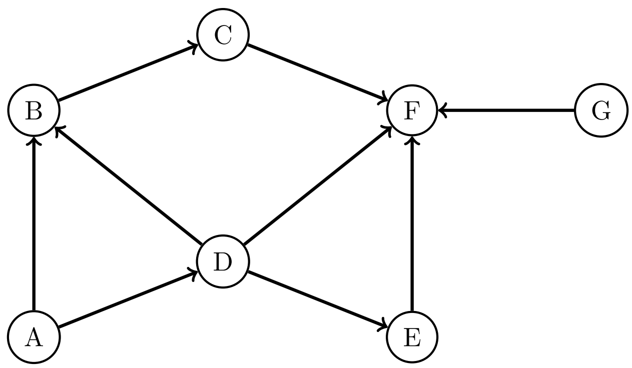
CS 61B
Spring 2019

DFS, BFS, SPs, MSTs

Discussion 10: April 1, 2019

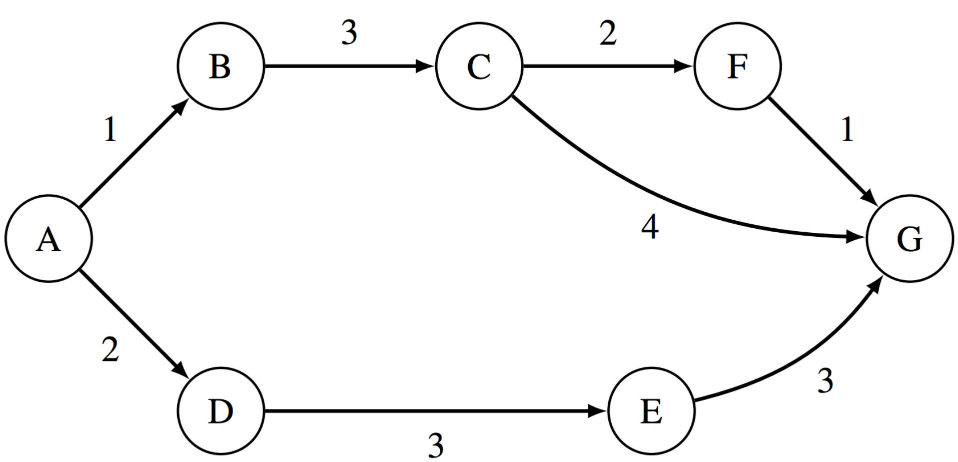
1 Graphs



1.1 Give the DFS preorder, DFS postorder, and BFS order of the graph traversals starting from vertex A. Break ties alphabetically.

2 Dijkstra's Algorithm

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`.



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

2.1 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 and make a table of current distances.

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)
```

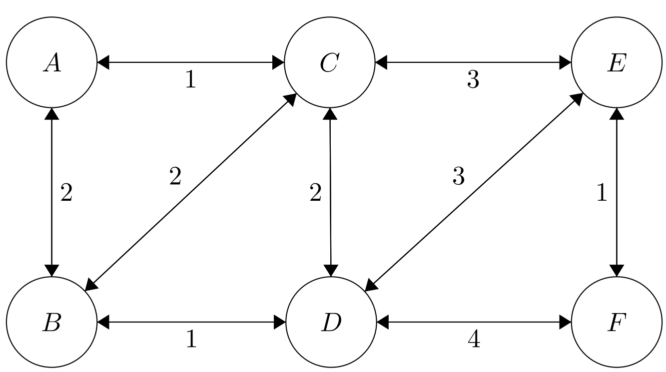
2.2 Given the weights and heuristic values for the graph below, what path would A* search return, starting from A and with G as a goal?

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))
```

2.3 Is the heuristic admissible? Why or why not?

3 Minimum Spanning Trees



3.1 Perform Prim's algorithm to find the minimum spanning tree. Pick A as the initial node. Whenever there is more than one node with the same cost, process them in alphabetical order.

3.2 Use Kruskal's algorithm to find a minimum spanning tree. When deciding between equiweighted edges, alphabetically sort the edge, and then pick in lexicographic order.

For instance, edges are always written as AB or AC, never BA or CA. If deciding between AB and AC, pick AB first.