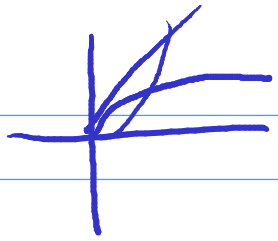
JoeyIsee

Everything You need to Know about Big O !



$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$$

1.

```
int sum()
{
    sum=0;
    for(int i=0; i<10; i++)
    {
      sum = sum+1;
    }
    return sum;
}
```

sum = 0

0 ... 10 0, 1, 2 ... 10

$O(10) \rightarrow O(c) \rightarrow O(1)$

```
1: for i = 1 to n do
2:   j = n
3:   while j ≥ 1 do
4:     Print (i, j)
5:     j = ⌊j/2⌋
6:   end while
7: end for
```

1 ... n $\rightarrow O(n)$

j = n

j = j/2

$\times$ $O(\log n)$

$O(n \log n)$

Input : The number n is an integer greater than one.

```
Sum = 0
i = 1
while i ≤ n do
    j = 1
    while j ≤ 100 do
        Sum = Sum + 1
        j = j + 2
    end while
    i = i + 1
end while
```

i: 1 ..... n $\rightarrow O(n)$

j: 1 ..... 100 $\rightarrow O(100)$

j + 2

i + 1

$O(1)$

$O(1) \times O(n)$

$\rightarrow O(n)$

Joey Issa

```
i=1;
while(i<=n){
    i=2*i;
}
```

$1 \dots n$

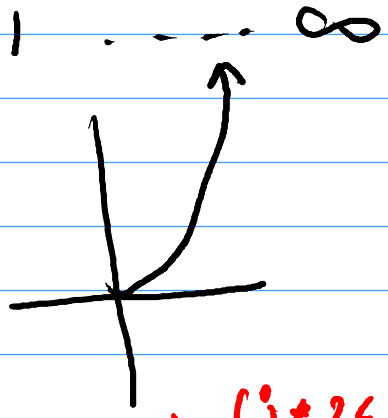
$i = 2*i$ → $O(\log_2 n)$

$i = 1 \quad (1)$
$i = 2 \quad (2)$
$i = 4 \quad (3)$
$i = 8 \quad (4)$
$i = 16 \quad (5)$

another example

while $(i \le \frac{n}{2})$ { $1 \dots \infty$

$O(\log_2 n)$



while $(i * 2 \le \frac{n}{2})$

$i * 2$

$O(\log n)$

```
for(i = 0;i < n; i++)
for(j = 0; j< n; j++)
    sum += i*j;
```

$0 \dots n$
$0 \dots n$

ignore

$O(n) \times O(n)$
$= O(n^2)$

```
int sum = 0;
for (int i = 1; i < N; i *=2)
{
    for (int j = 0; j < N; j++)
    {
        ArrayX[j]= ArrayY[i];
        sum++;
    }
}
```

Does not affect loop

$i : 1 \dots N$ $\quad i*2$ $\quad O(\log N)$

$j : 0 \dots N$ $\quad j++$ $\quad O(N)$

for loop 3
$0 \dots n$
3
for loop 2 $0 \dots n$)

$O(N \log n)$

Input: The number n is divisible by 4.

1: for $i = 2$ to n do
2: for $j = 0$ to n do
3: Print (i, j)
4: $j = j + \lceil n/4 \rceil$
5: end for
6: end for

$i : 2 \dots n$ $\quad O(n)$

$j : 0 \dots n$ $\quad O(n)$ $\quad O(1)$

$j + \lceil n/4 \rceil$

∞ $\quad n$

Bonus! =)

Answer $O(n)$

will only take 4 passes

Find the complexity of an algorithm that determines a patient between 100 Normal people using big O notation.

$O(100) \rightarrow O(1)$

n $\quad O(n)$

$n/2 \cdots n$

```
int mycode(int n) {
   int i, j, k = 0;
   for (i  = n/2; i <= n; i++)
      for (j = 2; j <= n; j = j * 2)
         k = k + 2;
   return k;
}
```

$n/2 \cdots n \rightarrow O(n)$

$2 \cdots n \rightarrow O(\log n)$

$j * 2$

i = half of ∞ / n → pass through the loop

∞ times

while of ∞

Joey Issa

```
1  /** Returns true if there are no duplicate elements in the array. */
2  public static boolean unique1(int[ ] data) {
3    int n = data.length;
4    for (int j=0; j < n−1; j++)
5      for (int k=j+1; k < n; k++)
6        if (data[j] == data[k])
7          return false;                    // found duplicate pair
8    return true;                           // if we reach this, elements are unique
9  }
```

O(n) — O(n) — 0 --- n-1 — 1 . . . n — $O\left(\frac{n(n-1)}{2}\right)$ = $O(n^2)$

O(n-1) O(n)

O(n) x O(n)

$O(n^2)$

i	j
0	1
1	2
1	3
3	4
4	5
n-1	n

PreOrder, InOrder, PostOrder

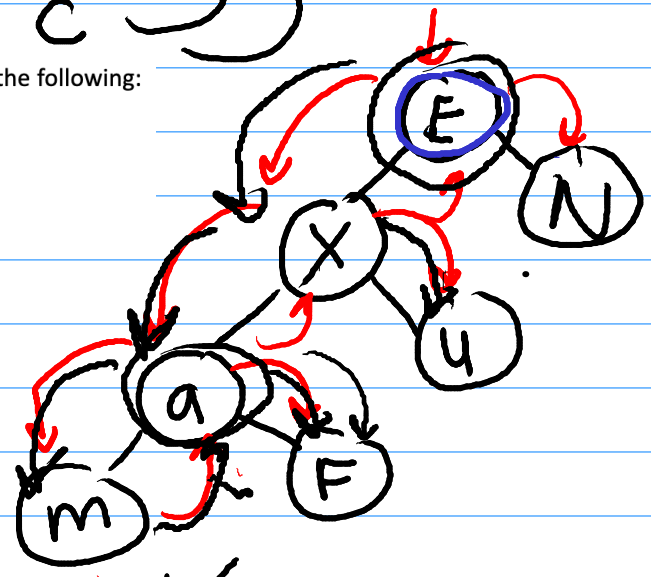
CLR LCR LRC

C — LR

★

R-8.22 Draw a binary tree *T* that simultaneously satisfies the following:

- Each internal node of *T* stores a single character.
- A *preorder* traversal of *T* yields EXAMFUN.
- An *inorder* traversal of *T* yields MAFXUEN



preorder CLR EXAMFUN (root)

Inorder LCR MAFXUEN

Test: Preorder: EXAMFUN ✓

print( )

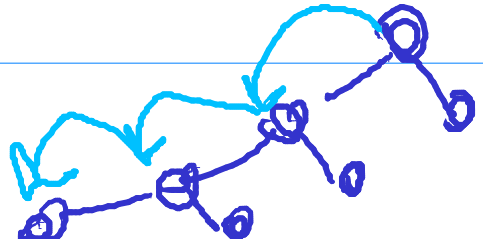
pre(node→left)

pre (node→Right)

Inorder:

LCR

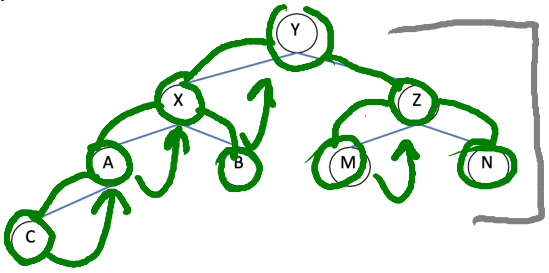
MAFXUEN



Joey Issa

Inorder → LCR PreOrder → CLR PostOrder → LRC

Traversals



Inorder → LCR

"print"

In() { Node → Z

In(node→Left)

print → Z

In(node→Right)

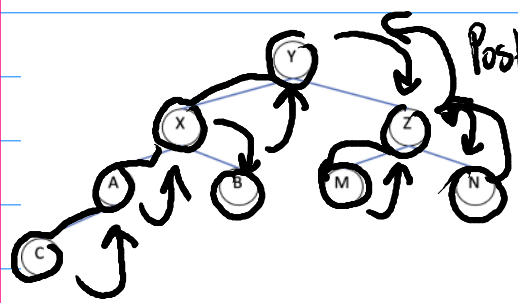
}

CAXBYMZN

M Z N

Left? Yes node → M

Right? Yes Z → N



Post: LRC CABXMNZY

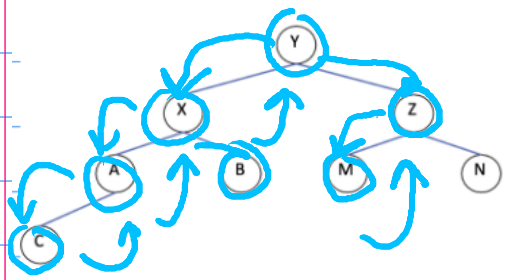
: CABMNZY

X

pos(node→Left)

pos(node→Right)

print(node)



CLR

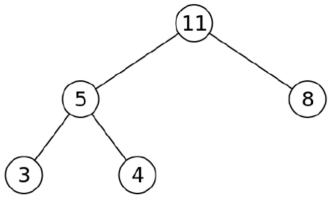
YXACBZMN

Heaps

JoeyTesa

Can you swap 4 & 8 =

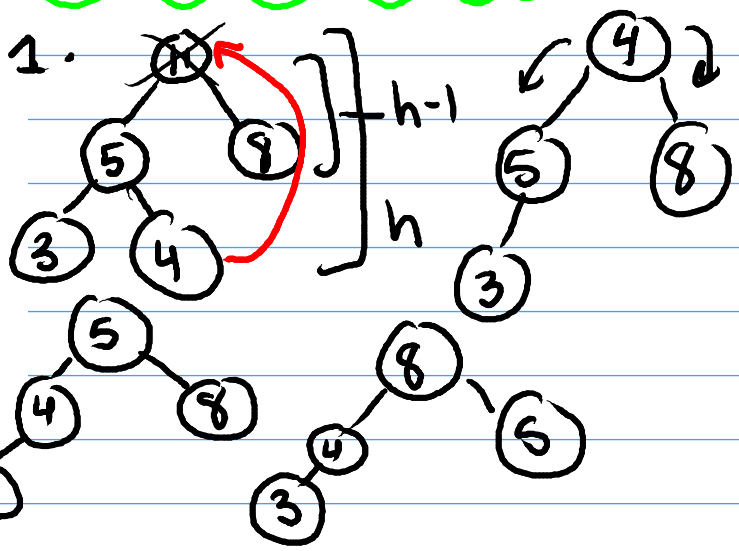
Question 11 [3 points] Consider the following max-heap:



and the following operations :

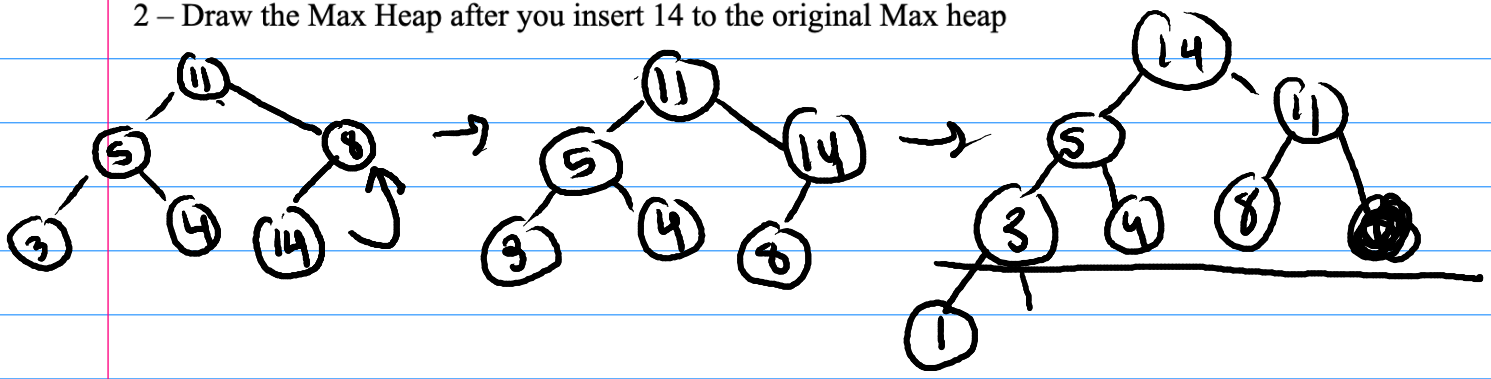
`insert(key)` : insert a node with the value key in the heap

`removeMax()` : delete and return the element of maximum key in the heap



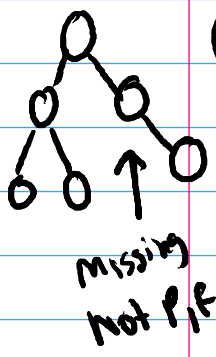
1- Draw the Max heap after Remove Max

2 – Draw the Max Heap after you insert 14 to the original Max heap



Joey Issa

Complete, full, Perfect <<Fast Review>>

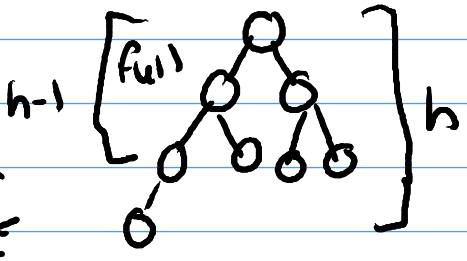


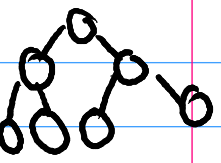
Complete – Perfect tree from Height (h-1) w >1 leaves @ h

full – each node has two children

Perfect → full

p→q	
TT	T
TF	F
FT	T
FF	T





Perfect – (full + Complete) full binary tree with all leaves @ the same level

* i = internal nodes n = total nodes
e = external nodes h = height of tree

Full tree [$i = (n-1)/2$ $e = i+1$ $n = i+e$ $n = 2e-1$

Perfect [* $n = 2^{h+1} - 1$ *

* What is the # of internal nodes if you have a Perfect tree with height = 3
↑ full

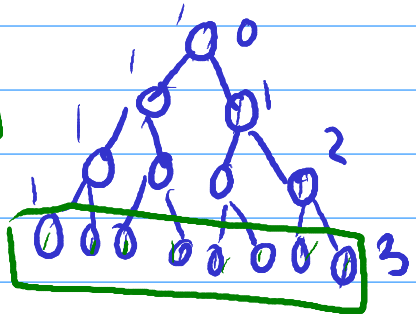
Ask for formula

$h = 2e+1$
$15 = 2e+1$
$14 = 2e$
$e = 7$

$e = i+1$

$n = 2e-1$
$n+1 = 2e$
$16/2 = 8$
$e = 8$

$n = 2^{h+1} - 1$
$n = 2^4 - 1$
$n = 15$



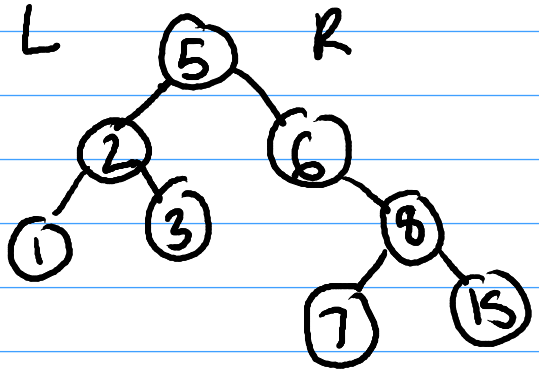
$e = i+1$
$8 = i+1$
$7 = i$
$i = 7$

You have numbers in this order: 5, 2, 6, 8, 15, 1

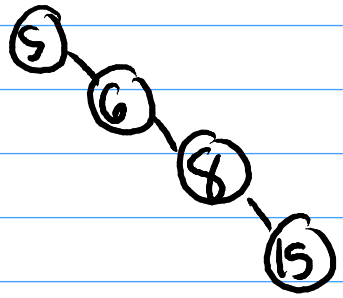
JoeyTesa

Create BST, Min Heap, Max heap, Avl

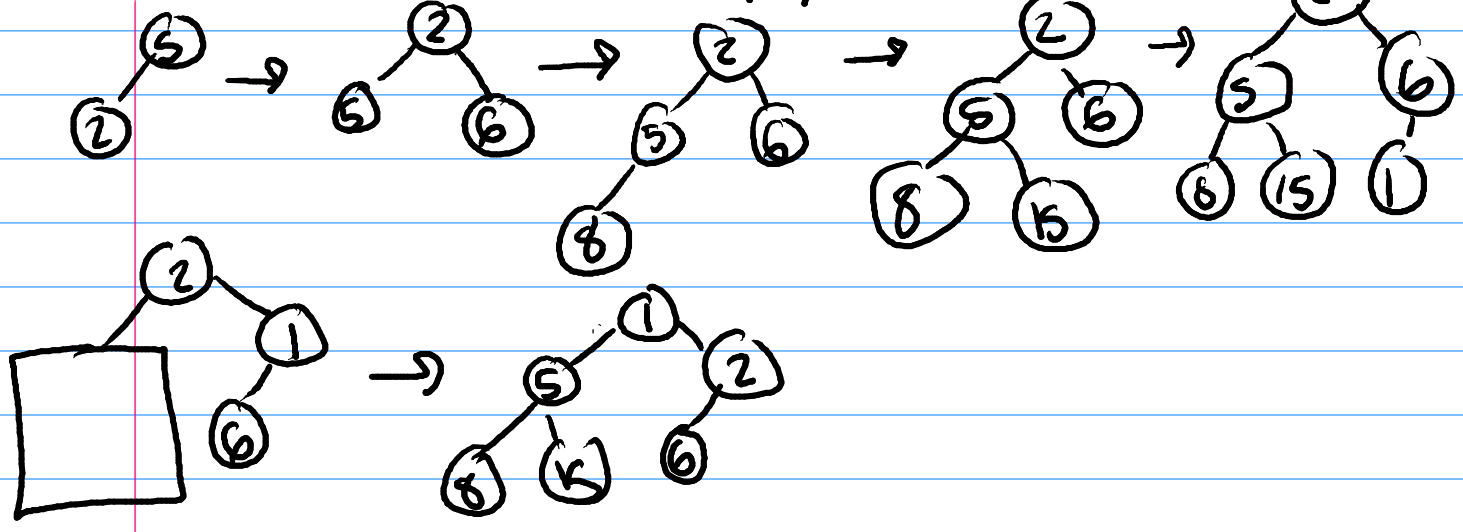
BST L R 7



5, 6, 8, 15

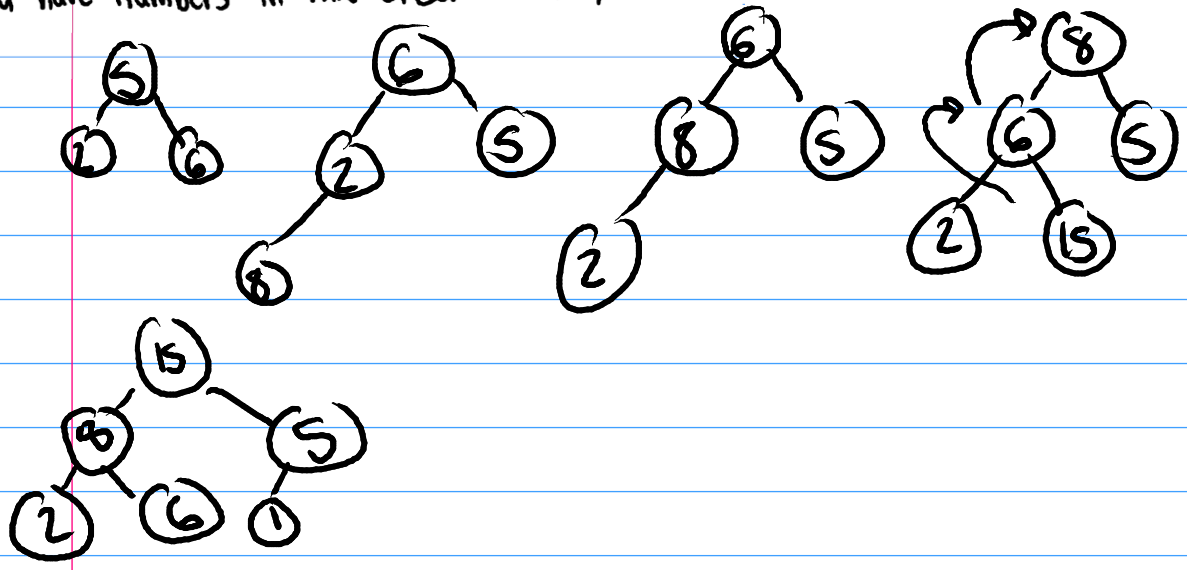


You have numbers in this order: 5, 2, 6, 8, 15, 1

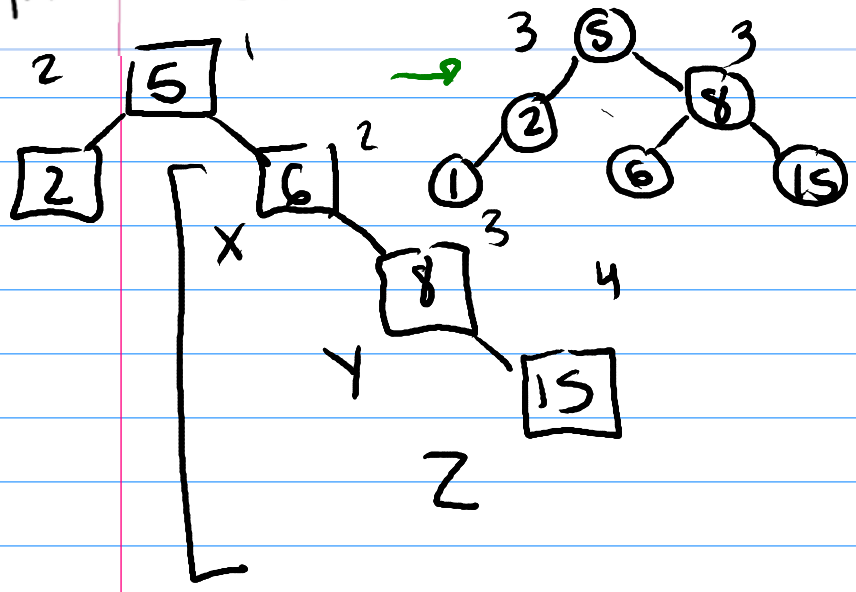


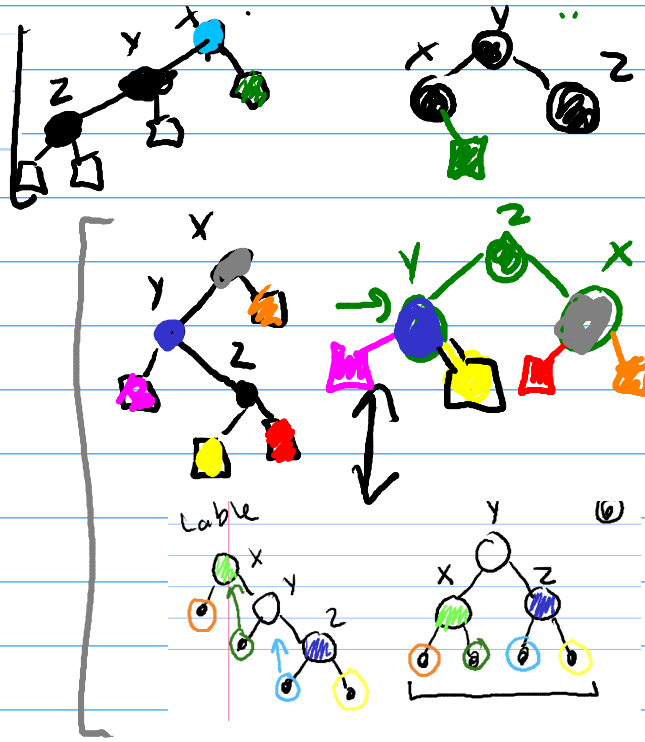
Joey Issa

You have numbers in this order: 5, 2, 6, 8, 15, 1

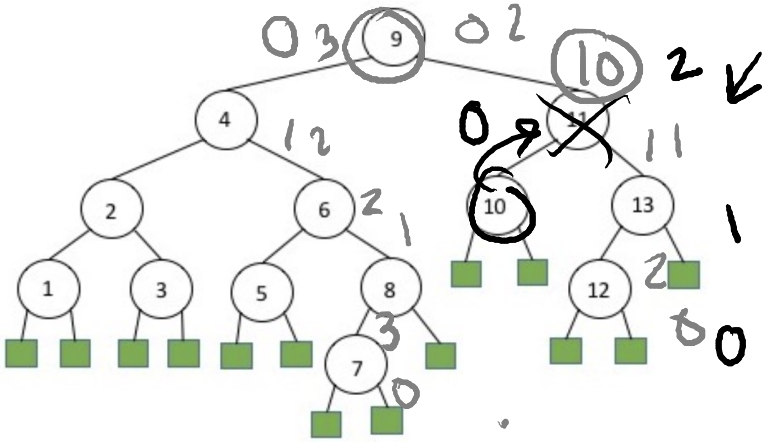


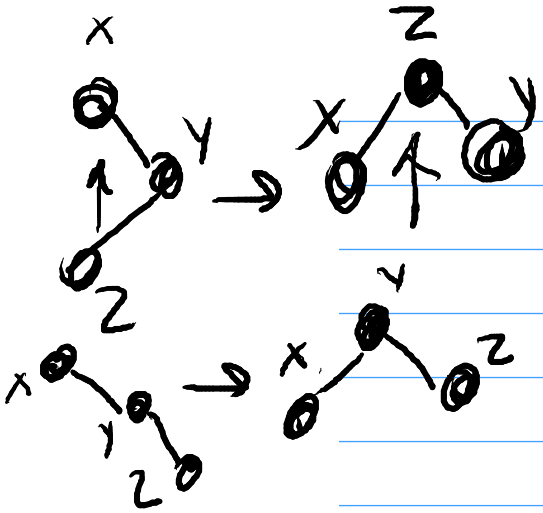
You have numbers in this order: 5, 2, 6, 8, 15, 1



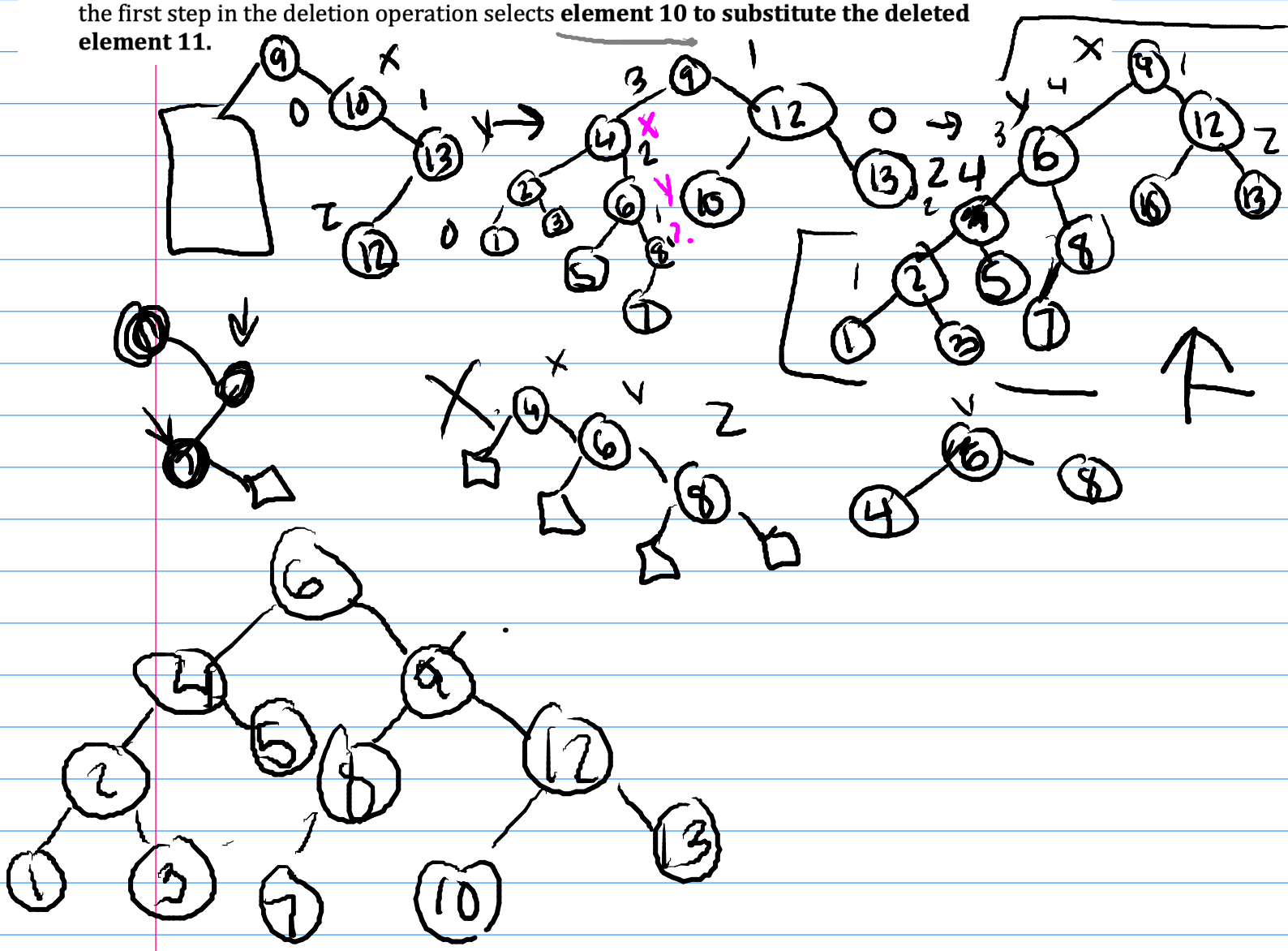


Question 17 [2 marks] Consider the following AVL tree.

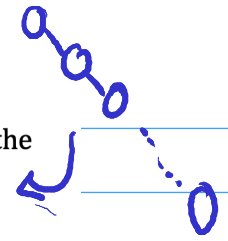




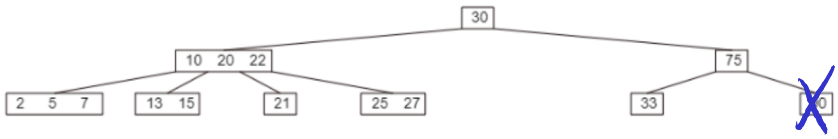
Show the relevant steps (tree transformations) after **the operation delete 11**, assuming the first step in the deletion operation selects **element 10 to substitute the deleted element 11.**



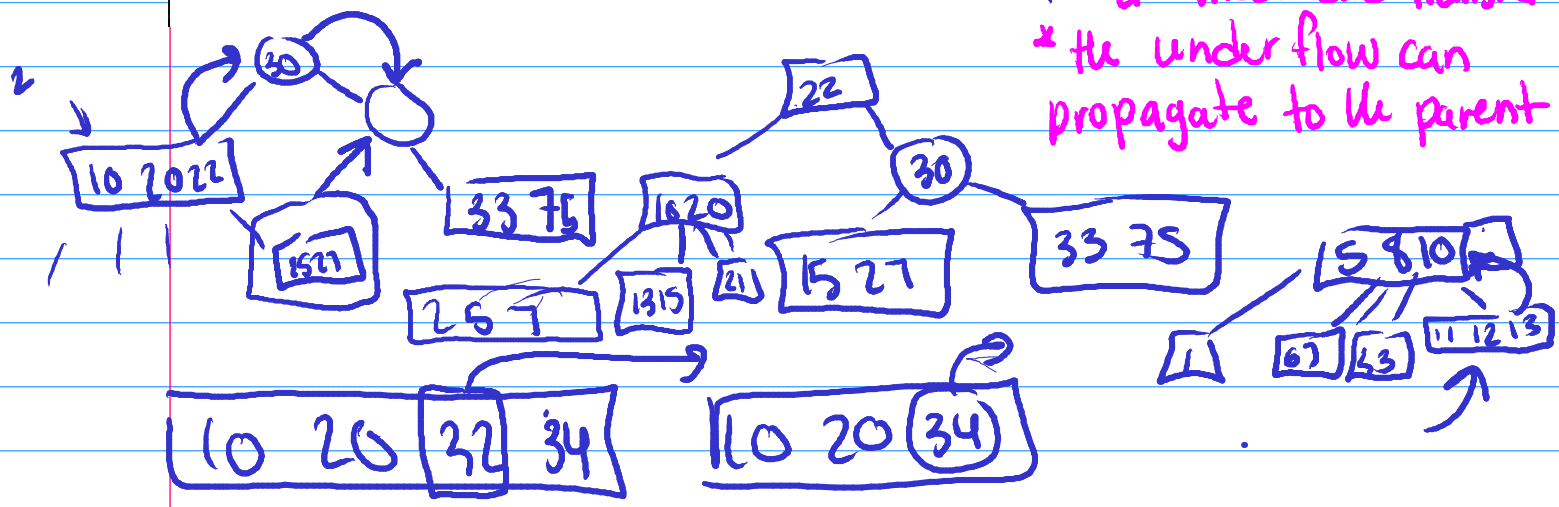
Which of the following alternatives is a true completion of the phrase:
The worst case running time of searching for a key in an AVL tree is (i) O(log n) but the worst case running time of searching for a key in a binary search tree is (ii) O(n)



c) Delete the key 90 from the following (2,4)-tree and show the resulting tree. Show any essential intermediate step.



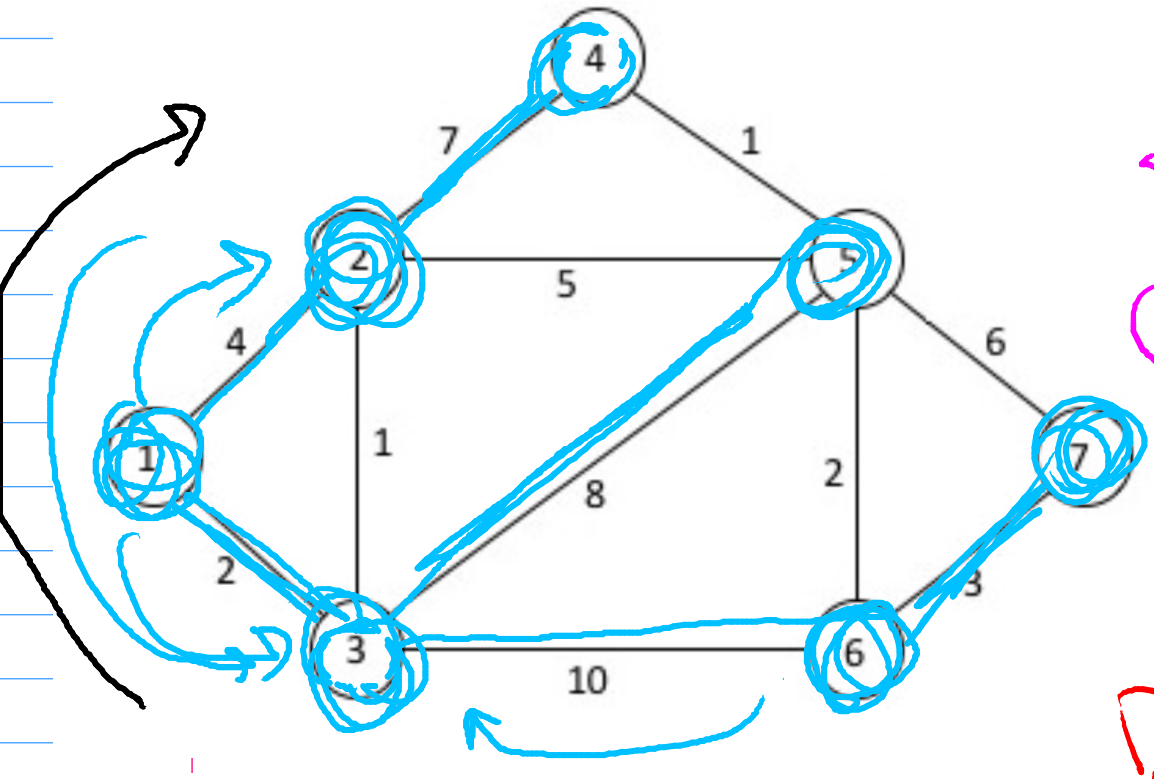
* every underflow can be solved using a series of Fusions & at most one transfer
* the underflow can propagate to the parent



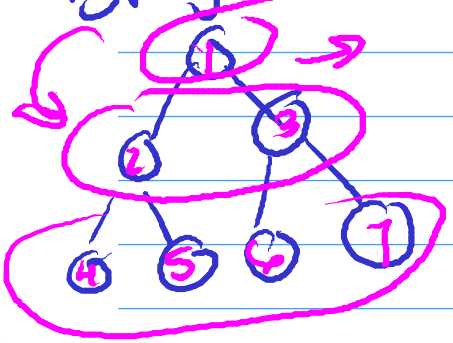
Dijkstra + Prim + Kruskal

JoeyIssa

BFS, DFS



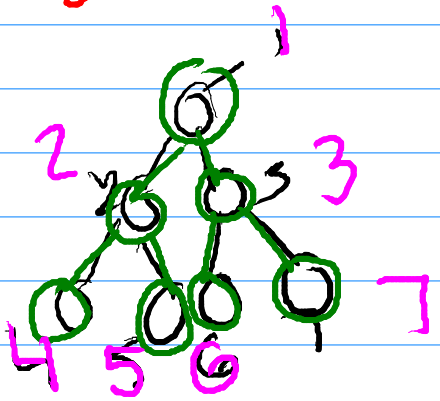
BFS



1 2 3 4 5

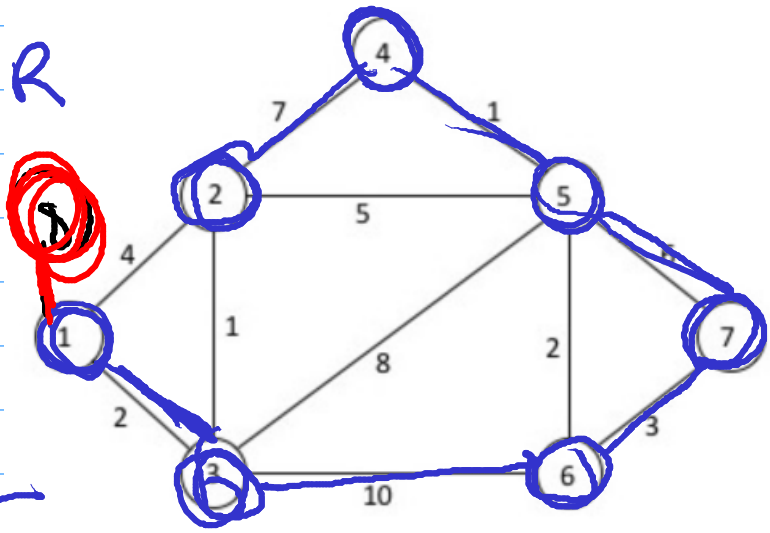
6 7

DFS



1 2 3 4 6 5 6 7

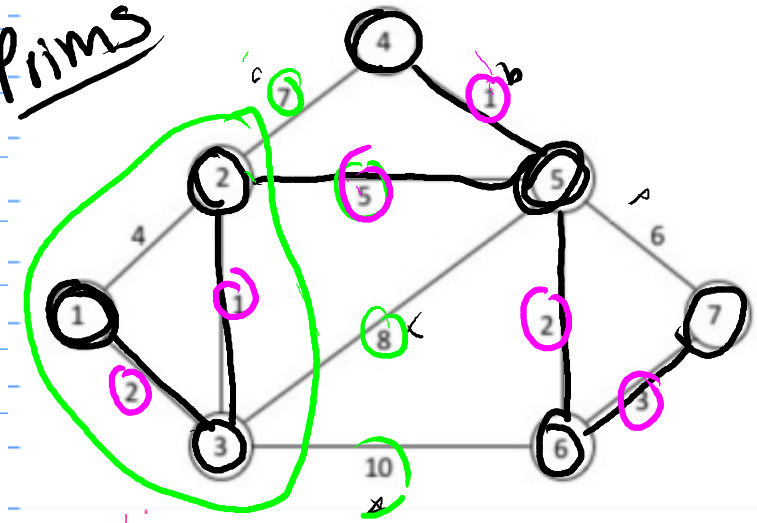
R



L

1 2 4 5 3 6 7

Prims



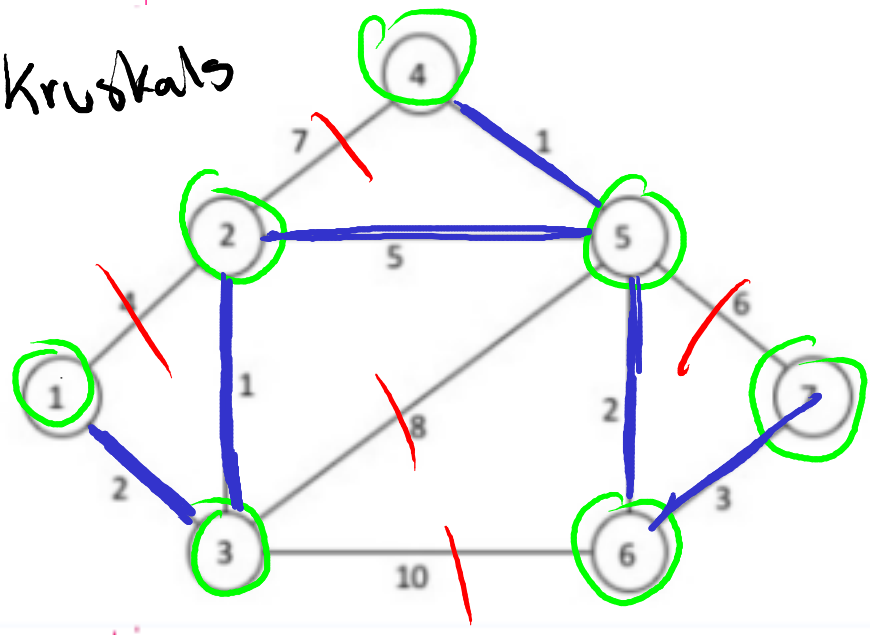
2+1+5
+1+2+3

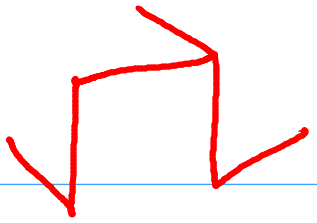
3+5+5

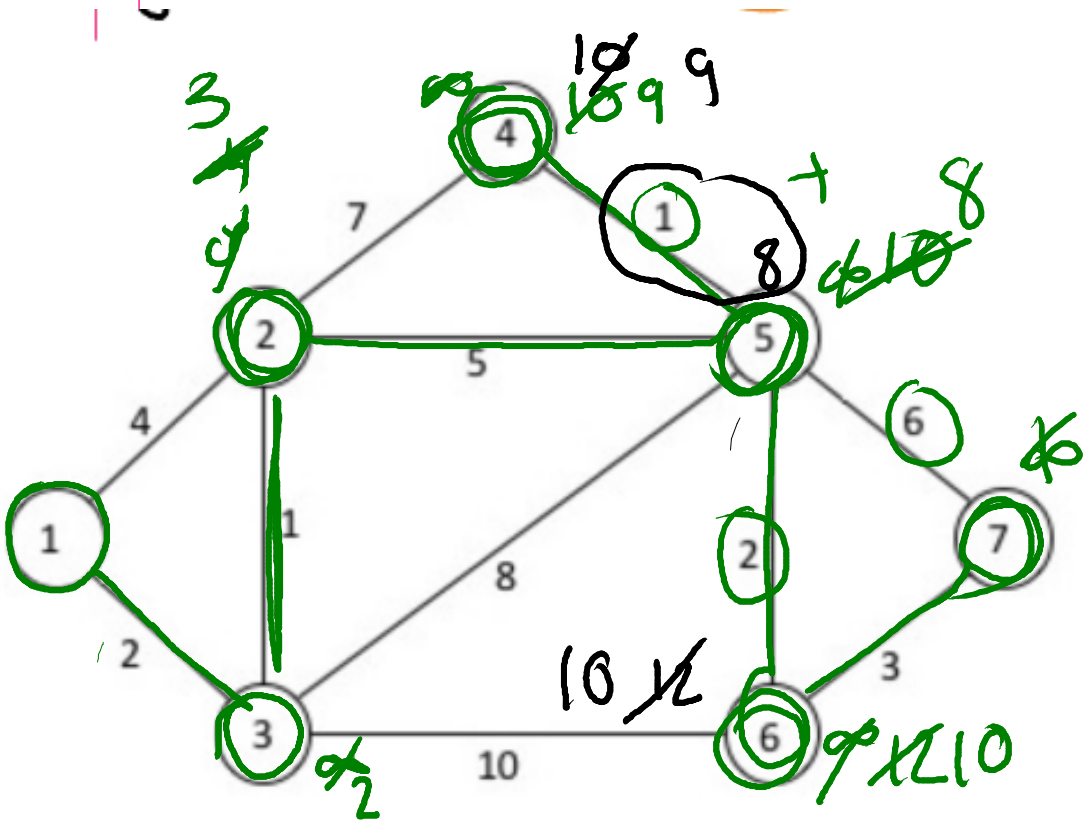
13

7
6
4
5
2
3
1

Kruskals







26+9
3 3
10 8
10 10
9
13 13

calculate

35

Hashing

yum!

Joey Issa

Question 10 [1 point] Consider the following Hash table where insertions are done using the hash function h(k)= k mod 7, and collisions are resolved with **quadratic probing**.

0	1	2	3	4	5	6
	15	1	8		5	

Regarding the ordering of insertion in this table, which answer is CORRECT?

A) Key 1 was the last key to be inserted.
B) Key 8 was the last key to be inserted.
C) Key 15 was the last key to be inserted.
D) It is impossible to determine which was the last key among 8 and 5.
E) None of the above is correct.

Question 11 [1 point] Consider the following Hash table where insertions are done using the hash function h(k)= k mod 7, and collisions are resolved with **linear probing**.

0	1	2	3	4	5	6
	15	1	8		5	

What is the **average number of probes** A for searching an existing key in this table?

A) A=1
B) 1 < A < 2
C) A= 2
D) A >2
E) None of the above is correct.

Question 12
Suppose you **insert element 2** in the table given in Question 11, still using linear probing. After this, you search for element 3. How many table positions must be probed until you conclude element 3 is not in the table?

A) 2 B) 3 C) 4 D) 5 E) 6 or more

Sorting Algorithms

Question 6 [1 point] The worst case running times of the following sort algorithms are:

	Insertion sort	Mergesort	Quicksort
A)	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n^2)$
B)	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n \log n)$
C)	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(n \log n)$
D)	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(n^2)$
E)	None of the above.		

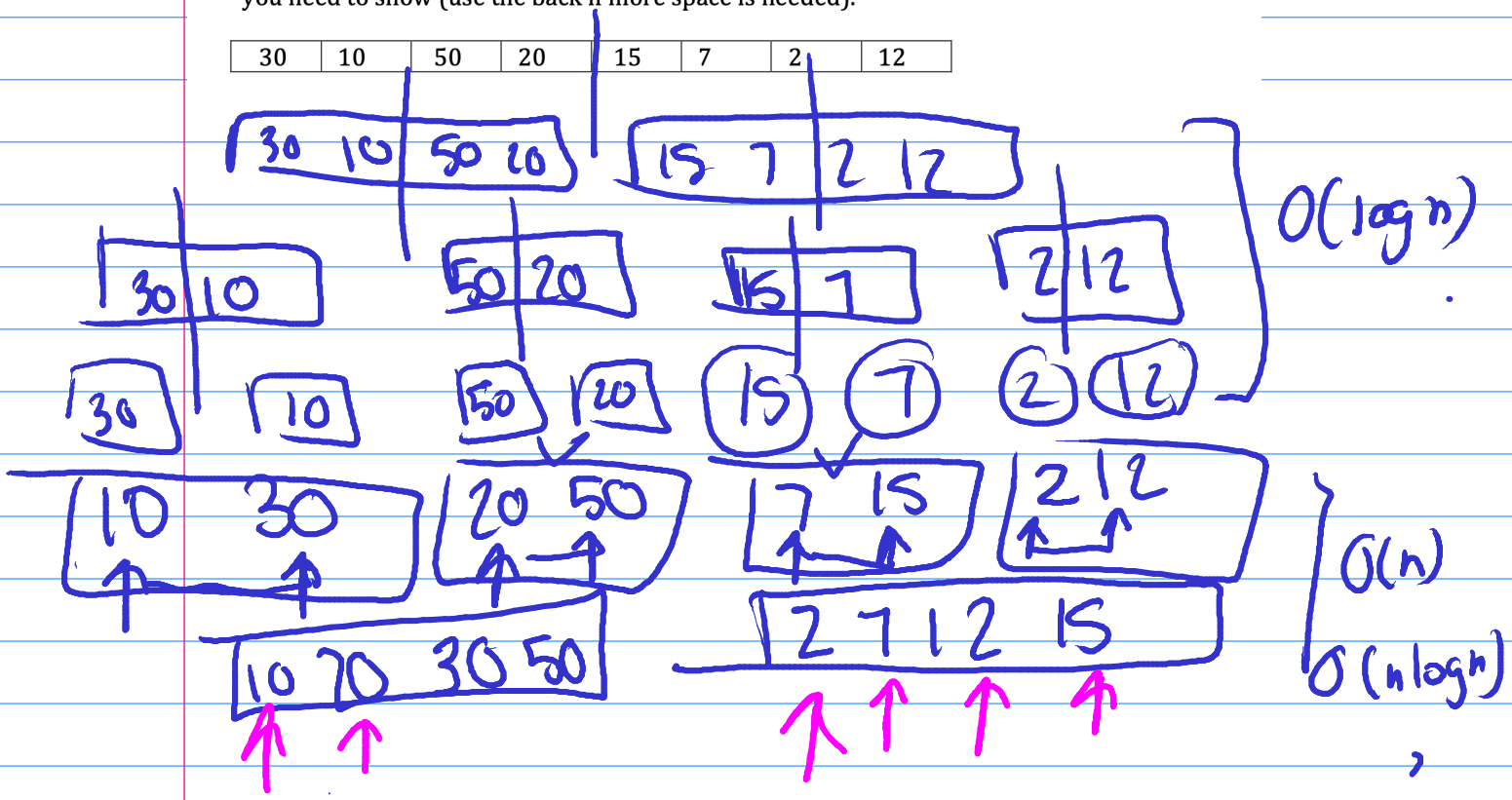
Insert $O(n^2)$
Bubble
Quick

$\Theta \sim O$

$O(n \log n)$

Question 20 [3 points] Mergesort
Simulate the execution of the in-place Mergesort algorithm for the given array below. Suppose that at the end of each "merge" step, the algorithm prints a line with the current contents of the full array.
Show each printout of this algorithm, and highlight the part of the array where the merge has been done. The number of blank arrays below may be more or less than the amount you need to show (use the back if more space is needed).

30	10	50	20	15	7	2	12



2 7 10 12 15 20 30 15