Hw4 - homework 4 solution

Data Structure and Algorithm (New York University)

1. A = <14, 12, 19, 5, 6, 9, 3, 4, 13, 7, 22, 16>

A(p) = pivot = 14

① i p 14 12 19 5 6 9 3 4 13 7 22 r 16 j

② p,i ⑦ 12 19 5 6 9 3 4 13 j ⑭ 22 r 16

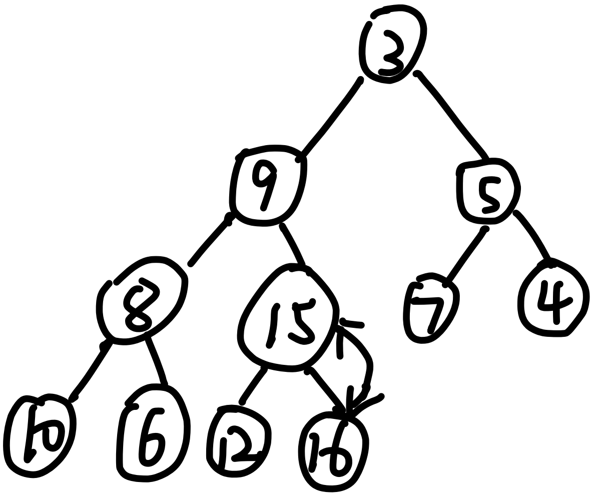
③ p 7 12 i ⑬ 5 6 9 3 4 j ⑲ 14 22 16

④ p 7 12 13 5 6 9 3 j 4 | i 19 14 22 16

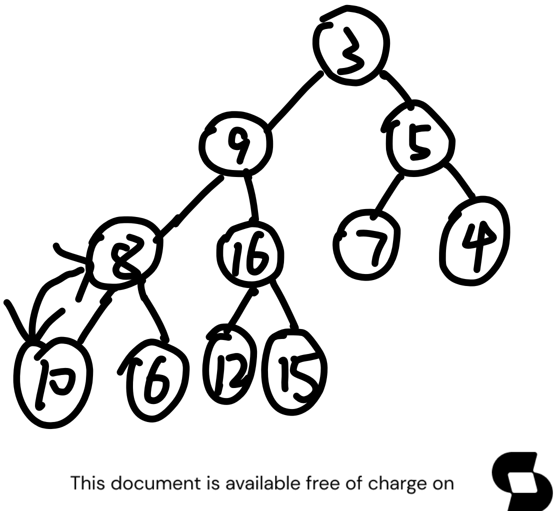
j > i → end

2. (a)

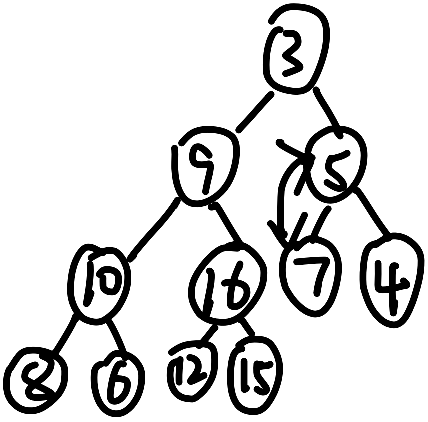
①



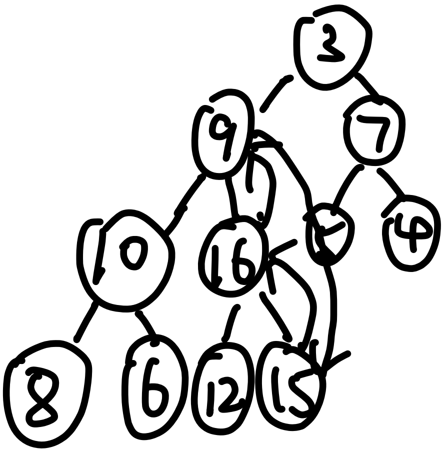
②



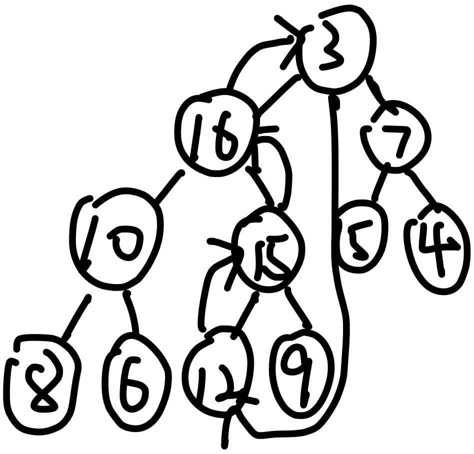
3


3
9
5
10
16
7
4
8
6
12
15

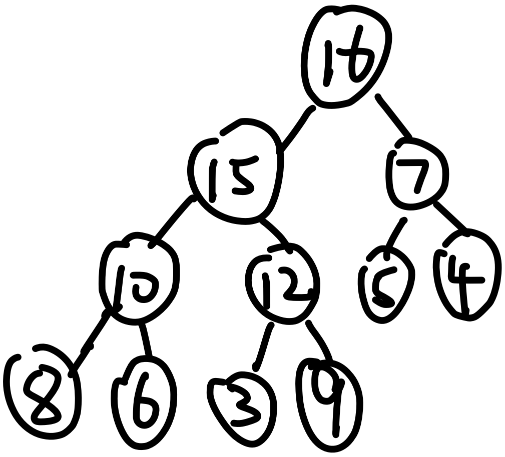
4


3
9
7
10
16
5
4
8
6
12
15

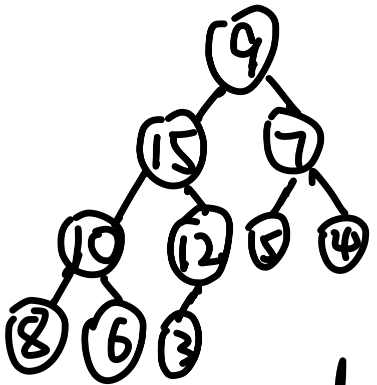
5


3
16
7
10
15
5
4
8
6
12
9

6

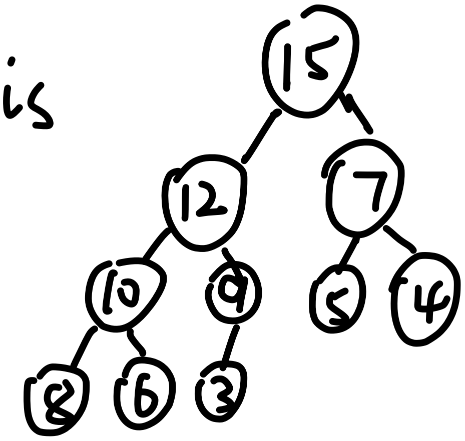

16
15
7
10
12
5
4
8
6
3
9

(b) ① remove A[1] , A[last] → A[1]



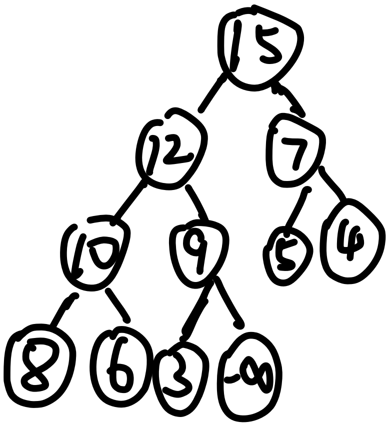
② for A(n-1) do MAX-HAEPIFY

then heap is

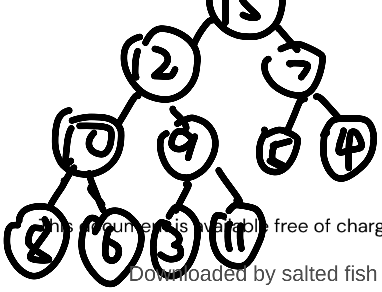


the array is <15, 12, 7, 10, 9, 5, 4, 8, 6, 3>

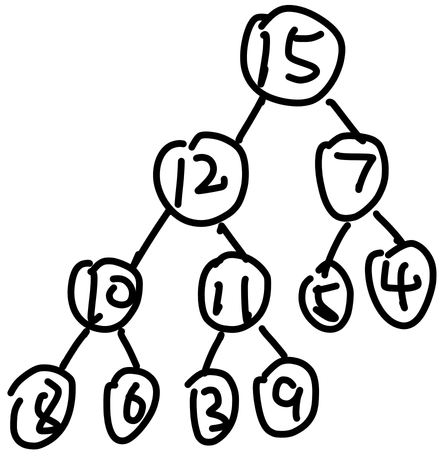
(c) ① insert -∞ in array



② increase key to 11



③ do function MAX-HAEPIFY



So, the array is <15, 12, 7, 10, 11, 5, 4, 8, 6, 3, 9>

3. Designed algorithm desmonstration:

① Create two heaps, the first is called big-heap, the other is called small-heap. For small-heap, we use BUILD-MAX-HEAP to manage it. For big-heap, we use BUILD-MIN-HEAP to manage it.

② We have an unsorted array A[n], traverse A[n]. First, using variable mid to store A[1]. ⟺ mid = A[1]
Then, traverse A[n] and if A[i] > mid, put A[i] into big-heap, else put A[i] into small-heap

③ For big-heap, we need MIN-HEAPIFY to make the minimum data be stored in the root. For small-heap, we need MAX-HEAPIFY to make the maximum data in the root node.

If |Elements(big-heap) − Elements(small-heap)| = 2;

a) Elements(big-heap) > Elements(small-heap)
(number) (number)

use MAX-HEAP-INSERT to insert mid to small-heap, and use HEAP-EXTRACT-MIN to extract the root element in big-heap and store it in mid.
⇒ mid = big-heap[1]

b) Elements(big-heap) < Elements(small-heap)
(number) (number

use MIN-HEAP-INSERT to insert mid to big-heap, and use HEAP-EXTRACT-MAX to extract root element in small-heap and make mid = small-heap[1].

④ repeat step ③ until all datas are stored in mid, big-heap and small-heap.

If n in A[n] is odd, median of the A[n] is mid.

If n is even, median of this array is (mid+ root node)/2

Element in root node is taken from the heap with more elements of big-heap and small-heap.

Thus, the array is only traversed one and the median of it can be found with two heaps.

hw 4

4

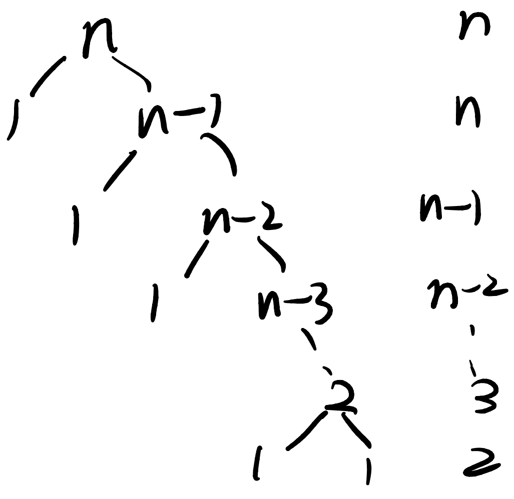
(a)

Worst case

Recurrence q=1

$T(n) = T(1) + T(n-1) + n$

$T(1) = \Theta(1)$



$T(n) = n + (\sum_{k=1}^{n} k) - 1 = \Theta(n) + \Theta(n^2) = \Theta(n^2)$

(b)

$\Theta(n\lg n)$

(c)

worst case: every time the median of the array is in $T(\frac{7n}{10})$ part.

$\therefore T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + O(n)$

assume $T(n) \leq cn$

$\therefore T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + O(n) \leq \frac{cn}{5} + \frac{7n}{10} + an$

$= \frac{9cn}{10} + an$

$= cn + (-\frac{cn}{10} + an)$

$$\therefore\ cn + c - \frac{cn}{10} + an) \leq cn$$

$$-\frac{cn}{10} + an \leq 0$$

$$c \geqslant 10a$$

$\therefore$ the worst case of quicksort (using BFPTR as the partition) is $O(n)$