Bus Dilemma

Need help? Get on call with TA/Helper

Flag Question

Problem Description

A bus travels to **N** different stops, and at each stop some amount of people get in and get out. You are given an array **A**, where integer **A[i]** gives the amount of people who have got onto the bus (if positive), or the amount of people who have gotten off the bus (if negative) at the stop i.

You are also given a positive integer **B**, that denotes the capacity of the bus, that is the maximum number of people the bus can hold. The bus can hold **0** to **B** number of people at any time.

Initially, the bus can have some number of people inside of it, you have to find the total number of possible ways of how many people were initially in the bus before the first stop, such that at any time there are always **0** to **B** number of people in the bus.

If it is not possible to find any valid number of ways, return **0**.

Problem Constraints

$-10^6 <= A[i] <= 10^6$

$1 <= B <= 10^9$

$1 <= |A| <= 1000$

min sum of all = -10^9

max sum of all = 10^9

Int range:

Eg: At any moment Bus should contain [0... B]

	0	1	2	3
A =	2	4	-1	3
P =	2	6	5	8

B = 10

Total ways to start = 3

min = 2, max = 8

mins = 0

mans : B - mans : mans - mins + 1

2 : 10 - 8 : 2 - 0 + 1 = 3

Ini

0 :	2	6	5	8
1 :	3	7	6	9
2 :	4	8	7	10
3 :	5	9	8	11

max of arr[] > 10 Exceeding capacity *

	0	1	2	3
A =	2	-1	3	3
P =	2	1	4	7

B = 10

min = 1, max = 7

mins = 0

maxs : B - max = mans - mins + 1

3 : 10 - 7 = 3 - 0 + 1 = 4

0 1 2 3

A = [-2 -4 5 3] B = 10

P = [-2 -6 -1 2] → min: -6, max: 2

S A = [-2 -4 5 3]

6 : 4 0 5 8

→ mins : -min = 6; maxs : B-max = 10-2 = 8; maxs - mins + 1 = 3

0 1 2 3

A = [-3 1 4 -6] B = 10

P = [-3 -2 2 -4] → min: -4, max: 2

→ mins : -min = 4; maxs : B-max = 8; maxs - mins + 1 = 5

0 1 2 3

A = [2 3 -1 2] B = 10

P = [2 5 4 6] → min: 2, max: 6

→ mins : -min = -2 → 0; maxs : B-max = 4; maxs - mins + 1 = 4-0+1 = 5.

0 1 2 3

A = [-3 1 -4 2] B = 10

P = [-3 -2 -6 -4] → min: -6, max: -2

→ mins : -min = 6; maxs : B-max = 10-(-2) = 12 > limit = 10

0 1 2 3

A = [-4 -2 3 -1] B = 10

P = [-4 -6 -3 -4] → min: -6, max: -3

↳ mins: -min = 6

maxs: B - max = 10 - (-3) = 13 > Limit = 10

0 1 2 3

A = [-2 3 -6 9] B = 4

P = [-2 1 -5 4] → min: -5, max: 4

↳ mins: -mins = 5

maxs: B - max = 0

0 1 2 3

A = [2 3 5 1] B = 6

P = [2 5 10 11] → min: 2, max: 11

↳ mins: 0

maxs: B - max = -5 : 0

```
int Bus(int ar[]){
    int n = ar.length;
    int pf[n];
    pf[0] = ar[0]
    int min = pf[0], man = pf[0];
    i = 1; i < n; i++){
        pf[i] = pf[i-1] + ar[i]
        if( min > pf[i]){
            min = pf[i]
        }
        if( man < pf[i]){
            man = pf[i]
        }
    }
    int mins = -1 * min;
    if( mins < 0 ){
        mins = 0
    }
    int mans = B - man;
    if( mans > B){
        mans = B
    }
    if( mins > B || mans < 0) // We cannot start
        return 0
    return mans - mins + 1
}
```

2 Cases

2 Cases

Max Submatrix Sum

Given a row-wise column wise sorted matrix return max Submatrix Sum

TL BR

Ex:

	0	1	2	3
0	-20	-16	-4	8
1	-10	-8	12	14
2	-1	6	21	30
3	5	7	28	42

→ $-8+12+14+6+21+30+7+28+42 = 152$

Ex2:

	0	1	2	3
0	-20	-16	-4	-1
1	-10	-8	-2	5
2	-4	2	4	8

→ $-2+5+4+8 = 15$

Ex3:

	0	1	2
0	-50	-40	-30
1	-35	-20	-15
2	-19	-14	-3

→ -3

Idea: Say mat[N][M]

obs1: BR of max Submatrix is N-1 M-1

obs2: TL of max Submatr can be anywhere

so, try every cell as TL.

```
int manSum(int mat[][]) {   TC: O(N*M)  SC: O(N*M)
    int ans = INT_MIN
    Construct pfmat[][].  ------>  TC: O(N*M)
    i=0; i<M; i++){  ------>  TC: O(N*M) * 1
        j=0; j<N; j++){
            // Top left: [i,j]  Bottom right: [N-1, M-1]
            // For above submatrix get sum.
            { Using pfmat[][] get the sum }
        }
    }
}
```

Max Subsequence Sum

Given ar[N], calculate sum of max of every subsequence

	0	1	2
A[ ] =	2	5	3

All Sub: → max

{ }	0
{2}	2
{5}	5
{3}	3
{2 5}	5
{5 3}	5
{2 3}	3
{2 5 3}	5

Sum of all sub = 28.

In how many Subseq 2 is max = 1 = 2*1 = 2

In how many Subseq 3 is max = 2 = 3*2 = 6

In how many Subseq 5 is max = 4 = 5*4 = 20

= 28

Idea:

For every number get its contribution & add in final.

= ar[i] * {count of Subsequences in which ar[i] is max}

Ex: ar[ ] = 2 6 4 8 7 3

→ : In how many subseq is 6 max :

{ 2 | 4 | 3 | 6 } = 8 = 2^3

Ex: ar[ ] = 2 6 4 3 5 7 8

→ : In how many subseq is 6 max

{ 2 | 4 | 3 | 5 | 6 }

4 elements → Sub: $2^4 = 16$

Ex: ar[] = | -3 | 6 | 7 | 2 | 10 |

→ : In how many subseq is 6 max

{ -3 , 2 , 6 }

2 ele ⟶ Sub: $2^2 = 4$

Idea:

For every number get its contribution & add in final.

= ar[i] * { count of subsequences in which ar[i] is max }

{ get count of ele < ar[i] in arr[] = x }

{ count of Subseq in which ar[i] max = 2^x }

Solve: For every ele get count of ele < ar[i] ?

ar[5] = { 10 3 5 -1 6 }
(0 1 2 3 4)

arr.sort() = { -1 3 5 6 10 }
in inc (0 1 2 3 4)

ele < = 0 1 2 3 4

ar[5] = { 6 11 2 4 12 }
(0 1 2 3 4)

arr.sort() = { 2 4 6 11 12 }
(0 1 2 3 4)

ele < = 0 1 2 3 4

```
int manSub(int ar[])
    Arrays.sort(ar) // sort given ar[] in inc order
    int ans=0;
    for(i=0; i<n; i++){
        // ele = ar[i]
        int less = i; // ele less than ar[i] = i
        int count= 1<<i {2^i} // in how many sub ar[i] is max
        int contri = count*ar[i]
        ans = ans+contri
    }
    return ans;
}
```

Dry run:

	0	1	2
A[] =	2	3	2

According to Code: n = 3

2	2	3

{ }	0
{2}	2
{3}	3
{2}	2
{2 3}	3
{3 2}	3
{2 2}	2
{2 3 2}	3

max : 18

ans = 0

for i = 0; i < 3; i++ ans = ans + 2^i * ar[i]

i = 0: ans = ans + 2^0 * 2 = ans = 2

i = 1: ans = ans + 2^1 * 3 = ans = 6

i = 2: ans = ans + 2^2 * 3 = ans = 18

- Sum of max of every Sub: ✓
- Sum of min of every Sub: TODO

Sum of (max - min) in Sub: Sum of max of Subseq - Sum of min of Subseq ?