**My Summery for Final Exam**

* **Lesson 11 Priority Queues and Heaps: (CHECK NOTEBOOK)**

Queues provide rapid insertions and removal of elements, adhering to the rule: “first in, first out.” In a Priority Queue, one is allowed to remove elements in any desired order (not based on when the element was inserted). Implementing this idea using the heap data structure results in **O(log n)** performance of the main operations.

* **Implementations of Priority Queues**

1. Unsorted array.

* removeMin takes Θ(n) on average.
* insertItem takes Θ(1) if implemented correctly

1. Sorted array

* removeMin takes Θ(1) if implemented properly (could arrange to have array in reverse sorted order so the min is always last element)
* insertItem takes Θ(n) on average, with many copies of array elements

1. Background data structure is a red-black tree

* removeMin takes O(log n) – remove the min, located at the end of leftmost branch (O(log n))
* insertItem takes O(log n) – this is just red-black tree insertion

1. Red-black tree performance is nearly optimal, but can be improved slightly using a different approach.

* **Heap**

1. A heap is a binary tree that has a structural property and an ordering property.

* a heap is a binary tree in which every level except possibly the bottom level is filled completely, and the bottom level is filled from left to right. Such trees are called **complete binary tree**
* The **Min-Heap** Order Property ensures **min key is at the root**. A variant is the **Max-Heap** Order Property that ensures the **max key is at the root**.

1. The structural property implies that if the heap has height h with n nodes, then

Taking logs establishes:

**Theorem**. If T is a heap with n nodes and height h, then h is O(log n) (with small constant factor).

1. Consequently, when a heap is used to implement a PriorityQueue, removeMin and insertItem run in O(log n) and the constant factors are smaller than if a Red-Black tree is used.
2. To **insert into heap**, start with root then fill tree from left to right (maintaining a complete binary tree), then restore heap order by applying **up-heap**, which is comparing the added node with parent, depending on if the heap order is max-to-min or vise-versa, if the ordering is wrong, swap child with parent, and keep doing so until the order of the tree is maintained.
3. To **remove from heap** (the root key will be removed), replace root with the last node in the heap, then restore heap order by applying **down-heap**, which is comparing the new root with its children, depending on if max-to-min order or vise-versa, if the ordering is wrong, swap parent (root) with the picked child (min/max child - depending on the heap ordering mechanism), and keep comparing until the order relationship between the parent and his children is maintained.
4. Build Heap Top-Down 🡪 O(nlog n)

build\_MaxHeap\_TopDown(A, n)

for (i <- 1 to n)

j <- i

while (j > 1 & A[j/2] < A[j])

swap(A[j], A[j/2])

j <- j/2

1. Build Heap Top-Down analysis
2. Build Heap Bottom-Up 🡪 O(n)

build\_MaxHeap\_BottomUp(A, n)

for (i <- n/2 to 1)

j <- i

k <- maxChildIndex(A, j)

while (k !=0)

swap(A[j], A[k])

j <- k

k <- maxChildIndex(A, j)

1. Build Heap Bottom-Up analysis

Since:

Hence:

Since:

Then:

* **Lesson 12 Graphs: (CHECK NOTEBOOK)**

Diagram, timeline

Description automatically generated

Timeline

Description automatically generated

Chart

Description automatically generated

Diagram

Description automatically generated with medium confidence

Diagram

Description automatically generated

Timeline

Description automatically generated

Chart, radar chart

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Text, letter

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

21 &22

Text, letter

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generatedText, letter

Description automatically generatedText, letter

Description automatically generated