COL 106 Lecture 12

Topic: Heap Operations

Recap:

- Binary heaps
- Enqueue, ExtractMin, ChangeKey.

Today:

- Proofs of Correctness
- More heap operations

HeapUp(u)

Pre-conditions:

① Almost-complete binary tree.

② $\forall$ nodes $v \neq u$: $v.parent.key \leq v.key$.

③ Keys of u's children $\geq u.parent.key$.

Post-condition : Heap

$u' \leftarrow u$

while $u'.parent.key > u'.key$:

    Exchange keys of u' and $u'.parent$

    $u' \leftarrow u'.parent$

Running time : $O(depth(u))$

Correctness of HeapUp

Keys whose parent key changed : x, y_m, children of x, $w_1 \cdots w_m$

$x \geq$ due to termination condition

$y_m \geq x$ because we swapped y_m, x.

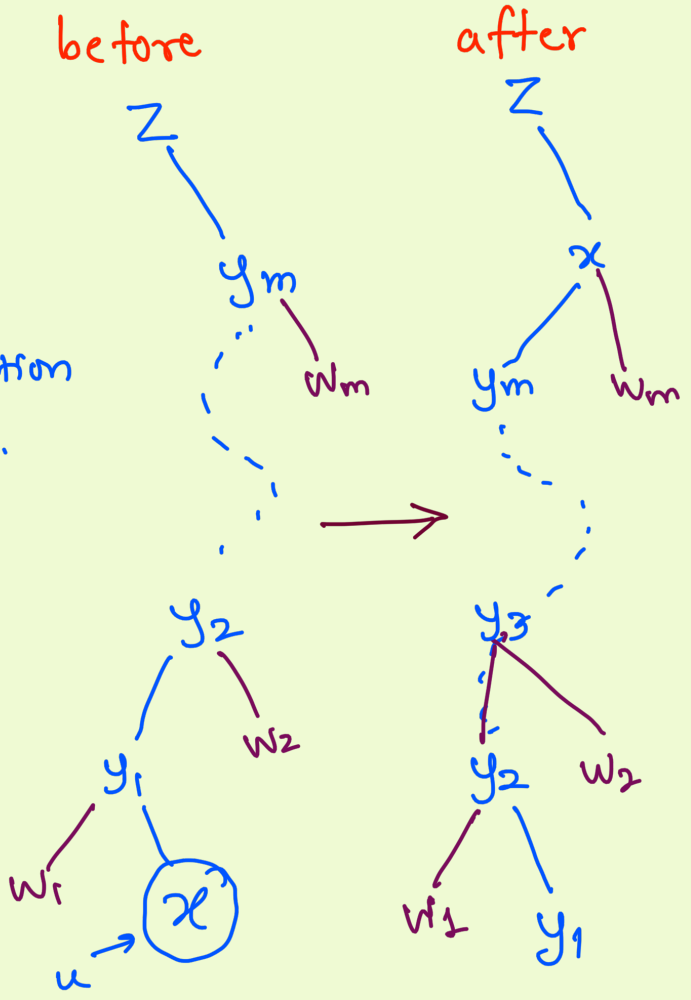
children of $x \geq y_1$ by precond.

$w_i \geq y_i \geq y_{i+1}$

$w_m \geq y_m \geq x$

precond

proved already.



Enqueue (x):

1. Attach a new node u with key x at the leftmost available place in the bottommost layer.

   (Create a new layer if needed)

   (ie. append (x) in the array / list representation.)

2. Heap up (u).

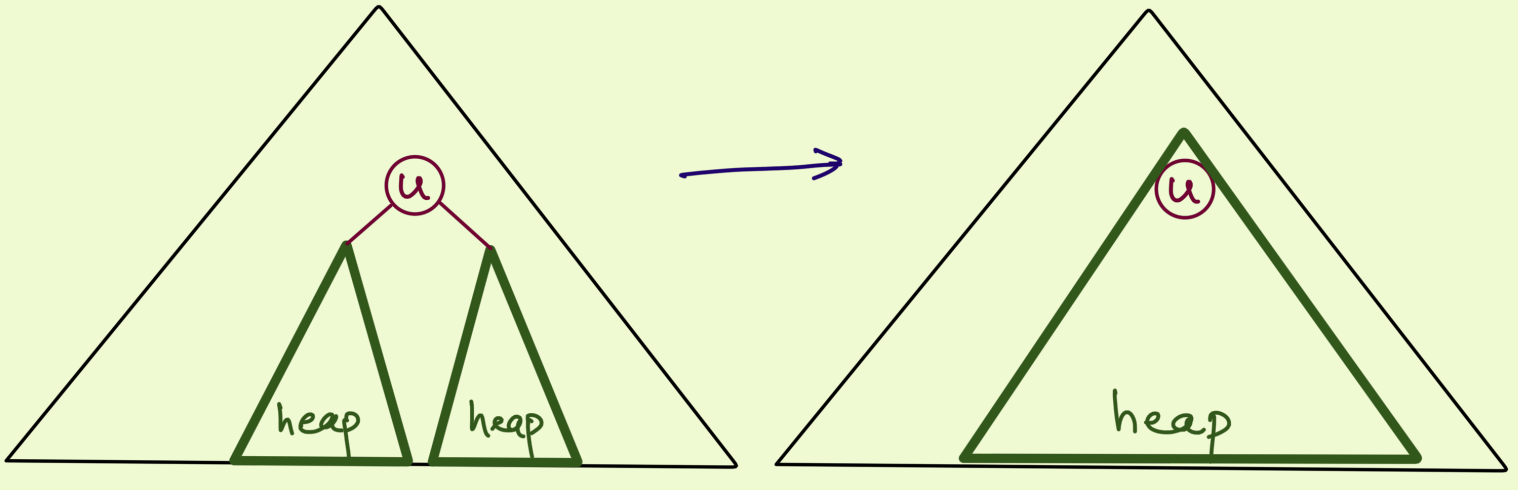
Heap Down (u)

Preconditions :

① Almost complete binary tree

② Left and right subtrees of u are both heaps.

Postcondition : Subtree rooted at u is a heap.



HeapDown(u)

$u' \leftarrow u$

while u' has a child with key $<$ u'.key:

  $v \leftarrow$ child of u' with smaller key.

  Exchange u'.key and v.key

  $u' \leftarrow v$

Running time : O(height of subtree rooted at u).

Correctness of HeapDown

Keys whose parent key changed: z, z', x, y_1

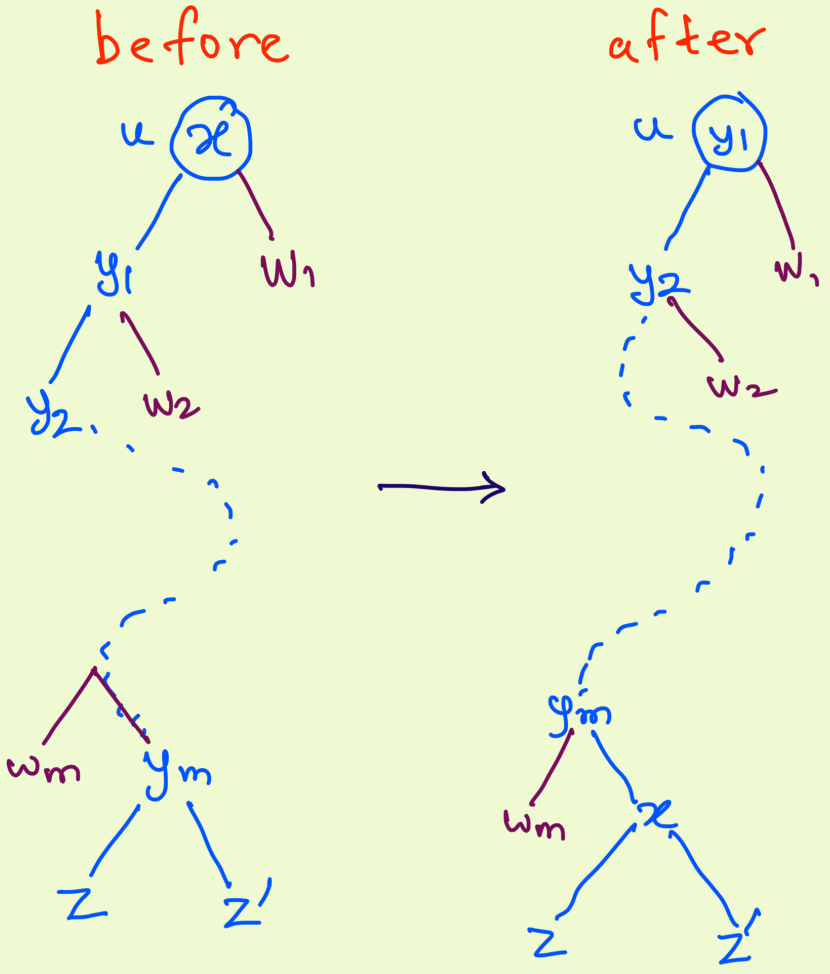
$w_1, \ldots, w_m$

$z, z' \geq x$ — termination condition

$x \geq y_m$ — we swapped x, y_m

y_1 — nothing to check because

$w_i \geq y_i$ because we swapped x with y_i, not w_i



Extract Min ()

1. $w \leftarrow$ root.key
2. Let u be the rightmost node in the bottommost layer. Let x be the key of u.
   Disconnect u from its parent.
   (pop() in the array / list representation)
3. root.key $\leftarrow x$
4. Heap Down (root)
5. Return w.

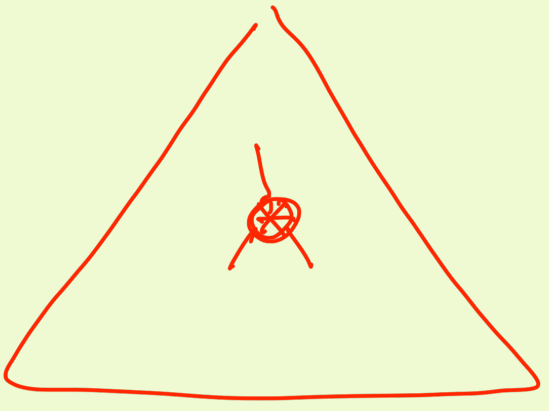
Changekey (u, x) (u: node ; replace u.key by x)

$x' \leftarrow u.key$

$u.key \leftarrow x$

If $x' > x$:
   Heap Up (u)

If $x' < x$:
   Heap Down (u) ← Prove that this makes the entire tree a heap



Build Heap (l) l: list of keys

Obvious algorithm: Enqueue each key in l, one by one

Running time : $O(n \log n)$

Claim: Running time is $\Omega(n \log n)$ (in the worst case).

$n, n-1, n-2, \ldots, n/2, \ldots, 1$

Enqueueing these takes $\Omega(\log n)$ time each.

Can we do better? → Next class