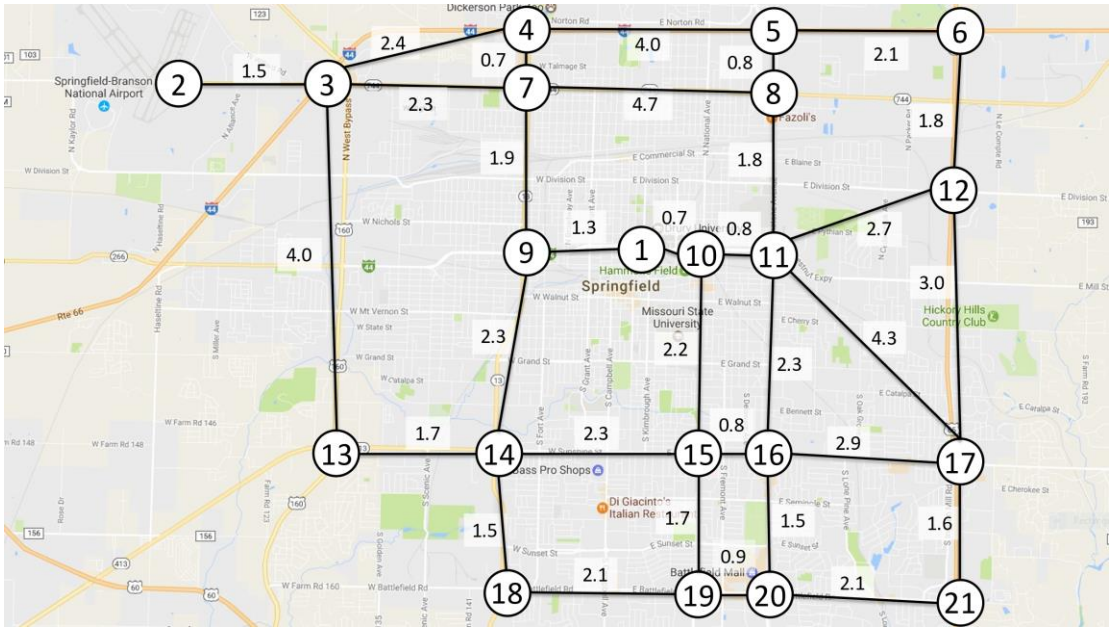



```
1  function Dijkstra(Graph, source):
2
3      for each vertex v in Graph.Vertices:
4          dist[v] ← INFINITY
5          prev[v] ← UNDEFINED
6          add v to Q
7      dist[source] ← 0
8
9      while Q is not empty:
10         u ← vertex in Q with minimum dist[u]
11         remove u from Q
12
13         for each neighbor v of u still in Q:
14             alt ← dist[u] + Graph.Edges(u, v)
15             if alt < dist[v]:
16                 dist[v] ← alt
17                 prev[v] ← u
18
19     return dist[], prev[]
```

Give the shortest path from a to each other vertex, along with the length of that path. List your results in the order in which Dijkstra's algorithm finds them.

(a -> b : 6) , (a -> d: 2). (a -> b : 6) > (a -> d: 2) Thus, path = (a -> d: 2)
(d -> c: 2), (d->b: 3), (d->e: 6). (d -> c: 2) < (d->b: 3), (d->e: 6)
Thus, path = (a -> d: 2), (d -> c: 2)
(c->e: 2) and e is goal thus, terminate and add. 2 + 2 + 2 = 6
(a, d, c, e) -> 6
The shortest path is (a, d, c, e: 6) from source A.

2. Create a Python implementation of A*. Run the program on the airport graph below. Your output should list the nodes in the shortest path in order of traversal, as well as the total cost of the path.



Submit a transcript of executing your program below and your source code in a separate file: