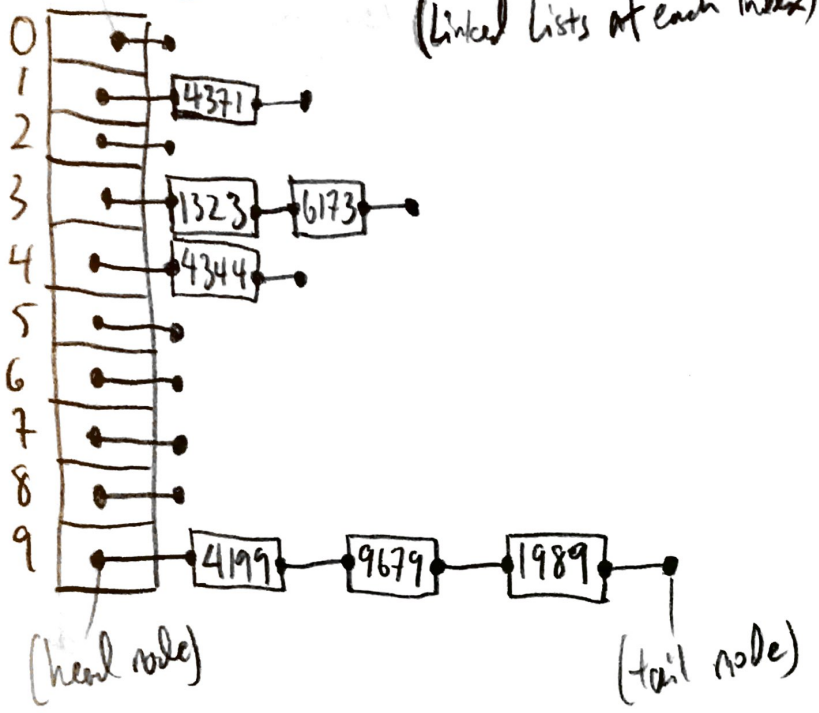
5.1 [11]

Conder Shon
CS3544

(A) $h(x) = x \bmod 10$
(Linked Lists at each index)



(B) Linear Probing

0	9679
1	4371
2	1989
3	1323
4	6173
5	4344
6	
7	
8	
9	4199

Collision resolution

$h_i(x) = h(x) + f(i)$

$f(i) = i$

(C) Quadratic Probing

For 1989

0	9679	i=1
1	4371	
2		
3	1323	i=2
4	6173	
5	4344	
6		
7		
8	1989	i=3
9	4199	

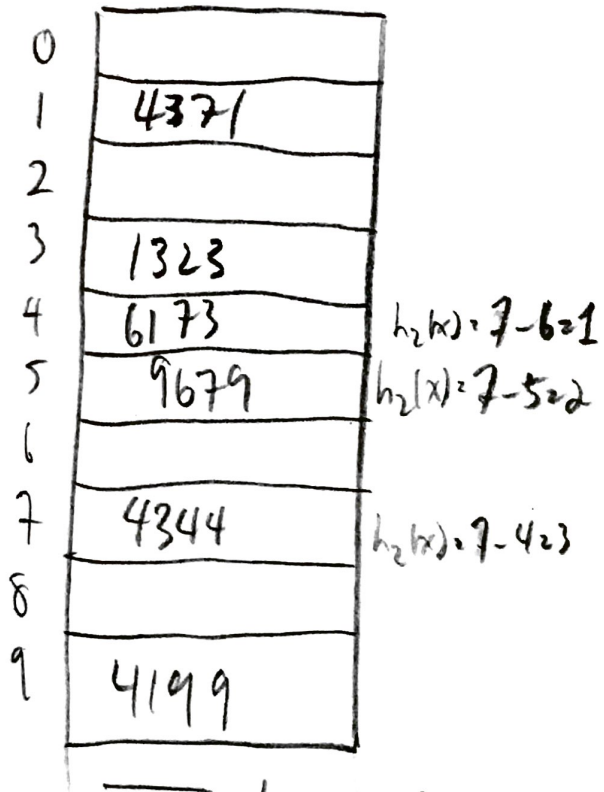
Collision Resolution

$h_i(x) = h(x) + f(i)$

$f(i) = f(i-1) + 2i - 1$

(d) $h_2(x) = 7 - (x \bmod 7)$

Rehashing



For 1989: $h_2(x) = 6$, never lands on empty space

1989 can't be inserted

5.2 [2]

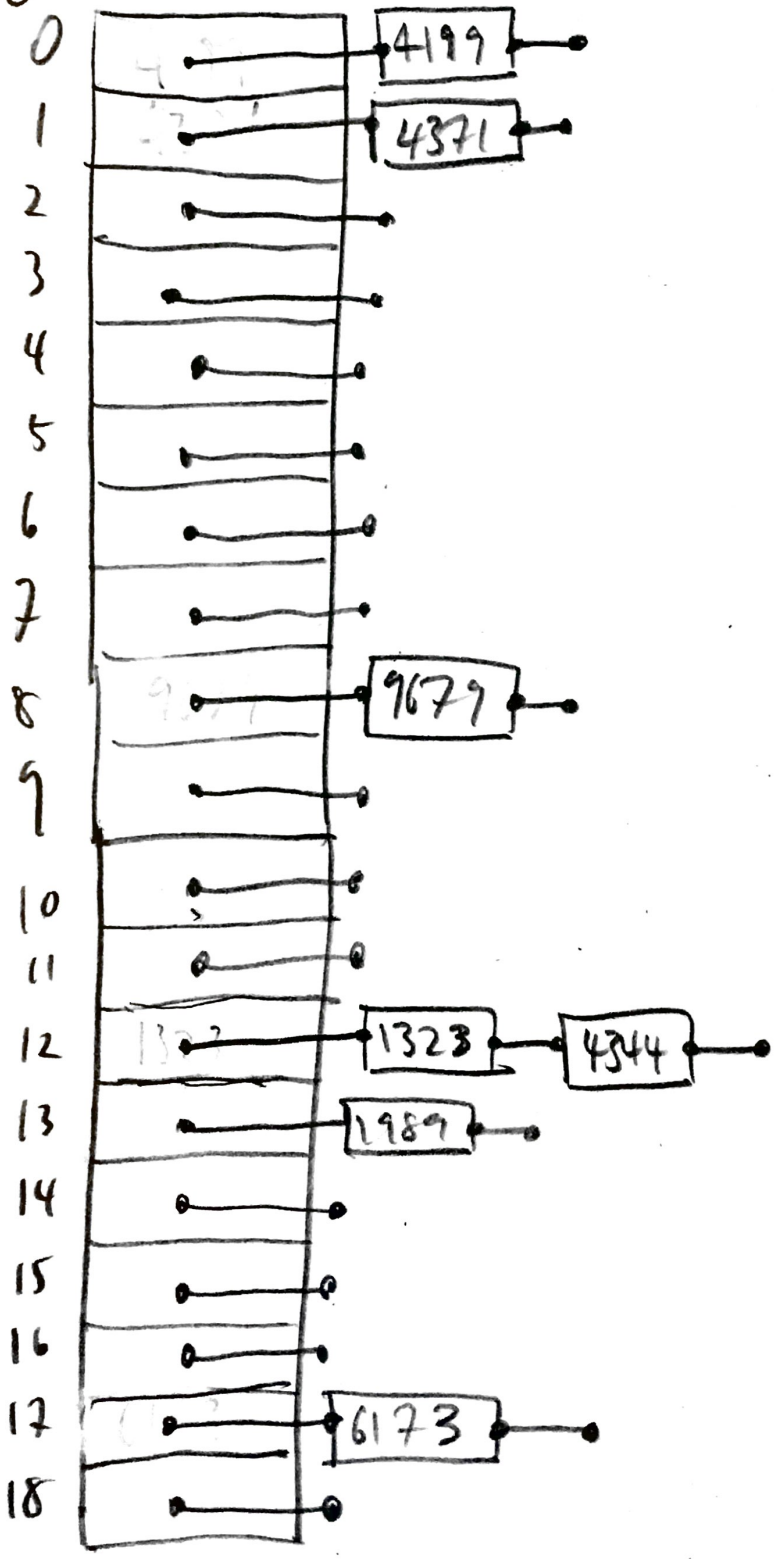
New Table Size: 19

$h_2(x)$ not changed

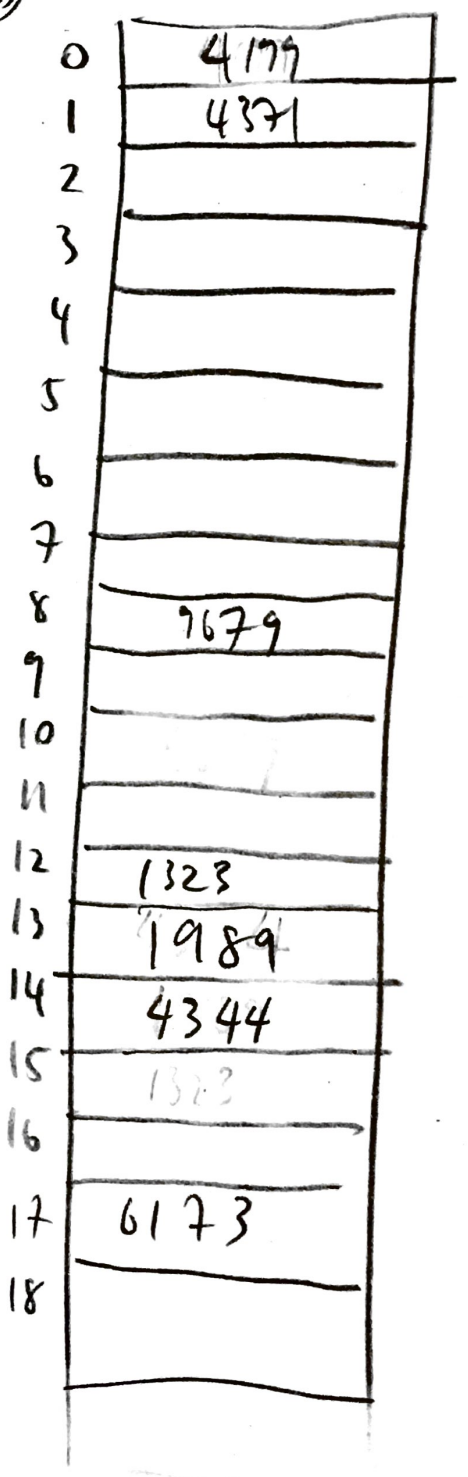
$h(x) = x \bmod 19$

Now the order of input is from scanning down the old tables from 5.1

(a)



(b) Linear Probing



5.2

(c) Quadratic Probing

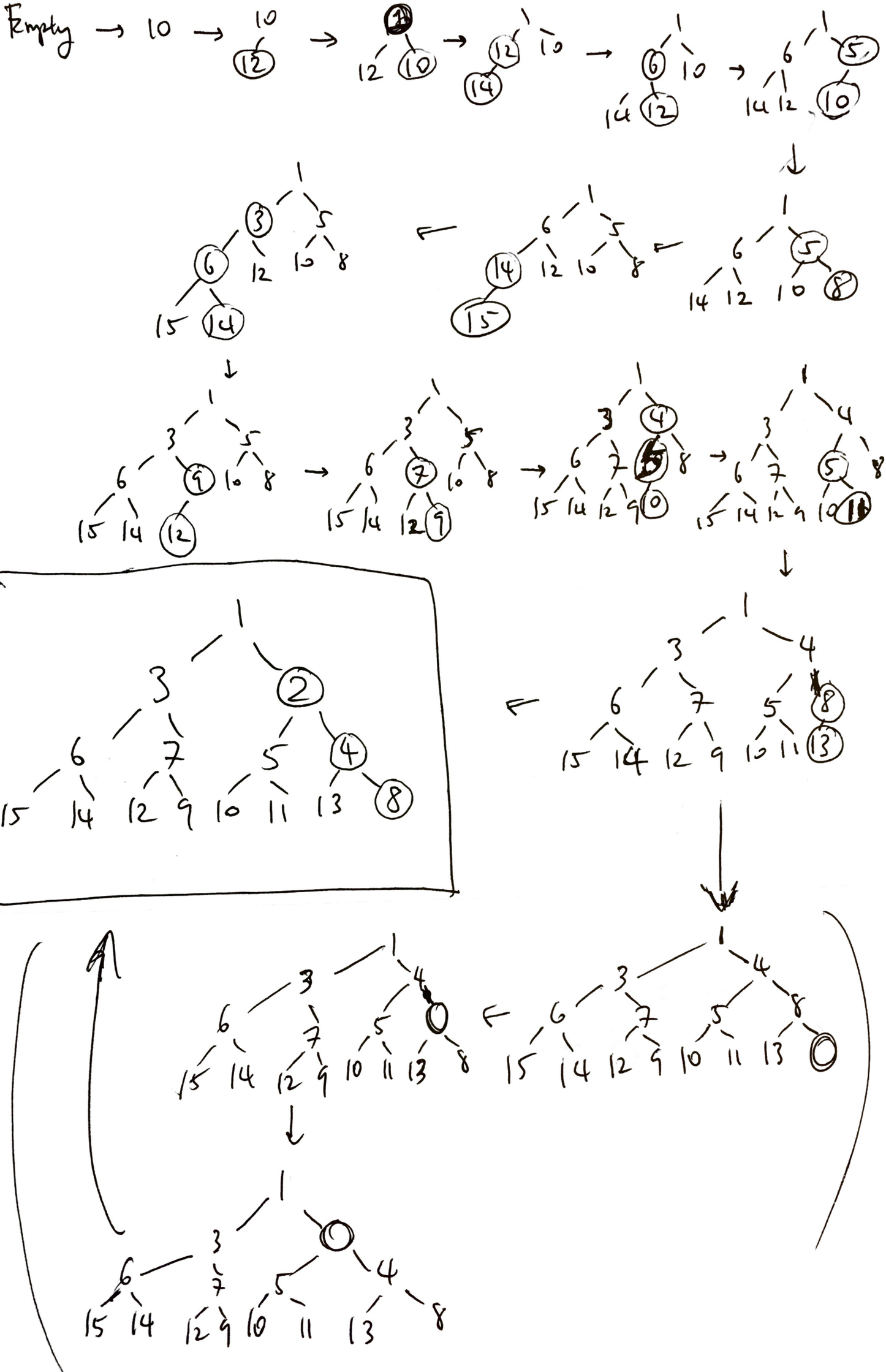
0	4199
1	4371
2	
3	
4	
5	
6	
7	
8	9679
9	
10	
11	
12	1323
13	1989
14	
15	
16	4344
17	6173
18	

(d) $h_2(x) = 7 - (x \bmod 7)$

0	4199
1	4371
2	
3	
4	
5	
6	
7	
8	9679
9	
10	
11	
12	1323
13	1989
14	
15	4344
16	
17	6173
18	

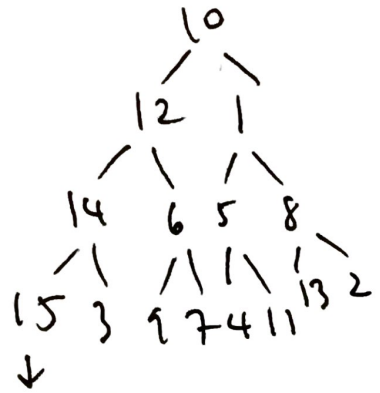
6.2 (a) [3]

Conder Show



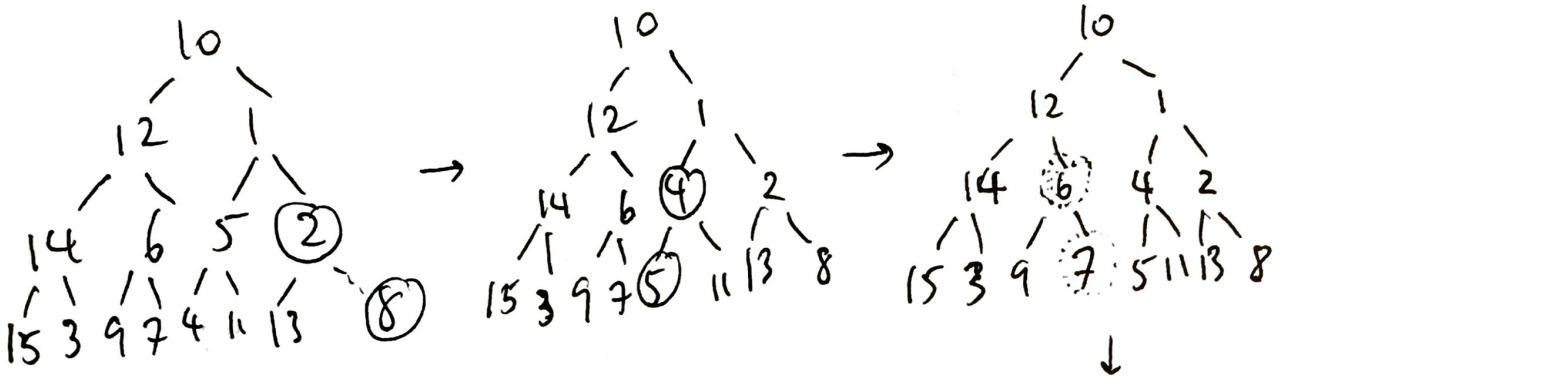
(16)

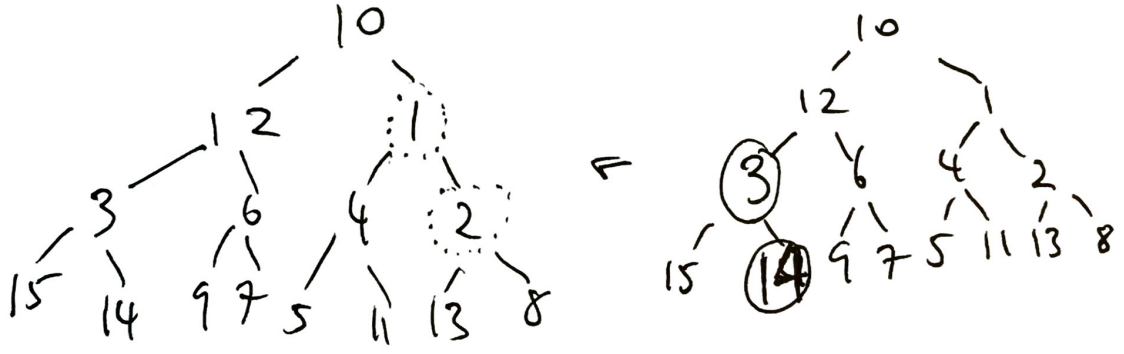
Initialize heap array with complete sequence

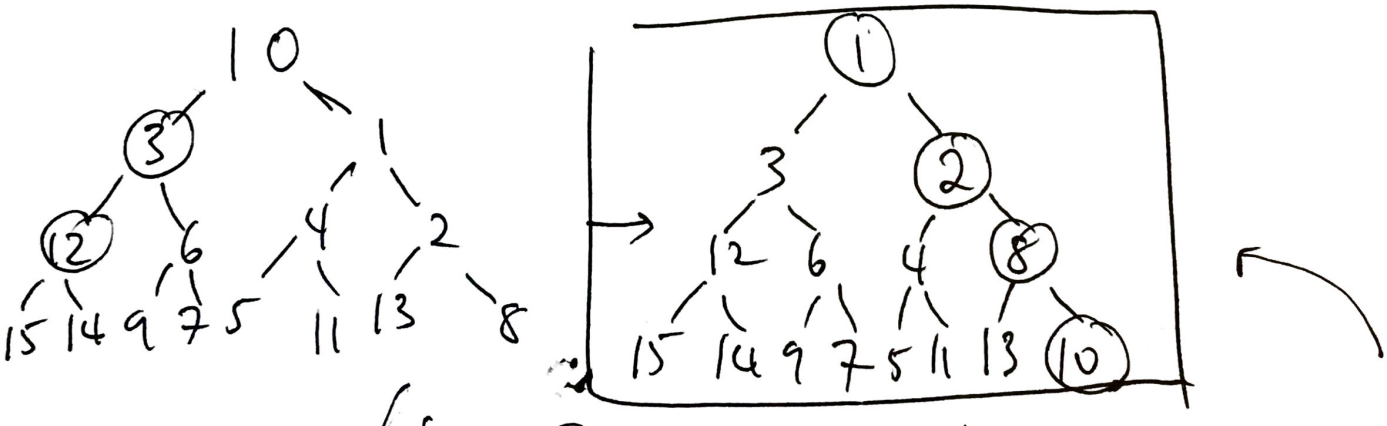


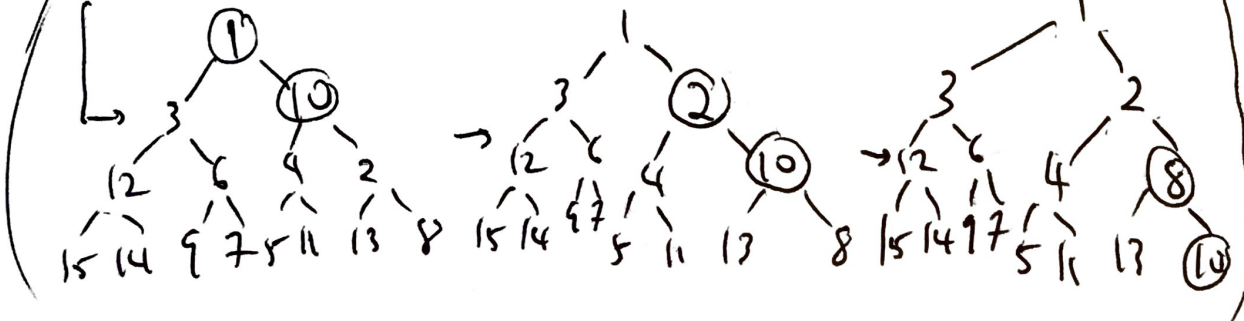
↓

PercolateDown(7) . . . .



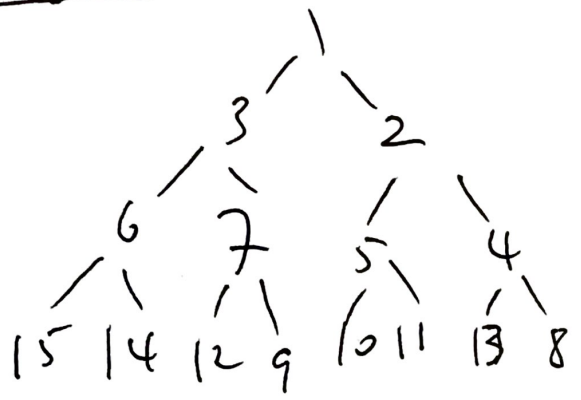




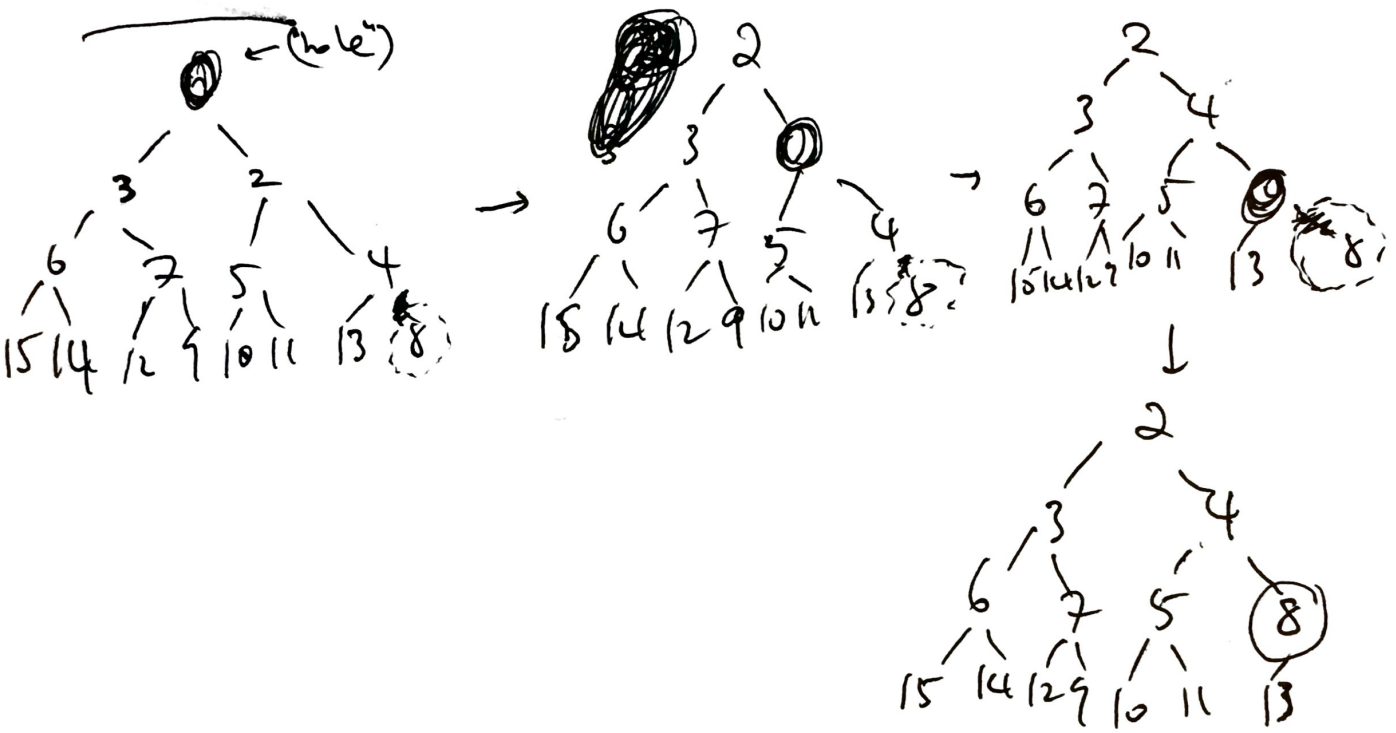


6.3 [4]

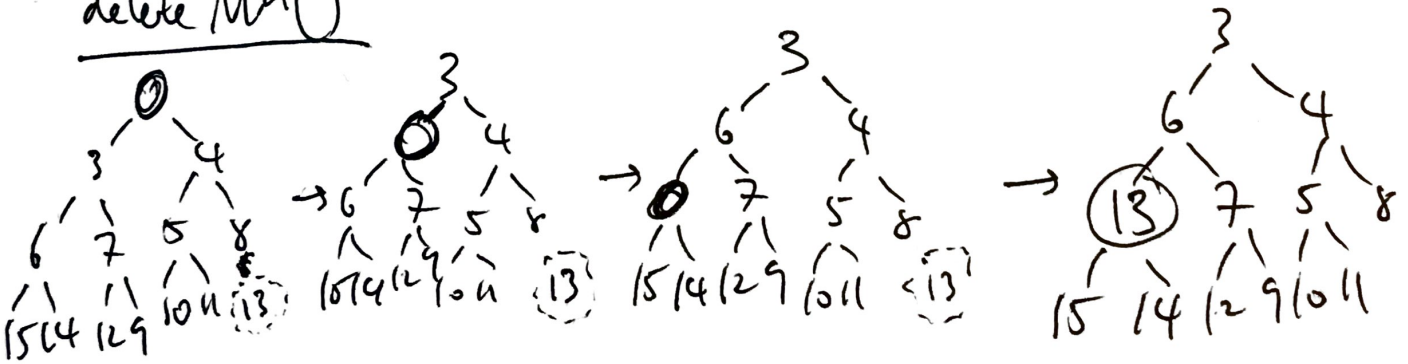
Initial Heap



1. deleteMin()

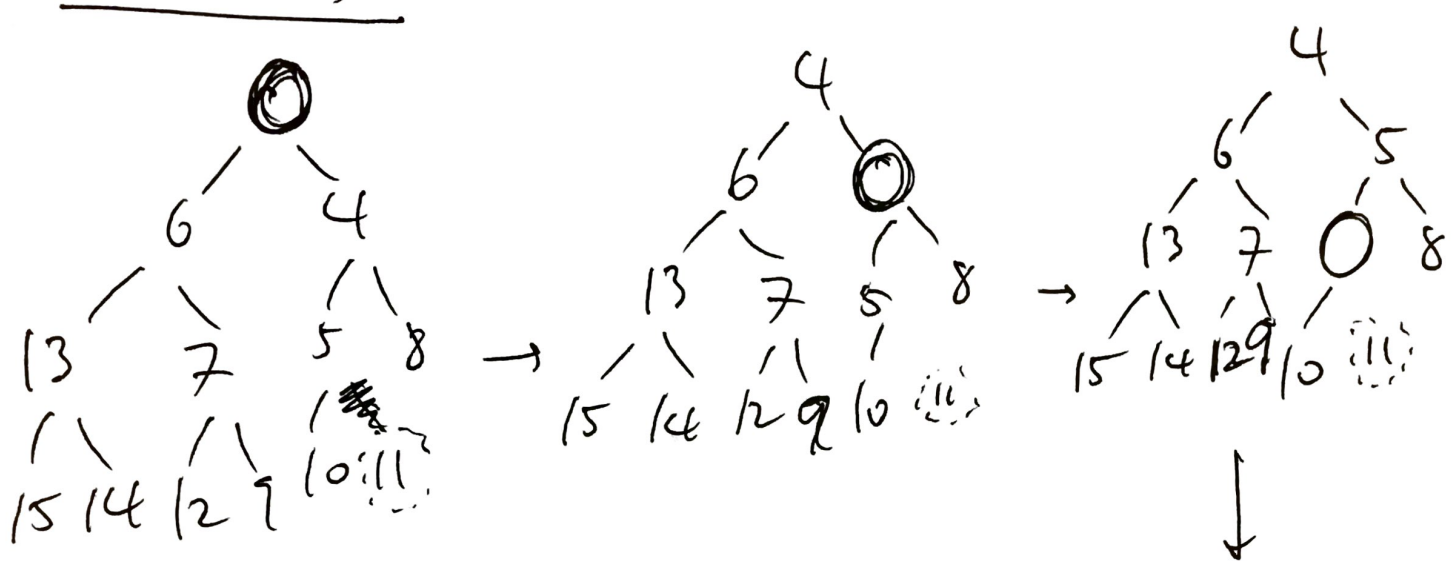


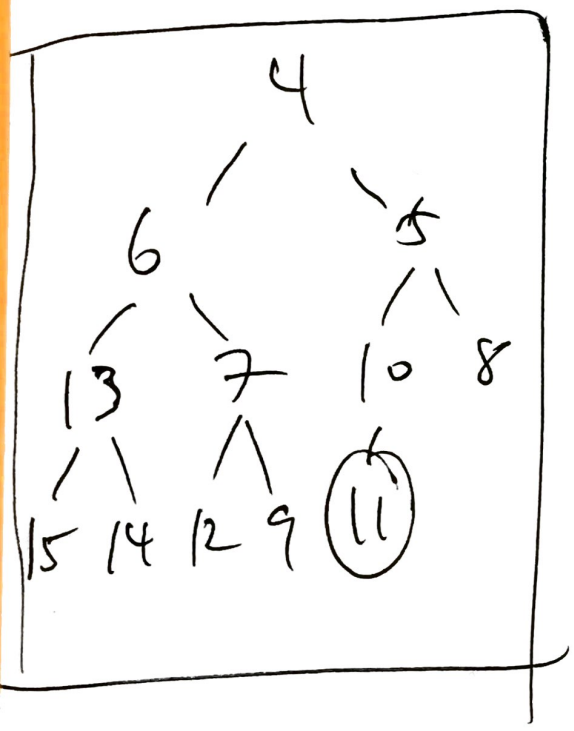
2. deleteMin()

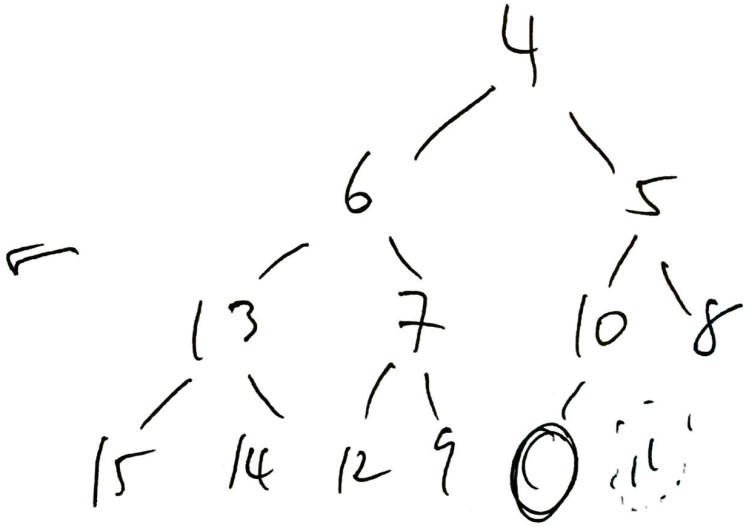


3.

deleteMin()


6 4
13 7 5 8
15 14 12 9 10 11
4
6 0
13 7 5 8
15 14 12 9 10 11
4
6 5
13 7 0 8
15 14 12 9 10 11


4
6 5
13 7 10 8
15 14 12 9 11


4
6 5
13 7 10 8
15 14 12 9 0 11

5.) **6.8 – Show the following regarding the maximum in the heap**

a. It must be at one of the leaves

The maximum item in the heap cannot be at any other location other than at one of the leaves because otherwise, if it had a child, then according to the heap-order property, it means that the child is supposed to be greater than the maximum item. This is not possible, because the heap-order property states that the parent node should always be smaller than the child node, and there is nothing greater than a maximum by its own definition.

Thus, a maximum node must be at one of the leaves. This has just been proven through a counterexample.

b. There are exactly [N/2] leaves

Let the last leaf of the heap be at index N.

Its parent is at index [N/2]

Thus, no nodes exist in which their parents are at or beyond index [N/2 + 1].

Therefore, any node at or beyond [N/2 + 1] has no child, and must be a leaf.

All leaves must then range from [N/2 + 1] to N.

Therefore, a heap has exactly [N/2] leaves.

c. Every leaf must be examined to find it

If we add a multitude of nodes to a heap, all the nodes of higher values will end up at the last level – at a leaf. They will end up there in the order in which they are inserted, but are not arranged in any particular order that tells us what the maximum element is. The maximum item could thus be placed at any one of the leaves, and thus each leaf must be examined to find the maximum value.