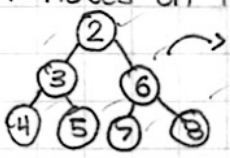
CS340 -> Assignment

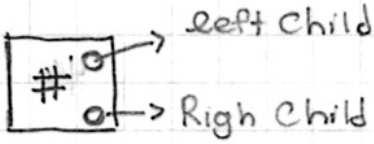
Maksim Sharoika

Problem 1)

Breadth first will start at the tree root, return all nodes on that level before moving on to the next level.



Queue is a FiFo structure that will use the push(add) and pop(remove) operations



Node

```
Algorithm levelOrder(root):
    if root is null
        return
    Queue "queue"
    tmp.push root
    while "queue" is not empty
        Node tmp = queue.front
        Output the value of tmp
        Pop the queue (removes tmp)
1)      if left node of tmp exists
            Push left node onto "queue"
2)      if the right node of tmp exists
            Push right node onto "queue"
```

if the root is none there is no tree for us to level order.

creating a temporary queue.

add the root to the queue.

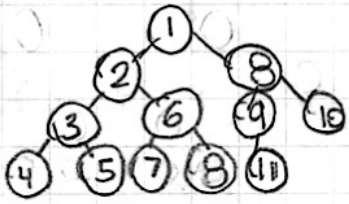
while the queue is not empty

take the node from the front of the queue or current leftmost node output it out and then pop it

if there is a left child add it to the queue

if there is a right child add it to the queue

Example:



Queue: 1 2 8 3 6 9 10 4 5 7 8 11

Output: 1 2 8 3 6 9

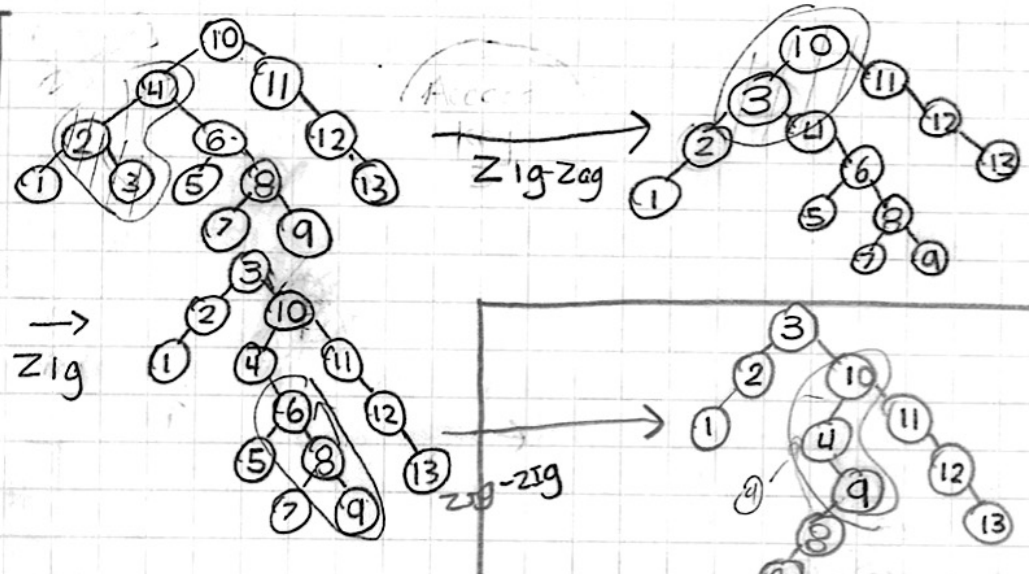
*we will stop here notice the nodes are queuing in level order, and the queue get larger compared to output

N is the number of nodes, with this algorithm for a level traversal we never have to visit a node more than once because we are working our way down the tree. Therefore at most we visit N nodes meaning this algorithm is O(N), to add onto this we are always moving from left to right and down, we never move up.

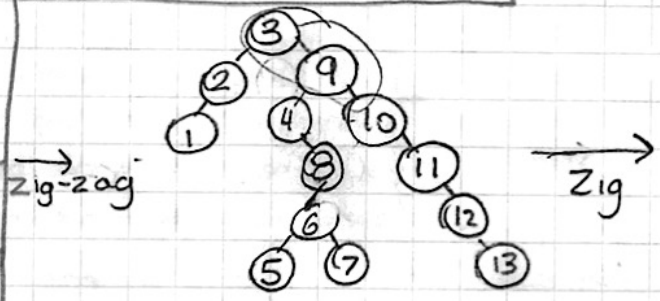
Problem 2)

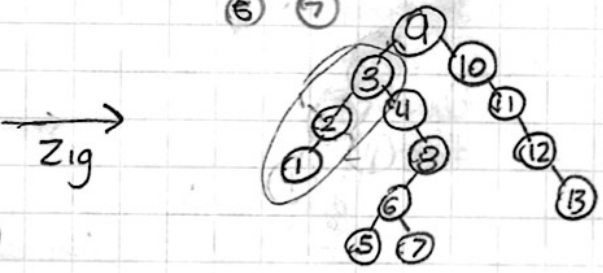
a)

Access Key 3

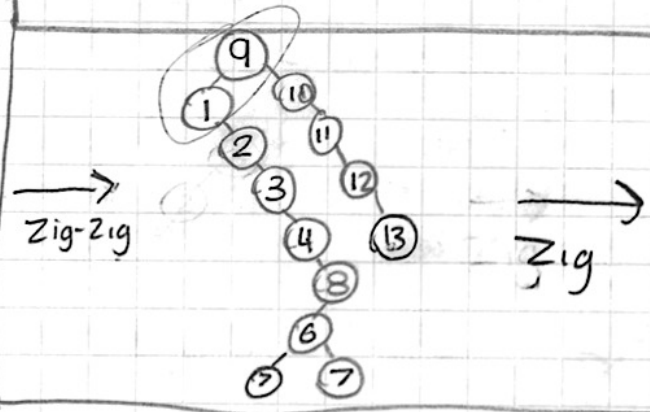


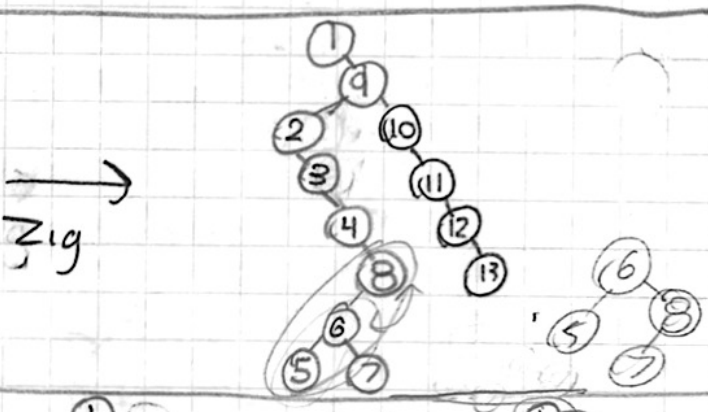
Access Key 9



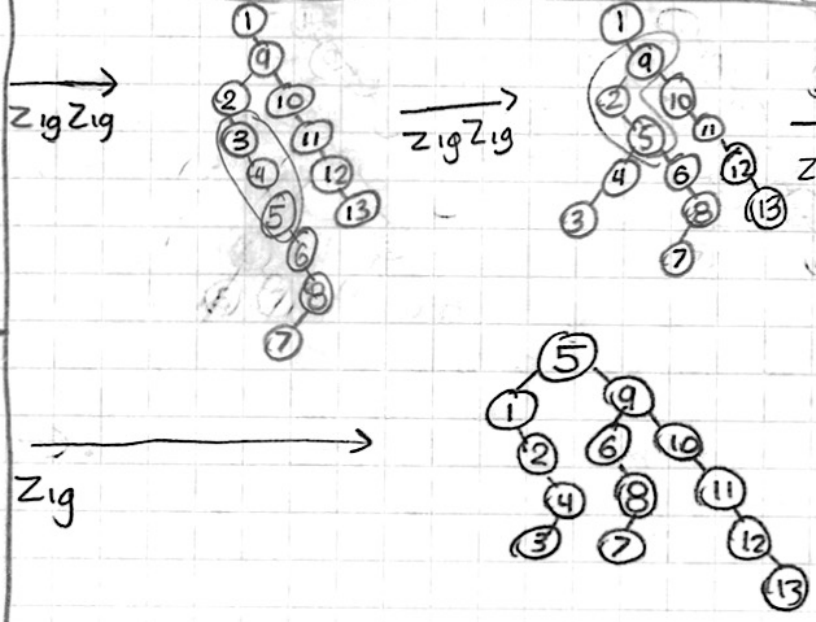


Access Key 1



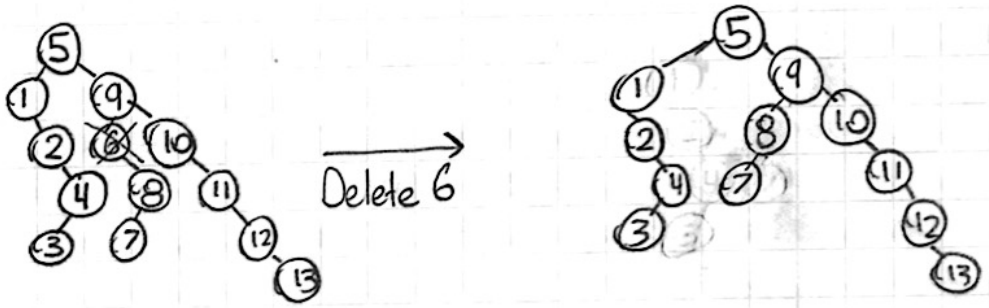


Access Key 5



Problem 2)

b)



Problem 3) (a) $T_{worst} = \Theta(N)$ $T_{AVG} = \Theta(N)$ $T_{Best} = \Theta(N)$

T_{worst} = regardless of the state of the array we will have to move all the current entries of the array to the right the average, best, and worst scenario are equal.

push 3

$S[] = \underset{0}{4}\ \underset{1}{5}\ \underset{2}{6}\ \underset{3}{4}$ $n=4$

$S[] = \underset{0}{3}\ \underset{1}{4}\ \underset{2}{5}\ \underset{3}{6}\ \underset{4}{4}$ $n=5$

↑ top ↑ Bottom

as you can see it is directly proportional to length.

(b)
0. insert 1 — — — 1 move
1. insert 2 2 → 1 — — 2 move
2. insert 3 3 → 2 → 1 — 3 move
3. insert 4 4 → 3 → 2 → 1 4 move

as we observe more of one operation we see it increases with size for that operation.

$\therefore \underset{0\text{th}}{1} + 2 + 3 + 4 + \underset{i\text{th}}{5} \ldots + N$ for each insertion there is 1 less movements.

$$\sum_{i=0}^{N} i = \frac{(n)(n+1)}{2} = \left[\frac{n^2+n}{2}\right] \quad \therefore \boxed{O(N^2)}$$

The total cost for N (insertions) operations is $O(N^2)$ therefore $N^2/N = N$

$\therefore$ Amortized running time $\Rightarrow$ $O(N)$

Problem 4)

(a) The worst case running time would be $\Theta(N)$ this will happen when the array is all 1's because the while loop will check all entries.

$\therefore \Theta(N)$

(b)

a[5]	a[4]	a[3]	a[2]	a[1]	a[0]	
0	0	0	0	0	0	0
0	0	0	0	0	1	1
0	0	0	0	1	0	2
0	0	0	0	1	1	3
0	0	0	1	0	0	4
0	0	0	1	0	1	5
0	0	0	1	1	0	6
0	0	0	1	1	1	7
0	0	1	0	0	0	8
0	0	1	0	0	1	9
0	0	1	0	1	0	10
0	0	1	0	1	1	11
0	0	1	1	0	0	12
0	0	1	1	0	1	13
0	0	1	1	1	0	14
0	0	1	1	1	1	15
0	1	0	0	0	0	16

* this example $k=5$

(c) every step a[0] is fliped, every second step a[1] is flipped every fourth. Step a[2] is flipped,

Therefore if we have N operation a[0] will flip N times, a[1] will flip N/2 time, a[2] will flip N/4 times.

↳ $N + N/2 + N/4 + N/8 \ldots N/2^i$

$i=0 \quad i=1 \quad i=2 \quad i=3$

let k be # of bits used. $\sum_{i=0}^{k-1} N/2^i = N\sum_{i=0}^{k-1} 1/2^i$

* we will over estimate ↳ $< \sum_{i=0}^{\infty} 1/2^i = 2$

* Pull out to front

$\therefore 2N$

Now we divide by cost of N operation $2N/N = 2$ → C

Amortized running time $\therefore \Theta(1)$