CSCA48 Winter 2018
Week 6:BT, Heap

Marzieh Ahmadzadeh, Nick Cheng

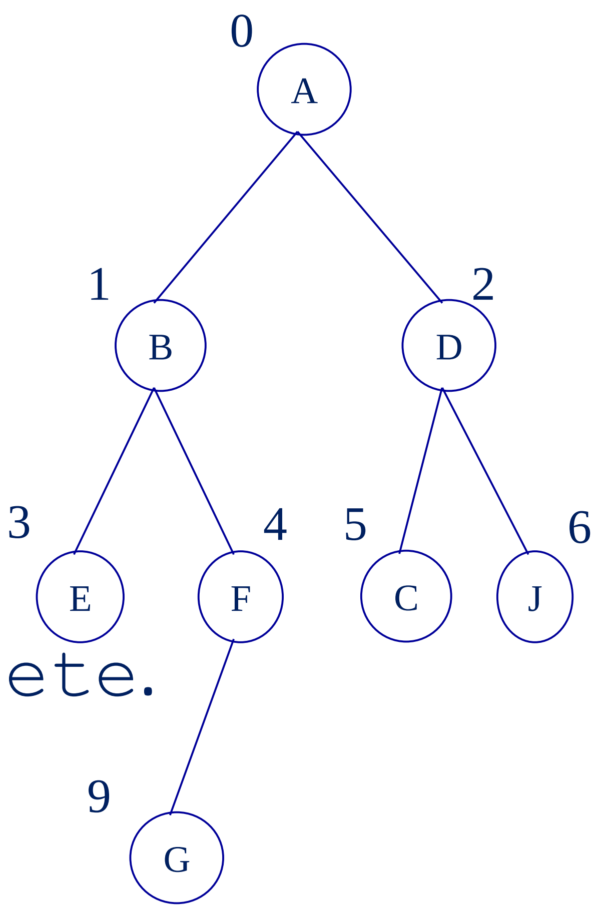
University of Toronto Scarborough

Administrative Detail

- TT1
  - Covers everything up to trees (trees excluded)
    - Including lectures, codes, exercises, readings and tutorials
  - Have your T-Card with you.
  - Know your tutorial # and TA's name
  - Be on-time
  - Attend for the room that allocated to you

List-based Representation of BTs

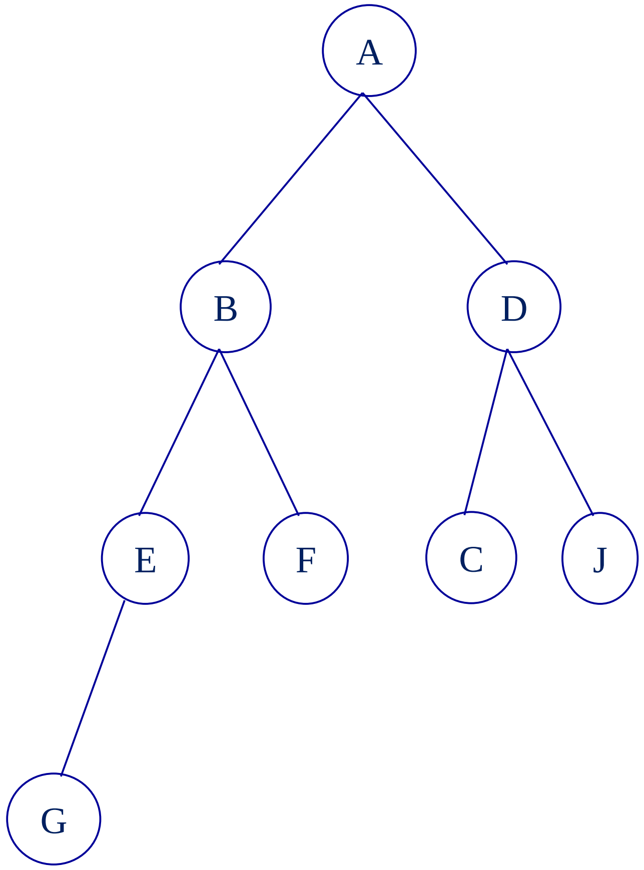
- Nodes are stored in a list based on their ranks.
  - rank(root) = 0
  - rank(left_child_of_a_node) = rank(node) * 2 + 1
  - rank(right_child_of_a_node) = rank(node) * 2 + 2



- It is a waste of space, however, if the tree is not `complete`.

Complete/Compact BT

- A tree with height of `h` is said to be complete if:
  - levels `0, 1, …, h-1` have the maximum number of nodes possible
  - nodes at level `h`, reside in leftmost possible position.
    - (or at level `h-1`, internal nodes are to the left of external nodes and also internal nodes with two children are to the left of internal nodes with one child)



Recall: PQs

- Each entry had a priority
- Operations:
    - insert(e, p)
    - min()
    - extract_min()
- The number of operations that is executed to insert or remove an entry depends on the number of inputs.

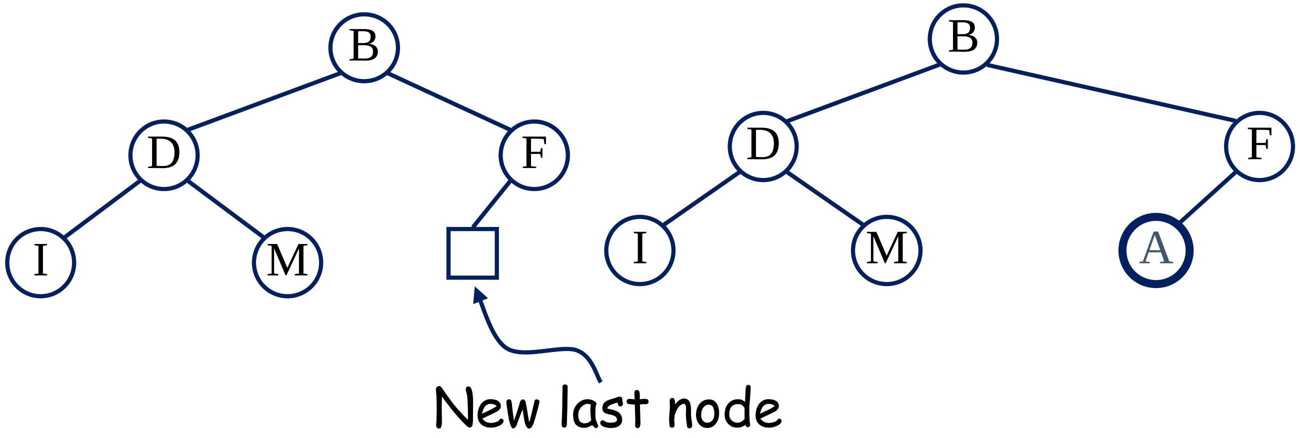
- Which data structure or ADT should we use?
    - None of what you’ve learnt is useful.
- Can we do better?

Heaps

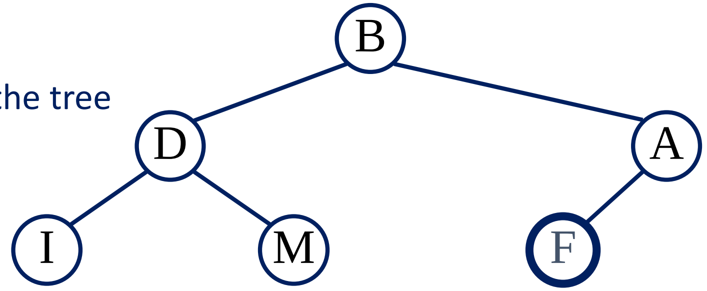
- Is a binary tree that stores `entries` in its node and satisfies two properties:
  - Heap order property: for every internal node `v` other than the root, $key(v) \geq key(parent(v))$
  - a heap is a `complete binary tree`.
    - AKA Compact tree
- Other facts about heap:
  - The rightmost node of depth ***h*** is called the ***last node*** of the heap.
  - The root contains the minimum key (or max key depending on how priority is defined) that is extracted next by extract_min()
  - Every subtree is a heap.
  - If you decide to have the highest key as the highest priority, then for every internal node v other than root, $key(v) \leq key(parent(v))$

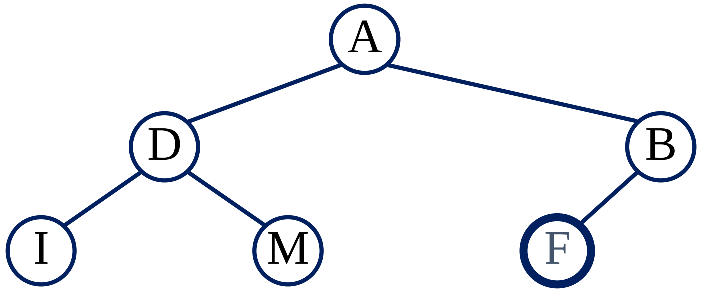
Insertion into a heap

- Has 4 steps:
  1. Find the new last node
  2. Insert data in this node
  3. Update last node
     - How?
  4. Restore the heap-order
     - Up-heap bubbling



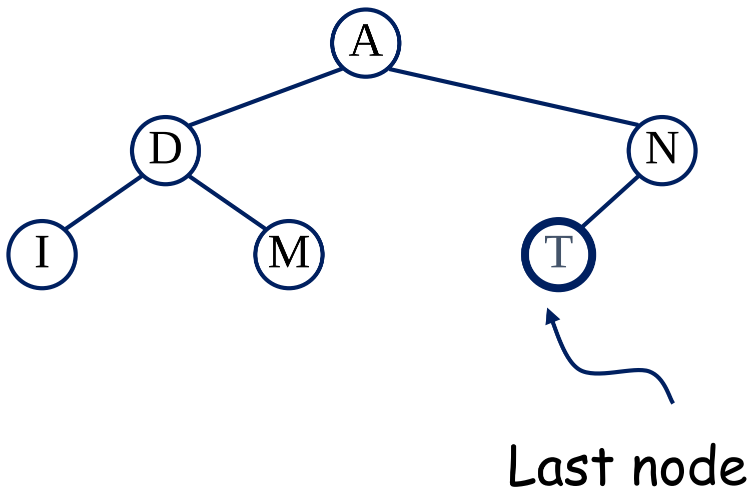
- For simplicity, we show only the keys in the tree

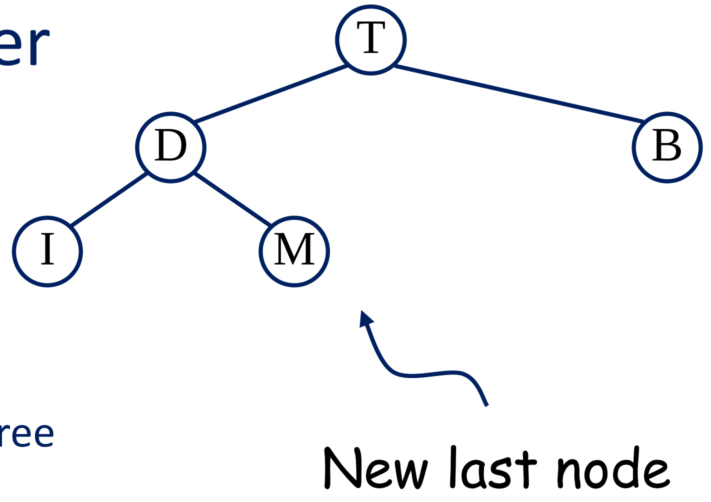


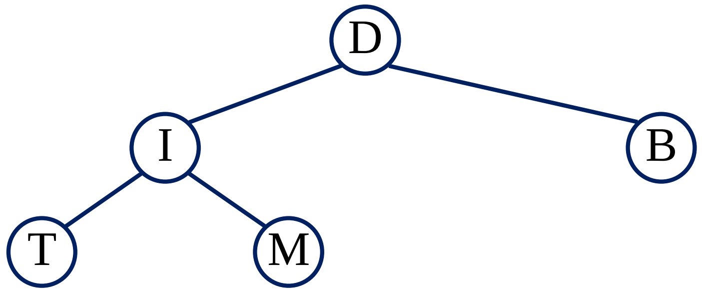


Removal from a heap

- Has 4 steps:
  1. Return the root data and replace the root with the last node's data
  2. Remove the last node
  3. Update last node
     - How?
  4. Restore the heap-order
     - Down-heap bubbling







- For simplicity, we only show the keys in the tree

Height of a heap

- Theorem: A heap storing n entries has height of at most $\log n$

  Proof:
  - Let h be the height of a heap storing n entries
  - Since a heap is a complete binary tree $\Rightarrow$ n = 1 + 2 + 4 + 2^{h-1} + number of nodes at height h
  - At height h, at least there is 1 node $\Rightarrow$ $n \geq 1 + 2 + 4 + \ldots + 2^{h-1} + 1$
  - Thus, $n \geq 2^h$, i.e., $h \leq \log n$

$$\sum_{k=0}^{n} ar^k = \frac{a(1-r^{n+1})}{1-r}$$

Sum of geometric series, where
a is the first term
n is the number of terms
r is the constant that each term is multiplied by to calculate the next term

Heap efficiency

- Since $h \leq \log n$ therefore, each insertion or removal, in worst case scenario takes *log n* with a heap (for up-heap or down-heap bubbling), which is much better than implementing PQs by a list.

- Using a sorted list for PQ implementation insert() needs n^2 operations
- Using an unsorted list for PQ implementation, extract_min needs n^2 operation