Implementing a Binary Heap: Takeaways

Syntax

- Implementing a min-heap:

```
class MinHeap:
    def __init__(self):
        self.values = []
    def _left_child(self, node):
        return 2 * node + 1
    def _right_child(self, node):
        return 2 * node + 2
    def _parent(self, node):
        return (node - 1) // 2
    def _swap(self, node1, node2):
        tmp = self.values[node1]
        self.values[node1] = self.values[node2]
        self.values[node2] = tmp
    def add(self, value):
        self.values.append(value)
        self._heapify_up(len(self.values) - 1)
    def _heapify_up(self, node):
        parent = self._parent(node)
        if node > 0 and self.values[parent] > self.values[node]:
            self._swap(node, parent)
            self._heapify_up(parent)
    def min_value(self):
        return self.values[0]
    def pop(self):
        self._swap(0, len(self.values) - 1)
        ret_value = self.values.pop()
        self._heapify_down(0)
        return ret_value
    def _heapify_down(self, node):
        left_child = self._left_child(node)
        right_child = self._right_child(node)
        min_node = node
        if left_child < len(self.values) and self.values[left_child] < self.values[node]:
            min_node = left_child
        if right_child < len(self.values) and self.values[right_child] <
self.values[min_node]:
            min_node = right_child
        if min_node != node:
            self._swap(node, min_node)
            self._heapify_down(min_node)
```

Concepts

- A heap is a binary tree in which every level except for the last one is full.
- A min-heap is a heap where the value of each node is smaller than or equal to the value of its children.
- A binary tree is complete if the tree's levels are filled in except for the last, which has nodes filled in from left to right.
- Heaps have many applications. In data engineering, we can use them to schedule tasks with priorities. In data science, we can use them to calculate order statistics on the data. They have many applications, including calculating the shortest routes in your GPS.
- The time complexity of heap operations is logarithmic in the worst case. This makes a heap very efficient data structure when we need to keep track of minimums (or maximums for a max-heap).

Resources

- Binary Heap
- Heap queue algorithm