CE2101/ CZ2101: Algorithm Design and Analysis

Week 3: Review Lecture

Ke Yiping, Kelly

Content

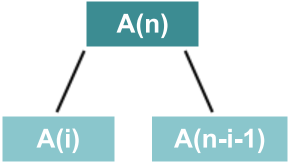
- Quicksort – Average Complexity
- Heapsort

Quicksort Complexity – Average Case

- Consider different final positions of pivot
- Each final position has an equal probability

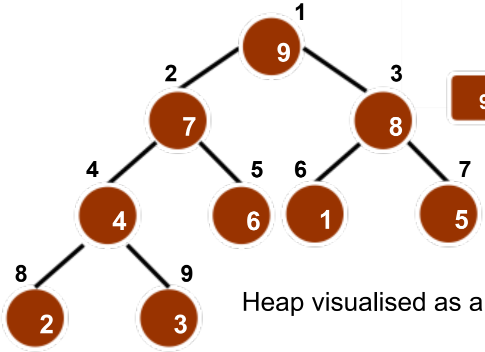
$$A(0) = A(1) = 0$$ No element or only one element.

$$A(n) = n - 1 + \frac{1}{n}\sum_{i=0}^{n-1}\left[A(i) + A(n-i-1)\right] = \Theta(n\lg n)$$



Heapsort – Heap Structure

- Content: partial order tree property
- Structure: binary tree that is complete till *h-1*



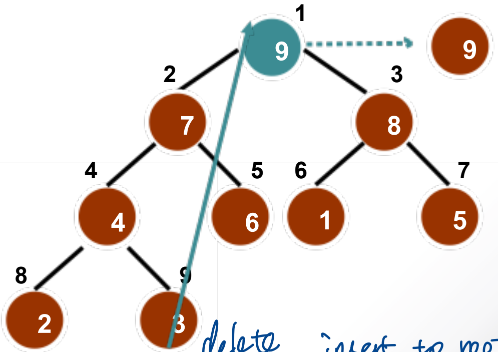
Heap visualised as a tree

Heap viewed as an array

Heapsort Method

```
heapSort (array, n)
{   construct heap H from array with n elements;
   for (i = n; i >= 1; i--)
   {   curMax = getMax(H);
       deleteMax(H);
       // as result, H has i – 1 elements
       array[i] = curMax;
       // insert curMax in sorted list
   }
}
```

Take out last and re-insert

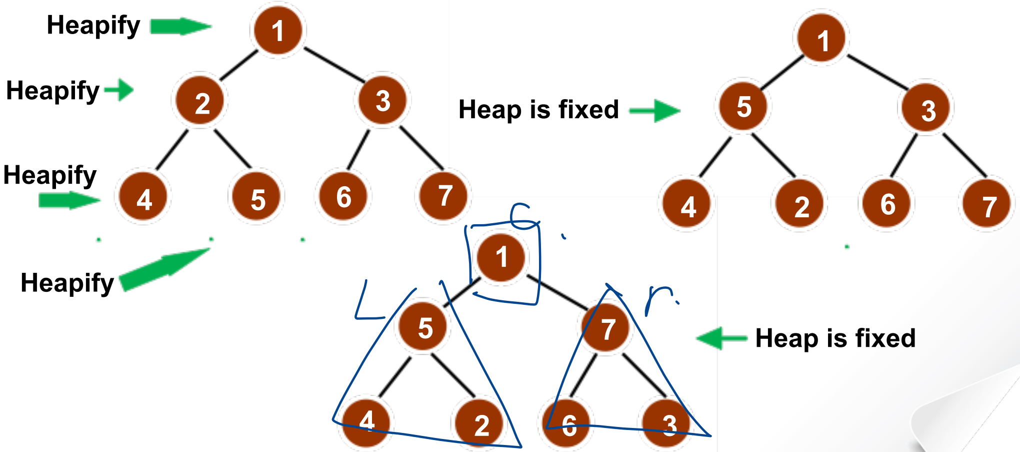


delete, insert to root, fix the heap

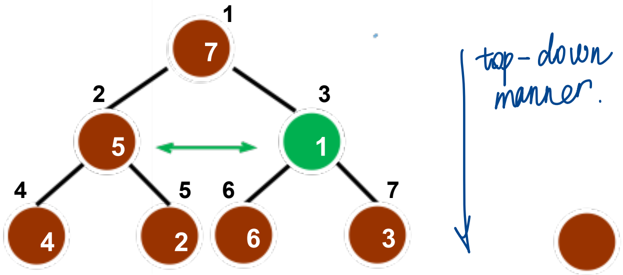
Heap Construction

Assume elements in initial arbitrary order: **1 2 3 4 5 6 7**

post-order traversal: l→r→c



fixHeap(H,"1")

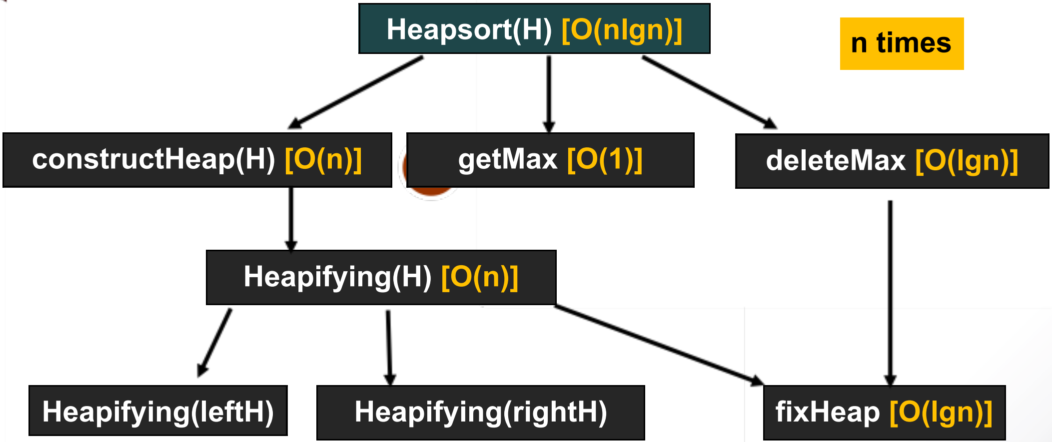


① ask the two child to fight first.

7 > 5 and 7 is also > 1; so 7 is inserted into Root, and the original slot of 7 becomes 1.

fixHeap is called again to reinsert 1 into the sub-heap.

Heapsort Performance



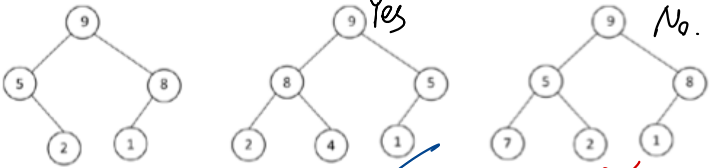
$$W(n) = 2W\left(\frac{n-1}{2}\right) + \underline{2\lg n}$$

worst case.

Exercise

- For each of the trees in the following figure, is it a maximizing heap? Briefly justify your answers. **[AY1617S2]**

< content
structure



Yes

No.

(A) (B) (C)

doesn't satisfy structure

最下一层要从左到右填满.

doesn't satisfy content

child node should always be smaller than parents

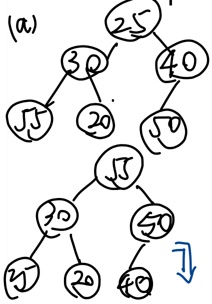
Exercise

- Suppose an array A = [25, 30, 40, 55, 20, 50] is given as input to Heapsort.

a) Show the contents of array A after the heap construction phase. How many key comparisons and swaps are done respectively to construct a maximizing heap from A? 7, 4

b) Show the contents of heap in the array after two calls of deleteMax() on the heap constructed in a). **[AY1819S1]**

(a)



[55, 30, 50, 25, 20, 40]

① heapifying (2,'30').

1. 55 vs 20.
2. 55 vs 30.

1. swap 55, 30

② heapifying (3,'40')

3. 40 vs 50.

2. swap 40. 50.

③ heapifying (1,'25')

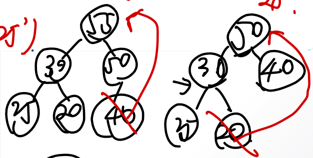
4. 55 vs 50
5. 55 vs 25

3. swap 25. 55.

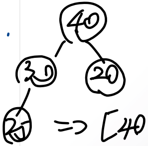
6. 30 vs 20.
7. 30 vs 25

4. swap 30. 25.

(b). after 1 call:



20.



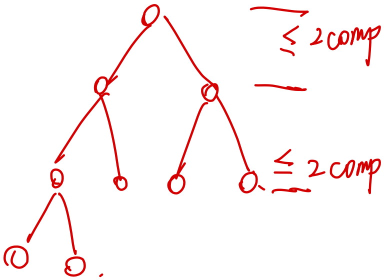
=> [40, 30, 20, 25, 50, 55]

Exercise

- What is the time complexity of fixHeap in Heapsort for an input array of n elements? Briefly justify your answer. **[AY1920S1]**

$O(\lg n)$

fixHeap doesn't change structure.



Key points of the question:

1. At every level, at most 2 comparisons are needed.
2. # of comparisons is up bounded by 2 × levels [height of the tree].
3. $\therefore O(2\lg n) = O(\lg n)$

top k

- Given an unsorted array of n integers, design an algorithm to find the k largest elements with the worst-case time complexity $O(n + k \lg n)$, where $1 \le k \le n$. You can use any algorithm learnt in the lectures as a subroutine of your algorithm (i.e. you need not write the pseudocode of the subroutine). Briefly explain the worst-case time complexity of your algorithm. **[AY1718S1]**

Use maximizing heap.

①. construct heap.

② do{ getMax(H)
deleteMax(H)
} for k times.