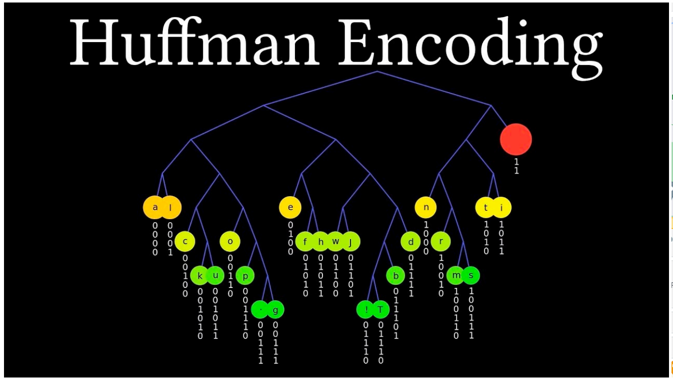
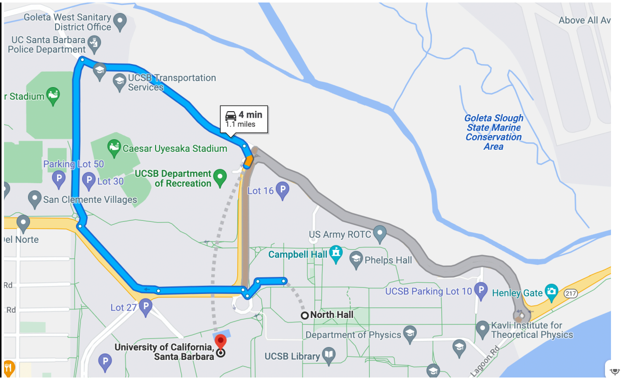
Lecture handout
https://bit.ly/CS24-binaryheap

PRIORITY QUEUES & BINARY HEAP

Problem Solving with Computers-II

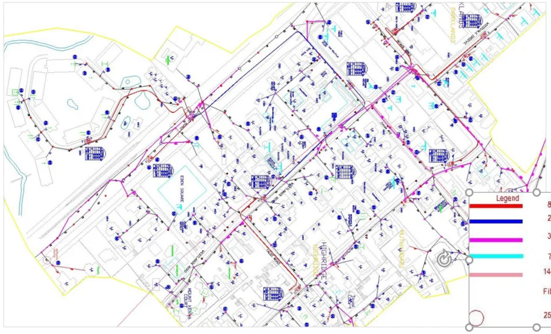


Data Compression



Navigation

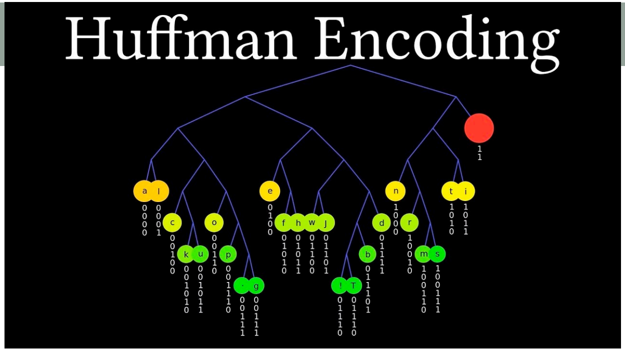
Planning Fiber Routes



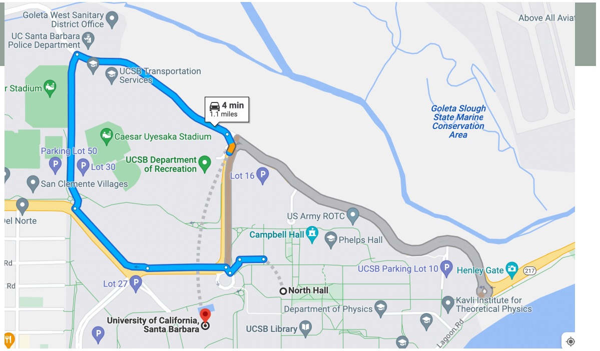
Network Design

Annoucements

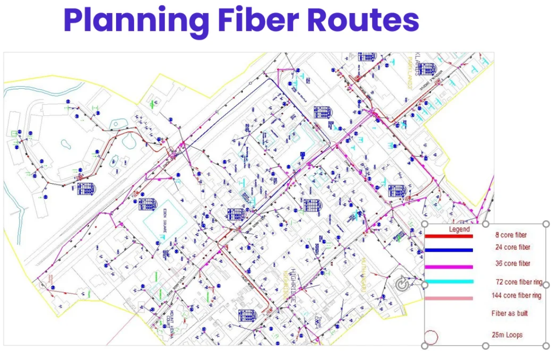
- **PA01 — extended deadline to next Friday 05/16**
- **Upcoming lab: implement a priority queue as a binary heap**
- **Midterm on Thursday (05/07)**
- **Extra office hours today after lecture**
  - **TA/LA Group office hours 2p - 3p in HFH 1152**
  - **Professor OH, 2p - 4p in HFH 1155 (instead of Thursday OH)**



Data Compression



Google Maps Navigation



Network Design

Algorithms: **Huffman Coding** \ | **Shortest Path** | / **Minimum Spanning Tree**

\ | /

ADT: **Priority Queue**

|

Data structure: Binary Heap

|

Complete Binary Tree

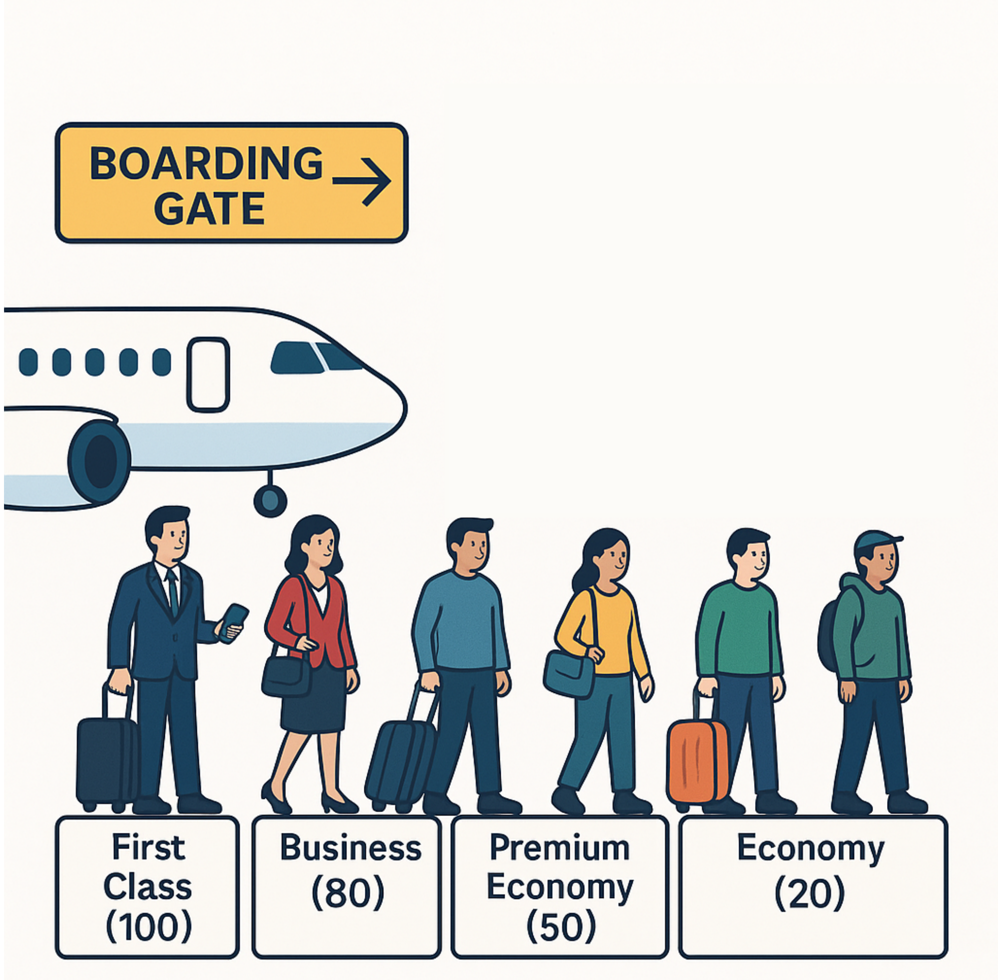
|

Vector

Many algorithms need to compute the min OR max repeatedly.

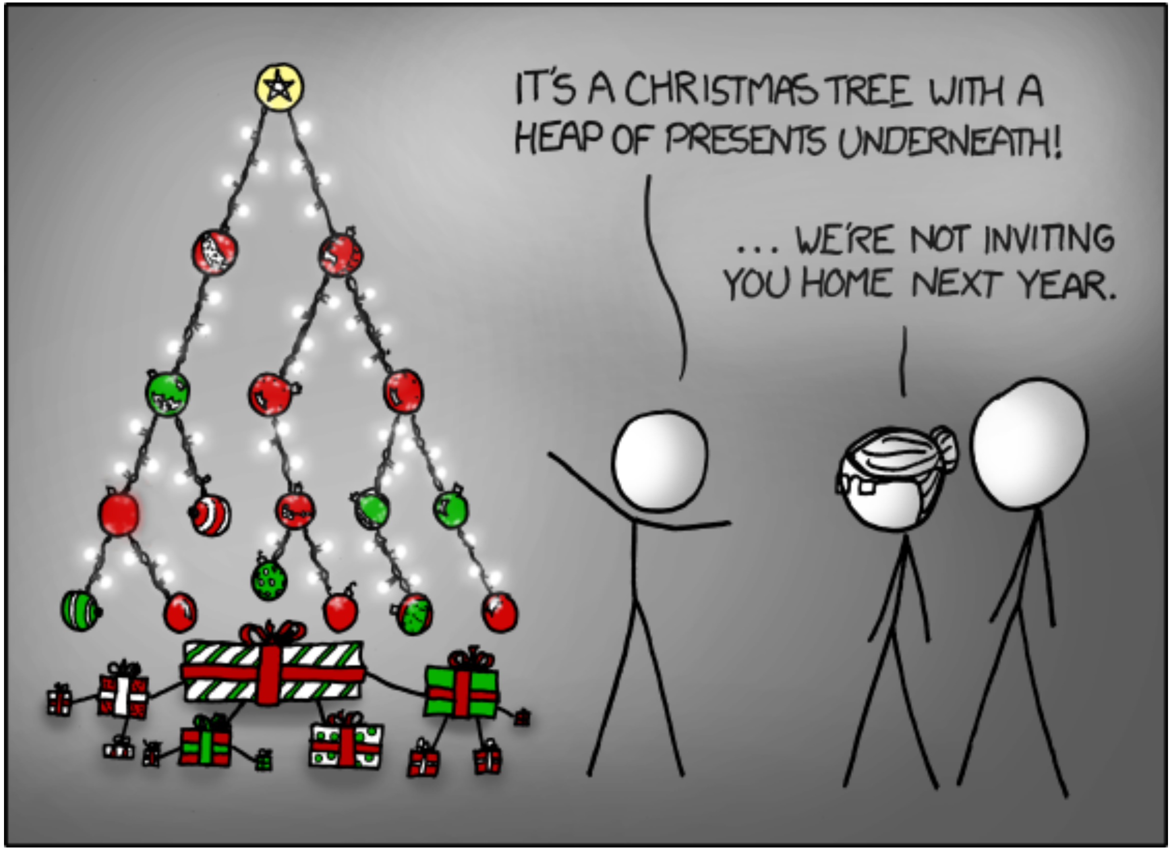
Priority Queue is used to speed up the running time!

C++ Priority Queue ≡ Airport Priority Boarding



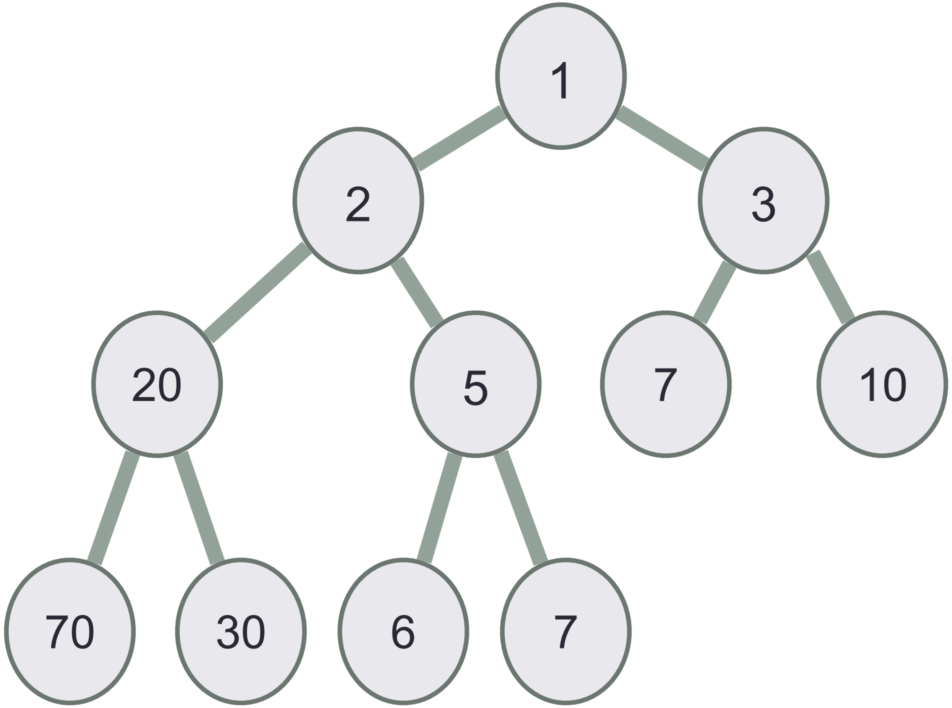

```
priority_queue<int> pq;

// New passengers arrivals
pq.push(20);
pq.push(20);
pq.push(80);
pq.push(50);
pq.push(100);

// Whose boarding next?
cout << pq.top();

// Next passenger to board
pq.pop();
```

priority_queue ADT is implemented as a Binary Heap Tree



Think of binary heap as a heap of presents!!

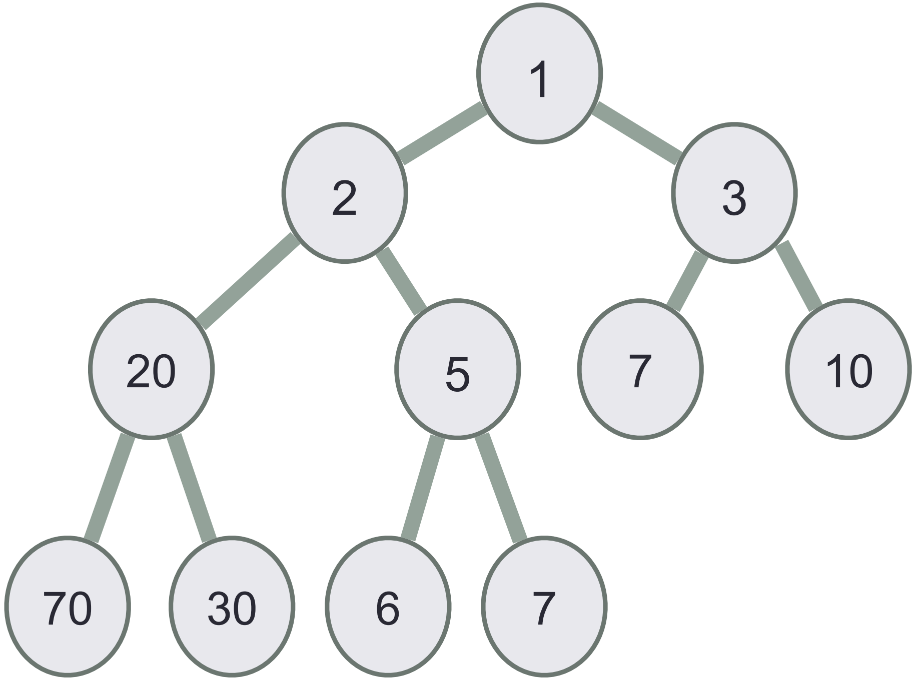
Two important properties of a binary heap tree



(1) Shape property:

(2) Heap property :

Two important properties of a binary heap tree



Example of a min-heap

(1) Shape property:

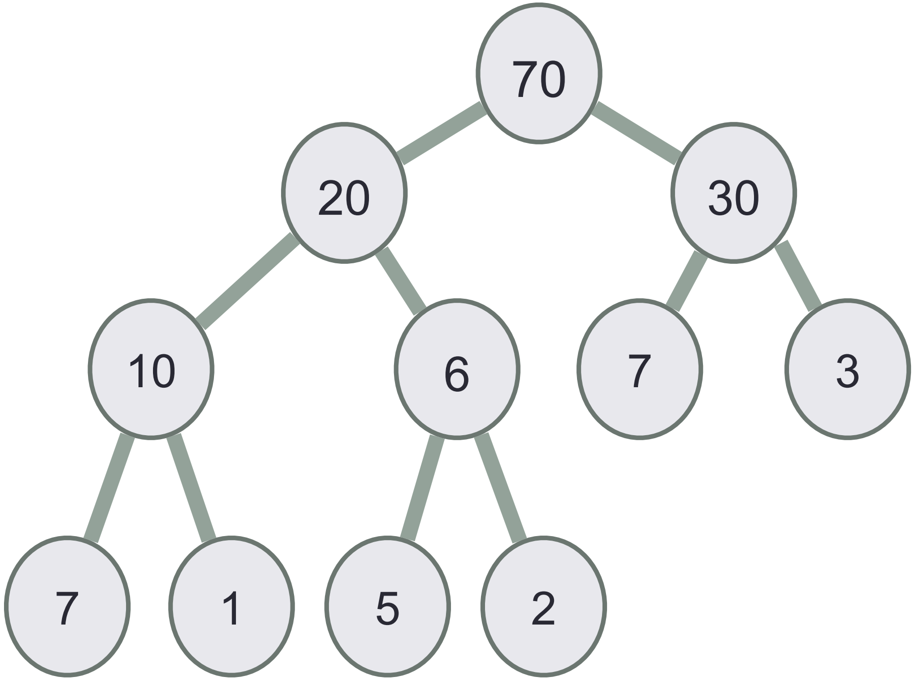
Internally, a heap is a **complete binary tree,** where each node satisfies the heap property

(2) Heap property:

In a min-heap, for each node (x):

key(x) <= key(children of x)

Two important properties of a binary heap tree



Example of a max-heap

(1) Shape property:

Internally, a heap is a **complete binary tree,** where each node satisfies the heap property

(2) Heap property:

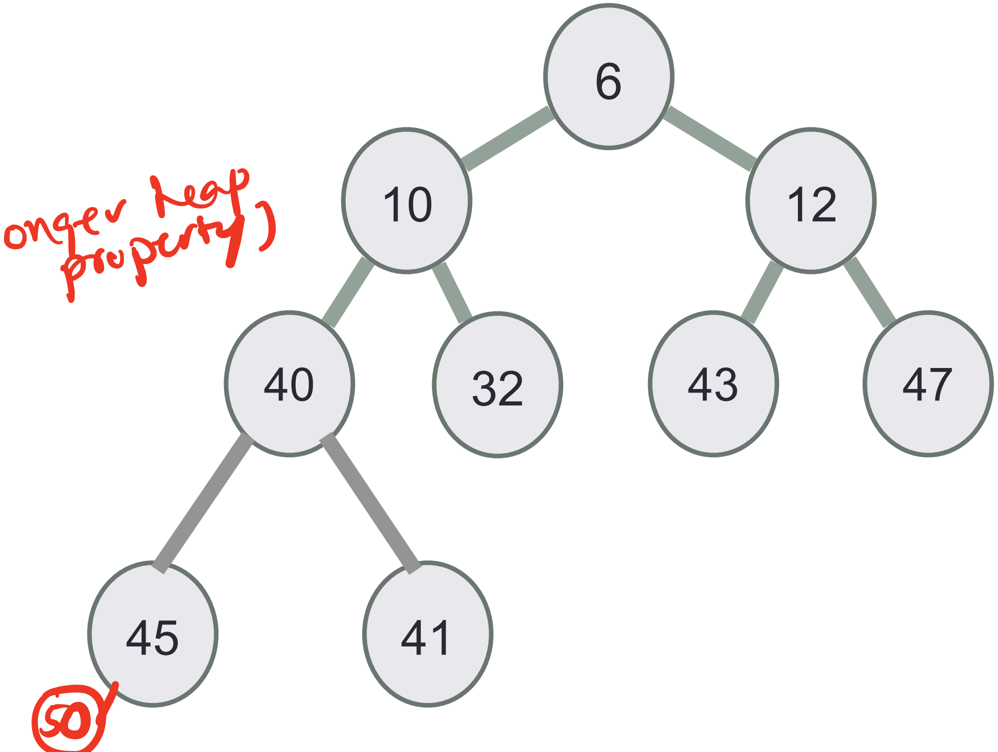
In a max-heap, for each node (x):

key(x) >= key(children of x)

Identifying heaps

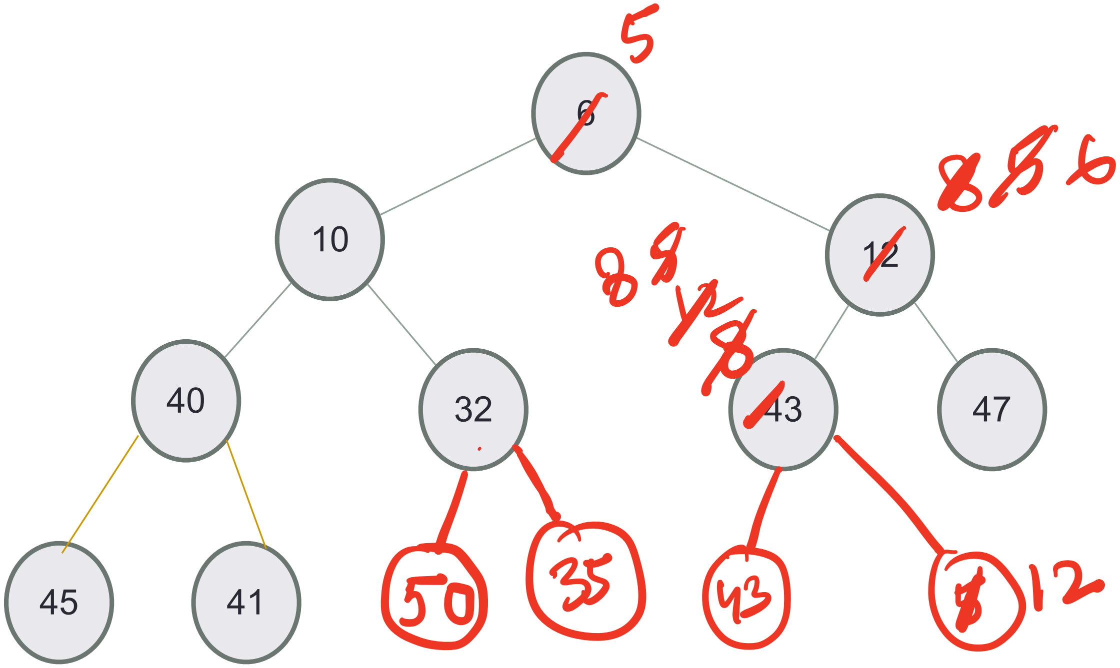
Starting with the following min-Heap which of the following operations will result in something that is NOT a min Heap

A. Swap the keys 40 and 32
B. Swap the keys 32 and 43
C. Swap the keys 43 and 40 (no longer heap property)
D. Insert 50 as the left child of 45
E. C&D



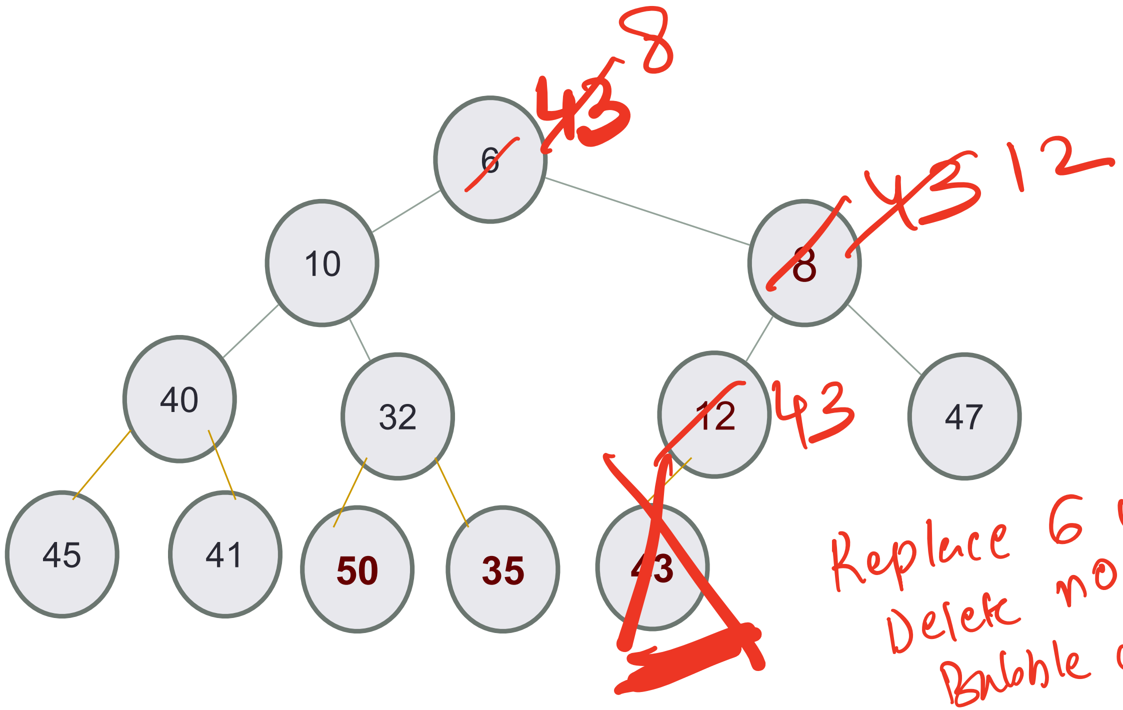
Insert into a min heap: push(50), push(35), push(8)

- Insert key x, preserving the complete structure of the heap : O(1) — why?
- If the heap property is not violated - Done
- Else "bubble up" key x (swapping with its parent's key) until the heap property is restored.



pop(): delete the key at the top() O(log n)

- Replace the root with another node to preserve the complete structure of heap: O(1)
- "Bubble down" i.e. swap keyof node with child that has the smallest key value until the heap property is restored



Replace 6 with 43.
Delete node with value 43
Bubble down '43' until heap property is restored

Practice inserting the values 20, 5, 7, 1, 3, 2 into an initially empty min-heap. But instead of drawing the results as a tree, draw the resulting vector that represents the binary heap tree

```
procedure push(x: key value)
  insert x in the first open spot in the tree
   while(x has a parent && parent(x) > x): //Bubble up!
    swap(x, parent(x))
  done
```

```
procedure max(a1,a2, … an: integers)
 max:= a1
 for i:= 2 to n
     if max < ai
         max:= x
return max{max is the greatest element}
```

What is the **best case** Big-O running time of max?

A. $O(1)$
B. $O(\log n)$
C. $O(n)$
D. $O(n^2)$
E. None of the above