12/11/20

Leetcode:

1. Array-based
2. Link-list

Review + new problem

* **New\_set:**
* for i,n in enumerate(**itertools.accumulate**(nums)

**1054: Distant Barcode**

**Input:** barcodes = [1,1,1,1,2,2,3,3]

**Output:** [1,3,1,3,1,2,1,2]

Rearrange the barcodes so that no two adjacent barcodes are equal.

1. Use heapq as **priority Q**.
2. 229. Bulls and Cows

**Input:** secret = "1807", guess = "7810"

**Output:** "1A3B"

Use collections.Counter()

Two Linear Scans

1. **300. Longest Increasing Subsequence**

Simply dp.

dp = [1] \* n

max\_len = 1

# Recursive back-track

**for i in range(n):**

**for j in range(i):**

**if nums[i] > nums[j]:**

len\_new = dp[j]+1

dp[i] = len\_new if len\_new > dp[i] else dp[i]

max\_len = max(max\_len, dp[i])

return max\_len

* + 1. **1546. Maximum Number of Non-Overlapping Subarrays With Sum Equals Target**

Use cumulative sum as an array, find diff between elements

**DP solution**

def maxNonOverlapping(self, nums: List[int], target: int) -> int:

# 5:38 11/2/20

def maxNonOverlapping(self, nums: List[int], **target**: int) -> int:

# 5:38 11/2/20

n = len(nums)

sums = [0]\*n

**res = [0] \* n**

max\_len = 0

**dic = {}**

**# cumulative SUM**

**for i in range(n):**

**sums[i] = sums[i-1] + nums[i] if i > 0 else nums[i]**

**for i, v in enumerate(sums):**

**val = v - target**

if val in dic:

res[i] = res[dic[val]] + 1

elif val == 0:

res[i] = 1

**dic[v] = i**

if i > 0:

res[i] = max(res[i], res[i-1])

return max(res)

* + 1. Pre-fix sum

out = 0

indexes = {0:-1}

lastIndexFoundAt = float("-inf")

for i,n in enumerate(itertools.accumulate(nums)):

prev = n-target

if prev in indexes and indexes[prev] >= lastIndexFoundAt:

lastIndexFoundAt = i

out+=1

indexes[n]=i

return out

* + 1. Greedy, best soln

dic = {0:1}

cnt = 0

cur\_sum = 0

for num in nums:

cur\_sum += num

prev\_sum = cur\_sum - target

if prev\_sum in dic:

cnt += 1

dic = {0:1}

cur\_sum = 0

else:

dic[cur\_sum] = 1

return cnt