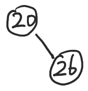
Lab 05

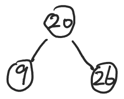
2024年10月10日 13:16

insert 20 :

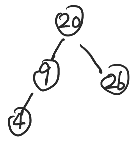
insert 26 :



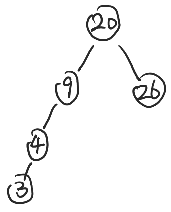
insert 9 :



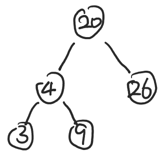
insert 4 :



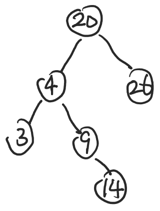
insert 3 :



right rotate 9 :

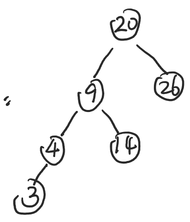


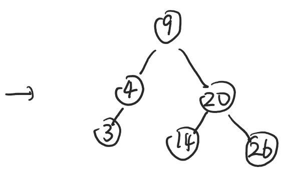
insert 14 :



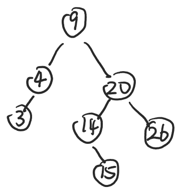
left rotate 4

then right rotate 20 :

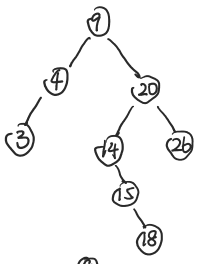




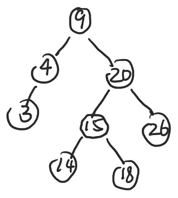
insert 15 :



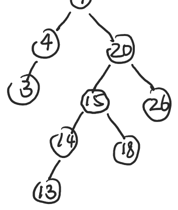
insert 18 :



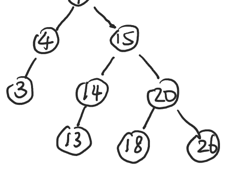
left rotate 14 :



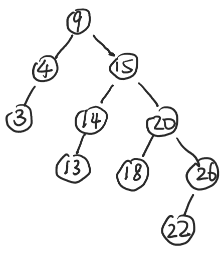
insert 13 :



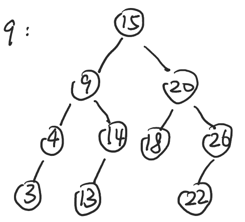
right rotate 20 :



insert 22 :



left rotate 9 :



4.1:

If $h = -1$, then the maximum and minimum number of elements are both 0.
If $h \geq 0$,then
When the layer from 0 to $h-1$ are full and there is only one elements in layer h , we get the minimum number of elements, which is $2^0 + 2^1 + \cdots + 2^{h-1} + 1 = 1 \times \frac{2^h-1}{2-1} + 1 = 2^h$
When all of h layers are full, we get the maximum number of elements, which is $2^0 + 2^1 + \cdots + 2^h = 2^{h+1} - 1$

4.2:

Let A_n be the original sequence.
There are two parts of heap sort, the first is building the heap, and the second is getting and delete the minimum value.
For the first part, the time complexity is $\Theta(n)$ (From the lecture slide)
For each delete operation, we need to execute Percolate Down once each time. For the i-th operation, the worst cases is that the times of execution of loop in Percolate Down equals to the current height of the heap, which is $\Theta\big(\log_2(n-i)\big)$
Then the total time is

$$\mathrm{T} = \mathrm{c} \cdot \left(1 + \overbrace{2+2}^{2\times 2} + \overbrace{3+3+3+3}^{2^2\times 3} + \overbrace{4+4+\cdots+4}^{2^3\times 4} + \cdots + \overbrace{i+\cdots+i}^{2^{i-1}\times i} + \cdots \overbrace{+\log_2 n + \log_2 n + \cdots + \log_2 n + \log_2 n}^{2^{\log_2 n-1}\times(\text{number of element of layer } \lceil\log_2 n\rceil)}\right)$$

let $k = \lfloor \log_2 n \rfloor$, then

$$f(k) = \sum_{i=0}^{k} 2^i \cdot (i+1) = 2^{k+1} \cdot k + 1$$

and then $c \cdot f(k) \leq T \leq c \cdot f(k+1) \Rightarrow T$ is in $\Theta\big(n \cdot \log_2 n\big)$
Thus the time complexity of the worst cases is $\Theta\big(n \cdot \log_2 n\big)$.