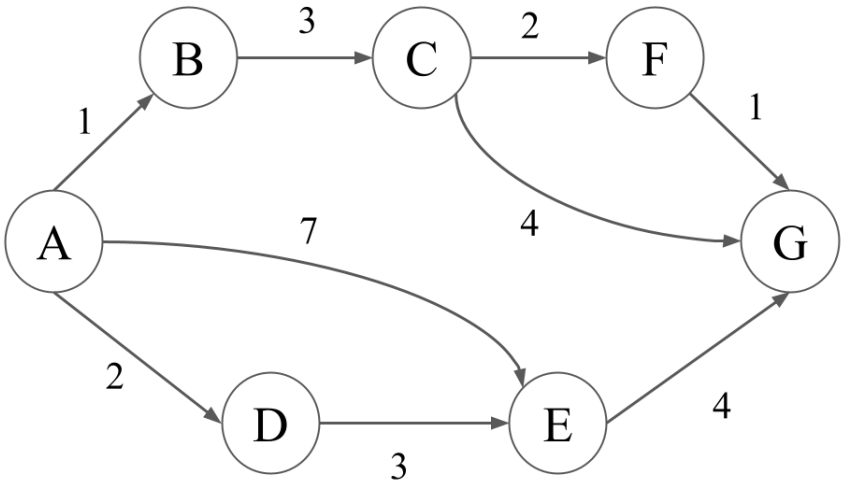
CS 61B Spring 2023

Shortest Paths, MSTs

Discussion 10: March 20, 2023

1 The Shortest Path To Your Heart

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`.



Below, the pseudocode for Dijkstra's and A* are both shown for your reference throughout the problem.

Dijkstra's Pseudocode

```
PQ = new PriorityQueue()
PQ.add(A, 0)
PQ.add(v, infinity) # (all nodes except A).

distTo = {} # map
distTo[A] = 0
distTo[v] = infinity # (all nodes except A).

while (not PQ.isEmpty()):
  poppedNode, poppedPriority = PQ.pop()

  for child in poppedNode.children:
    if PQ.contains(child):
      potentialDist = distTo[poppedNode] +
        edgeWeight(poppedNode, child)
      if potentialDist < distTo[child]:
        distTo.put(child, potentialDist)
        PQ.changePriority(child, potentialDist)
        edgeTo[child] = poppedNode
```

A* Pseudocode

```
PQ = new PriorityQueue()
PQ.add(A, h(A))
PQ.add(v, infinity) # (all nodes except A).

distTo = {} # map
distTo[A] = 0
distTo[v] = infinity # (all nodes except A).

while (not PQ.isEmpty()):
  poppedNode, poppedPriority = PQ.pop()
  if (poppedNode == goal): terminate

    for child in poppedNode.children:
      if PQ.contains(child):
        potentialDist = distTo[poppedNode] +
          edgeWeight(poppedNode, child)

        if potentialDist < distTo[child]:
          distTo.put(child, potentialDist)
          PQ.changePriority(child, potentialDist + h(child))
          edgeTo[child] = poppedNode
```

(a) Run Dijkstra's algorithm to find the shortest paths from A to every other vertex. You may find it helpful to keep track of the priority queue. We have provided a table to keep track of best distances, and the adjacent vertex that has an edge going to the target vertex in the current shortest paths tree so far.

	A	B	C	D	E	F	G
DistTo							
EdgeTo							

(b) Given the weights and heuristic values for the graph above, what path would A* search return, starting from A and with G as a goal? Note that the edge weights provided below for your convenience are the same as in the image.

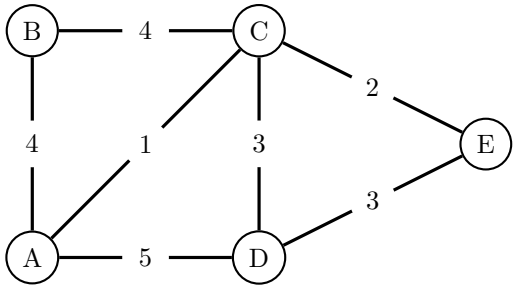
Edge weights	Heuristics
$g(A,B) = 1$	$h(A,G) = 7$
$g(B,C) = 3$	$h(B,G) = 6$
$g(C,F) = 2$	$h(C,G) = 3$
$g(C,G) = 4$	$h(F,G) = 1$
$g(F,G) = 1$	$h(D,G) = 6$
$g(A,D) = 2$	$h(E,G) = 3$
$g(D,E) = 3$	
$g(E,G) = 4$	
$g(A,E) = 7$	

	A	B	C	D	E	F	G
DistTo							
EdgeTo							

(c) Based on the heuristics for part b, is the A* heuristic for this graph good? In other words, will it always give us the actual shortest path from A to G? If it is good, give an example of a change you would make to the heuristic so that it is no longer good. If it is not, correct it.

2 Minimalist Moles

Circle the mole wants to dig a network of tunnels connecting all of his secret hideouts. There are a set few paths between the secret hideouts that Circle can choose to possibly include in his tunnel system, shown below. However, some portions of the ground are harder to dig than others, and Circle wants to do as little work as possible. In the diagram below, the numbers next to the paths correspond to how hard that path is to dig for Circle. Lucky for us, he knows how to use MSTs to optimize the tunnel paths!



Below, the pseudocode for Kruskal's and Prim's are shown for your reference throughout the problem.

Kruskal's Pseudocode

```
while there are still nodes not in the MST:
        Add the lightest edge
            that does not create a cycle.
        Add the new node to the
            set of nodes in the MST.
```

Prim's Pseudocode

```
Start with any node.
Add that node to the set of nodes in the MST.
While there are still nodes not in the MST:
    Add the lightest edge from a node in the MST
        that leads to a new node that is unvisited.
    Add the new node to the set of
        nodes in the MST.
```

(a) Find a valid MST for the graph above using Kruskal's algorithm, then Prim's. For Prim's algorithm, take A as the start node. In both cases, if there is ever a tie, choose the edge that connects two nodes with lower alphabetical order.

(b) Are the above MSTs different or the same? More generally, is the MST for this graph unique? Describe a different tie-breaking scheme or make edits to the edge weights that would change your answer.