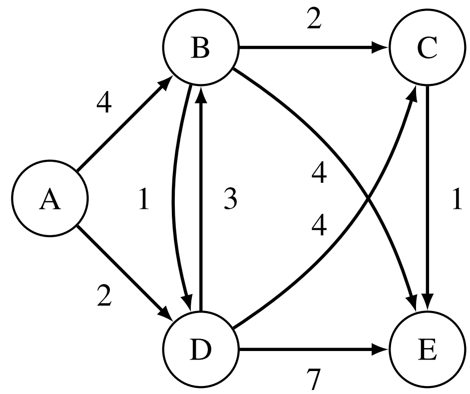
1 Dijkstra's Algorithm

(a) Given the following graph, run Dijkstra's algorithm starting at node A. For each iteration, write down the entire state of the algorithm. This includes the value `dist(v)` for all vertices v as well as what node was popped off of the fringe for that iteration.

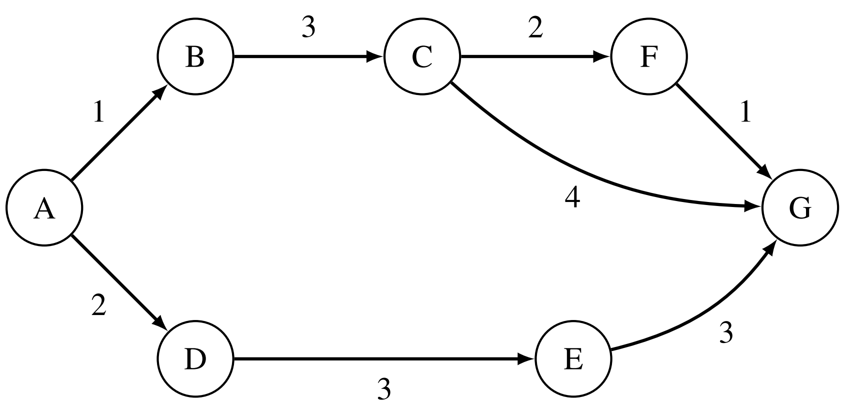
Note: If you want to keep track of the vertices traversed along the shortest paths from A to every other node in the graph, you will need to maintain an edgeTo array.



(b) What must be true about our graph in order to guarantee Dijkstra's will return the shortest path's tree to every vertex? Draw an example of a graph that demonstrates why Dijkstra's might fail if we do not satisfy this condition.

2 A* Search

For the graph below, let `g(u, v)` be the weight of the edge between any nodes `u` and `v`. Let `h(u, v)` be the value returned by the heuristic for any nodes `u` and `v`. Remember the heuristic serves to estimate the distance between two nodes `u` and `v`.



```
Edge weights:   Heuristics:
g(A, B) = 1     h(A, G) = 8
g(B, C) = 3     h(B, G) = 6
g(C, F) = 2     h(C, G) = 5
g(C, G) = 4     h(F, G) = 1
g(F, G) = 1     h(D, G) = 6
g(A, D) = 2     h(E, G) = 3
g(D, E) = 3
g(E, G) = 3
```

(a) Given the weights and heuristic values for the graph above, what would A* search return as the shortest path from A to G?

(b) Is the heuristic admissible? Why or why not? A heuristic is admissible if it never overestimates the distance it is estimating.