Use recurrences to show that binary search on a sorted array runs in $O(\log n)$ time

Pseudo code ← if array size is 1 check that element

NR { for a lo and hi
calculate a mid

NR - if toFind > mid
R → recurse on 2nd half
NR - if toFind < mid
R → recurse on first half
NR - if toFind == mid
return true;

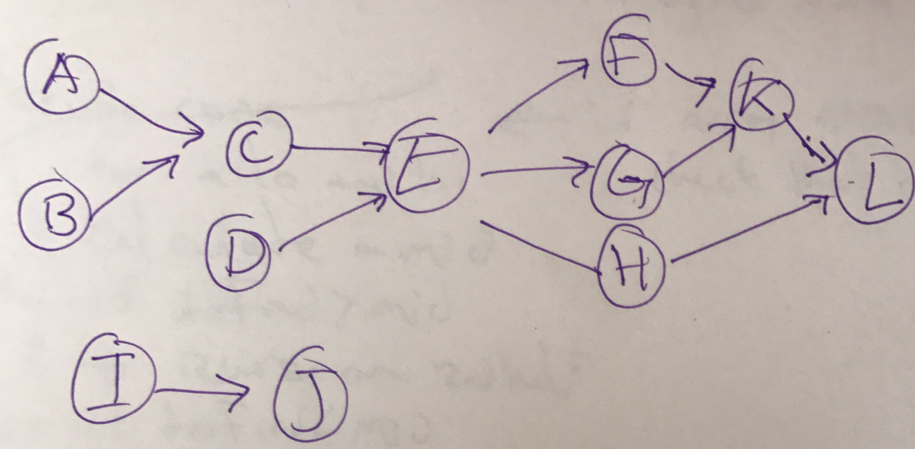
$$T(n) = O(1) + T(n/2)$$

$$T(n) = O(1) + T(n/2)$$
$$= O(1) + O(1) + T(n/4)$$
$$= O(1) + O(1) + O(1) + T(n/8)$$ ← → $\log_2 n$ divisions for $n < 2$

$$= \log_2 n \cdot O(1)$$

$$T(n) = O(\log n)$$

Provide two topological orderings of the following graph. Show steps



A	B	C	D	E	F	G	H	I	J	K	L
∅	∅	~~2~~	∅	~~2~~	~~1~~	~~1~~	~~1~~	∅	~~1~~	~~2~~	~~2~~
		0		~~1~~	~~0~~	~~0~~			0		
				0							
					0	0	0				
										0	~~1~~
											0

ABDI, CJ, E, FGH, K, L

JC

Starting with an array of size 7, insert the following elements into a hash table using Linked-list chaining.

Use k%7 as the hash function

51, 14, 91, 16, 3, 11, 28, 2, 23, 37

0 → 14 → 91 → 28

1

2 → 51 → 16 → 2 → 23 → 37

3 → 3

4 → 11

5

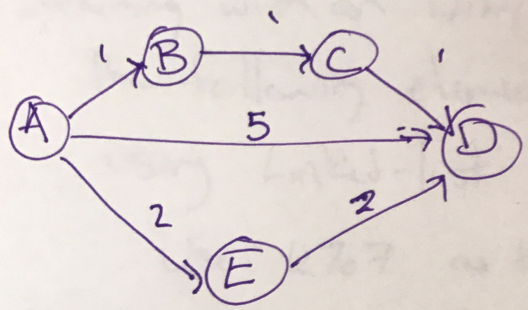
6

Discuss any interesting results as far as runtime. Is a resize advisable at any point?

$5/9$ elements are in index 2

O(n) find WC

Resize: recognize that 2 is overloaded



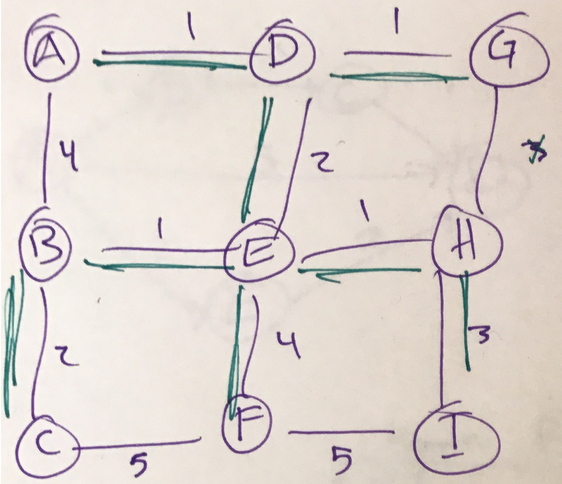
Use Dijkstra's algorithm to find the shortest path from A to D

Show intermediate steps

	Known	Path	Previous
A	1	0	X
B	2	1	A
C	4	2	B
D	5	~~5~~ ~~4~~ 3	~~A~~ ~~E~~ C
E	3	2	~~B~~ A

(A, B), (B, C), (C, D)

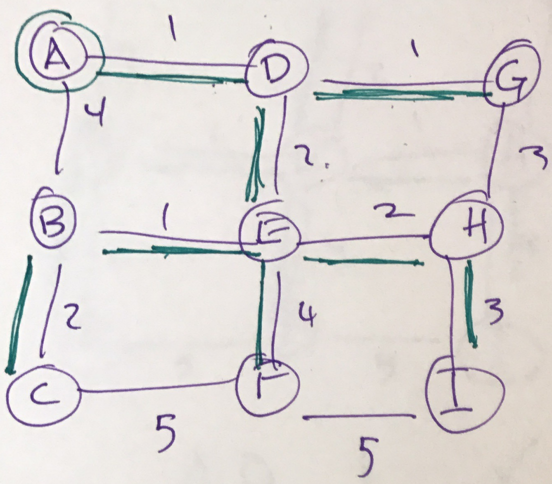
cost: 3



Provide the MST using kruskal's algorithm

A, D	1	✓	
D, G	1	✓	
B, E	1	✓	
E H,	1	✓	
D, E	2	✓	
B, C	2	✓	
G, H	3	X	D, G, E, H
H, I	3	✓	
A, B	4	X	A, B, D, E
EF	4	✓	
CF	5		
FI	5		

V-1 edges



Provide the MST for this graph using Prim's algorithm. Show steps: Indicate if multiple MSTs may exist and why

	K	edge	prev
A	1		
B	5	~~4~~,1	~~A~~,E
C	6	2	B
D	2	1	A
E	4	2	
F	9	4	
G	3	1	D
H	7	~~3~~ 2	
I	8	3	H

Select A

A	x
D	(A,D) 1
G	(D,G) 1
E	(D,E) 2
B	(E,B) 1
C	(B,C) 2
H	(E,H) 2
I	(I,H) 3
F	(E,F) 4

76

9	4	12	2	10	6	13	1	5	8	3	7	11

Use Quicksort and median of 3 approach to sort the following list. Show steps.

9	4	7	2	10	6	13	1	5	8	3	12	11

9 4 7 2 3 6 13 1 5 8 10 12 11

9 4 7 2 3 6 8 1 5 13 10 12 11

5 4 7 2 3 6 8 1   9   13 10 12 11

2 4 7 5 3 6 8 1   11 10 12 13

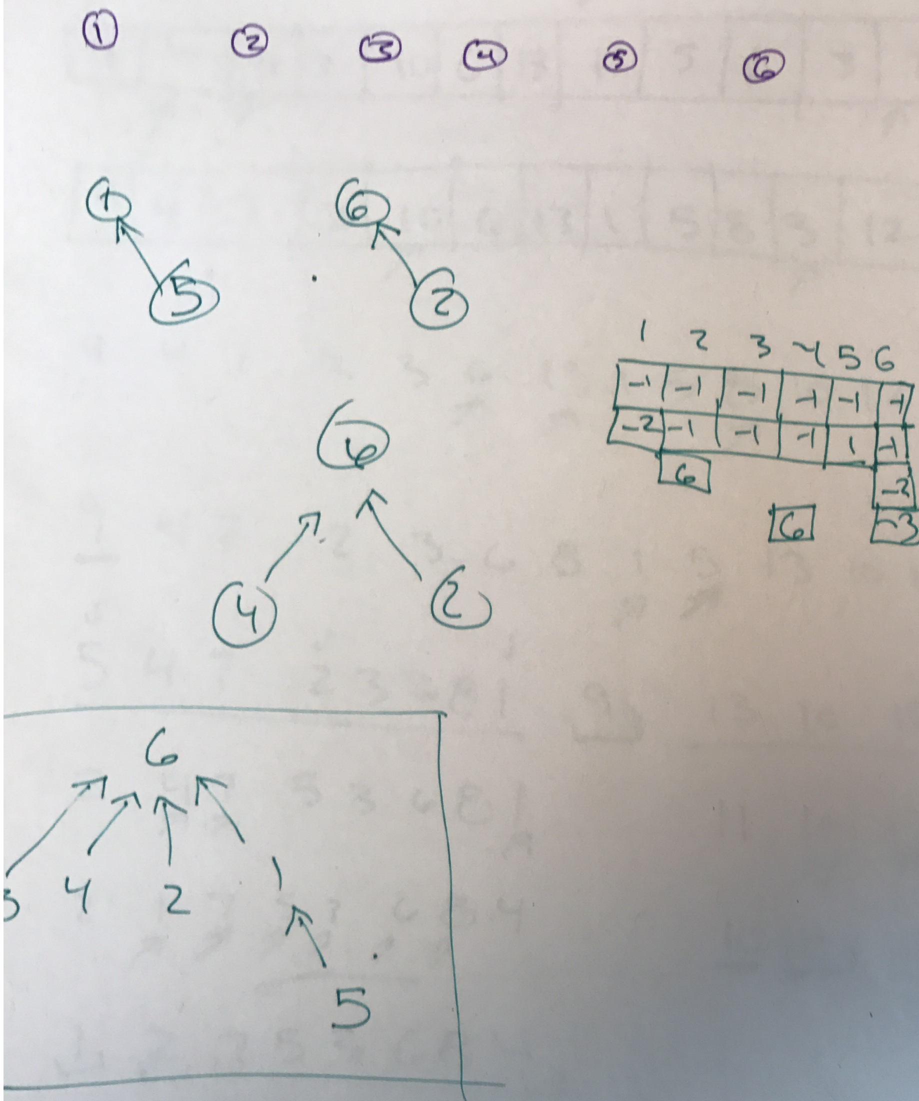
2 1 7 5 3 6 8 4   10 11 12 13

1 2 7 5 3 6 8 4

4 5 3 6 8 7

3 3 5 6 7 8

3 4



Show the uptree after the following operations.
Use weighted union and path compression.

union(1,5)
union(6,2)
union(4,~~2~~2)
union (1,2)
find (6)
find (3)
union(3,2)
find(1)

In case of ties, let the first argument of union be the representative

Design an algorithm which determines whether 2 integers share any common factors.

Recall, that if two numbers are prime and not equal, they share no common factors.

Also recall that if a number n is composite (non-prime) it must have a factor $k \leq \sqrt{n}$

All numbers have a unique prime factorization

j, k are the numbers

if $j \% p$ and $k \% p = 0$, then they have a common factor p

if $j > k$,

check all p from 2 to k

check all p from 2 to $\sqrt{k}$

,3

6