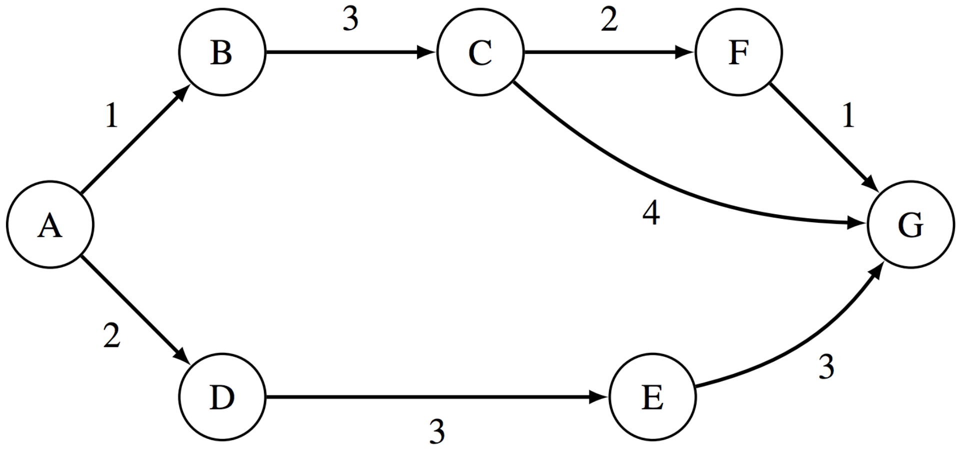
CS 61B
Spring 2018

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.

vertex	edgeTo	distTo
✓ A	/	0
✓ B	A	1
✓ C	B	4
✓ D	A	2
✓ E	D	5
✓ F	C	6
✓ G	~~C~~ F	~~8~~ 7

$$\begin{tikzcd} & B \arrow[r] & C \arrow[r] & F \arrow[r] & G \; ✓ \\ A \arrow[ur] \arrow[dr] & & & & \\ & D \arrow[r] & E & & \end{tikzcd}$$

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?

我觉得是走到G就结束了，再好的也不管了… ✓

In A* algo not every vertex visited …

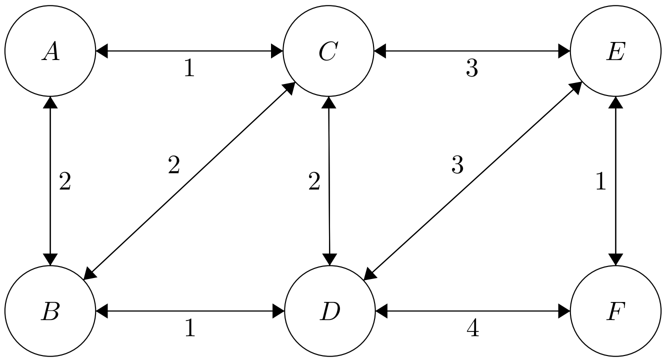
vertex	edgeTo	priority
✓ ~~A~~	~~/~~	~~8~~
✓ ~~B~~	~~A~~	~~7~~
✓ C	B	9
✓ ~~D~~	~~A~~	~~8~~
✓ ~~E~~	~~D~~	~~8~~
F		
✓ G	E	8

$A \to D \to E \to G$ ✓

1.3 Is the heuristic admissible? Why or why not?

No, because h(c, g) overestimates
h(a, g) also overestimates

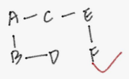
2 Minimum Spanning Trees



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.

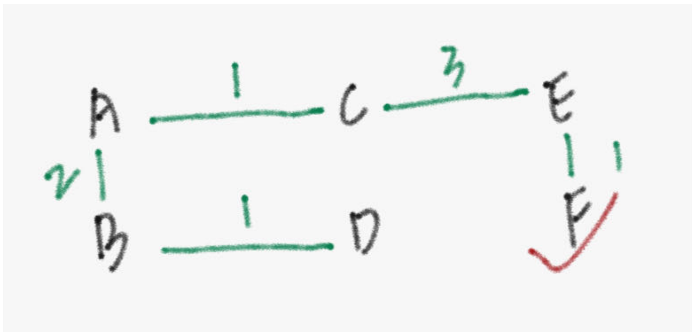
vertex	edgeTo	distTo
✓A	/	0
✓B	A	2
✓C	A	1
✓D	B	1
✓E	C	3
✓F	E	1

what's different from Dijkstra:
it's distance to the MST under construction
not to the initial vertex

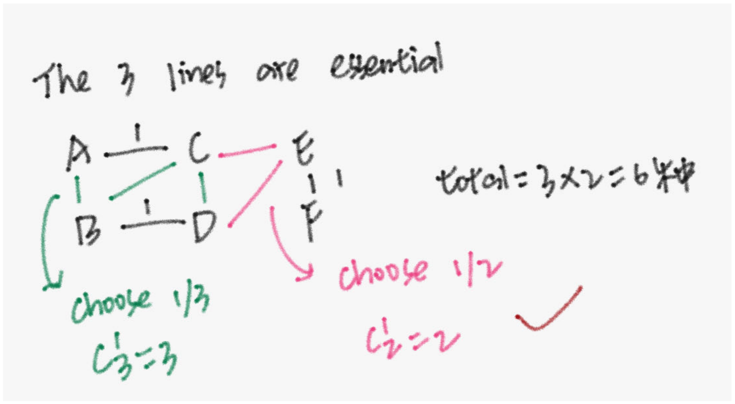


The 3 lines are essential

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?



The 3 lines are essential

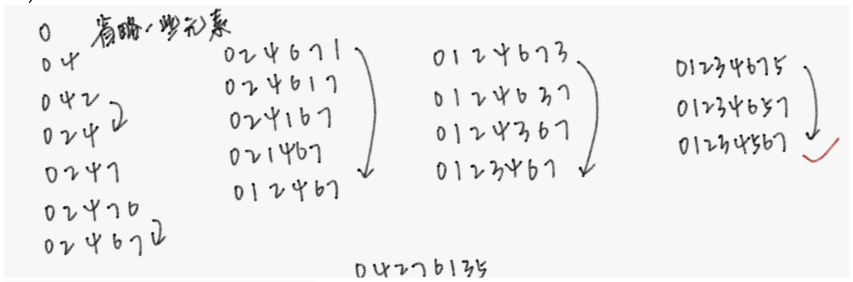
total = 3×2 = 6种

3 Mechanical Sorting

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

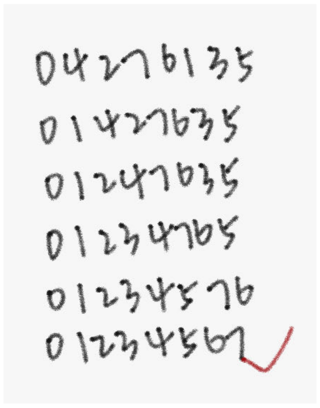
`0, 4, 2, 7, 6, 1, 3, 5`

(a) Insertion sort



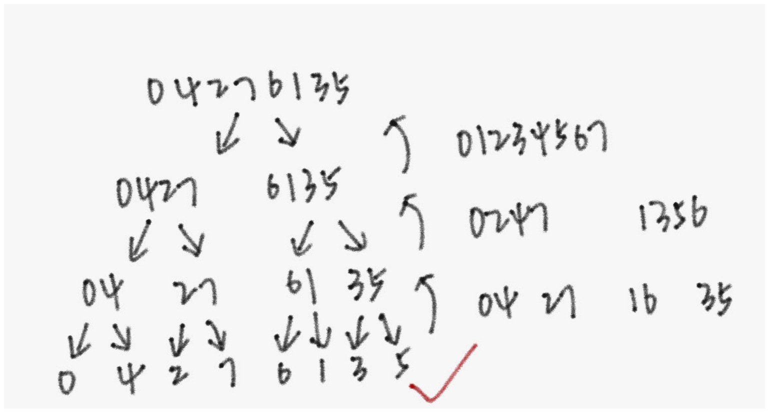
0 省略一些元素
04
042
024
0247
02476
02467
024671
024617
024167
021467
012467
0124673
0124637
0124367
0123467
01234675
01234657
01234567 ✓
04276135

(b) Selection sort



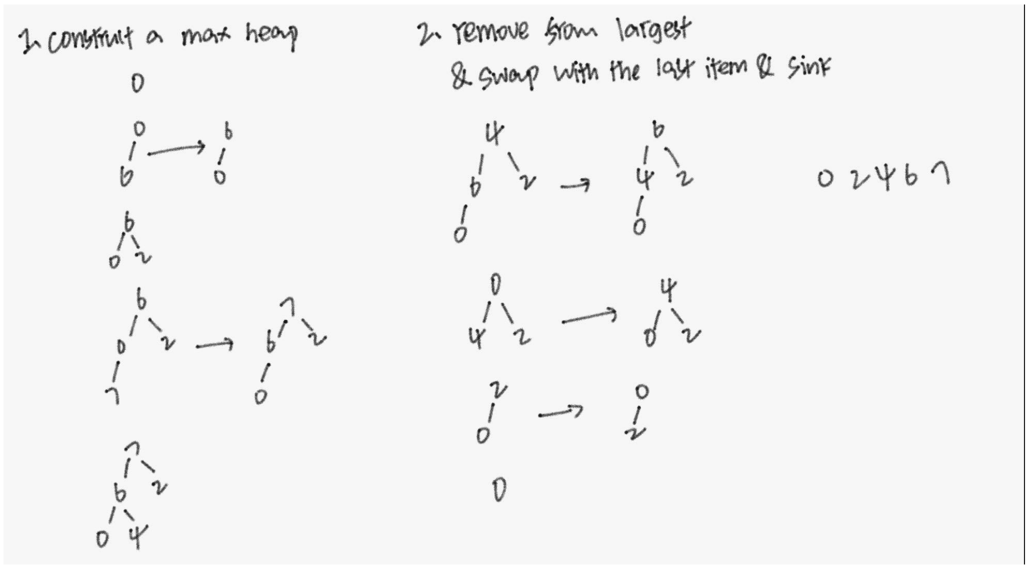
04276135
01427635
01247635
01234765
01234576
01234567 ✓

(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`


1. construct a max heap
2. remove from largest
& swap with the last item & sink
0 2 4 6 7