Can always make it with log(N) time

**BINARY HEAP**

Log( n ) to add/remove min for operations in a collection of size N

Binary heap is a binary tree with propery at a node is ordered before the value of its children;

Ordering:

Greater than or equal to 🡪 max-heap

Less than or equal to 🡪 min-heap

Heap property is recursive

No guarantee about the rest of the ordering beyond the root

As we add/remove elements,wemaintain 1. Heap order’g’

Complete TPEs.

ADD:

To maintain complete tree structure, we’ll always add new eleenm to first available slot

Need to fix broken heap property

Sift-uo: compare node value with its parent vale and swap if ordering is broken, repeat as necessary, up to to top of free

Minheal:

[26,43,9,34,17,51,6,120A

Complexity:

1. Start with a o of n elements

1. locate je;l ;f o(log(n))

2. ,army structure o(1)

2. fix heap

Sequence of compare/shield updatetions

Each compare/swap is o(1)

At most o(log(n)) stages

Missing 1 line

**Remove-min:**

Either root node of tree, first node in array,

Identifying part to mofonce again, need to fix head

Move last heap element to root

While heap is broken,

(order between node/children broken)

Swap node value with smaller childs

6 remove, what now?

[43,17,34,12,1,12,9,51,26,

It maters wjich child is smaller

To keep tree balanced and correct/complete, take that lowest value in heap furthest from root, and swap it to balance heat

Then swap numbers tp jap s

HEAP SORT:

Stage 1: build heap

N steps (additions)

Each takes o(log(n)) work

Total work: N \* log(N)

Stage 2: deconstruct heap

N steps

Each step is O(log(N))

Total work: o(N \* log(N))

Total work for heap sort is o(NlogN)

\*best you can do using comparison based sorting algorithms

Implementing heap sort:

If max size is changing:

Linked data structure

Root, node needs value associated with it, 3 references: parent, left, right (slots)

If max heap size is known, dump into arrau