Data Structures

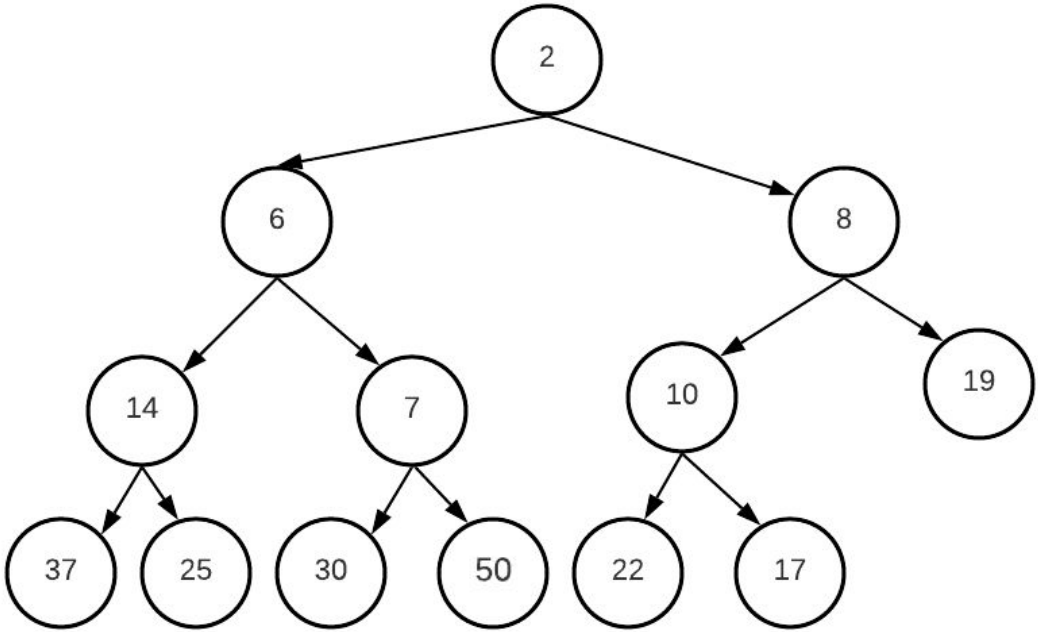
Heap Deletion

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

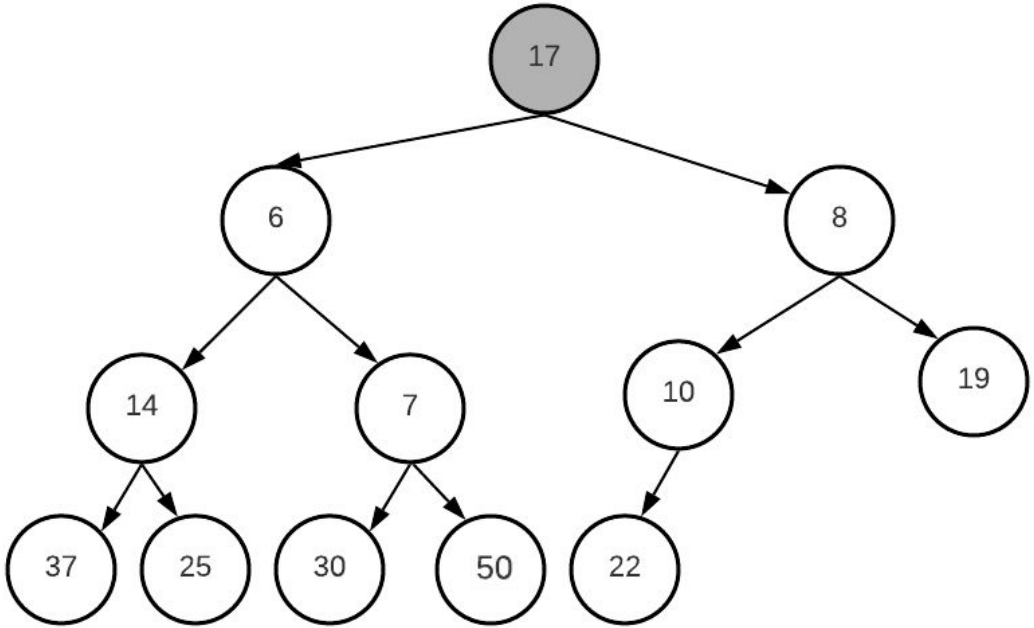
Let’s delete the minimum

- Assume we want to remove the minimum element, update the tree to remain heap
- Root = min, so can get it
- Let’s follow again **update and fix** approach
- We will get the last node (17) and make it the temp root
  - Then fix the tree
  - How can we push 17 to right place?
  - Think for 10 minutes



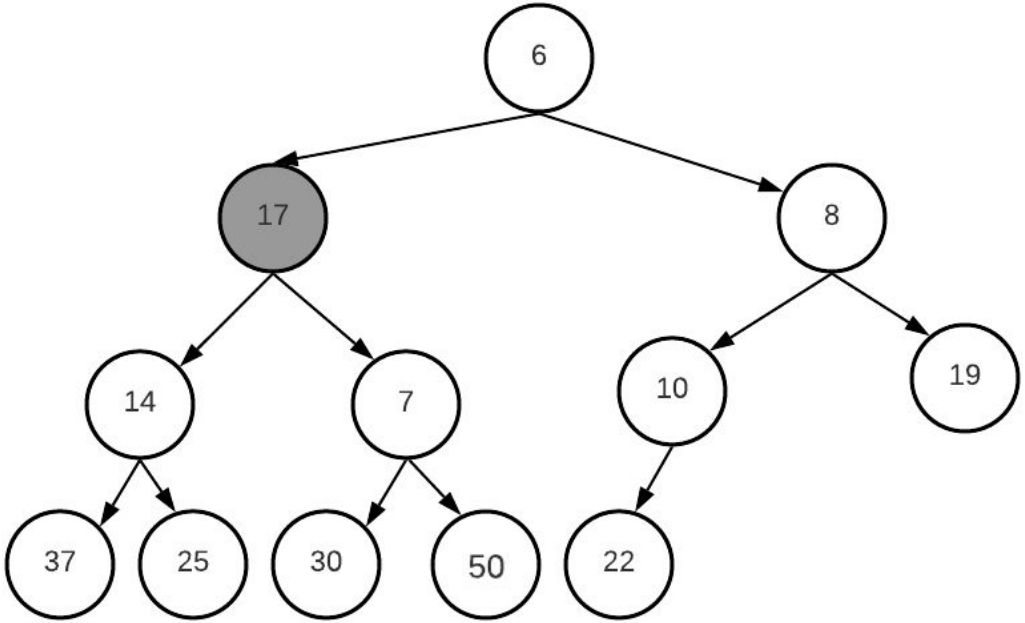
Fix the tree

- We want to move 17 to its right location
- Think level by level
- 17 as parent for (6, 8)
- How to fix?
- The min among them must be root
- In other words, swap 17 with the minimum child (6)



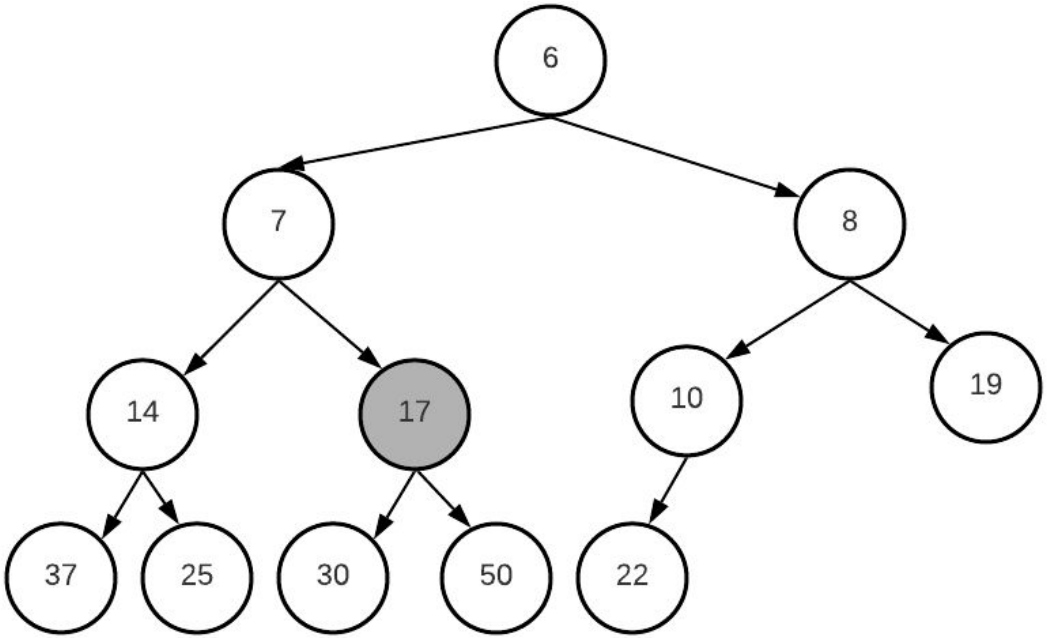
Fix the tree

- 17 as parent for (14, 7)
- 17 again can’t be parent
- min(7, 14) should be the new parent
- Swap 17 with 7

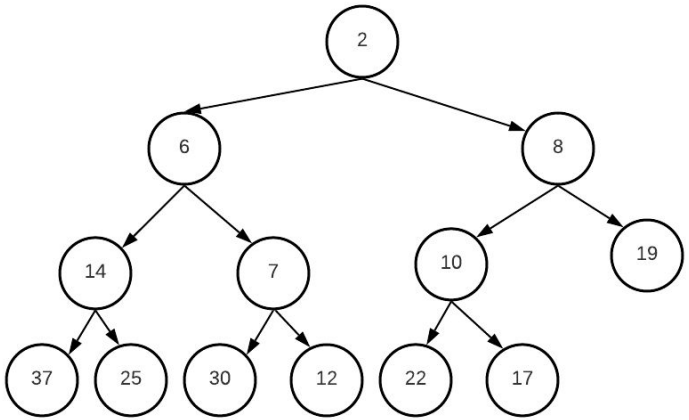


Fix the tree

- 17 as parent for (30, 50)
- This time 17 is in a perfect position: We can **stop**
- This is called **heapify-down**
- It means take invalid value and move it down to its right place

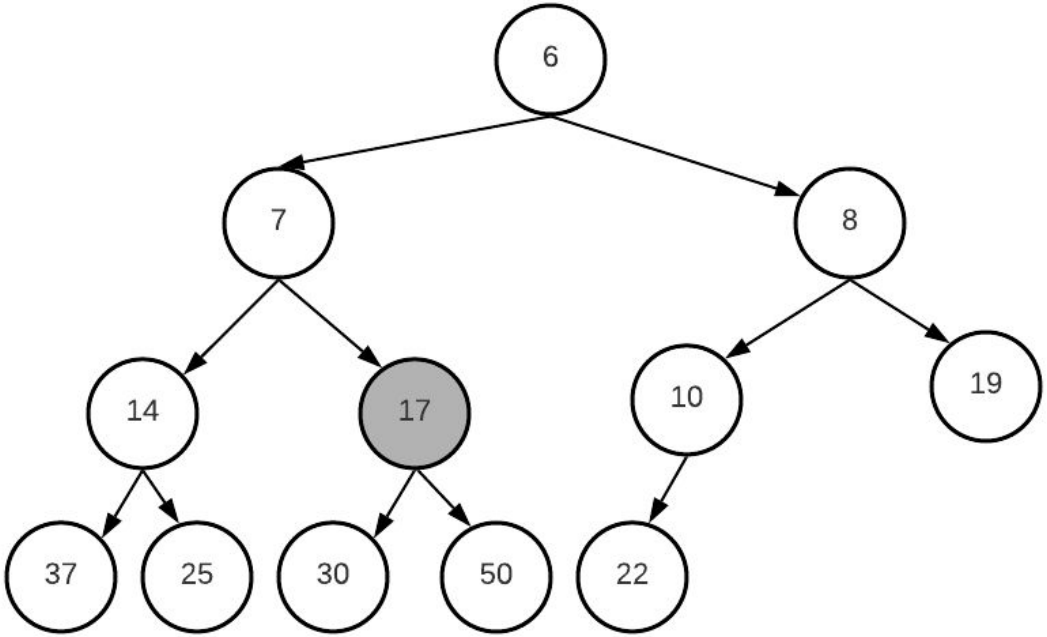


```
void heapify_down(int parent_pos) { // O(logn)
    int child_pos = left(parent_pos);
    int right_child = right(parent_pos);

    if (child_pos == -1) // no children
        return;
    // is right smaller than left?
    if (right_child != -1 && array[right_child] < array[child_pos])
        child_pos = right_child;

    if (array[parent_pos] > array[child_pos]) {
        swap(array[parent_pos], array[child_pos]);
        heapify_down(child_pos);
    }
}

void pop() {
    assert(!isempty());
    array[0] = array[--size];
    heapify_down(0);
}
```



Let’s remove all elements

- Now we got
  - Value 2 (min)
  - Fixed tree
- Imagine, we kept popping all elements until heap is empty
- Can you guess the output?



Let’s remove all elements

```
85 void creat_heap_nlogn() {
86     MinHeap heap;
87
88     vector<int> v { 2, 17, 22, 10, 8, 37,
89         14, 19, 7, 6, 5, 12, 25, 30 };
90
91     for (int i = 0; i < v.size(); ++i)
92         heap.push(v[i]);
93
94     while (!heap.isempty()) {
95         cout << heap.top() << " ";
96         heap.pop();
97     }
98     // 2 5 6 7 8 10 12 14 17 19 22 25 30 37
99 }
```

Heap sort

- You will get the array content but **sorted from small to large**!
  - Or large to small in max-heap
- This is called heap sort!
  - It can be done in-place. See homework
- To sort data, we add them to heap and then get them all
  - Loop to add O(n)
    - Push is O(logn)
  - Loop to remove O(n)
    - Pop is O(logn)
- Total is O(nlogn) for sorting n numbers!

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."