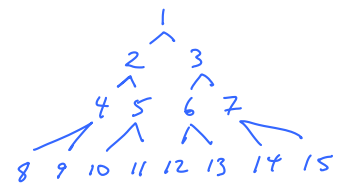
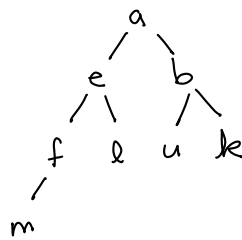


lecture 2.4

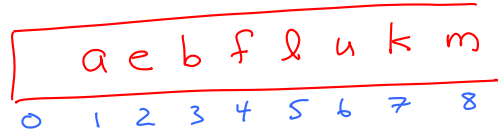
(min) heaps

- add + upHeap
- remove + downHeap
- heap sort

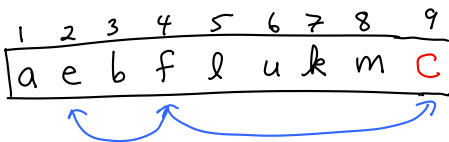
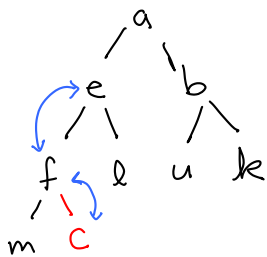
Review: Heaps



level
0
1
2
3



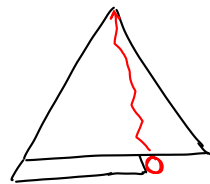
Example: add(c)



Recall: Building a heap

for $i = 1$ to n
upheap(i)

Worst Case: array $a[1..n]$ is a decreasing sequence

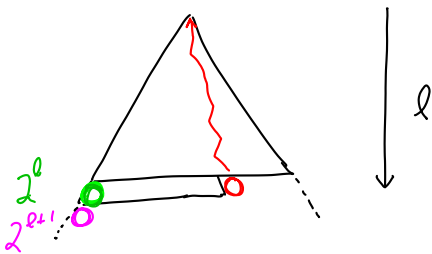


- Why?
- How many swaps would be required?

Let node i be at level l .

Then,

$$2^l \leq i < 2^{l+1}$$



$add(i)$ takes $\lfloor \log i \rfloor$ swaps (worst case)

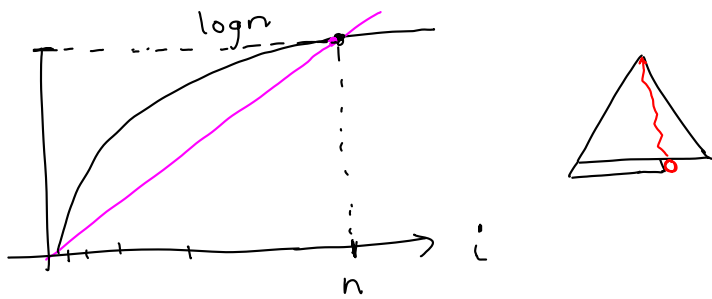
Building a heap

for $i = 1$ to n
upheap(i)

Worst case:

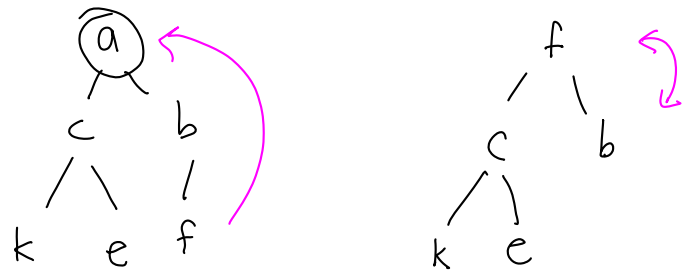
$$t(n) = \sum_{i=1}^n \lfloor \log i \rfloor$$





$$\frac{1}{2} n \log n < \sum_{i=1}^n \log i < n \log n$$

Recall: remove Min()



```

remove Min () {
  n ← a.size
  element ← a[1]
  a[1] ← a[n]
  downHeap ( 1, n-1 )
  a.size --
  return element
}

```



```

downHeap ( i, n ) {
  // pre-condition: children of
  // node i are roots of
  // (possibly empty) heaps
  // (only consider nodes i to n.)
  // post-condition: index i
  // is a root of a heap
}

```

```

downHeap ( i, n ) {
  if ( 2*i ≤ n ) {
    child ← 2*i // left child
    if ( child < n ) {
      if ( a[child] > a[child+1] )
        child ++
    }
    if ( a[child] < a[i] ) {
      swap ( i, child )
      downHeap ( child, n )
    }
  }
}
}

```

Heapsort


```

given a[], build a heap
for i = 1 to n {
  swap ( 1, n+1-i )
  downHeap ( 1, n-i )
}

```


Example

1 2 3 4 5 6 7 8 9
① d b e l u k f w
w d b e l u k f | a



Example

1 2 3 4 5 6 7 8 9
② d k e l u w f | a
f d k e l u w | b a
d e k f l u w | b a



Heapsort

for $i = 2$ to n } $\sum_{i=1}^n \log i$
 upheap(i) }
for $i = 1$ to n { } $\sum_{i=1}^n \log i$
 swap($1, n+1-i$) }
 downHeap($1, n-i$) }
}

Recall $\frac{n \log n}{2} < \sum_{i=1}^n \log i < n \log n$