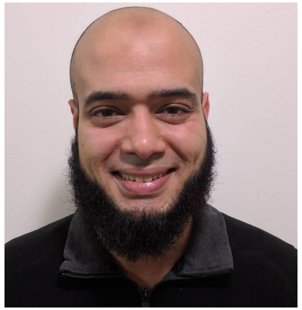
Data Structures

Heap Creation 1

Mostafa S. Ibrahim
Teaching, Training and Coaching for 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)

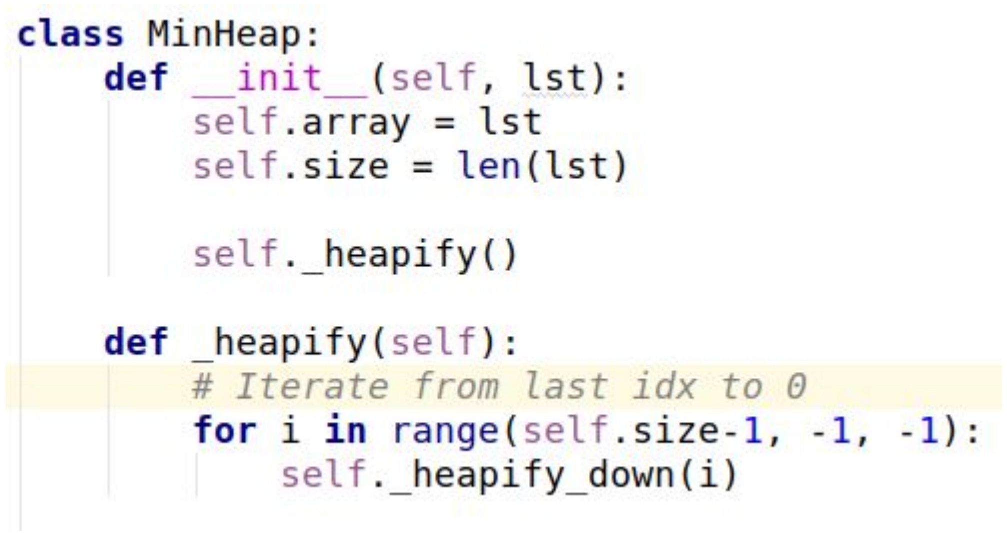


Let’s create a heap from an array

- def __init__(self, lst):
- This initi should take the array and build a heap using it
- We know a simple way to do so:
  - Iterate on elements, and keep pushing using **heapify_up**
  - Correct, but time complexity O(nlogn)
- Floyd described another simple O(n) algorithm that uses heapify_down
  - Build the tree level by level, but starting from the **bottom level**
    - E.g. iterate from the last number to the first number
  - With each number’s index, just fix its sub-tree with **heapify_down**
  - This can (and should!) all be done **in-place**
  - The proof for the time complexity is out of our scope (estimate #comparisons)

Floyd heapfiy algorithm

- Iterate from the end to start and keep fixing it
    - This means lower subtree levels will always be correct after each completed step
- Think about WHY this is correct for 10 minutes



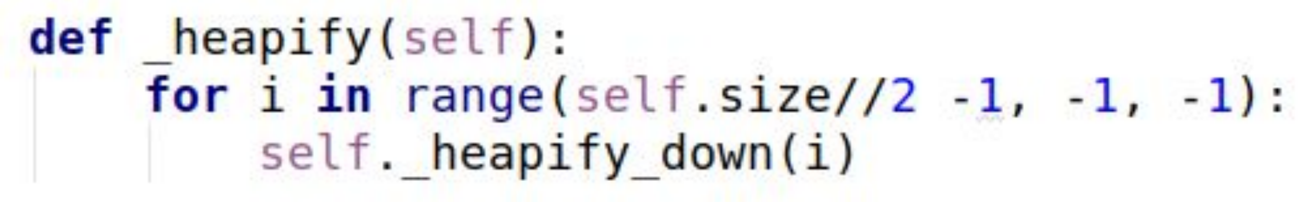
```python
class MinHeap:
    def __init__(self, lst):
        self.array = lst
        self.size = len(lst)

        self._heapify()

    def _heapify(self):
        # Iterate from last idx to 0
        for i in range(self.size-1, -1, -1):
            self._heapify_down(i)
```

Optimization

- Can you think for 5 min why the following code is a proper optimization?
    - The answer is in the next video
    - Hint: leaf vs non-leaf node



```
def _heapify(self):
    for i in range(self.size//2 -1, -1, -1):
        self._heapify_down(i)
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."