DATE ____/____/____

1. (a) T
(b) T
(c) F
(d) F
(e) F
(f) F
(g) F
(h) T
(i) F
(j) F

2. b, d

3. b, d

4. a, b, c, d

5. a, c, d

6. b, d.

MON TUE WED THU FRI SAT SUN

DATE ____/____/____

7. insertion sort is to suppose the first is always sorted and to insert the rest

we have [71], 25, 40, 7, 60, 13, 20, 80

① 25 [71] [60] 7 60 13 20 80

next

② 25 40 71 7 60 13 20 80

③ 7, 25, 40, 71, 60 13 20 80

④ 7, 25, 40, 60 [71] [13] 20 80

⑤ 7 13, 25, 40, 60, [71] 20 80

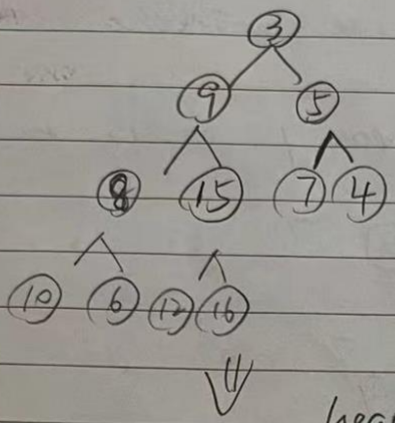
⑥ 7 13 20, 25, 40 60 71, 80

[sorted]

DATE ____/____/____ 3 9 5 8 15 7 4 10 6 12 16

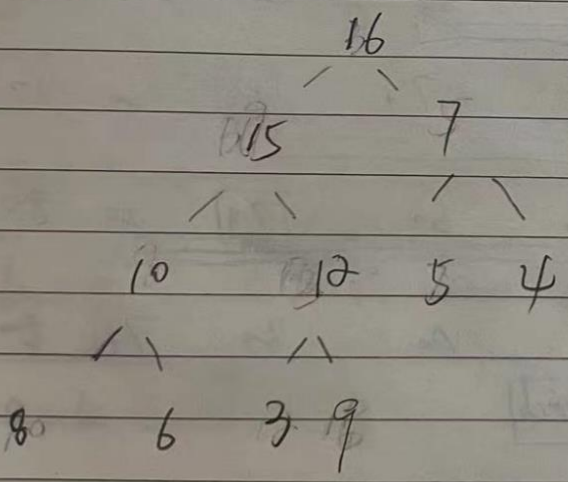
MON TUE WED THU FRI SAT SUN

(a) the original Tree is



heapify to max-heap

⇒



⇒ this is the initial max-heap built from original Array.

MON TUE WED THU FRI SAT SUN

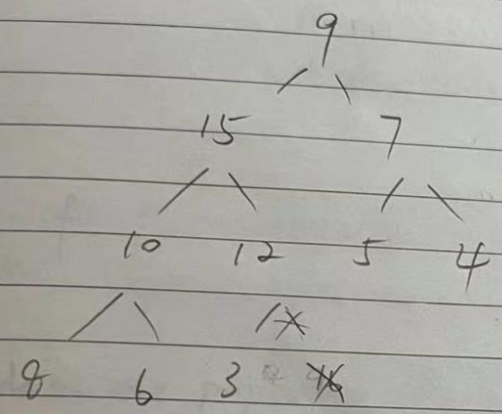
DATE ____/____/____

8 (b). extracts the max one

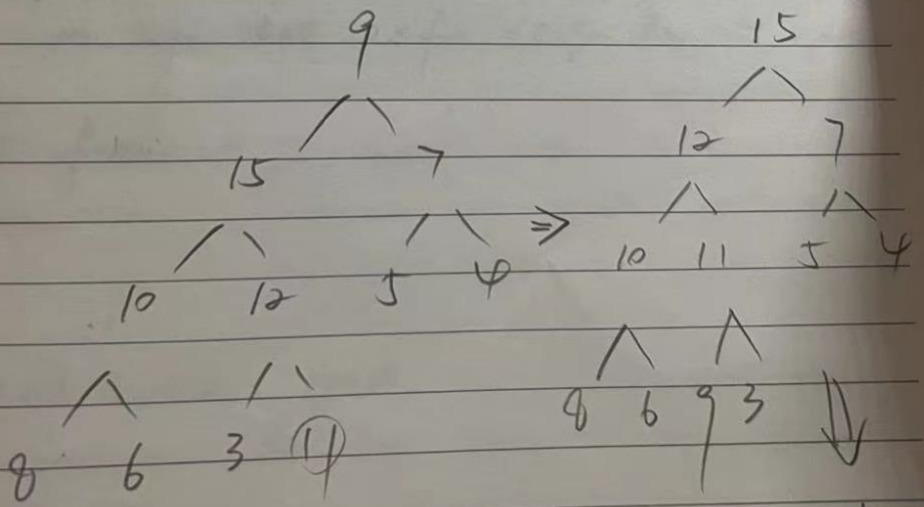
① swap the last one with the top one

② then remove the last one.

①



② insert 11, put it at the last node. the repeat the heapify.

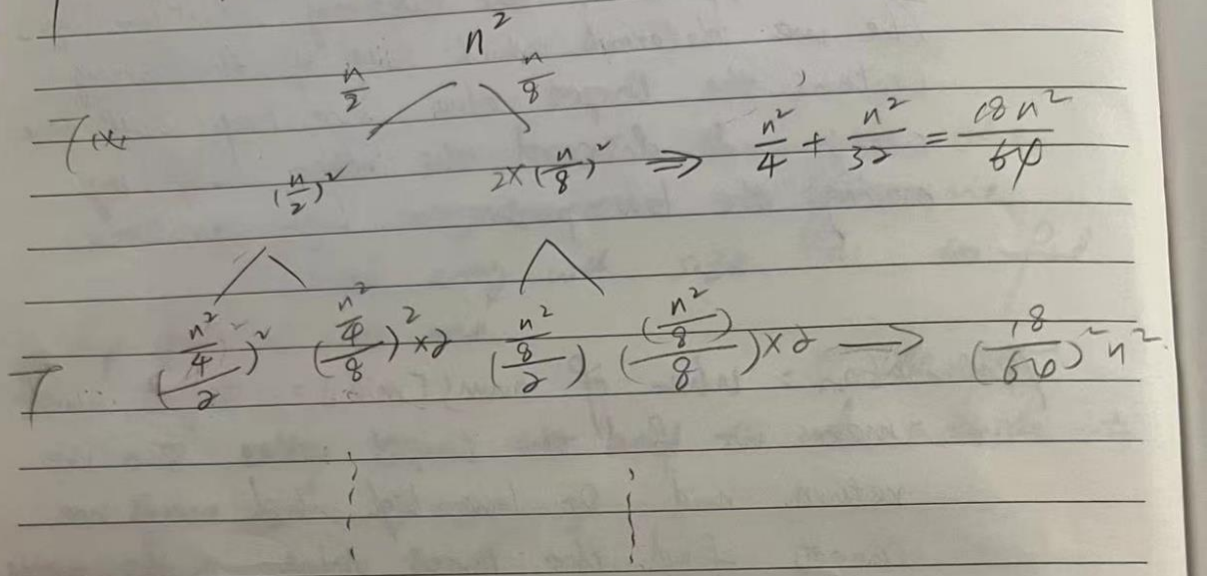


max-heap.

MON TUE WED THU FRI SAT SUN

DATE ____/____/____

9 (a) $T(n) = T(\frac{n}{2}) + 2T(\frac{n}{8}) + n^2$.



$\therefore$ the $T(n) = n^2(1 + \frac{18}{64} + \frac{18}{(64)^2} + \frac{18}{(64)^3} + \dots + \frac{18}{(64)}^{\log_2 n})$

$= n^2 \cdot \frac{1}{1 - \frac{18}{64}}$

$\approx T(n) = O(n^2)$

BOSHI PAPER

MON TUE WED THU FRI SAT SUN

DATE ____/____/____

q. (b) ① we set $T(k) \le dk^2$.

$$T(n) = T\left(\frac{n}{2}\right) + 2T\left(\frac{n}{8}\right) + n^2$$

$$\le d\left(\frac{n}{4}\right)^2 + 2d\frac{n^2}{64} + n^2$$

$$\le dn^2$$

$$d \times \frac{18n^2}{64} + n^2 \le dn^2$$

$$d\left(1 - \frac{18}{64}\right) \le d$$

$$\frac{1}{1 - \frac{18}{64}} \le d$$

② $T(k) \ge dk^2$ for all $k < n$.

$$T(n) \ge \frac{n^2}{4}d + 2d\cdot\frac{n^2}{64} + n^2 \ge dn^2$$

$$d \le \frac{1}{1 - \frac{18}{64}}$$

proved.

q (c) $T(n) = 4T\left(\frac{n}{2}\right) + n^2 + n^{1.5}$

master theorem.

we set ① $a = 4$, $b = 2$, $d = 2$

② $a = 4$, $b = 2$, $d = 1.5$

① $\alpha \frac{a}{b^d} = 1$ $n^2 \log n$

② $\frac{a}{b^d} < 1$ $= n^{1.5}$ (ignore)

∴ Time Complexity: $n^2 \log n$.

BOSHI PAPER

MON TUE WED THU FRI SAT SUN

DATE ____/____/____

10. (a) disaprove.

we have $n+\sqrt{n} = w(n)$

$\Rightarrow$

$\because n+\sqrt{n} < n$

$\sqrt{n} < 0$

but $\sqrt{n}$ can't smaller the zero

(b) if $f(n) = w(g(n))$

$g(n) = o(f(n))$

based on the property, if $f(n) = w(g(n))$, there exists positive constants c, n_0 such that $0 \le c \cdot g(n) < f(n)$ for all $n \ge n_0$

if $f(n) = o(g(n))$ there exists positive constant c, n_0 such that $0 \le f(n) \le c \cdot g(n)$ for all $n \ge n_0$

since $f(n) = w(g(n))$

for all $c > 0$, exist $n_0 \ge 0$ such that

$f(n) < c \cdot (g(n))$

$\Rightarrow$ if $n \ge n_0 \Rightarrow \frac{1}{c} f(n) \le g(n)$ $\because \frac{1}{c}$ is also constant

$\Rightarrow g(n) = o(f(n))$

BOSHI PAPER

DATE ____/____/____

MON TUE WED THU FRI SAT SUN

11. (a) the worst case of Binary search is when the element is the first of the last one.

Since in the worst case, it require the case require binary search $O(\log n)$ time on a whole sorted array with size n, to find the first or last one.

Since every time we need to ~~d~~ s divide into $\frac{n}{2} \rightarrow O(\log_2 n)$

(b) Loop-invariants: at the start of each iteration of while loop, the target value must be in the subarray nums[low, high].
also, the array should be sorted.
and low is always smaller equal to high.

~~maintance~~

initialization: before the first iteration, low = 0, high = n-1. the target value must be in the subarray nums[1, n-1]

DATE ____/____/____

maintance: while in the iteration, we first check the mid point of the subarray, nums[low, high]. The we determin which half of the array contains the target value. We keep half the array and discard the other half by moving the low pointer or high pointer.

Termination: when if nums[mid] == target, which means we find the target value. The we return mid. Or low > high, which means we cannot find the target value in the nums array, return -1

DATE ____/____/____

MON TUE WED THU FRI SAT SUN

12.

(a) after the first call rand

since left = 0, right = n-1

the left partition is $\frac{n-1}{2}$.

the probability is $p = \frac{n-1}{2}$

(b) after the first one

since it is right.

$$\Rightarrow p = \frac{i-1}{n} + \frac{n-i-k}{n} = \frac{n-k-1}{n}$$

after the second call

$$p = \frac{n-k-1}{n} \times \frac{n-k-1}{2n}$$

Question13:
Do know how to do.