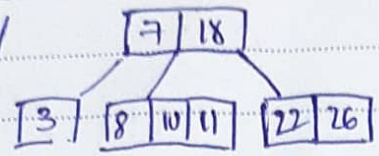
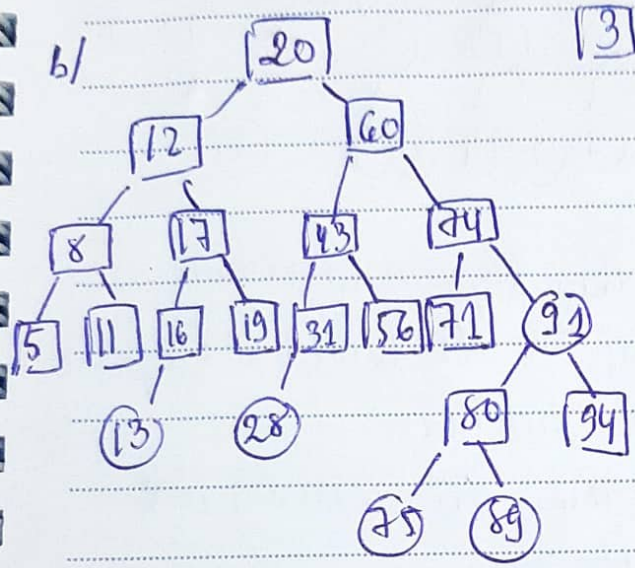
Date: / /

fit@hcmus

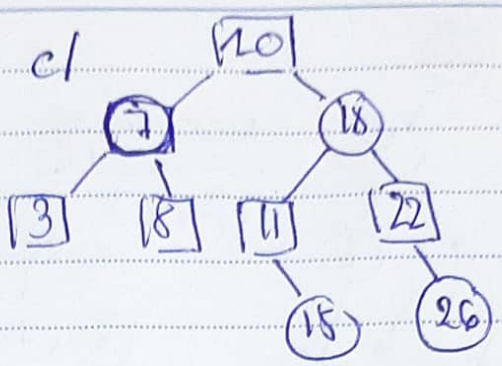
3.2 a/



b/



c/



3.3/

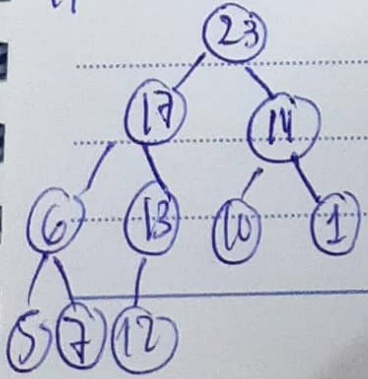
1/ the minimum number of element is 2^h

maximum number of element is $2^{h+1} - 1$

2/ the smallest element must be a leaf node

3/ yes, a sorted array is a min heap

4/ is not a max-heap



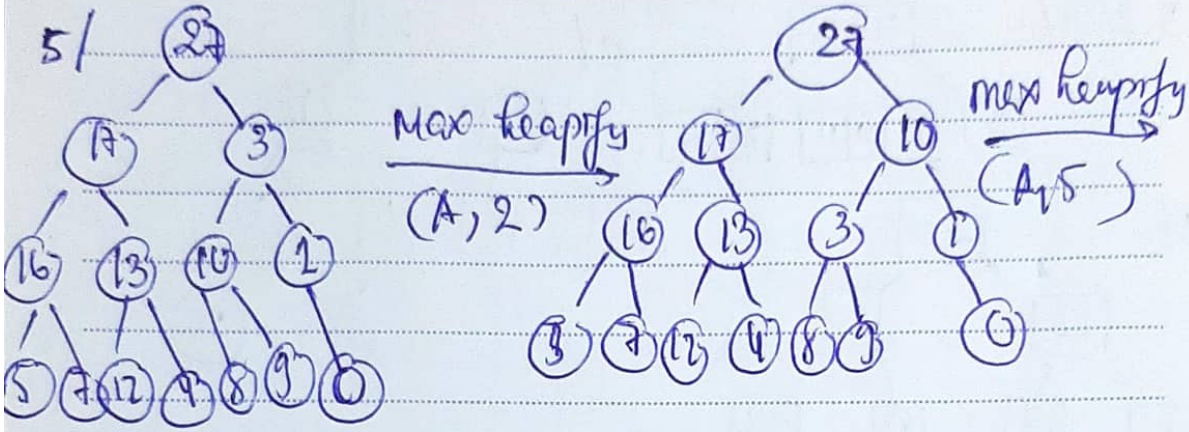
in this heap, 7 is greater than its parent (7 > 6)

⇒ is not a max heap

Date: / /

fit@hcmus

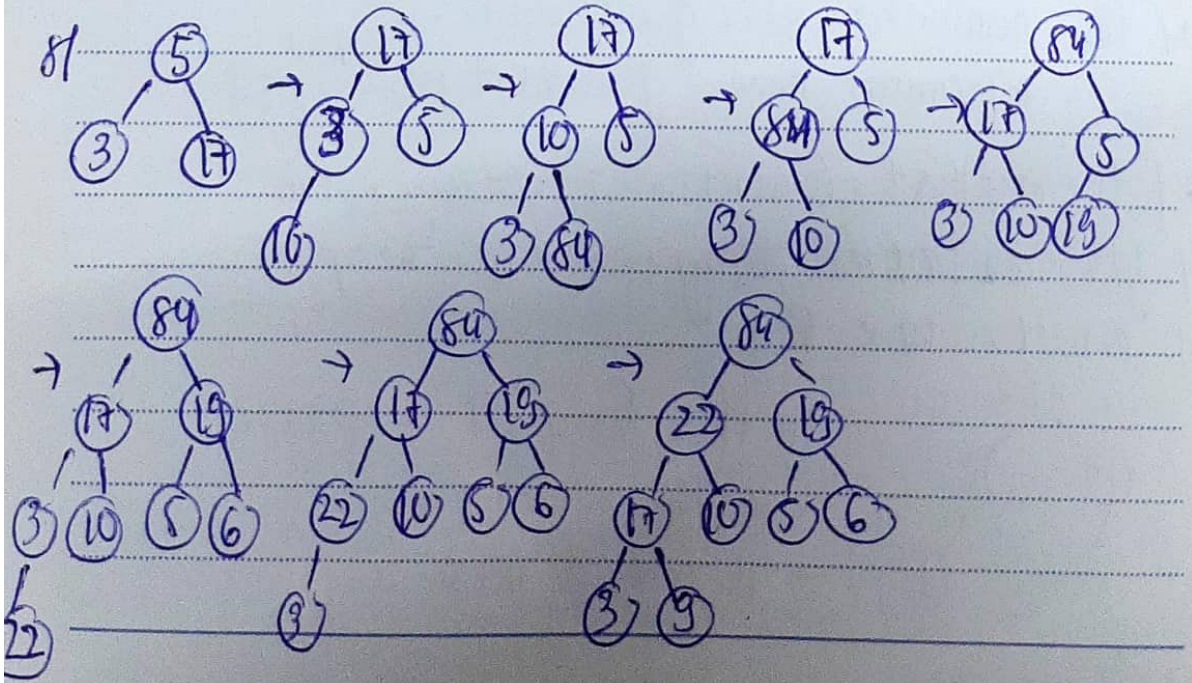
5/



6/ nothing happens when call max heapify to A[i] which is larger then its children.

7/ Nothing, the elements are all leaves.

8/

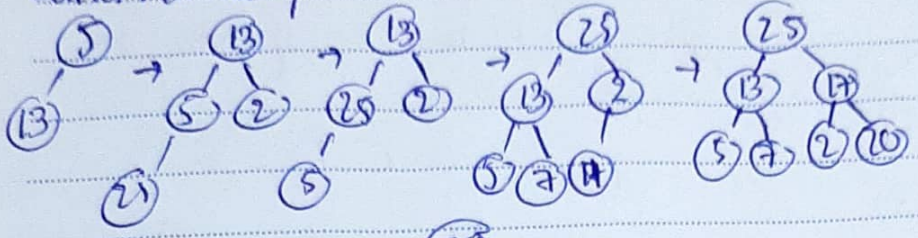


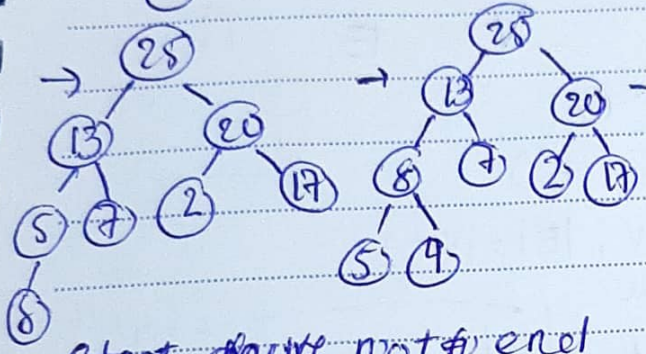
9/ Date:

Build

Date: fit@hcmus

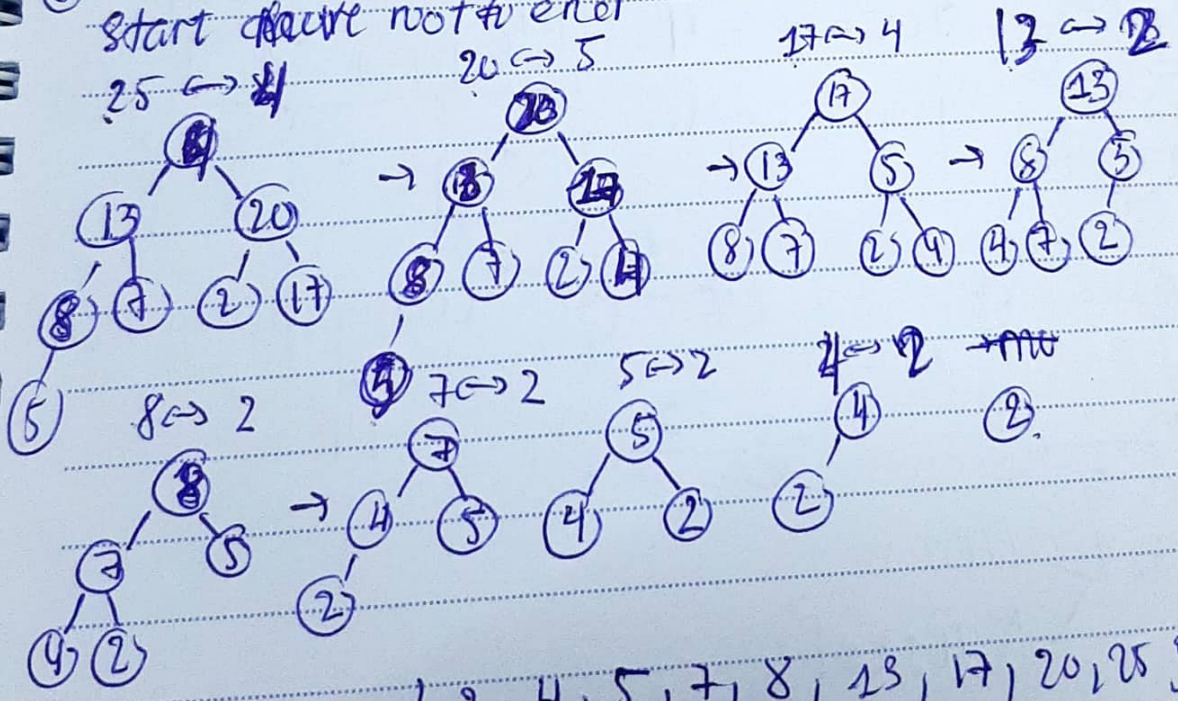
9/ Build Maxheap





→ array = {25, 13, 20, 8, 7, 2, 17, 5, 4}

Start from root to end



→ array = {2, 4, 5, 7, 8, 13, 17, 20, 25}

20/ Both of them or O(n lgn)

```
int left(int parent, int n)
{
	int left = parent * 2 + 1;
	if (left < n) return left; else return 0;
}
int right(int parent, int n)
{
	int right = parent * 2 + 2;
	if (right < n) return right; else return 0;
}
int compare(int a[], int n, int i)
{
	int left_val = left(i, n);
	int right_val = right(i, n);
	if (!left_val && !right_val)return 0;
	if (left_val && right_val)
	{
		if (a[left_val] > a[right_val]) return left_val;
		else return right_val;
	}
	if (left_val) return left_val; else return right_val;
}
void swap(int &a, int &b)
{
	int t = a;
	a = b;
	b = t;
}
void maxHeapify(int a[], int n, int i)
{
	int position_new = compare(a, n, i);
	int position_cur = i;
	while (position_new && a[position_cur] < a[position_new])
	{
		swap(a[position_cur], a[position_new]);
		position_cur = position_new;
		position_new = compare(a, n, position_cur);
	}
}
void buildMaxHeap(int a[], int n)
{
	for (int i = n / 2 - 1; i >= 0; --i)
		maxHeapify(a, n, i);
}
void heapSort(int a[], int n)
{
	buildMaxHeap(a, n);
	for (int i = n - 1; i > 0; --i)
	{
		swap(a[0], a[i]);
		maxHeapify(a, i, 0);
	}
}
```