DSA Problems

Agenda

- Problem 1: Pick From Both Sides
- Problem 2: Alternative Subarrays
- Problem 3: Leaders in the Array
- Problem 4: Bulbs
-
- Problem 5: Even Subarrays

Problem 1: Pick From Both Sides

You are given an integer array A of size N.

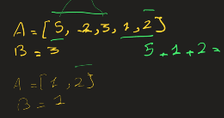
You have to pick B elements in total. Some (possibly 0) elements from left end of array A and some (possibly 0) from the right end of array A to get the maximum sum.
Find and return this maximum possible sum.

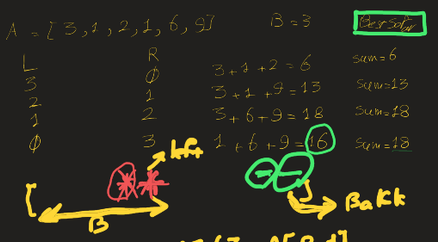
NOTE: Suppose B = 4, and array A contains 10 elements, then

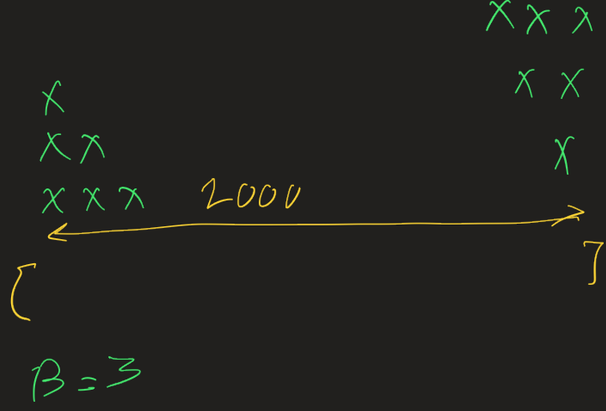
You can pick the first four elements or can pick the last four elements, or can pick 1 from front and 3 from the back, etc. You need to return the maximum possible sum of elements you can pick.

Problem Constraints

- 1 <= N <= 105
- 1 <= B <= N
- -103 <= A[i] <= 103







cur = sum A[0] : A[B-1]

cur = cur - A[B-1] + A[Back]

Cur = cur - A[B-2] + A[Back-1]

Code

```
int PickFromBothSides(vector<int> & A, int B)
    int n = A.size()
    int cur = 0                          int ans = Min.INT
    // Pick all from left
    for(int i=0; i<B; i++)
      cur += A[i];
    }
    int left = B-1
      ans = max(ans, cur)
  * for(back = n-1; back >= n-0; back--)
    {
       cur = cur + A[back] - A[left]
       left --;
       ans = max(ans, cur)
    }
    return ans
}
```

remove one element from left and add one element from right

TC : O(B)

SC : O(1)

Problem 2: Alternative Subarrays

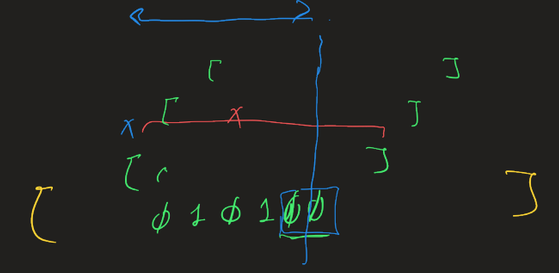
Problem Description

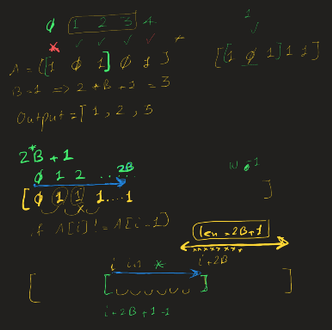
You are given an integer array A of length N comprising of 0's & 1's, and an integer B.You have to tell all the indices of array A that can act as a center of 2 * B + 1 length 0-1 alternating subarray.

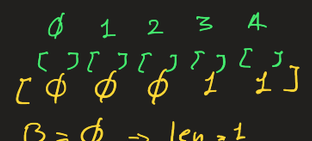
A 0-1 alternating array is an array containing only 0's & 1's, and having no adjacent 0's or 1's. For e.g. arrays [0, 1, 0, 1], [1, 0] and [1] are 0-1 alternating, while [1, 1] and [0, 1, 0, 0, 1] are not.

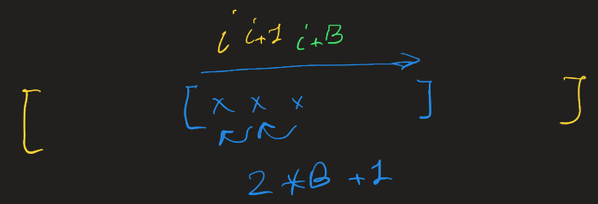
Problem Constraints

1 <= N <= 103
A[i] equals to 0 or 1.
0 <= B <= (N - 1) / 2









```
Code
Vector<int> Solution(Vector<int>& A, int B)

Vector result
int n = A.size()
int len = 2*B+1
for(int i=0; i<n-len+1; i++){
  int prev = -1;
  int flag = 1;
  //assume that i is the start of subarray
  for(int j=i; j<i+len; j++){
    if( A[j] >= prev){
      flag = 0
      break
    }
    prev = A[j];
  }
  if( flag == 1){
    result.push_back(i+B) // add middle index
  }
}
return result
}
```

len = 1

O(N) · O(B)

TC: O(N*B)

SC: O(1)

Lets take a short break for 5 min

Problem 3: Leaders in the Array

Problem Description

Given an integer array A containing N distinct integers, you have to find all the leaders in array A.

An element is a leader if it is strictly greater than all the elements to its right side.

NOTE: The rightmost element is always a leader.

Problem Constraints

- 1 <= N <= 105
- 1 <= A[i] <= 108

Problem 4: Bulbs

Problem Description

A wire connects N light bulbs. Each bulb has a switch associated with it; however, due to faulty wiring, a switch also changes the state of all the bulbs to the right of the current bulb.

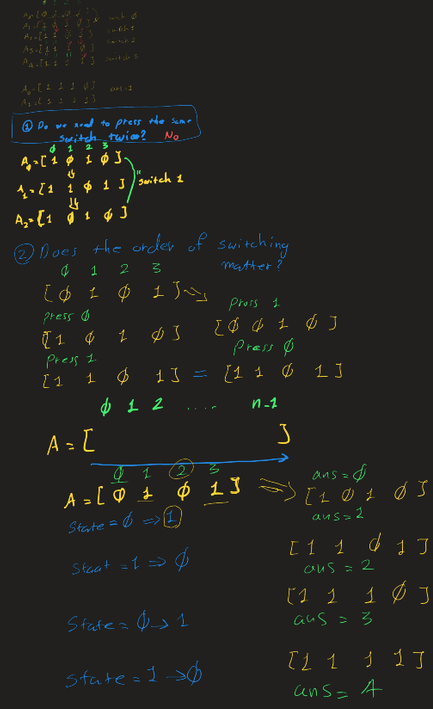
Given an initial state of all bulbs, find the minimum number of switches you have to press to turn on all the bulbs.
You can press the same switch multiple times.

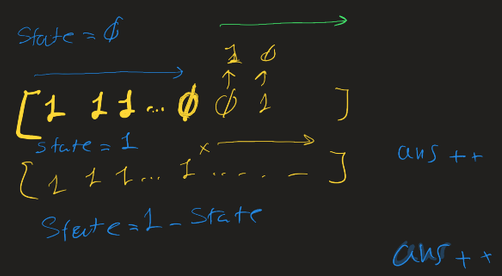
Note: 0 represents the bulb is off and 1 represents the bulb is on.

Problem Constraints

0 <= N <= 5×105

0 <= A[i] <= 1





Code

```
int bulbs(vector<int>& A)
    int ans = 0
    int state = 0
    for(int i=0; i<A.size(); i++) {
        if (A[i] == state) {
            ans++;
            state = 1 - state
        }
    }
    return ans;
}
```

TC: O(N)

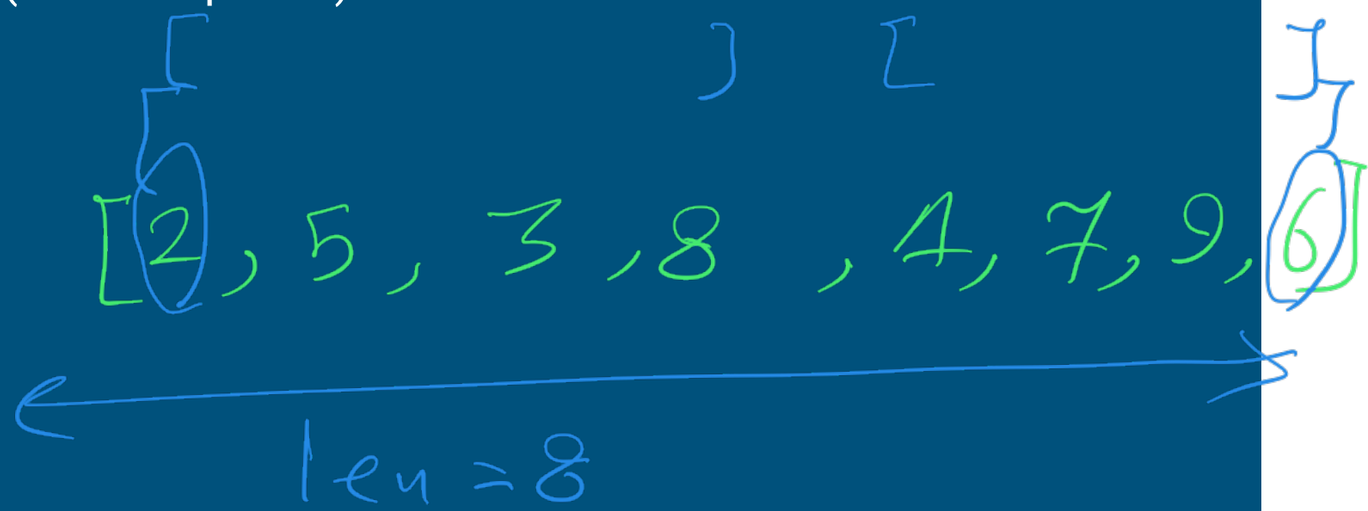
SC: O(1)

Problem 5: Even Subarrays

Problem Description

You are given an integer array A.

Decide whether it is possible to divide the array into one or more subarrays of even length such that the first and last element of all subarrays will be even.
Return "YES" if it is possible; otherwise, return "NO" (without quotes)



Problem Constraints

1 <= |A|, A[i] <= 106

A = [2, 4, 8, 6] Yes
len = 4

A = [2, 4, 8, 7, 6] NO
, len = 5 odd

A = [2, 5, 8] NO

1. [0 0] if A[0] % 2 != 0 || A[N-1] % 2 != 0

2. [<->] odd
(A.size)(N % 2 != 0)

sum(len1, len2, ... lenk) = even
even even even

Code

```
bool isValidSubarray(vector<int>& A)
    int len = A.size()
    if ((A[0] % 2 == 0) &&
        (A[len-1] % 2 == 0) &&
        (len % 2 == 0))
        return True
    return False
```

TC: O(1)
SC: O(1)

[2, 4, 6]

[2, 4, 6] *

[2], [4, 6]

[2, 4], [6]

[2, 4, 5, 6]

len = 4 % 2 = 0

[2 3 5 8]