Data Structures & Algorithms

Heaps

B.Bhuvaneswaran, AP (SG) / CSE

9791519152

bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

Heaps

- A heap is a data structure that is an implementation of the priority queue.
- Note that a priority queue is an abstract data structure.
- A heap is one of many ways to implement a priority queue.
- However, people often use the two terms interchangeably.

Heaps

- A heap is a container that stores elements, and supports the following operations:
  - Add an element in O(logn)
  - Remove the minimum element in O(logn)
  - Find the minimum element in O(1)

Note

- A heap can also find the max elements instead of the min elements.
- If a heap is configured to find/remove the min element, it's called a min heap.
- If it's configured to find/remove the max element, it's called a max heap.

Heaps

- The ability to find the max/min element in constant time, while only needing logarithmic time to maintain this ability through changes makes a heap an extremely powerful data structure.

How is a heap implemented?

- Like a hash map, all major programming languages will have support for a heap, so you don't need to implement it yourself.
- In terms of solving algorithm problems, you only really care about the interface, not how it is implemented.
- But like with hash maps, it's still good to understand the implementation in case you are asked about it in an interview.

How is a heap implemented?

- There are multiple ways to implement a heap, although the most popular way is called a binary heap using an array.

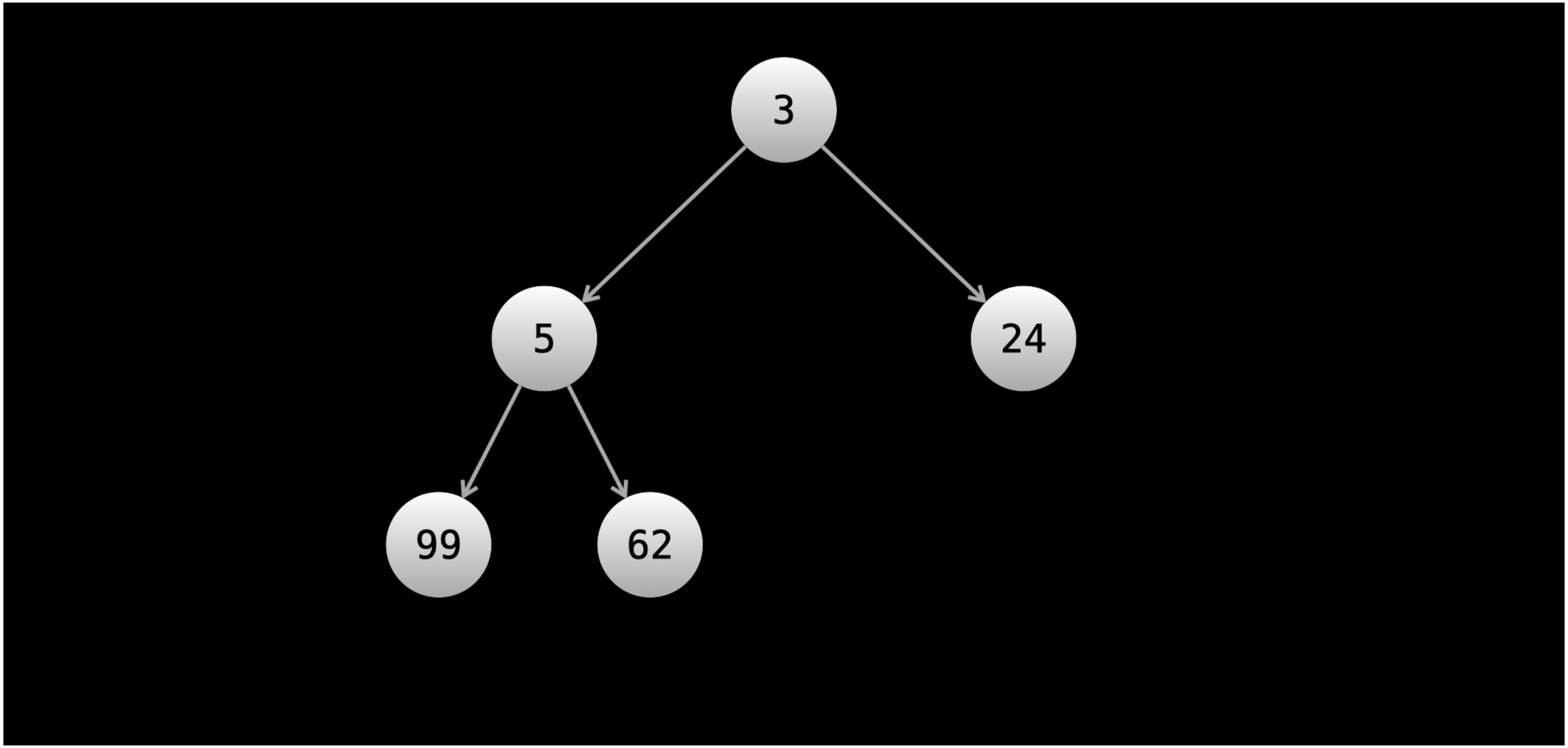
How is a heap implemented?

- A binary heap implements a binary tree, but with only an array.
- The idea is that each element in the array is a node in the tree.
- The smallest element in the tree is the root, and the following property is maintained at every node:
  - if A is the parent of B, then A.val <= B.val.
- Notice that this property directly implies that the root is the smallest element.
- Another constraint is that the tree must be a complete tree.

How is a heap implemented?

- The parent-child relationships are done using math with the indices.
- The first element at index 0 is the root, then the elements at indices 1 and 2 are the root's children, the elements at indices 3 and 4 are the children of the element at index 1 and the elements at indices 5 and 6 are the children of the element at index 2, and so on. If a node is at index i, then its children are at indices 2i + 1 and 2i + 2.
- When elements are added or removed, operations are done to maintain the aforementioned property of parent.val <= child.val.
- The number of operations needed scales logarithmically with the number of elements in the heap, and the process is known as "bubbling up".

How is a heap implemented?



Note

- An existing array of elements can also be converted into a heap in linear time, although the process is complicated.
- Luckily, some major programming languages have built-in methods to do this.
- Remember: you shouldn't worry too much about how heaps are implemented. The important thing is that you understand the interface.

Note

- In many problems, using a heap can improve an algorithm's time complexity from $O(n^2)$ to $O(n \log n)$, which is a massive improvement (for n = 1,000,000, this is 50,000 times faster).
- A heap is a great option whenever you need to find the maximum or minimum of something repeatedly.

Interface guide

- Note: some languages implement a min heap by default, while some implement a max heap by default.
- If you're dealing with numbers and you want to deal with the opposite type of heap that your language implements, an easy way to do this is to multiply all numbers by -1

Interface guide

```
// In Java, we will use the PriorityQueue interface and the
// PriorityQueue implementation. By default, this implements
// a min heap
PriorityQueue<Integer> heap = new PriorityQueue<>();

// Add to heap
heap.add(1);
heap.add(2);
heap.add(3);
```

Interface guide

```
// Check minimum element
heap.peek(); // 1

// Pop minimum element
heap.remove(); // 1

// Get size
heap.size(); // 2

// Bonus: if you want a max heap instead, you can pass
// Comparator.reverseOrder() to the constructor:
PriorityQueue<Integer> maxHeap = new
PriorityQueue<>(Comparator.reverseOrder());
```

Heap Examples

- A heap is an amazing tool whenever you need to repeatedly find the maximum or minimum element.

Last Stone Weight

- You are given an array of integers stones where stones[i] is the weight of the i^{th} stone.
- On each turn, we choose the heaviest two stones and smash them together. Suppose the heaviest two stones have weights x and y with x <= y.
- If x == y, then both stones are destroyed.
- If x != y, then x is destroyed and y loses x weight.
- Return the weight of the last remaining stone, or 0 if there are no stones left.

Example

- Input:
  - stones = [2, 7, 4, 1, 8, 1]
- Output:
  - 1

Minimum Operations to Halve Array Sum

- You are given an array nums of positive integers.
- In one operation, you can choose any number from nums and reduce it to exactly half the number.
- Return the minimum number of operations to reduce the sum of nums by at least half.

Example

- Input:
  - nums = [5, 19, 8, 1]
- Output:
  - 3

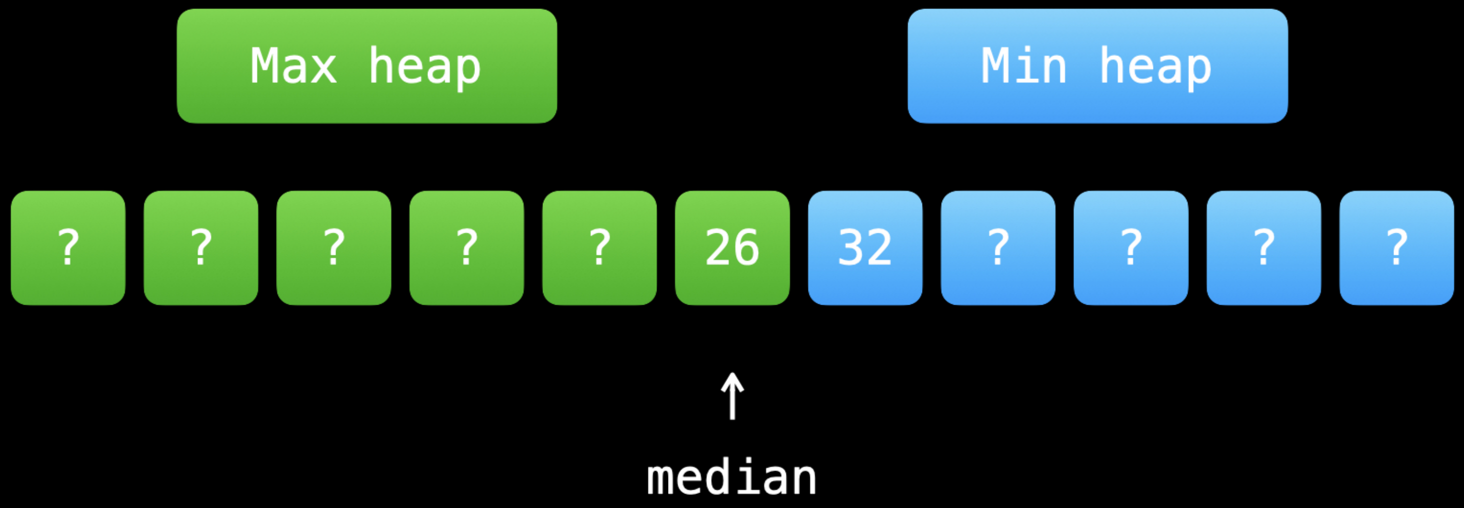
Two heaps

- Using multiple heaps is uncommon and the problems that require it are generally on the harder side.
- If a problem involves finding a median, it's a good thing to think about.

Find Median from Data Stream

- The median is the middle value in an ordered integer list.
- If the size of the list is even, the median is the average of the two middle values. Implement the MedianFinder class:
- MedianFinder() initializes the MedianFinder object.
- void addNum(int num) adds the integer num to the data structure.
- double findMedian() returns the median of all elements so far.

Example



Elements not at the top of the heap are reprensented as ? as their exact values are not important for the purpose of the demonstration.

All "?" in the min heap is greater than or equal to 32
All "?" in the max heap is less than or equal to 26

Code

```
class MedianFinder {
    private PriorityQueue<Integer> minHeap = new PriorityQueue<>();
    private PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Comparator.reverseOrder());;

    public void addNum(int num) {
        maxHeap.add(num);
        minHeap.add(maxHeap.remove());
        if (minHeap.size() > maxHeap.size()) {
            maxHeap.add(minHeap.remove());
        }
    }

    public double findMedian() {
        if (maxHeap.size() > minHeap.size()) {
            return maxHeap.peek();
        }

        return (minHeap.peek() + maxHeap.peek()) / 2.0;
    }
}
```

Queries?

Thank You…!