CS340 – Advanced Data Structures and Algorithm Design – Fall 2020
Assignment 2 – September 18, 2020

Dr. Sandra Zilles, Department of Computer Science, University of Regina

answer key

Problem 0 (1 mark). Everyone gets 1 mark, even if they don't send a joke/cartoon.

Problem 1 (6 marks). **algorithm** Let `q` be an initially empty queue. Call `levelOrder(r)`.

```
levelOrder(n)
  q.enqueue(n)
  while not q.empty do
    node := q.dequeue()
    output(node->element)
    if (node->left != NULL)
      q.enqueue(node->left)
    if (node->right != NULL)
      q.enqueue(node->right)
```

(4 marks – give partial marks for reasonable but incomplete approaches)

Since every node is enqueued exactly once and in every iteration of the while loop one node is dequeued, the while loop runs through $N = O(N)$ iterations. Every iteration consumes a constant amount of time ($O(1)$). Hence the running time of the algorithm is $O(N \cdot 1) = O(N)$. (2 marks – subtract one mark if the argument is incomplete but at least partly reasonable)

Problem 2 (6+2 marks). see hand-written sheets

Problem 3 (3+4 marks).

(a) The running time depends only on the length of the array. Therefore the worst case, average case and best case are all equal. [1 mark]
The time to insert is $\Theta(N)$. [1 mark]
This is because the N elements in the array all have to be moved by one slot. [1 mark]

(b) The cost of the sequence of insertions is $1 + 2 + 3 + \ldots + N$ units, because for the i^{th} insertion $i - 1$ entries first have to be moved. [1 mark]
This equals $\frac{N(N+1)}{2} = O(N^2)$. [1 mark]
This means that the amortized running time over N insertions is $O(N)$. [2 marks]
(Actually, here no real amortization happens, because worst case cost equals amortized cost.)

Problem 4 (2+2+4 marks). (a) The worst case running time of a single execution is $\Theta(N)$. [1 mark]
This happens when the array contains only `1`s. [1 mark]

(b)

`a[5]`	`a[4]`	`a[3]`	`a[2]`	`a[1]`	`a[0]`
0	0	0	0	0	0
0	0	0	0	0	**1**
0	0	0	0	**1**	**0**
0	0	0	0	1	**1**
0	0	0	**1**	**0**	**0**
0	0	0	1	0	**1**
0	0	0	1	**1**	**0**
0	0	0	1	1	**1**
0	0	**1**	**0**	**0**	**0**
0	0	1	0	0	**1**
0	0	1	0	**1**	**0**
0	0	1	0	1	**1**
0	0	1	**1**	**0**	**0**
0	0	1	1	0	**1**
0	0	1	1	**1**	**0**
0	0	1	1	1	**1**
0	**1**	**0**	**0**	**0**	**0**

Subtract 1 mark if it is not correctly marked which bits are flipped. Subtract 1 mark if there is more than only a tiny mistake in the table.

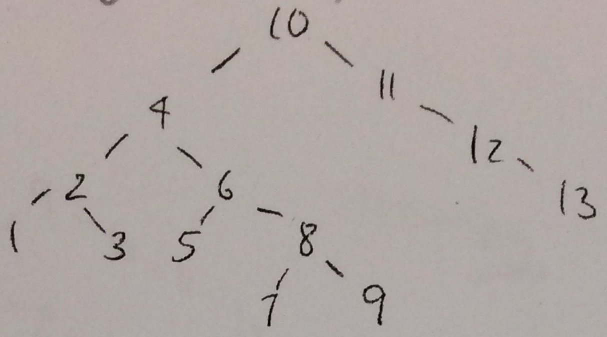
(c) It is obvious in (b) that in every step the bit in `a[0]` is flipped, in every second step the bit in `a[1]` is flipped, in every fourth step, the bit in `a[2]` is flipped, and so on. Therefore, in N operations, `a[0]` flips N times, `a[1]` flips $\lfloor N/2 \rfloor$ times, and so on, i.e., `a[i]` flips $\lfloor N/2^i \rfloor$. [1 mark]
The total number of flips is therefore

$$\sum_{i=0}^{k-1} \lfloor \frac{N}{2^i} \rfloor < N \sum_{i=0}^{k-1} \frac{1}{2^i} = 2N\,.$$

[1 mark]
The worst case time for a sequence of N operations starting from the empty array is therefore $O(N)$. [1 mark]
The amortized cost is therefore $O(1)$. [1 mark]

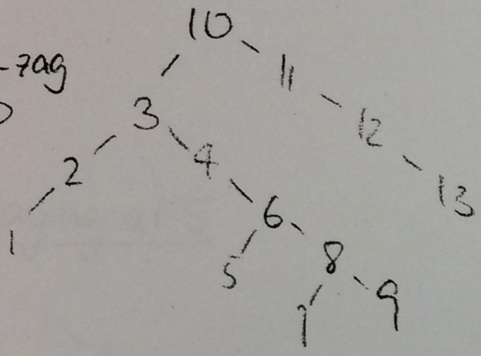
assignment 2, Problem 2

ial tree:

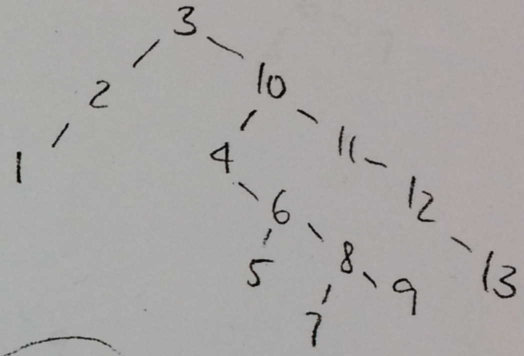


splaying at 3:

zig-zag $\rightarrow$



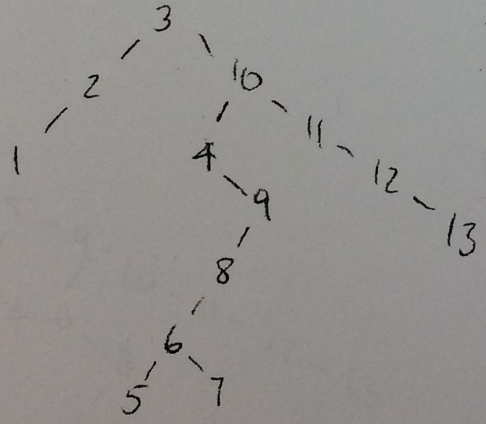
zig $\rightarrow$



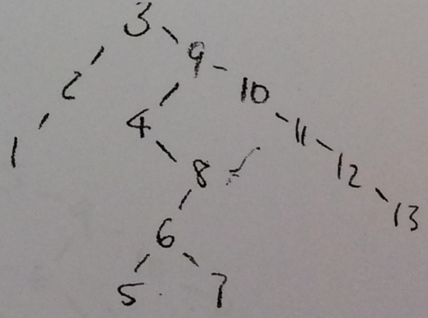
1 mark

splaying at 9:

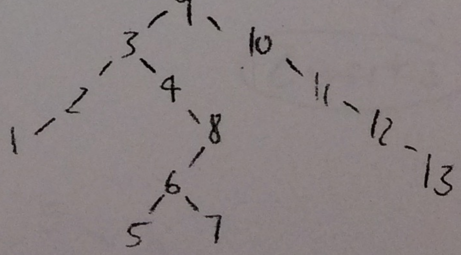
zig-zig $\rightarrow$



zig-zag $\rightarrow$

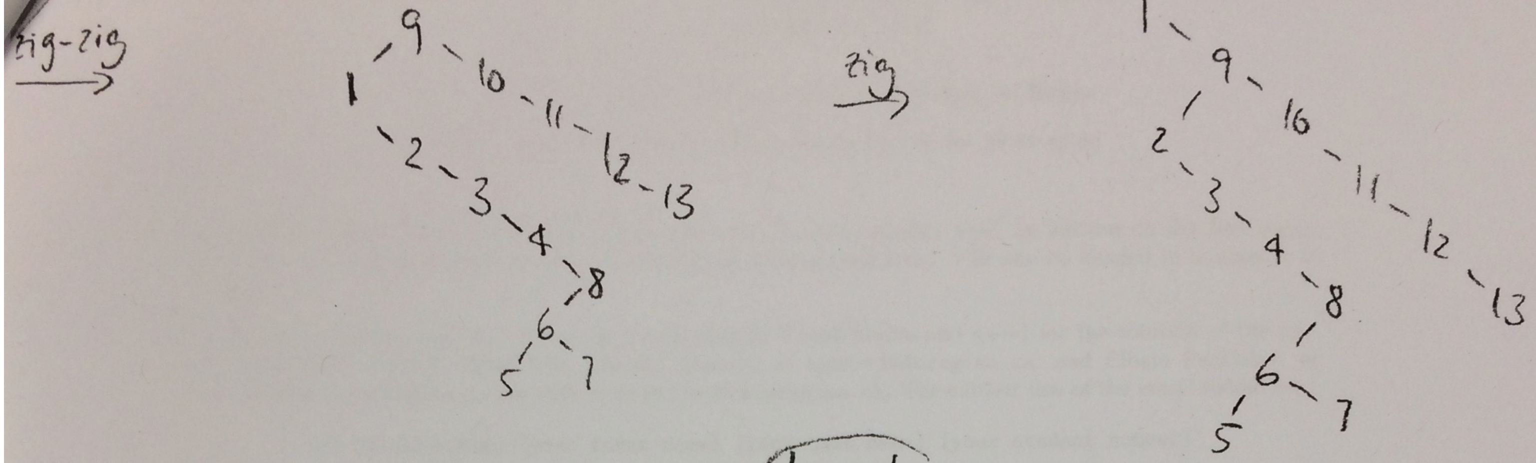


zig $\rightarrow$



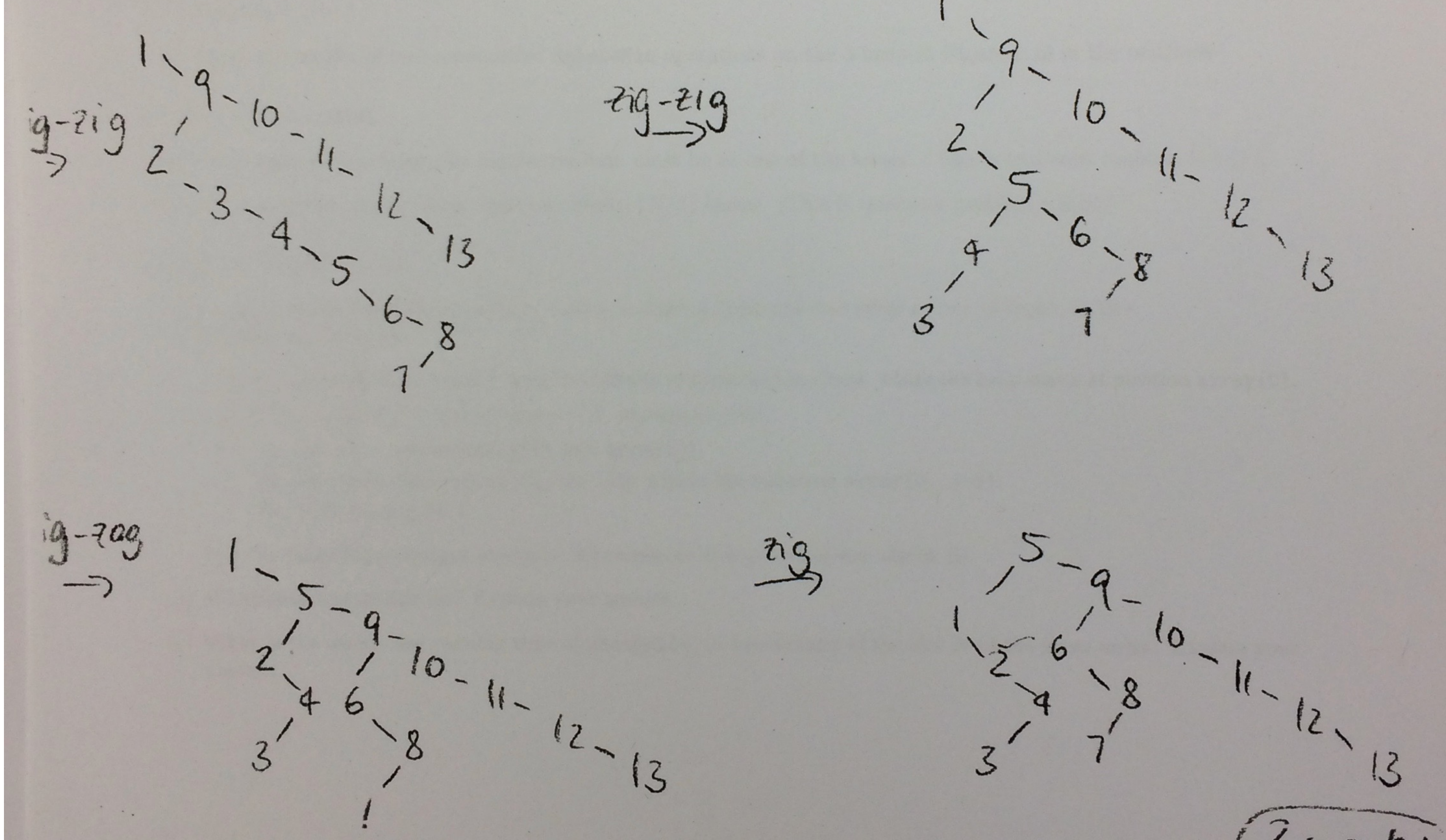
(2 marks)

g at 1



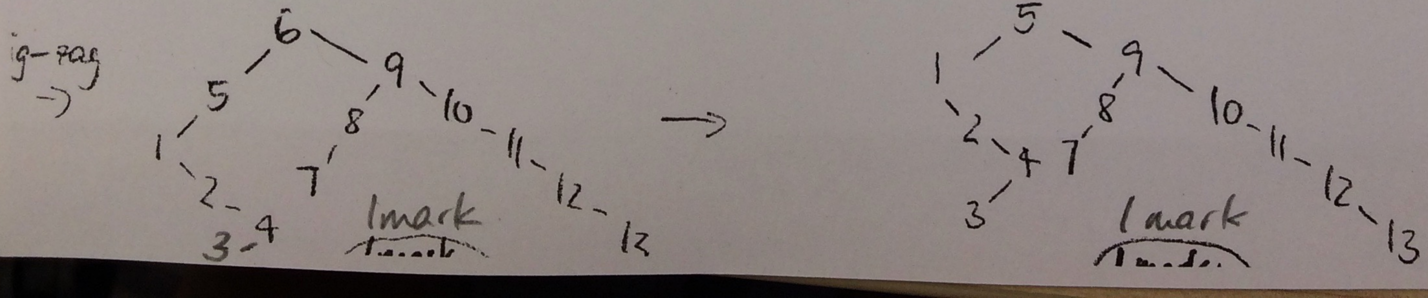
1 mark

splaying at 5



2 marks

)) delete 6 : first splay at 6, then remove



1 mark

1 mark