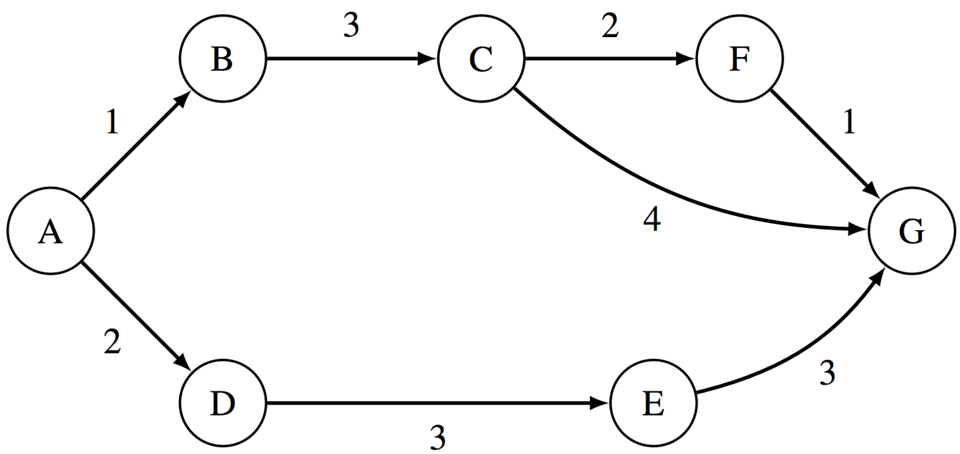
CS 61B Spring 2018 — Graphs & Sorting

Discussion 12: April 10, 2018

1 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`.



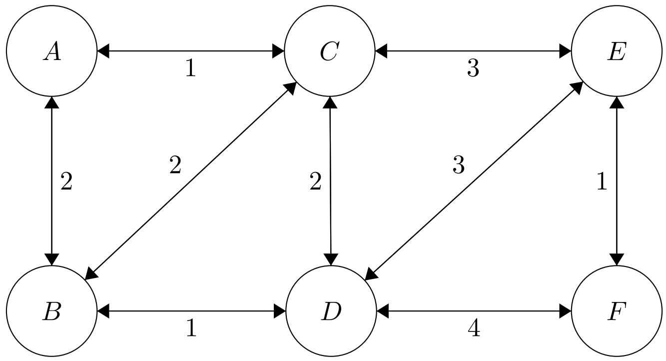
Edge weights	Heuristics
$g(A,B)=1$	$h(A,G)=8$
$g(B,C)=3$	$h(B,G)=6$
$g(C,F)=4$	$h(C,G)=5$
$g(C,G)=4$	$h(F,G)=1$
$g(F,G)=1$	$h(D,G)=6$
$g(A,D)=2$	$h(E,G)=3$
$g(D,E)=3$	
$g(E,G)=3$	

1.1 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.

1.2 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?

1.3 Is the heuristic admissible? Why or why not?

2 Minimum Spanning Trees



2.1 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.

2.2 Use Kruskal's algorithm to find a minimum spanning tree.

2.3 There are quite a few MSTs here. How many can you find?

3 Mechanical Sorting

3.1 Show the steps taken by each sort on the following unordered list:

`0, 4, 2, 7, 6, 1, 3, 5`

(a) Insertion sort

(b) Selection sort

(c) Merge sort

(d) Use heapsort to sort the following array (hint: draw out the heap). Draw out the array at each step:

`0, 6, 2, 7, 4`