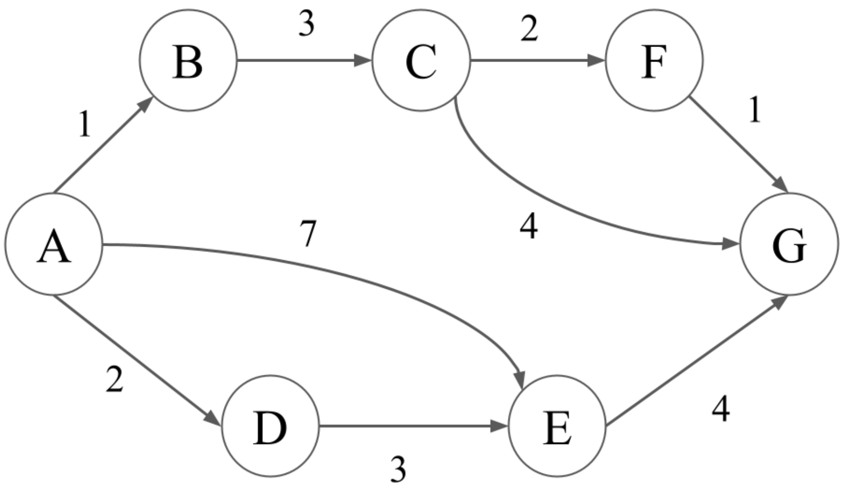
CS 61B
Spring 2021

Shortest Paths and MSTs

Discussion 10: March 29, 2021

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()):
  popNode, popPriority = PQ.pop()

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

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))
```

(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 edge leading to each vertex on the currently known shortest paths.

	A	B	C	D	E	F	G
DistTo	0	1	4	2	5	6	7
EdgeTo	-	A	B	A	D	C	F

(b) *Extra:* 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.

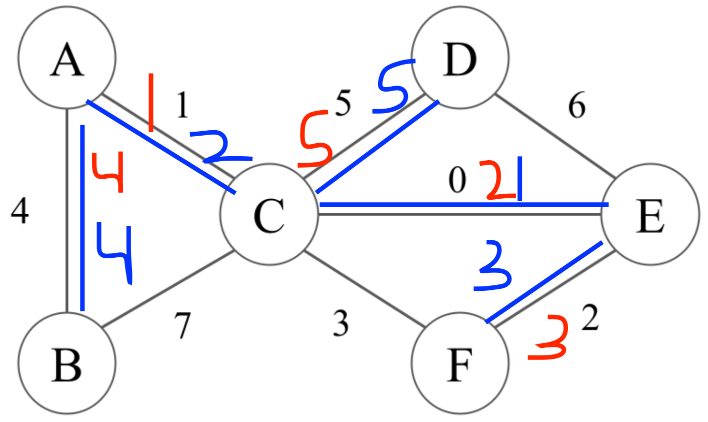
Edge weights	Heuristics
$g(A,B) = 1$	$h(A,G) = 7$
$g(B,C) = 3$	$h(B,G) = 6$
$g(C,F) = 2$	$h(C,G) = 3$
$g(C,G) = 4$	$h(F,G) = 1$
$g(F,G) = 1$	$h(D,G) = 6$
$g(A,D) = 2$	$h(E,G) = 3$
$g(D,E) = 3$	
$g(E,G) = 4$	
$g(A,E) = 7$	

	A	B	C	D	E	F	G
DistTo	0	1	4	2	7	6	7
EdgeTo	-	A	B	A	A	C	F

PQ: [G:7, D:8, E:10] -> Then pop G off the Queue and stop

2 Minimalist Moles

Mindy the mole wants to dig a network of tunnels connecting all of their secret hideouts. There are a set few paths between the secret hideouts that Mindy can choose to possibly include in their tunnel system, shown below. However, some portions of the ground are harder to dig than others, and Mindy 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 Mindy.



(a) How can Mindy figure out a tunnel system to connect their secret hideouts while doing minimal work?

Mindy can do this by using the minimum spanning tree

(b) *Extra:* Find a valid MST for the graph above using Kruskal's algorithm, then Prims. 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.

MST in Blue above, numbered by order on Kruskal's

Red Numbers are Prim's

(c) *Extra:* Are the above MSTs different or the same? Is there a different tie-breaking scheme that would change your answer?

MST's are the same, no tiebreaking scheme for this tree, as all weights are different