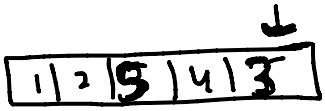
CLASS 68

25 April 2023 21:08

Priority Queue

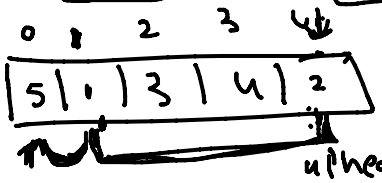
upHeapify, downHeapify, BuildHeap using upHeapify & DownHeapify

upHeapify




```
Void upHeapify (vector<int>& heap, int idx)
{
    if(idx==0)
        return;
    int Parent = (idx-1)/2;
    if (heap[Parent] < heap[idx])
    {
        swap(heap[Parent], heap[idx]);
        upHeapify(heap, Parent);
    }
    else
        return;
}
```

while loop



upHeapify(heap,4)

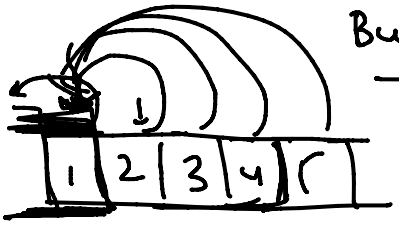
upHeapify(heap,1)

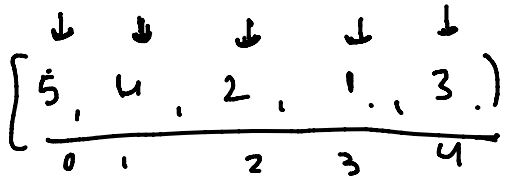
upHeapify(heap,0);

4-1/2 = 1

1-1/2 = 0

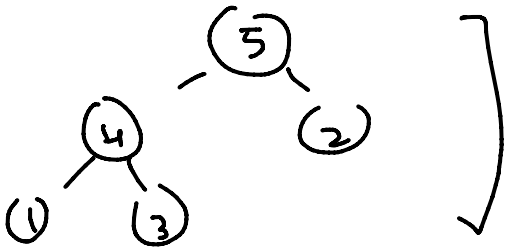
Build a heap using upHeapify





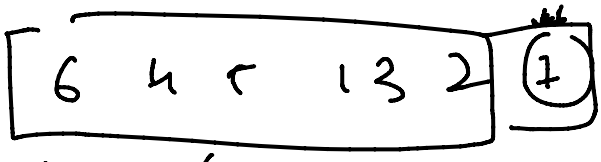
2-1/2 = 0 = 3-1/2 = 1

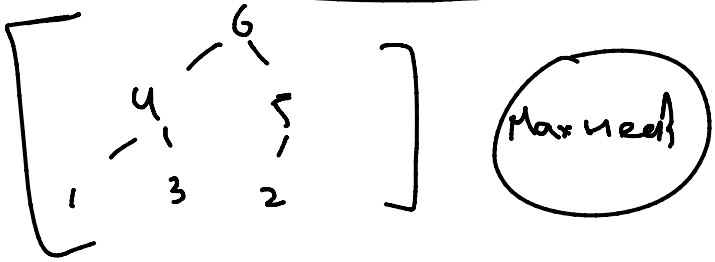
1-1/2 = 0



MaxHeap

$\frac{4-1}{2} = \frac{3}{2} = 1$





MaxHeap

Insertion in Heap

DownheapIfy

```
void downheapify (vector <int>& heap, int idx)
{
    int left = 2*idx + 1;
    int right = 2*idx + 2;
    int largest = idx;
if ( left >= heap.size() and right >= heap.size())
{
    return;
}
if ( left < heap.size()  && heap[left] > heap[idx] )
```

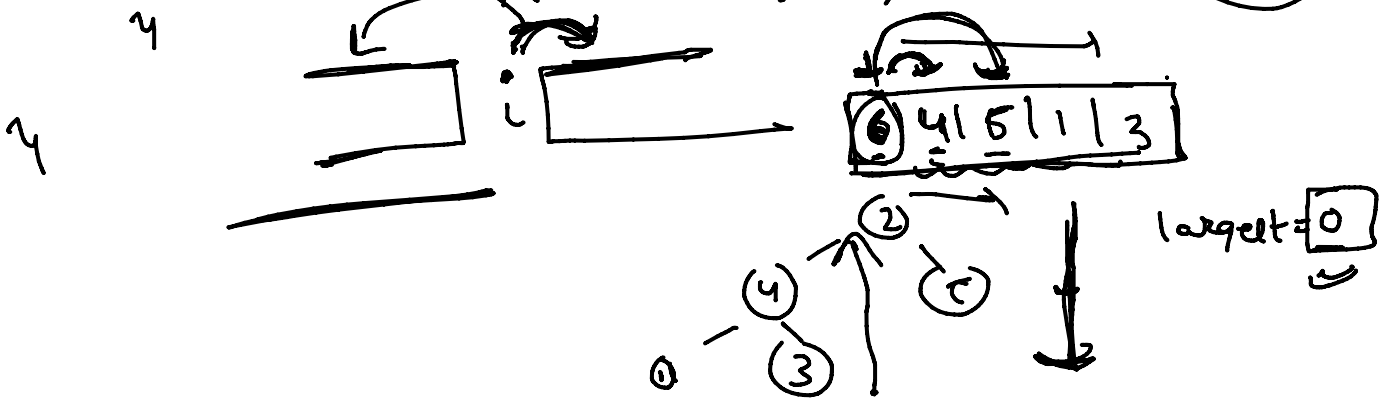
{ largest = left;
}

if (right < heap.size && heap[right] > heap[largest])
{ largest = right;
}

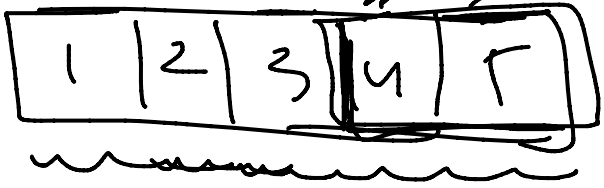
if (largest != idx)
{ swap(heap[idx], heap[largest]);
downHeapify(heap, largest);
}

downHeapify

Heap

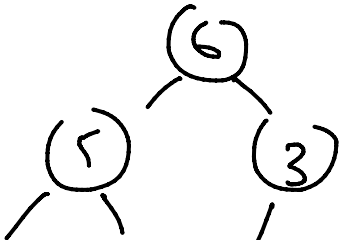


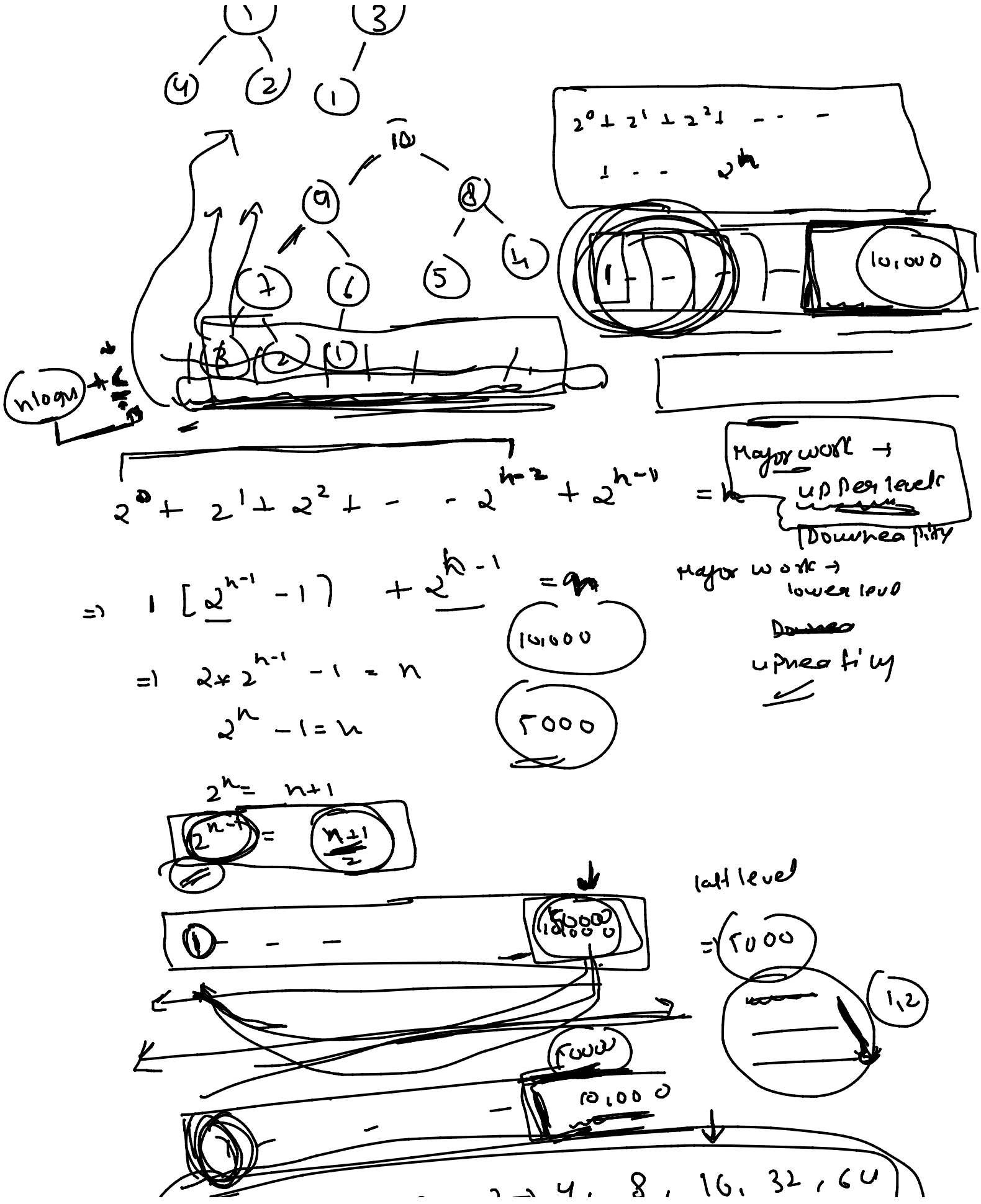
Build heap using Downheapify ()

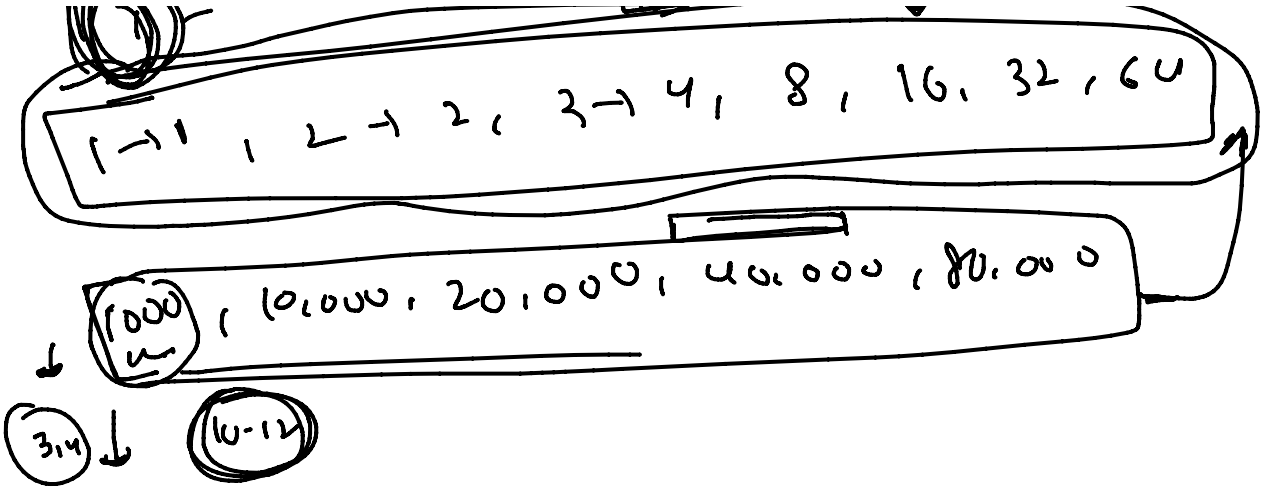


1 5 3 4 2 6

Deletion

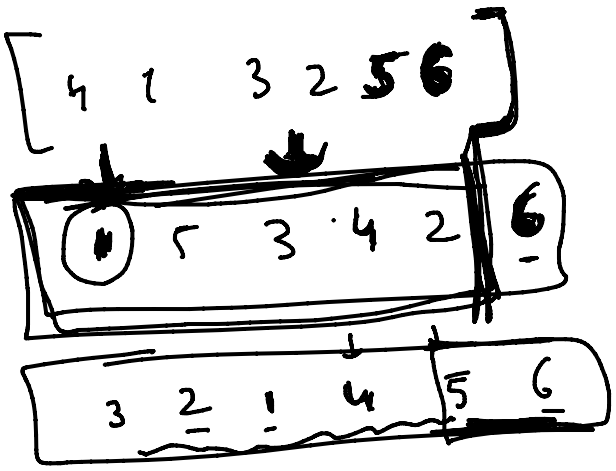



1
3
4
2
1
10
9
8
7
6
5
4
3
2
1
nlogn
$2^0 + 2^1 + 2^2 + \dots + 2^{h-2} + 2^{h-1} = n$
$2^0 + 2^1 + 2^2 + \dots + 1 \dots 2^h$
10,000
Major work → upper level
Downheaping
$\Rightarrow 1[2^{h-1} - 1] + 2^{h-1} = n$
$\Rightarrow 2 * 2^{h-1} - 1 = n$
$2^h - 1 = n$
$2^h = n+1$
$2^{h-1} = \frac{n+1}{2}$
Major work → lower level
Downheaping
10,000
5000
last level
5000
1,2
5000
10,000
2 → 4, 8, 16, 32, 64

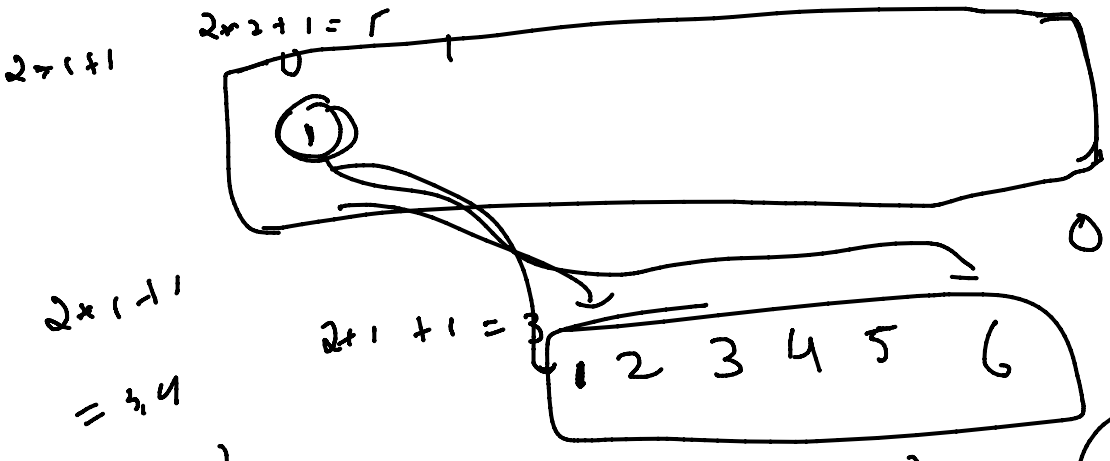


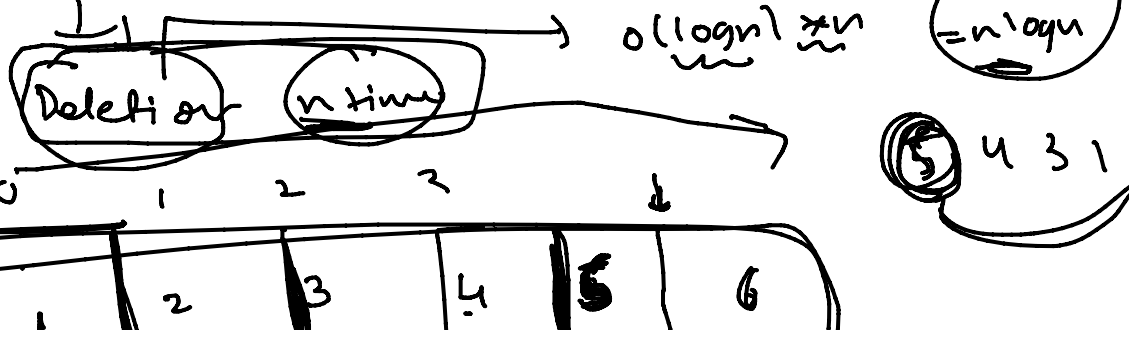
Heap Sort

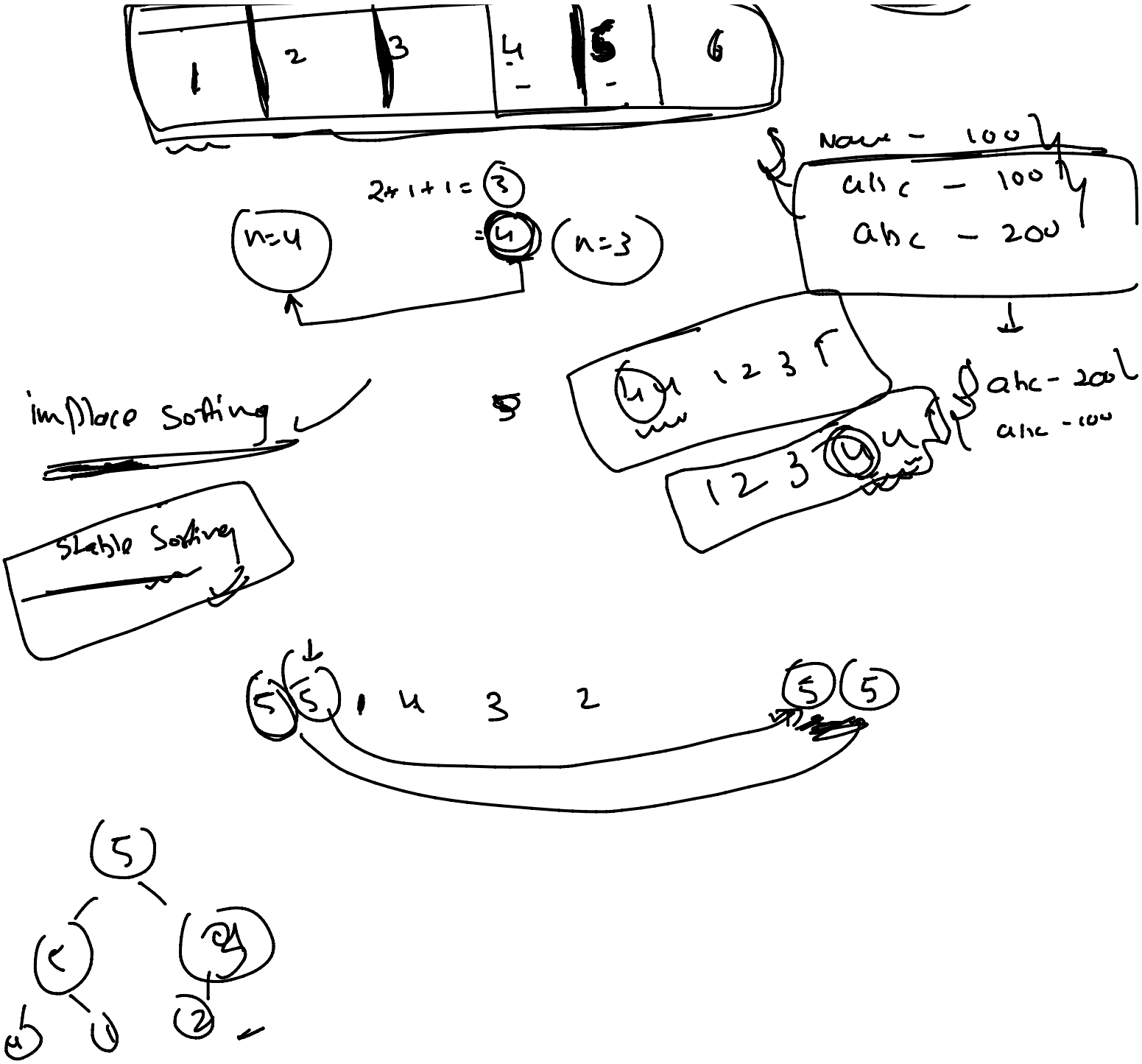
Sorting technique using Heaps



MaxHeap






1
2
3
4
5
6
Name - 100
abc - 100
abc - 200
2*1+1=3
n=4
=4
n=3
1 2 3
abc-200
abc-100
1 2 3
inplace Sorting
Stable Sorting
5 5 1 4 3 2 5 5