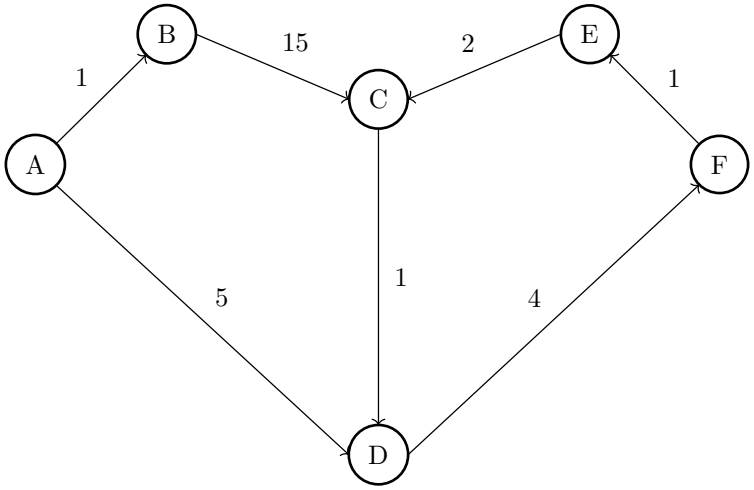
CS 61B Spring 2025

Shortest Paths, MSTs

Discussion 09: March 31, 2025

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

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

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

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

(b) Given the weights and heuristic values for the graph above, what path would A$*$ search return, starting from A and with F as a goal?

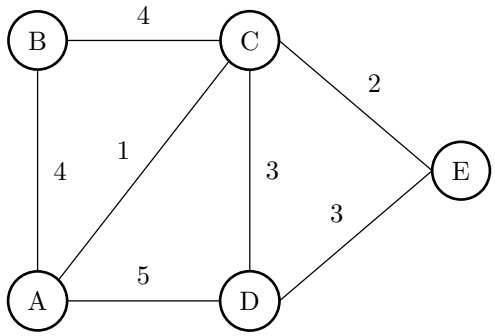
Edge Weights	**Heuristics**
$g(A, B) = 1$	$h(A, F) = 8$
$g(A, D) = 5$	$h(B, F) = 16$
$g(B, C) = 15$	$h(C, F) = 4$
$g(C, D) = 1$	$h(D, F) = 4$
$g(D, F) = 4$	$h(E, F) = 5$
$g(F, E) = 1$	
$g(E, C) = 2$	

	A	B	C	D	E	F
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 F? 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

Karen the mole wants to dig a network of tunnels connecting all of her secret hideouts. There are a few set paths between the secret hideouts that Karen can choose to possibly include in her tunnel system, shown below. However, some portions of the ground are harder to dig than others, and Karen 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. Lucky for us, she knows how to use MSTs to optimize the tunnel paths!



Below, the pseudocode for Kruskal's and Prim's are both 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? If different, describe a tie-breaking scheme that would make them the same. If the same, describe a tie-breaking scheme that would make them different.

3 Sticky Flights

Your airline company has been contracted to fly a large shipment of honey from Honeysville to the 61Bees in Goldenhive City. However, the airplane doesn't have enough fuel capacity to fly directly to Goldenhive City so it will stop at at least one of n airports along the way to refuel. Refueling takes an hour, and if the airport is one of $k < n$ airports, your airplane will be grounded for six hours due to curfews (refueling is included in the six hours). The 61Bees want their honey as soon as possible so please design an algorithm to find the route that will allow your airplane to reach Goldenhive City in the least amount of hours.

Hint: Think of the n airports as a graph, where the paths between them are edges of weight equivalent to the number of hours it takes to fly from airport A to airport B. You may assume that the amount of time it takes to fly from A to B is equal to the amount of time it takes to fly from B to A.