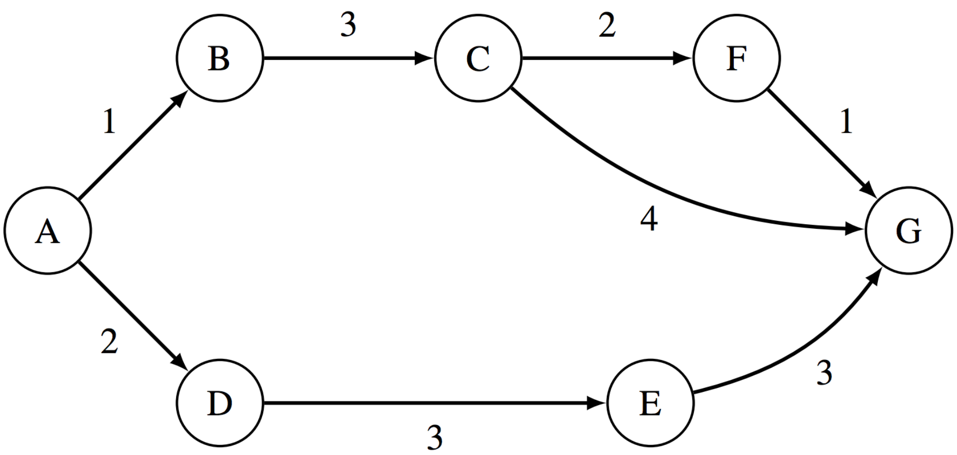
CS 61B
Spring 2018

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 (A, 0)
② (B, 1), (D, 2)
③ (D, 2), (C, 4)
④ (C, 4), (E, 5)
⑤ (E, 5), (F, 6), (G, 7)
⑥ (F, 6), (G, 7)
⑦ (G, 7)
⑧ /

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

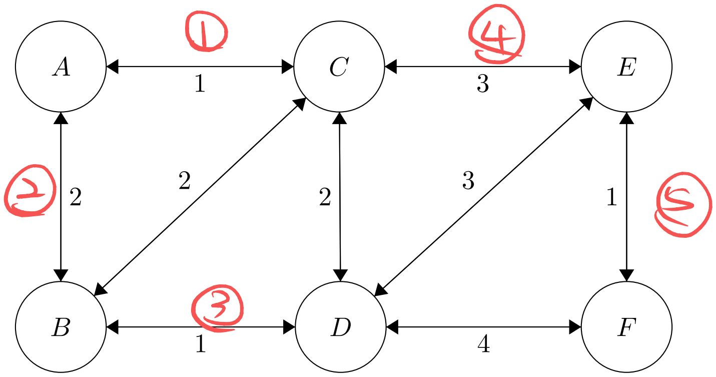
1.2 Given the weights and heuristic values for the graph below, what path would A* search return, starting from A and with G as a goal?

① (AB, 1+6), (AD, 2+6)
② (AD, 2+6), (ABC, 4+5)
③ (ADE, 5+3) (ABC, 4+5)
④ (ADEG, 8) DONE.

1.3 Is the heuristic admissible? Why or why not?

Not admissible, the h functi is really bad as it does not really reflect the distance from a specifi vertex to the target

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.

① (C,1), (B,2)

② (B,2), (D,2), (E,3) add AC

③ (D,1), (E,3) add AB

④ (E,3), (F,4) add BD

⑤ (F,1) add CE

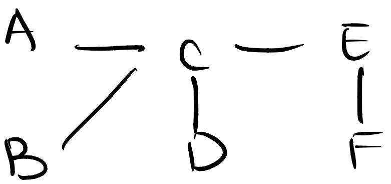
⑥ add EF

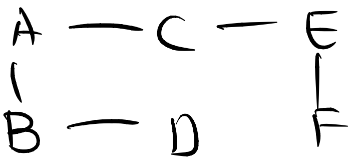
2.2 Use Kruskal's algorithm to find a minimum spanning tree.

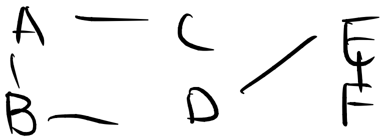
(AC,1) (BD,1) (EF,1) (AB,2) (BC,2) (CD,2) (CE,3) (DE,3) (DF,4)

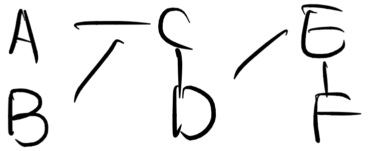
① ② ③ ④ ✗ ✗ ⑤

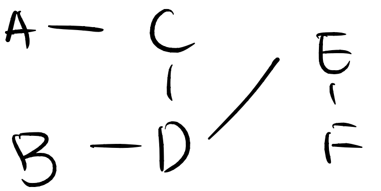
2.3 There are quite a few MSTs here. How many can you find?

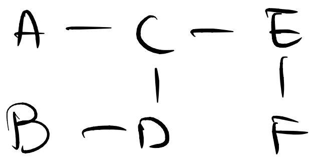












3 Mechanical Sorting

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

0 4 2 7 6 1 3 5

0 4 2 7 6 1 3 5

0 2 4 7 6 1 3 5

0 2 4 7 6 1 3 5

0 2 4 6 7 1 3 5

0 1 2 4 6 7 3 5

0 1 2 3 4 6 7 5

0 1 2 3 4 5 6 7

(b) Selection sort

0 4 2 7 6 1 3 5

0 1 2 7 6 4 3 5

0 1 2 7 6 4 3 5

0 1 2 3 6 4 7 5

0 1 2 3 4 6 7 5

0 1 2 3 4 5 7 6

0 1 2 3 4 5 6 7

(c) Merge sort

0 4 2 7 6 1 3 5

04 27 61 35

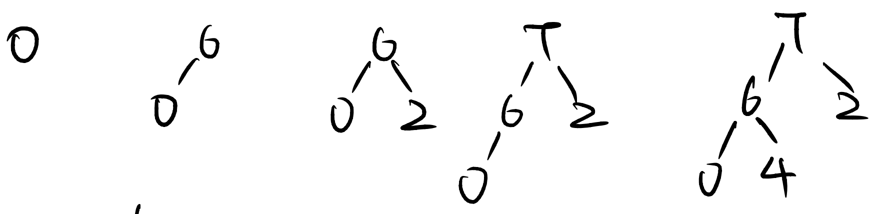
04 27 16 35

0247 1356

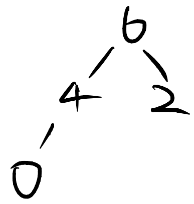
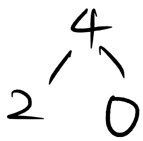
0123456

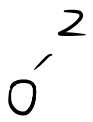
(d) Use heapsort to sort the following array (hint: draw out the heap). Draw out the array at each step:

`0, 6, 2, 7, 4`



① 76204

⇒  ② 64207 ⇒  ③ 42067

⇒  ④ 20467 ⇒ 0 02467.