
zscaler™
Experience your world, secured.™

HackShare stock. The PnL of his portfolio is represented by an array *pnl* where *pnl[i]* represents the profit earned in the i^{th} month that can possibly be negative indicating a loss.

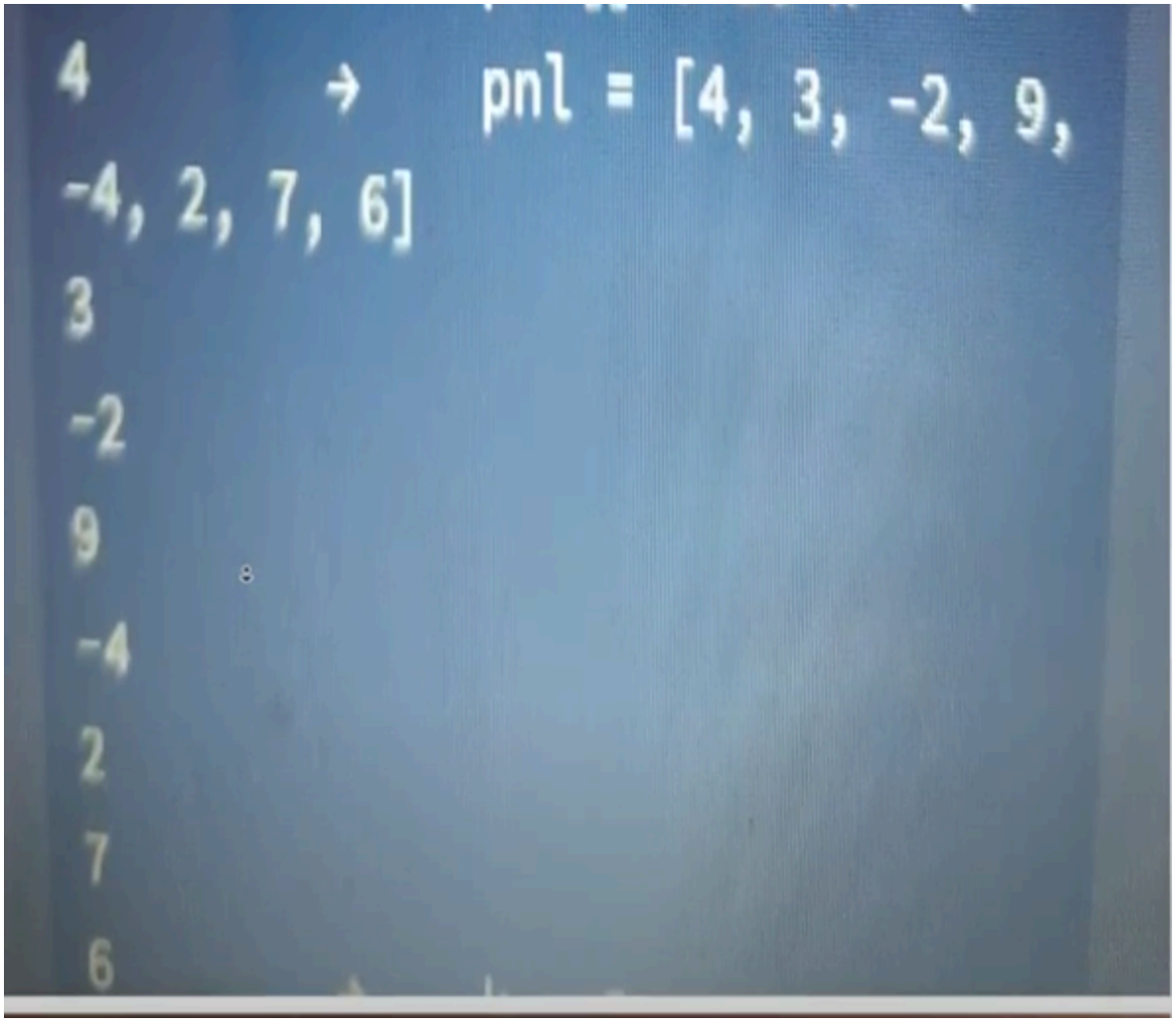
Given the *pnl* array of *n* months, find the maximum net profit (sum of profits) gained in any contiguous segment of months i.e. a subarray of months such that the number of

earned in the i^{th} month that can possibly be negative indicating a loss.

Given the *pnl* array of *n* months, find the maximum net profit (sum of profits) gained in any contiguous segment of months i.e. a subarray of months such that the number of months in the segment does not exceed a given integer *k*.

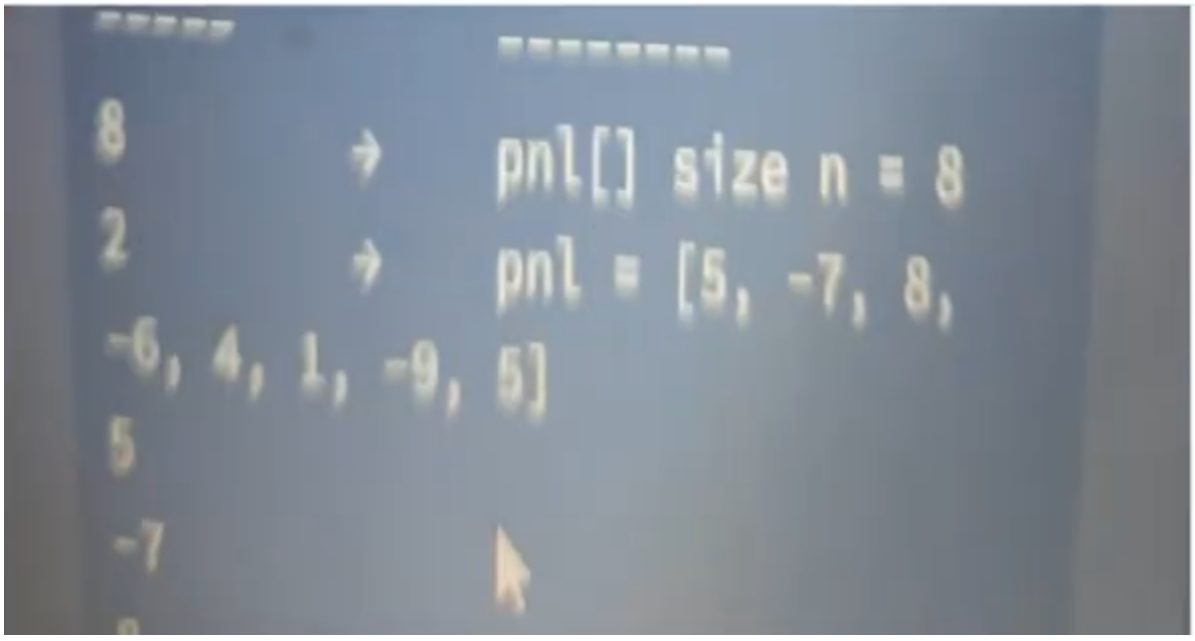
Given $n = 6$, $prf = [-3, 4, 3, -2, 2, 5]$, $k = 4$.

We can select the subarray [3, -2, 2, 5] with a total profit $3 + (-2) + 2 + 5 = 8$. Note that while the maximum net profit of a subarray can be 12 for the subarray [4, 3, -2, 2, 5] but the length of the subarray will exceed $k = 4$.


4 → pnl = [4, 3, -2, 9,
-4, 2, 7, 6]
3
-2
9
-4
2
7
6

Given $n = 7$, $pnl = [4, 3, -2, 9, -4, 2, 7]$, $k = 6$.
We can select the subarray [3, -2, 9, -4, 2, 7] with sum = 15 and size 6 which is equal to $k = 6$.

Hence, the answer is 15. It can be shown that the answer cannot be greater than 15.



Given an array of size “N”; find the largest sum subarray in it but please make sure its size <=K

Brute force :- Find sum of all subarrays whose size <=K -> print maximum of all those sums.

TC - O(N*N)

->n,k

->b[n]

```
sum=0
max_sum = INT_MIN;
for(i=0;i<n;i++){
   sum=0
   for(j=i;j<n;j++){
      //[i.....j]
      sum = sum + b[j]
      size = abs(i-j) + 1
      if(size<=K){
      max_sum=max(max_sum,sum)
```

```
    }
    }
}
print(max_sum)

//RRRRR
```

Let's think in the terms of prefix sum:-

p[i] = sum of all numbers till index "i"

I am only interested in the subarray which is ending at index "j"

So which is the largest sum subarray of size <=K which is ending at index "j".

[i……j]
sum = p[j] - p[i-1]

-> J is fixed and p[j] is fixed; so minimizing p[i-1] will give the maximum sum subarray ending at index "j"

Sum = P8 - min(p7,p6,p5,p4,p3)

If you do this for each index "j"; your job is done.

C++ - https://ideone.com/VEHWf8
Java - https://ideone.com/Mzw2KQ
Py - https://ideone.com/qxQpZn

TC - O(N*K)
O(N) size for the p array

We are always trying to find a minimum of k continuous numbers. Basically for each sliding window of size "k" you have to find the minimum number in it

You can do it using a priority queue.

Strong IntuItion :- When you remove an element from priority queue:-> It is guaranteed that it can never become the correct answer for now and for future.

C++ - https://ideone.com/zIIUGn
Java - https://ideone.com/UDVUy4

Py - https://ideone.com/eO14H5