Build Priority Queue.

```
self.heap[parent], self.heap[idx] = self.heap[idx], self.heap[parent]
            self.upheapify(parent)

    def remove(self):
        if self.size() == 0:
            return 'Underflow'
        self.heap[0], self.heap[-1] = self.heap[-1], self.heap[0]
        temp = self.heap.pop()
        self.downheapify(0)
        return temp

    def downheapify(self, idx):
        if idx>=self.size():
            return

        leftChild = 2 * idx + 1
        rightChild = 2 * idx + 2
        if leftChild < self.size() and rightChild < self.size():
            if self.heap[leftChild] < self.heap[rightChild] and
self.heap[leftChild] < self.heap[idx]:
                self.heap[leftChild], self.heap[idx] = self.heap[idx],
self.heap[leftChild]
                self.downheapify(leftChild)
            elif self.heap[rightChild] < self.heap[leftChild] and
self.heap[rightChild] < self.heap[idx]:
                self.heap[rightChild], self.heap[idx] = self.heap[idx],
self.heap[rightChild]
                self.downheapify(rightChild)
```

Here add takes logN and remove takes logN as well.

But the below addition takes linear time to heapify an entire array. Without the modification, it would have taken nlogn time to heapify the array.

```
def constantHeapify(self, arr):
        temp = arr
        n = len(temp)
        for i in range(n // 2 - 1, -1, -1):
```

```
            self.downheapify(i)
        self.heap = temp[:]
```