Lecture 21 - April 3

Priority Queues, Heap, Heap Sort

PQ: List Implementations
Heap: Structure, Relational Properties
Heap: Insertion, Deletion
Heap Sort

Announcements/Reminders

- Assignment 4 (on linked Trees) released
- Makeup Lecture (for ProgTest2) posted
- Bonus opportunity: Final Course Evaluation
- Office hours 3pm Thu this week
- Office hours, review session, ex. questions to be releasd
- Lecture notes template, Office Hours, TA Contact

List-Based Implementations of Priority Queue (PQ)

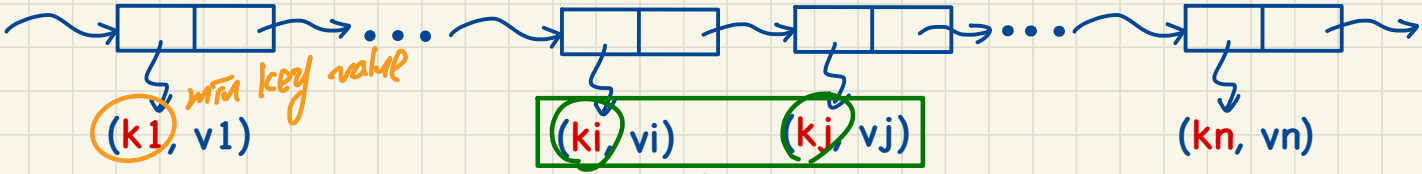
PQ Method	List Method	
	SORTED LIST	UNSORTED LIST
size	list.size O(1)	
isEmpty	list.isEmpty O(1)	
min	list.first O(1)	search min O(N)
insert	insert to "right" spot O(N)	insert to front O(1)
removeMin	list.removeFirst O(1)	search min and remove O(N)

existing space for true

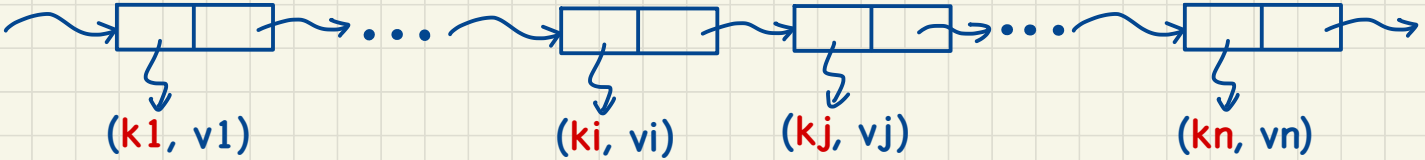
(2) infrequent expansion to P.Q.

Approach 1: Sorted List

more suitable: (1) frequent retrieval/removal of top-prior. entries



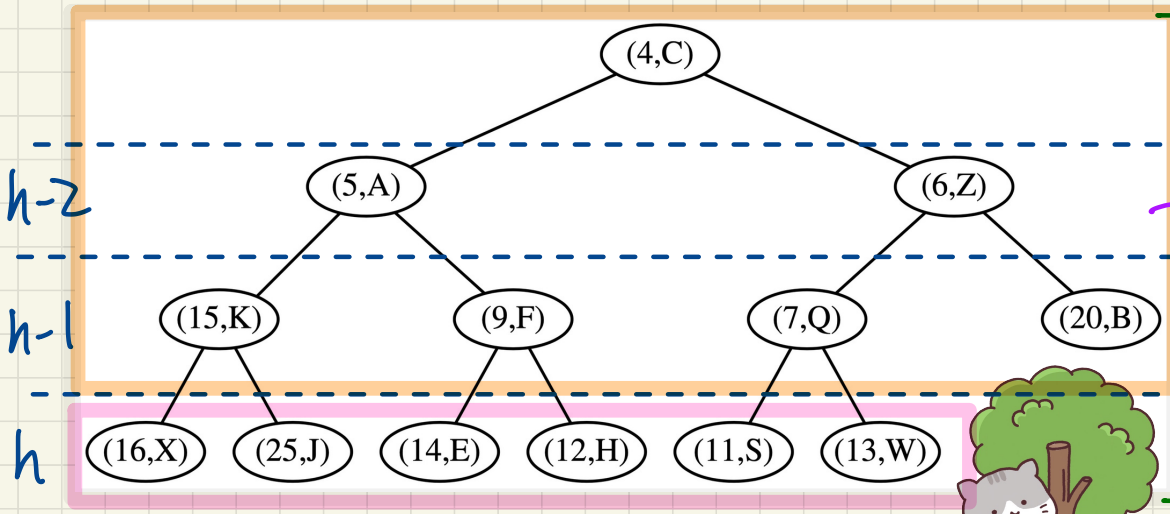
Approach 2: Unsorted List



Heaps: Structural Properties of Nodes

→ height: $\log N$.

Property: The tree is a complete Binary Tree



$h = \lfloor \log N \rfloor$

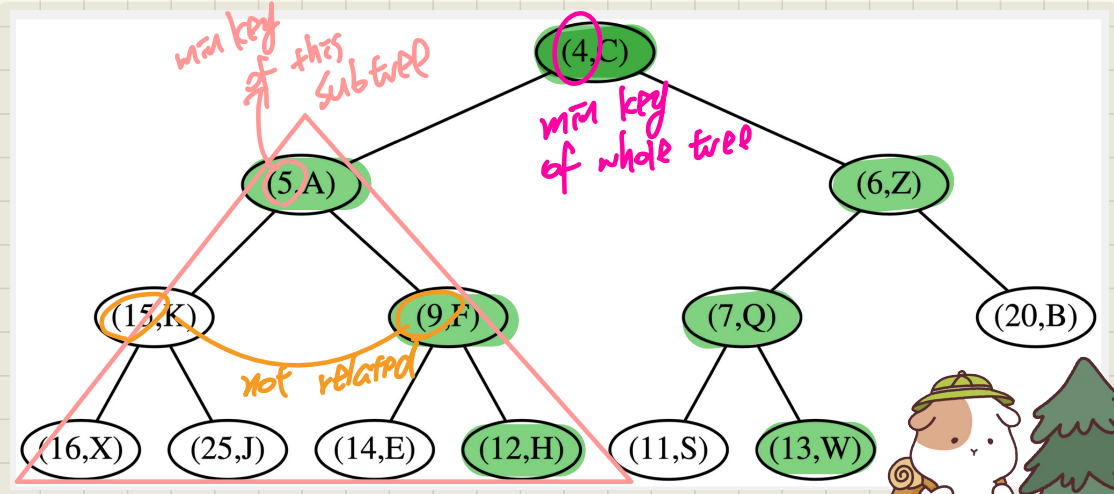
→ # nodes from levels 0 to $h-1$: $2^h - 1$

Min # Nodes: $(2^h - 1) + 1$ 2^h

Max # Nodes: $(2^h - 1) + 2^h$ $2^{h+1} - 1$

Heaps: Relational Properties of Keys

Property: Each non-root node n is s.t. key(n) ≥ key(parent(n))



P1. Any leaf-to-root path has a sorted seq. of keys (non-ascending order)

P2. min key exists in the root

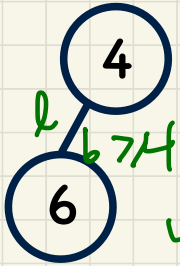
P3. keys between LST and RST are not related.

Example Heaps

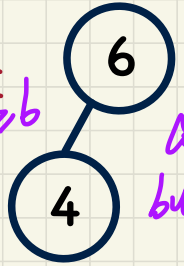
Example 1

smallest possible one-node heap

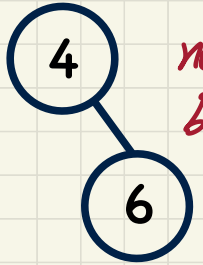
Example 2



Example 3

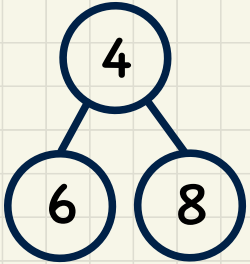


Example 4

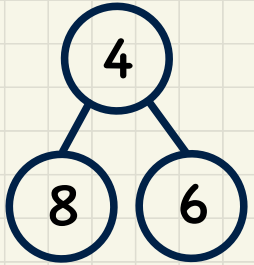


not complete but sat. HOP

Example 5



Example 6



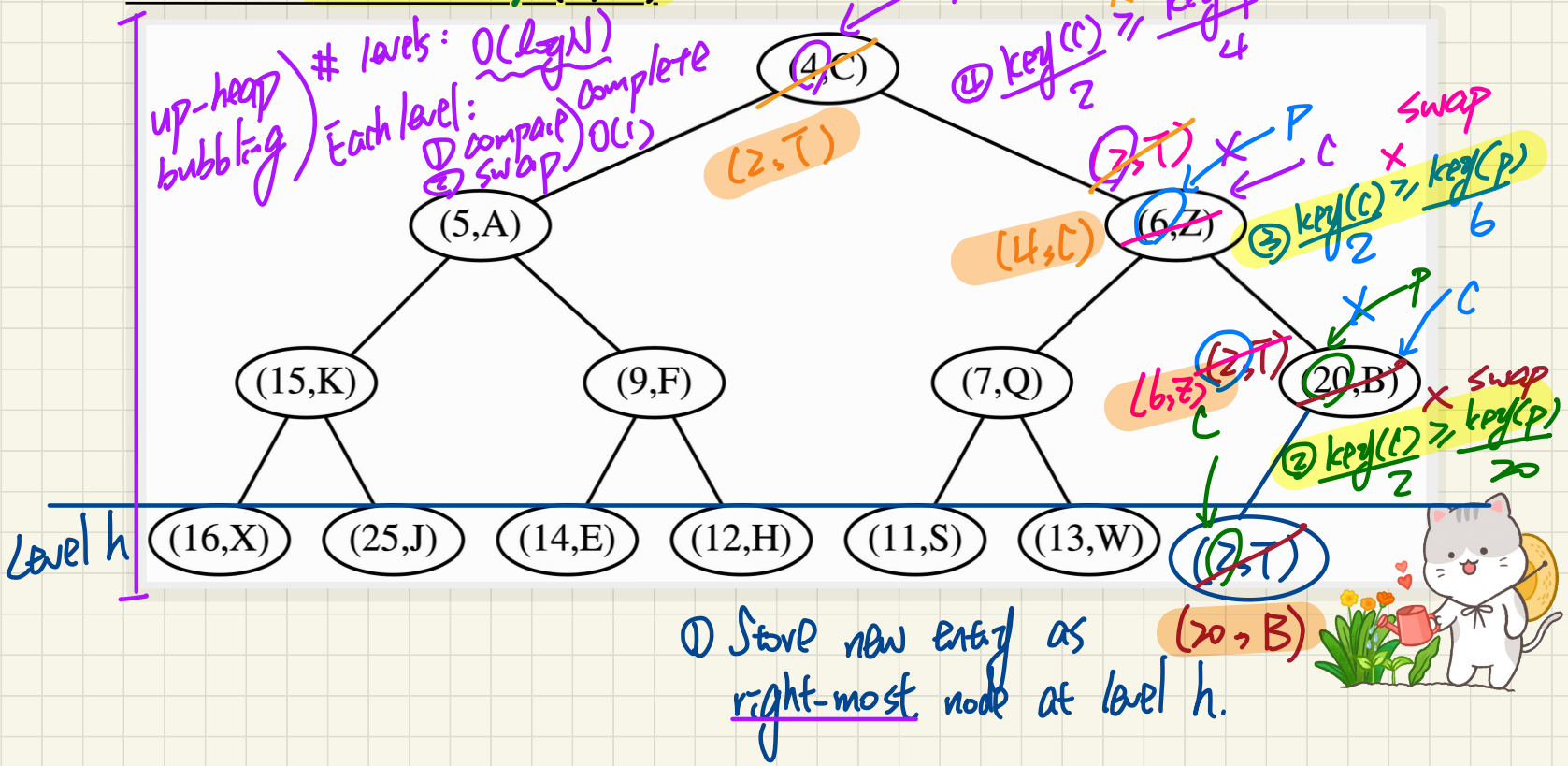
heaps!

full BT

⇒ complete BT

Heap Operations: Insertion

Insert a new entry (2, T)



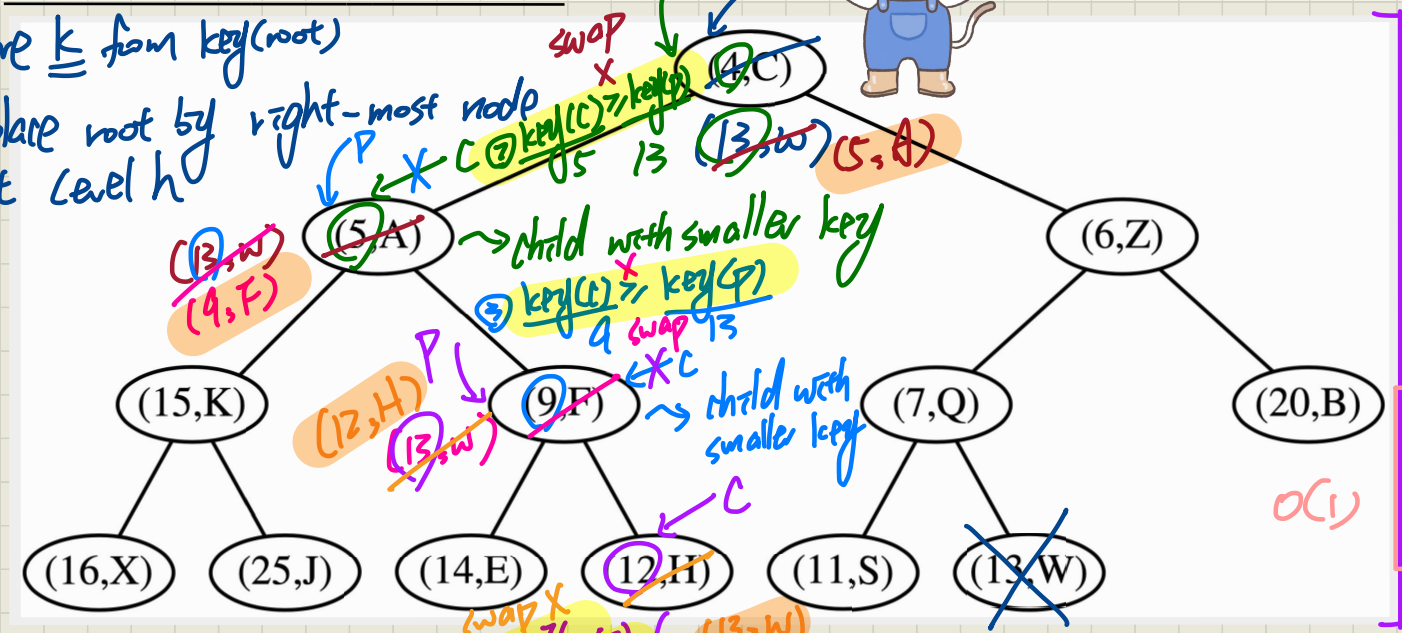
Heap Operations: Deletion

entry with min key (root)

Delete the root/minimum

① store k from key(root)

Replace root by right-most node at level h



down-heap bubbling.

levels: O(log N)

Each level:
① choose c
② compare
③ swap

O(1)

⑤ return k. O(1 · log N)

deletion ← O(log N)

Heap Sort: Ideas

$O(N \cdot \log N)$

construct a heap out of N entries

(A) Top-Down

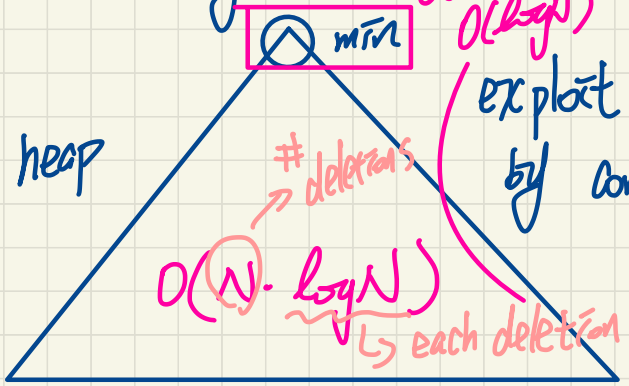
$O(N \cdot \log N)$

(B) Bottom-Up

$O(N)$

$\sim$ selection sort

↳ selection: $O(N)$



extract: $O(\log N)$

exploit the Hop (relational) by continuing to delete/extract min/root until trap is empty.

Exam

- ~ 1~2 review sessions
- ~ PDF guide
- ~ question booklet (answer booklets)
  - ↳ no calculator
  - ↳ little to none multiple choice Qs.
  - ↳ definitions
  - ↳ short answers (explanations, justifications)
  - ↳ coding/tracing
  - ↳ proofs (e.g., asymptotic U.B., trees)

~ Jeff
Sunila
Jadoo

- ~ slides/iPad notes
- ~ example code
  - ↳ BST/Node
- ~ assignments