Arrays, Strings & Linked Lists Lecture 11

Wednesday, 14 August 2024 6:21 AM

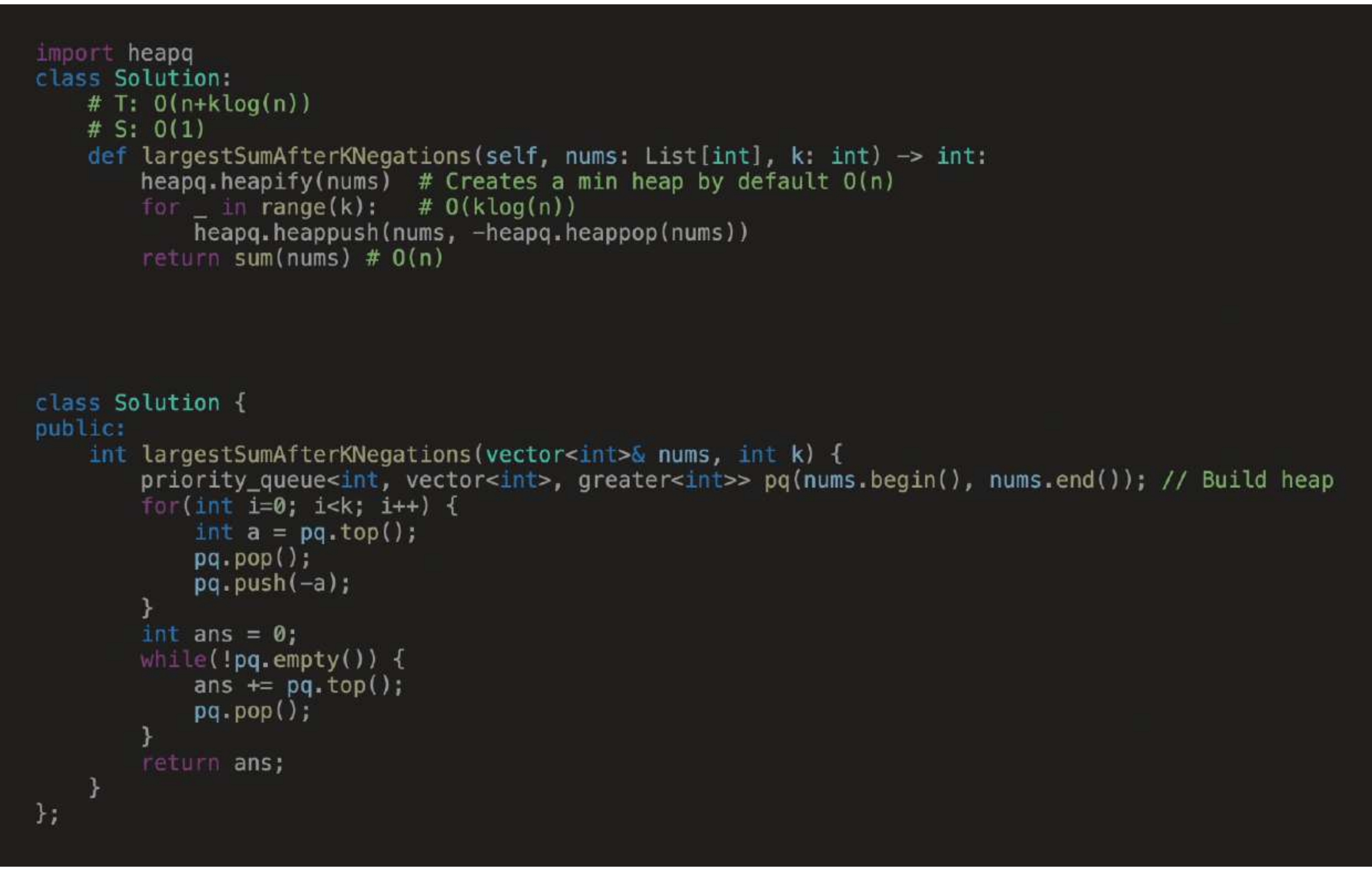
https://leetcode.com/problems/maximize-sum-of-array-after-k-negations/

Greedy step:- k times

Min Heap ← {
- Find the min element in the array
- change its sign
- Push it back to the array

- Build Heap(n) → $O(n)$
- k times, top(), pop(), push() → $\log(n)$
- Pop all elements & return their sum → $n\log(n)$

$$\boxed{T = O(n\log n)}$$



```python
import heapq
class Solution:
    # T: O(n+klog(n))
    # S: O(1)
    def largestSumAfterKNegations(self, nums: List[int], k: int) -> int:
        heapq.heapify(nums)  # Creates a min heap by default O(n)
        for _ in range(k):   # O(klog(n))
            heapq.heappush(nums, -heapq.heappop(nums))
        return sum(nums) # O(n)
```

```cpp
class Solution {
public:
    int largestSumAfterKNegations(vector<int>& nums, int k) {
        priority_queue<int, vector<int>, greater<int>> pq(nums.begin(), nums.end()); // Build heap
        for(int i=0; i<k; i++) {
            int a = pq.top();
            pq.pop();
            pq.push(-a);
        }
        int ans = 0;
        while(!pq.empty()) {
            ans += pq.top();
            pq.pop();
        }
        return ans;
    }
};
```

https://leetcode.com/problems/candy/

$[\boxed{3}, \boxed{2}, 1, 1, 2, 2, 2, 3, \boxed{4}, \boxed{5}, \overset{\downarrow}{\boxed{6}}, \boxed{5}, \boxed{1}, \boxed{1}, 4]$

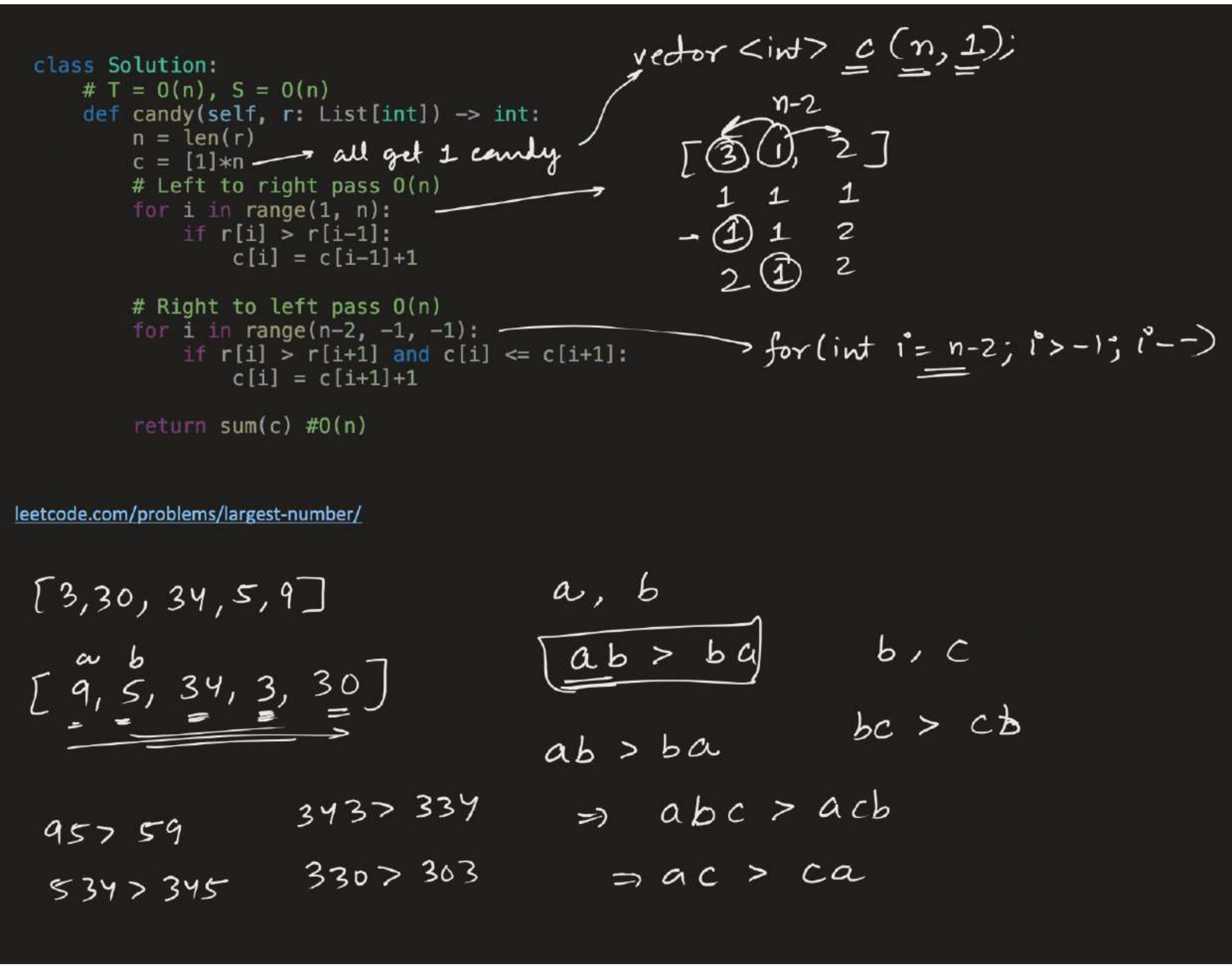
$\underline{1}, 1, 1, 1, 2, 1, \underline{1, 2, 3, 4, \boxed{5}}, 1, 1, 1, 2$ $\downarrow$ Do not decrease # candies

$3, \underline{2}, 1, 1, 2, 1, 1, 2, 3, 4, 5, \boxed{2}, 1, 1, 2$

```
Left → Right
    if r[i+1] > r[i]
        c[i+1] = c[i]+1

Right → left
    if r[i-1] > r[i] and c[i-1] < c[i]
        c[i-1] = c[i]+1

return sum(c)
```



```
class Solution:
    # T = O(n), S = O(n)
    def candy(self, r: List[int]) -> int:
        n = len(r)
        c = [1]*n
        # Left to right pass O(n)
        for i in range(1, n):
            if r[i] > r[i-1]:
                c[i] = c[i-1]+1

        # Right to left pass O(n)
        for i in range(n-2, -1, -1):
            if r[i] > r[i+1] and c[i] <= c[i+1]:
                c[i] = c[i+1]+1

        return sum(c) #O(n)
```

all get 1 candy

vector <int> c (n, 1);

n-2

[3, 1, 2]

1 1 1

1 1 2

2 1 2

for(int i= n-2; i>-1; i--)

leetcode.com/problems/largest-number/

[3, 30, 34, 5, 9]

$[\overset{a}{9}, \overset{b}{5}, 34, 3, 30]$

a, b

$\boxed{ab > ba}$

ab > ba

b, c

bc > cb

95 > 59

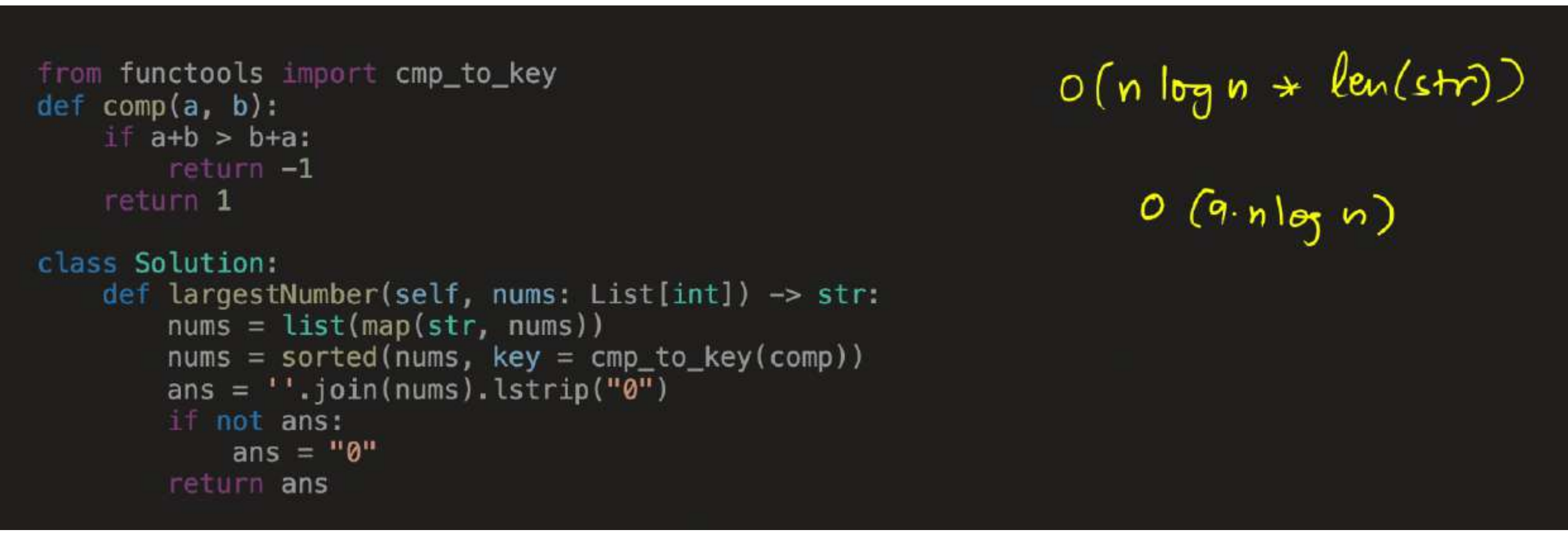
534 > 345

343 > 334

330 > 303

$\Rightarrow$ abc > acb

$\Rightarrow$ ac > ca



```python
from functools import cmp_to_key
def comp(a, b):
    if a+b > b+a:
        return -1
    return 1

class Solution:
    def largestNumber(self, nums: List[int]) -> str:
        nums = list(map(str, nums))
        nums = sorted(nums, key = cmp_to_key(comp))
        ans = ''.join(nums).lstrip("0")
        if not ans:
            ans = "0"
        return ans
```

$O(n \log n * \text{len}(\text{str}))$

$O(9 \cdot n \log n)$