



香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Optimization

Christy Cheng {christycty}

2022-02-12

Motivation

C++11	Time Limit Exceeded
C++11	Time Limit Exceeded
C++11	Time Limit Exceeded
C++11	Time Limit Exceeded
C++11	Time Limit Exceeded
C++11	Time Limit Exceeded
C++11	Time Limit Exceeded
C++11	Time Limit Exceeded
C++11	Time Limit Exceeded

➤ We need to reduce time complexity by **avoiding repeated computation**

some rule-of-thumb in OI:

$O(N!)$ → $N \leq 10$

$O(2^N)$ → $N \leq 20$

$O(N^2)$ → $N \leq 5000$

$O(N \lg N)$ → $N \leq 10^6$

$O(N)$ → $N \leq 10^7$

Task List

Explained in lecture

S164 Alice's Meal

T062 Tappy World

M0652 Museum

T042 Game of Life IV

Extra Exercises

20108 Maximum Sum

I0111 Mobile Phones

J051 Pattern Matching

J134 Lucky Rainbow

J144 Fair Santa Claus

M1613 Fair Patrol

M1513 Advanced
Optimization Coding

S032 Project

S072 Partners

S113 Gravity Game

S152 Apple Garden

S211 Skyscraperhenge

**Feel free to head to the advanced session if
you know how to solve all these!**

Outline

- ❑ Partial Sum
- ❑ Difference Array
- ❑ Precomputation
- ❑ Sliding Window (Two Pointers)
- ❑ Batching Step
- ❑ Discretization

Outline

- ❑ **Partial Sum**
- ❑ Difference Array
- ❑ Precomputation
- ❑ Sliding Window (Two Pointers)
- ❑ Batching Step
- ❑ Discretization

Problem

Given an array of N integers and Q queries.

For each query, find the sum of the L^{th} to R^{th} element (*inclusive*) of the array.

i	1	2	3	4	5	6	7	8	9
a[i]	3	-2	6	1	0	5	-5	2	4

Example: sum of 3rd to 6th element = $a[3] + a[4] + a[5] + a[6]$
 $= 6 + 1 + 0 + 5$
 $= 12$

Naive Solution

```
for i ← 1 to N do  
    input a[i]
```

```
for i ← 1 to Q do  
    input L and R  
    sum = 0  
    for j ← L to R do  
        sum += a[j]  
    output sum
```

What is the problem of this solution?

It is too slow :(

Time complexity $O(NQ)$

(Imagine all queries ask for $a[1]$ to $a[N]$)

Naive Solution

i	1	2	3	4	5	6	7	8	9
a[i]	3	-2	6	1	0	5	-5	2	4



We can see that the sum of $a[4]$ to $a[6]$ is calculated many times...
Can we perform the calculation once only?

1D Partial Sum - Idea

$c[i]$ = sum of the first i elements ($a[1]+a[2]+\dots+a[i]$)

i	1	2	3	4	5	6	7	8	9
$a[i]$	3	-2	6	1	0	5	-5	2	4
$c[i]$	3	1	7	8	8	13	8	10	14

$$\begin{aligned} \text{e.g. } c[3] &= a[1] + a[2] + a[3] \\ &= 3 + (-2) + 6 = 7 \end{aligned}$$

1D Partial Sum - Idea

$c[i] = \text{sum of the first } i \text{ elements } (a[1] + a[2] + \dots + a[i])$

i	1	2	3	4	5	6	7	8	9
a[i]	3	-2	6	1	0	5	-5	2	4
c[i]	3	1	7	8	8	13	8	10	14

Using array c only, how can we find the values of a[4]?

Hint

$$c[4] = a[1] + a[2] + a[3] + a[4]$$

$$c[3] = a[1] + a[2] + a[3]$$

1D Partial Sum - Idea

Using array c only, how can we find the values of $a[4]$?

i	1	2	3	4	5	6	7	8	9
$a[i]$	3	-2	6	1	0	5	-5	2	4
$c[i]$	3	1	7	8	8	13	8	10	14

$$c[4] = a[1] + a[2] + a[3] + a[4]$$

$$\Rightarrow a[4] = c[4] - c[3]$$

$$c[3] = a[1] + a[2] + a[3]$$

We can generalize it to $a[x] = c[x] - c[x-1]$ (*proof left as exercise :P*)

1D Partial Sum - Idea

How can we find the values of $a[L] + a[L+1] + a[L+2]$?

i	1	2	3	4	5	6	7	8	9
a[i]	3	-2	6	1	0	5	-5	2	4
c[i]	3	1	7	8	8	13	8	10	14

Let's use our previous formula $a[x] = c[x] - c[x-1]$

$$a[L] = c[L] - c[L-1]$$

$$a[L+1] = c[L+1] - c[L]$$

$$a[L+2] = c[L+2] - c[L+1]$$

Sum them up and we will get

$$a[L] + a[L+1] + a[L+2]$$

$$= c[L+2] - c[L-1]$$

1D Partial Sum - Idea

What if we would like to find values of $a[L] + a[L+1] + \dots + a[R]$?

i	1	2	3	4	5	6	7	8	9
a[i]	3	-2	6	1	0	5	-5	2	4
c[i]	3	1	7	8	8	13	8	10	14

We obtained: $a[L] + a[L+1] + a[L+2] = c[L+2] - c[L-1]$

Generalize it: $a[L] + a[L+1] + \dots + a[R] = c[R] - c[L-1]$

1D Partial Sum - Idea

Apply $a[L] + \dots + a[R] = c[R] - c[L-1]$, find sum of $a[3]$ to $a[6]$

i	1	2	3	4	5	6	7	8	9
a[i]	3	-2	6	1	0	5	-5	2	4
c[i]	3	1	7	8	8	13	8	10	14

$L = 3, R = 6$

$\text{ans} = c[6] - c[3-1]$
 $= 13 - 1$
 $= 12$

Verify

$a[3] + a[4] + a[5] + a[6]$
 $= 6 + 1 + 0 + 5$
 $= 12$

1D Partial Sum - Implementation

```
for i ← 1 to N do
    c[i] = c[i-1] + a[i]
```

First, compute array c .

Why we use $c[i-1] + a[i]$, instead of $a[1] + a[2] + \dots + a[i]$?

Looping through the first i elements each time is slow! Time complexity $O(N^2)$

Note that $c[i-1] = a[1] + \dots + a[i-1]$,

$$\begin{aligned} c[i] &= a[1] + \dots + a[i-1] + a[i] \\ &= c[i-1] + a[i] \end{aligned}$$

1D Partial Sum - Implementation

```
for i ← 1 to N do  
    c[i] = c[i-1] + a[i]
```

```
for i ← 1 to Q do  
    input L and R  
    output c[R]-c[L-1]
```

Time Complexity

$O(N)$ partial sum precomputation

$O(1)$ per query \rightarrow total $O(Q)$

$= O(N + Q)$

Reminder:

- Use long long if value of $a[i]$ is large
- Initialize $c[0]$ first
- Be aware of 0/1-based issue

Problem

Now, let's extend everything to 2-dimension

-3	1	2	0	2
2	0	4	-3	1
-1	3	0	1	4
5	-2	1	4	2

Given an array a with N rows and M columns and some queries.

In each query, you need to find the sum of elements from (S_r, S_c) to (E_r, E_c) .

(r, c) = element at r^{th} row and c^{th} column

What is the sum from $(1, 2)$ to $(2, 4)$?

$$1 + 2 + 0 + 0 + 4 + (-3) = 4$$

Naive Solution

```
for i ← 1 to Q do
  input Sr, Sc, Er, Ec
  sum = 0
  for j ← Sr to Er do
    for k ← Sc to Ec do
      sum += a[j][k]
  output sum
```

Loop over the required area in each query
and sum up the numbers

Time Complexity?

$O(NMQ)$

Imagine all queries ask for $(1, 1)$ to (N, M)

1D Partial Sum Solution

Can we applied what we learned just now?

-3	1	2	0	2
2	0	4	-3	1
-1	3	0	1	4
5	-2	1	4	2

1D Partial Sum Solution

Compute a 1D partial sum array for each row

-3	1	2	0	2
2	0	4	-3	1
-1	3	0	1	4
5	-2	1	4	2



-3	-2	0	0	2
2	2	6	3	4
-1	2	2	3	7
5	3	4	8	10

1D Partial Sum Solution

```
for i ← 1 to N do
  for ← 1 to M do
    c[i][j] = c[i][j-1] + a[i][j]

for i ← 1 to Q do
  input Sr, Sc, Er, Ec
  sum = 0
  for j ← Sr to Er do
    sum += c[j][Ec] - c[j][Sc-1]
  output sum
```

Compute array c

- Note that here we arrange the partial sum arrays horizontally (i.e. one row = one array)
- depends on range of N and M, you may arrange it vertically (i.e. one column = one array)

Perform query

1D Partial Sum Solution

```

for i ← 1 to N do
  for ← 1 to M do
    c[i][j] = c[i][j-1] + a[i][j]

for i ← 1 to Q do
  input Sr, Sc, Er, Ec
  sum = 0
  for j ← Sr to Er do
    sum += c[j][Ec] - c[j][Sc-1]
  output sum

```

Time Complexity?

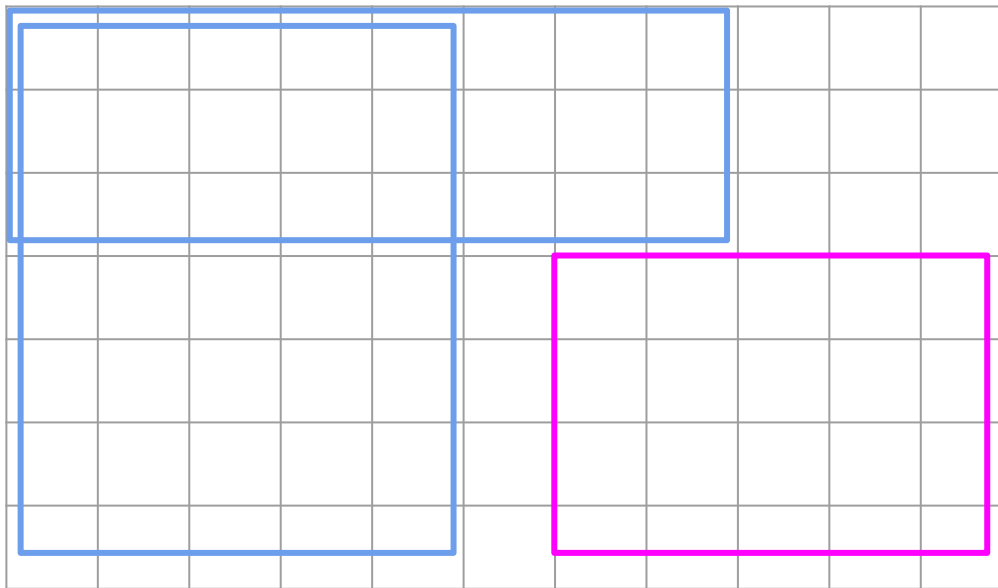
$O(NM)$ precomputation

$O(\min(N, M))$ per query

$= O(NM + \min(N, M) * Q)$

Not bad, but can it be better?

2D Partial Sum - Idea

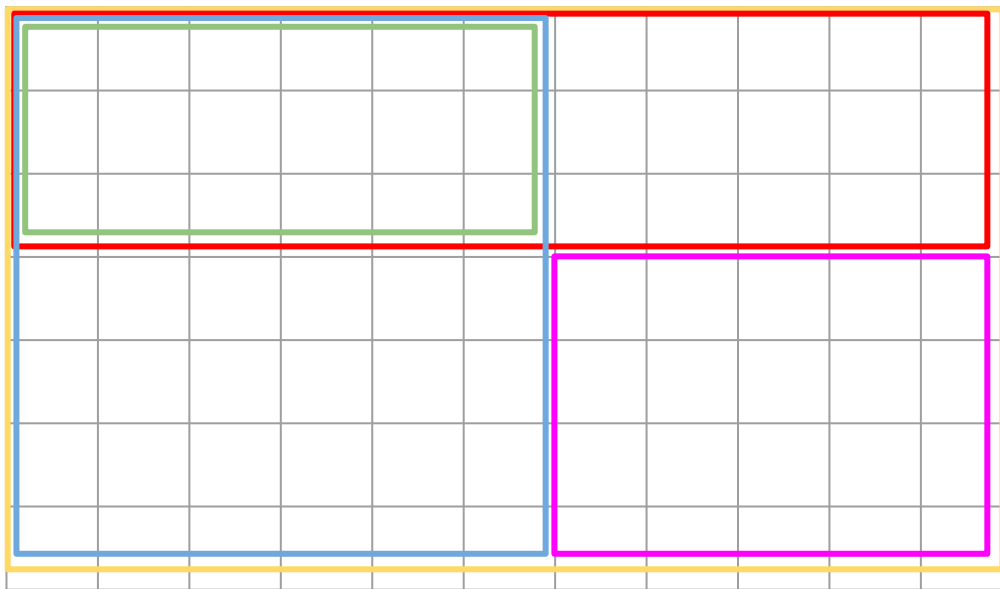


Given that we have area of all rectangles with top-left corner at $(1, 1)$ (such as the blue ones)

How can we find the area of the magenta rectangle?

2D Partial Sum - Idea

Magenta = Yellow - Red - Blue + Green



1	2
3	4

1 + 2 + 3 + 4

- 1 - 2

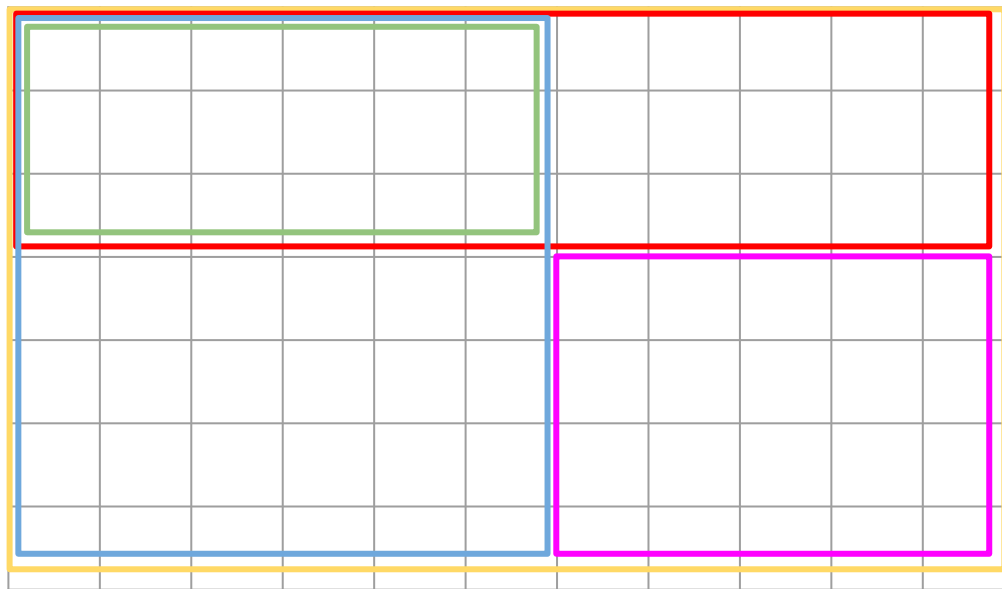
- 1 - 3

+ 1

= 4

2D Partial Sum - Idea

Magenta = **Yellow** - **Red** - **Blue** + **Green**



Let's verify that by calculation!

$$7 * 11 = 77$$

$$3 * 11 = 33$$

$$7 * 6 = 42$$

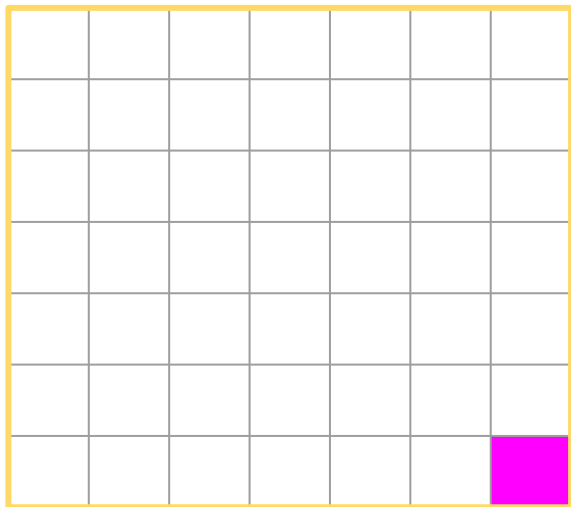
$$3 * 6 = 18$$

$$4 * 5 = 20$$

$$77 - 33 - 42 + 18 = 20$$

2D Partial Sum - Idea

How to precompute the area of each rectangle touching $(1, 1)$?



Given that we have areas of all rectangles touching $(1, 1)$ within the table, except that at (N, M) .

We also have area of magenta grid.

How can we compute area of the table (yellow)?

2D Partial Sum - Idea

How to precompute the area of each rectangle touching (1, 1)?



$$\text{Yellow} = \text{Magenta} + \text{Red} + \text{Blue} - \text{Green}$$

1	2
3	4

$$\begin{aligned}
 &4 \\
 &+ 1 + 2 \\
 &+ 1 + 3 \\
 &- 1 \\
 &= 1 + 2 + 3 + 4
 \end{aligned}$$

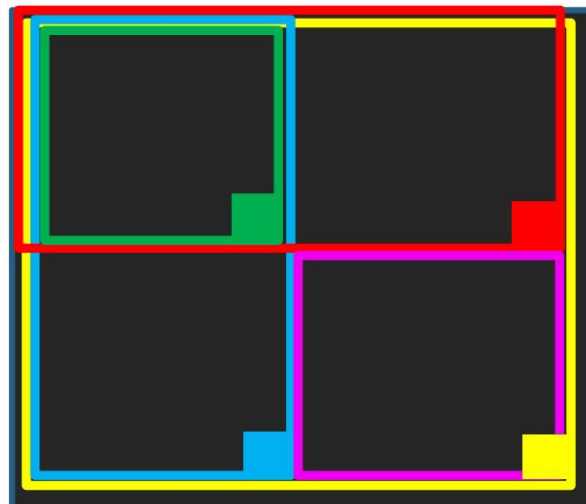
2D Partial Sum - Idea

Or mathematically...

$$\text{Magenta} = \text{Yellow} - \text{Red} - \text{Blue} + \text{Green}$$

after rearranging some terms, we have:

$$\text{Yellow} = \text{Magenta} + \text{Red} + \text{Blue} - \text{Green}$$



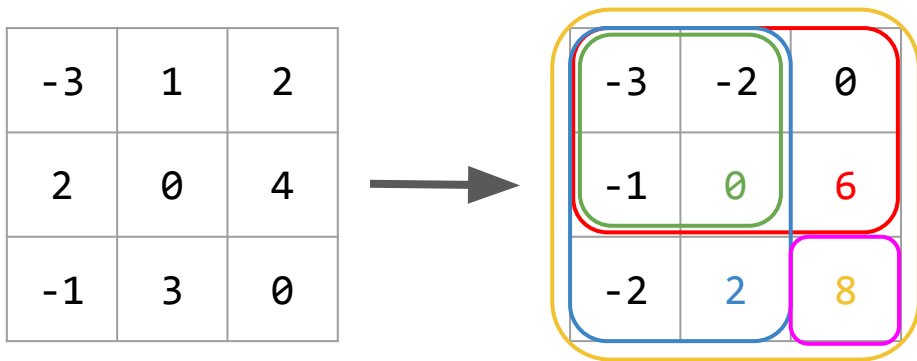
2D Partial Sum - Precomputation

Let $c[i][j]$ = sum of elements from $a[1][1]$ to $a[i][j]$

for $i \leftarrow 1$ to N do

for $j \leftarrow 1$ to M do

$$c[i][j] = a[i][j] + c[i-1][j] + c[i][j-1] - c[i-1][j-1]$$



2D Partial Sum - Query

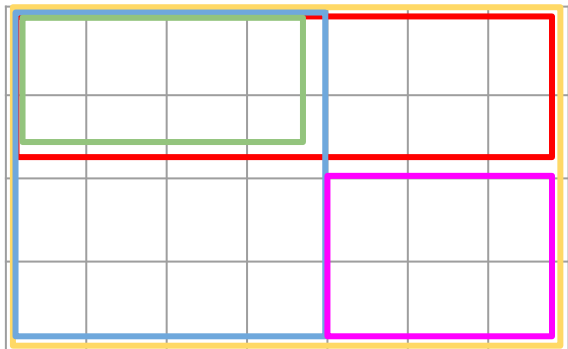
Magenta: (Sr, Sc) to (Er, Ec)

```
for i ← 1 to Q do
```

```
    input Sr, Sc, Er, Ec
```

```
    sum = c[Er][Ec] - c[Sr-1][Ec] - c[Er][Sc-1] + c[Sr-1][Sc-1]
```

```
    output sum
```



2D Partial Sum - Query

Time Complexity?

```
for i ← 1 to Q do
    input Sr, Sc, Er, Ec
    sum = c[Er][Ec] - c[Sr-1][Ec] - c[Er][Sc-1] + c[Sr-1][Sc-1]
    output sum
```

$O(NM)$ precomputation and $O(1)$ per query
= $O(NM + Q)$

2D Partial Sum - Example

Query (2, 3) to (4, 4)

-3	1	2	0	2
2	0	4	-3	1
-1	3	0	1	4
4	2	-2	1	3

2D Partial Sum - Example

Query (2, 3) to (4, 4)

$$\begin{aligned}
 & c[4][4] - c[1][4] - c[4][2] + c[1][2] \\
 &= 11 - 0 - 8 + (-2) \\
 &= 1
 \end{aligned}$$

-3	1	2	0	2
2	0	4	-3	1
-1	3	0	1	4
4	2	-2	1	3

-3	-2	0	0	2
-1	0	6	3	6
-2	2	8	6	13
2	8	12	11	21

2D Partial Sum - Example

Query (2, 2) to (3, 5)

-3	1	2	0	2
2	0	4	-3	1
-1	3	0	1	4
4	2	-2	1	3

2D Partial Sum - Example

Query (2, 2) to (3, 5)

$$\begin{aligned} & c[3][5] - c[1][5] - c[3][1] + c[1][1] \\ &= 13 - 2 - (-2) + (-3) \\ &= 10 \end{aligned}$$

-3	1	2	0	2
2	0	4	-3	1
-1	3	0	1	4
4	2	-2	1	3

-3	-2	0	0	2
-1	0	6	3	6
-2	2	8	6	13
2	8	12	11	21

**attendance taking + 5-minute break
(until 10:37)**

check out 20108 Maximum Sum if you are finding sth to do :)

Example - HKOJ S164 Alice's Meal

After attending Dr. Jones' class, Alice feels very tired and hungry. She goes to a restaurant to have a meal. There are N dishes on the menu. The dishes on the menu are numbered from 1 to N from top to bottom. After reading the whole menu, Alice knows how delicious the dishes are and gives each dish a delicious level. Dish i has a delicious level of D_i .

Since Alice is in a hurry, she has only K minutes to eat, and each minute she can finish exactly one dish. Moreover, the later she eat, the happier she would become, even for the same dish. This is because she is so hungry that she enjoy the first few dishes less. Specifically, if Dish i is eaten during the t^{th} minute, Alice will gain $t \times D_i$ happiness from the dish. The total happiness Alice can gain from the meal is the sum of the happiness from each dish.

Alice is too lazy to order the dishes one by one, so she wants to pick exactly K consecutive dishes in the menu, and eat them according to the menu order from top to bottom. At the same time, she wants to gain as much happiness as possible. For example, there are 5 dishes in the menu, and their delicious level D_1, \dots, D_5 are 2, 4, 1, 5 and 3. If Alice has 3 minutes to eat, she should choose Dish 2 to 4, which can give her $1 \times 4 + 2 \times 1 + 3 \times 5 = 21$ happiness.

As a friend of Alice, help her find the maximum total happiness she can gain from the meal.

$$1 \leq N \leq 100000$$

$$1 \leq K \leq N$$

$$0 \leq D_i \leq 1000$$

INPUT

The first line contains two integers, N and K .

The second line contains N integers. The i^{th} integer is the happiness value of Dish i : D_i .

OUTPUT

Output one integer, the maximum possible value of the total happiness.

Example - HKOJ S164 Alice's Meal

tl;dr version:

- Given an array d of integers
- We would like to find a continuous sequence of length K s.t. the sum of $d[x] * 1 + d[x+1] * 2 + \dots + d[x+K-1] * K$ is maximized

$$1 \leq N \leq 100000$$

$$1 \leq K \leq N$$

$$0 \leq D_i \leq 1000$$

Note that $O(NK)$ won't work

Observation

Let's look at the happiness of difference sequences

Happiness of sequence starting from $x - 1$

$$d[x-1] * 1 + d[x] * 2 + \dots + d[x-1+K-1] * K$$

Happiness of sequence starting from x

$$d[x] * 1 + d[x+1] * 2 + \dots + d[x+K-1] * K$$

Any observations?

Observation

Let's reformat it a bit

$$d[x-1]*1 + d[x]*2 + d[x+1]*3 + \dots + d[x-1+K-1]*K$$

$$d[x]*1 + d[x+1]*2 + \dots + d[x-1+K-1]*(K-1) + d[x+K-1] * K$$

Difference?

$$-d[x-1]*1 - d[x]*1 - d[x+1]*1 - \dots - d[x-1+K-1]*1 + d[x+K-1] * K$$

$$= -d[x-1] - \dots - d[x-1+K-1] + d[x+K-1]*K$$

Solution

Define $f(x)$ as sum obtained from sequence starting at $d[x]$

Using our observation just now,

$$f(x) - f(x-1) = -d[x-1 \dots x-1+K-1] + d[x-K-1] * K$$

Rearrange the terms, $f(x) = f(x-1) - d[x-1 \dots x-1+K-1] + d[x-K-1] * K$

sum of $d[x-1 \dots x-1+K-1]$ sounds familiar... How can we compute it?

Solution

Using 1D partial sum, we can compute $d[x-1 \dots x-1+K-1]$ at $O(1)$!

After finding $f(1)$, we can compute all sequences onwards easily

\Rightarrow exhaust all sequences at $O(N)$!

Outline

- ❑ Partial Sum
- ❑ **Difference Array**
- ❑ Precomputation
- ❑ Sliding Window (Two Pointers)
- ❑ Batching Step
- ❑ Discretization

Problem

Given P actions. In each action, add v_i to $a[L \dots R]$ of array a (size N). Find the final value of array a .

Add 2 to $a[2 \dots 4]$

i	1	2	3	4	5	6
$a[i]$	0	2	2	2	0	0

Add 3 to $a[3 \dots 6]$

i	1	2	3	4	5	6
$a[i]$	0	2	5	5	3	3

Naive Solution

```
for i ← 1 to P do  
    input L, R, V  
    for j ← L to R do  
        a[j] += V
```

```
for i ← 1 to N do  
    output a[i]
```

Simulate the operation, adding v_i to all $a[j]$ with $L \leq j \leq R$ each action

Time Complexity?

$O(PN)$

(Imagine all actions act on $a[1]$ to $a[N]$)

1D Difference Array - Idea

Assume the actions are:

1. add **2** for $a[2]$ to $a[6]$
2. add **3** for $a[4]$ to $a[8]$
3. add **4** for $a[7]$ to $a[7]$

Try to compute the final result

1D Difference Array - Idea

Assume the actions are:

1. add **2** for $a[2]$ to $a[6]$
2. add **3** for $a[4]$ to $a[8]$
3. add **4** for $a[7]$ to $a[7]$

Try to compute the difference between adjacent values, any observations?

Final Result:

i	1	2	3	4	5	6	7	8	9
$a[i]$	0	2	2	5	5	5	7	3	0

difference **+2** 0 **+3** 0 0 +2 **-4** **-3**

$$(+2 = +4 -2)$$

1D Difference Array - Idea

Lets mark the differences as an array d.

i	1	2	3	4	5	6	7	8	9
d[i]		+2		+3			+2	-4	-3
a[i]	0	2	2	5	5	5	7	3	0

- $d[i] = a[i] - a[i-1]$

1D Difference Array - Idea

for each action i , it adds v_i to $d[L]$ and subtracts v_i from $d[R+1]$

i	1	2	3	4	5	6	7	8	9
$d[i]$		+2		+3			-2 +4	-4	-3
$a[i]$	0	2	2	5	5	5	7	3	0

- add **2** for $a[2]$ to $a[6]$
- add **3** for $a[4]$ to $a[8]$
- add **4** for $a[7]$ to $a[7]$

1D Difference Array - Idea

Can we compute value of array a based on array d?

i	1	2	3	4	5	6	7	8	9
d[i]		+2		+3			+2	-4	-3
a[i]	0	2	2	5	5	5	7	3	0

Yes! $a[i] = a[i-1] + d[i]$

1D Difference Array - Idea

Instead of updating the whole contiguous section, we can just mark the value at start and end points using array d.

array a can be computed based on d after all actions are done.

i	1	2	3	4	5	6	7	8	9
d[i]		+2		+3			-2 +4	-4	-3
a[i]	0	2	2	5	5	5	7	3	0

- add **2** for a[2] to a[6]
- add **3** for a[4] to a[8]
- add **4** for a[7] to a[7]

1D Difference Array - Implementation

```
for i ← 1 to P do
```

```
    input L, R, V
```

```
    d[L] += V
```

```
    d[R+1] -= V
```

```
for i ← 1 to N do
```

```
    a[i] = a[i-1] + d[i]
```

```
    output a[i]
```

Time Complexity

= $O(N+P)$

Example - T062 Tappy World

Tappy World is a 2-D world, which only has the directions `up`, `down`, `left` and `right`. It has an infinite height, but a finite width of N . You can view the world as a 2D grid. The columns are labeled consecutively from 1 to N .

Let us assume that the land was initially at zero height when Tappy World was created. The Tappy World God has the power to raise or sink a range of the land at a time. For example, if he wished to raise the land two units from column 3 to 8, the results would be like this:

```

4
3
2  .....
1  .....
0  .....
-1 .....
-2 .....
-3 .....
-4 .....
123456789

```

If he then wished to sink the land 4 units from column 2 to 5, the results would be like this:

```

4
3
2      ...
1      ...
0 .    ....
-1 .   ....
-2 .  ....
-3 .  ....
-4 .  ....
123456789

```

The final height of the columns would be 0, -4, -2, -2, 2, 2, 2, 0, respectively.

Given a list of actions the Tappy World God has done on Tappy World, you are asked to find what the world finally looks like.

- $1 \leq N, M \leq 100,000$.
- $1 \leq X, Y \leq N$.
- D is a 32-bit signed integer.

Example - T062 Tappy World

tl;dr version:

- Given an array (let's denote as a) of size N with all values initialized as 0
- M actions are performed, each action adds D (32-bit integer) to $a[X..Y]$
- What is the resulting array a ?

looks familiar? it is same as the problem for 1D difference array!

Example - T062 Tappy World

```
for i ← 1 to M do
```

```
    input D, X, Y
```

```
    d[X] += D
```

```
    d[Y+1] -= D
```

```
for i ← 1 to N do
```

```
    a[i] = a[i-1] + d[i]
```

Time complexity $O(M+N)$

Reminder

- use long long for a
- initialize $a[0]$

Problem

Now, let's extend the question to 2-dimension

0	2	2	2	0
0	2	2	2	0
0	0	0	0	0
0	0	0	0	0

Given an array a with N rows and M columns and some actions.

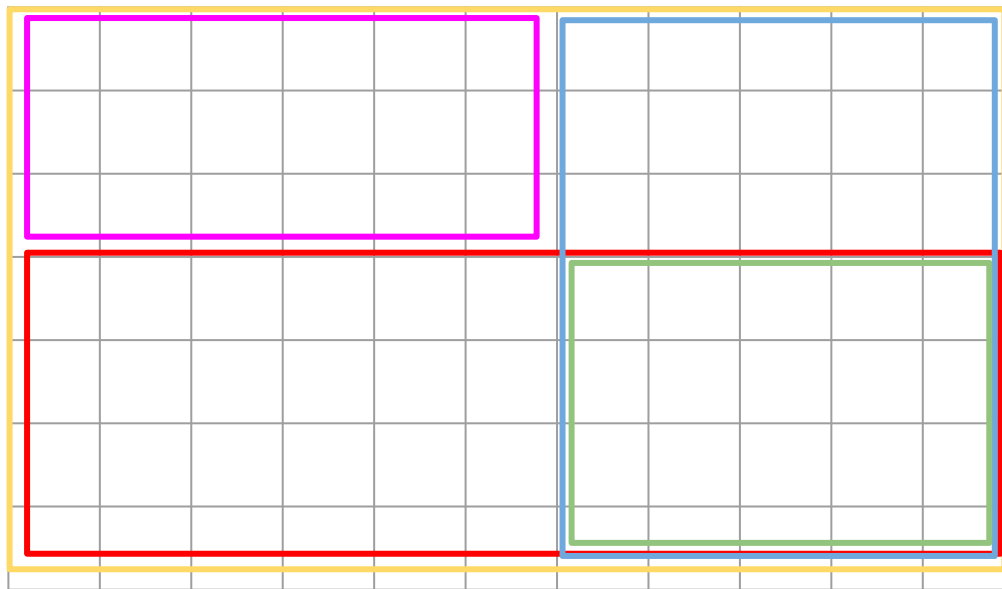
In each action, you need to add v_i all to elements from (Sr, Sc) to (Er, Ec) .

example:

add 2 to all elements from $(1, 2)$ to $(2, 4)$

2D Difference Array - Idea

This picture again... but this time rotated

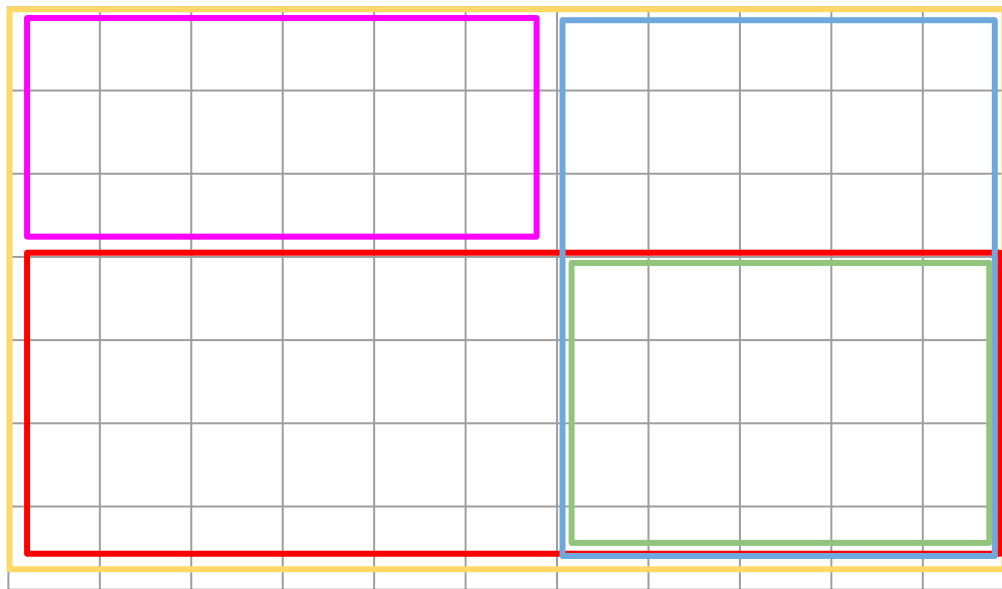


Assume we can get the “area” of each rectangle touching (N, M)

How can we get the “area” of magenta?

2D Difference Array - Idea

magenta = yellow - red - blue + green



4	3
2	1

1 + 2 + 3 + 4

- 1 - 2

- 1 - 3

+ 1

= 4

2D Difference Array - Idea

Now, let's define an array d such that

$$d[i][j] = a[i][j] - a[i][j-1] - a[i-1][j] + a[i-1][j-1]$$

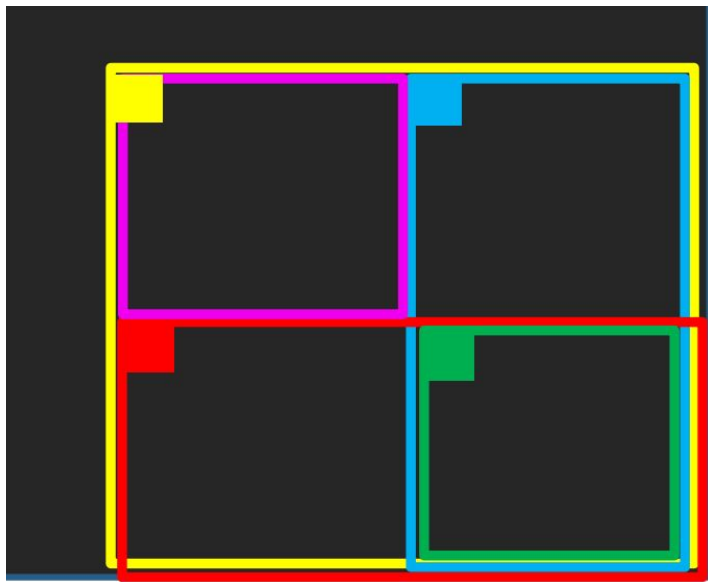
0	2	2	4
2	5	2	3
2	6	1	5
3	4	0	1



0	2	0	2
2	1	-3	-1
0	1	-2	3
1	-3	1	-3

If you don't get the formula, just bear in mind that $d[i][j]$ affects all value from (i, j) to (N, M) .

2D Difference Array - Idea



Imagine the small filled squares as $d[i][j]$, and the rectangle is the area affected by $d[i][j]$.

To update $[Sr...Er, Sc...Ec]$, we first add v to $d[Sr][Sc]$.

Then, we block the increase outside the area by subtracting v from $d[Sr][Ec+1]$ and $d[Er+1][Sc]$.

Since v is subtracted twice from $[Er+1...N, Ec+1...M]$, we add v to $d[Er+1][Ec+1]$.

2D Difference Array - Idea

Now, let's see how we can compute array a from d .

In the previous slides, we defined:

$$d[i][j] = a[i][j] - a[i][j-1] - a[i-1][j] + a[i-1][j-1]$$

By rearranging the equation, we get:

$$a[i][j] = d[i][j] + a[i][j-1] + a[i-1][j] - a[i-1][j-1]$$

2D Difference Array - Action

```
for i ← 1 to P do
  Input Sx, Sy, Ex, Ey
  d[Sx][Sy] += v
  d[Sx][Ey+1] -= v
  d[Ex+1][Sy] -= v
  d[Ex+1][Ey+1] += v
```

What is the array d
corresponding to
this array?

0	0	0	0
5	5	5	0
5	5	5	0
0	0	0	0



0	0	0	0
+5	0	0	-5
0	0	0	0
-5	0	0	+5

2D Difference Array - Compute Result

```
for i ← 1 to N do
  for j ← 1 to M do
    a[i][j] = d[i][j] + a[i][j-1] + a[i-1][j] - a[i-1][j-1]
  output a[i][j]
```

Time Complexity = $O(NM + P)$

Outline

- ❑ Partial Sum
- ❑ Difference Array
- ❑ **Precomputation**
- ❑ Sliding Window (Two Pointers)
- ❑ Batching Step
- ❑ Discretization

Problem

Given a string that consists of 'A', 'B' and some queries.

For each query q_i , output the closest 'B' whose index $\leq q_i$.

Example:

string = "AABAAABBA"

query 3 \rightarrow output 3

query 6 \rightarrow output 3

query 9 \rightarrow output 8

Naive solution

```
for i ← 1 to Q do
  input qi
  for j ← qi downto 1 do
    if (s[j] = 'B')
      output j
      break
```

Notations:

s = given string

N = length of string

Time complexity = $O(QN)$

- Remember to handle the case where 'B' is not found

Precomputation - Solution

Build an array a where $a[i]$ = the last "B" which index $\leq i$

For example, string = "AABAAABBA":

i	1	2	3	4	5	6	7	8	9
a[i]	-1	-1	3	3	3	3	7	8	8

- -1 is used to indicate that there is no 'B' with index $\leq i$.

Precomputation - Solution

What is the array a for string = "ABABABBA"?

i	1	2	3	4	5	6	7	8	9
a[i]	-1	2	2	4	4	6	7	7	7

What is the algorithm used to compute $a[i]$?

Precomputation - Solution

Note that if $\text{string}[i] = \text{'B'}$, then $a[i] = i$, else $a[i] = a[i-1]$

i	1	2	3	4	5	6	7	8	9
a[i]	-1	2	2	4	4	6	7	7	7

string = "ABABABBA"

Precomputation - Solution

```
a[0] = -1
for i ← 1 to N do
    if (s[i] = 'B')
        a[i] = i
    else
        a[i] = a[i-1]
```

```
for i ← 1 to Q do
    input qi
    output a[qi]
```

Total time complexity = $O(N + Q)$

Precomputation

Some useful arrays to be precomputed

- ❑ Prefix / suffix sum (partial sum)
- ❑ Prefix / suffix max / min / special element
- ❑ Prefix / suffix XOR sum
- ❑ Prefix / suffix index of last special element
- ❑ Prefix / suffix count (e.g. number of odd numbers / number of "*"s)

Practice

20108 Maximum Sum

I0111 Mobile Phones

J051 Pattern Matching

J134 Lucky Rainbow

M1513 Advanced Optimization Coding

S032 Project

S113 Gravity Game

S164 Alice's Meal

T062 Tappy World

attendance + 5-minute break (until 11:44)

Outline

- ❑ Partial Sum
- ❑ Difference Array
- ❑ Precomputation
- ❑ **Sliding Window (Two Pointers)**
- ❑ Batching Step
- ❑ Discretization

Problem

Given two sorted array a and b ,
find the number of pair (i, j) such that $a[i] + b[j] = C$

i	1	2	3	4	5	6	7	8	9
$a[i]$	1	5	8	10	12	14	15	18	25
$b[i]$	3	6	6	7	13	14	15	18	19

If $C = 20$, $\text{ans} = 4$ $[(1, 9), (2, 7), (6, 2), (6, 3)]$

Naive Solution

```
for i ← 1 to N do
  for j ← 1 to N do
    if (a[i] + b[j] = C)
      ans += 1
output ans
```

Exhaust all pairs of (i, j) and count how many of them satisfy $a[i] + b[j] = C$

Time Complexity = $O(N^2)$

Hint: the arrays are **sorted**

Binary Search Solution

For each element in a ,
binary search the number of $b[j]$ such that $b[j] = C - a[i]$

Two binary search (one for lower bound, one for upper bound) are needed if the elements in b are not distinct.

Time complexity = $O(N \log N)$

Good, but can it be better?

Sliding Window - Observation

Find the number of pair (i, j) such that $a[i] + b[j] = C$

If $C = 20$, ans = 5 $[(1, 9), (2, 7), (6, 2), (6, 3), (4, 5)]$

i	1	2	3	4	5	6	7	8	9
a[i]	1	5	8	10	12	14	15	18	25
b[i]	3	6	6	8	13	14	15	18	19

Let's look at the array again, are there any pattern in the pairs' position?
When $a[i]$ increases, its paired $b[j]$ decreases!

Sliding Window - Observation

1	5	8	10	12	14	15	18	25
3	6	6	7	13	14	15	18	19

1	5	8	10	12	14	15	18	25
3	6	6	7	13	14	15	18	19

1	5	8	10	12	14	15	18	25
3	6	6	7	13	14	15	18	19

1	5	8	10	12	14	15	18	25
3	6	6	7	13	14	15	18	19

- i increases
- $a[i]$ increases
- $C - a[i]$ decreases
- $b[j]$ decreases
- j decreases

Sliding Window - Observation

from last slide, we know that when i increases, j decreases

⇒ when $a[i]$ is selected as the x th pair, $b[j]$ must be less than or equal to $(x-1)$ th pair's j value

⇒ we can eliminate all possibility of j that is $>$ last pair's j value!

Sliding Window - Implementation

```
j = N
for i ← 1 to N do
    while (j > 0) and (a[i] + b[j] > C)
        j -= 1
    while (j > 0) and (a[i] + b[j] = C)
        ans += 1
output ans
```

Instead of storing each pair's j value, we let j be decreasing.

Time Complexity = $O(N)$

Sliding Window - Demonstration

Find number of pairs (i, j) such that $a[i] + b[j] = C$. Assume $C = 20$.

	↓	↓	↓	↓	↓	↓	↓	↓	↓
a	1	5	8	10	12	14	15	18	25
b	3	6	6	7	13	14	15	18	19
	↑	↑	↑ ₊₁	↑ ₊₁	↑	↑	↑	↑ ₊₁	↑ ₊₁

```

for i ← 1 to N do
    while (j > 0) and (a[i] + b[j] > C) j -= 1
    while (j > 0) and (a[i] + b[j] = C) ans++, j -= 1
    
```

Sliding Window - HKOJ M0652 Museum

Given N , M and an array of integers a .

Find the shortest consecutive segment that contains all numbers $1..M$.

$1 \leq N \leq 1000000$, $1 \leq M \leq 2000$

Example:

$N = 12$, $M = 5$

2	5	3	1	3	2	4	1	1	5	4	3
---	---	---	---	---	---	---	---	---	---	---	---

Sliding Window - M0652 Museum

We can set the start and end of the segment as our two pointers.

```
j = 1
for i ← 1 to N do
    while (a[i..j] does not contain all m integers) do
        j += 1
    ans = min(ans, j - i + 1)
```

- Counting array can be used to maintain if current segment contains all m integers.
- Remember to handle cases when $j \geq N$.
- Note that j would not decrease when i increases.

Sliding Window - M0652 Museum

[illegible]

Sliding Window - Application

- Input is two array or one array that references itself
- the arrays are sorted, or
- the things we want to find have monotonicity (e.g. sum / count of sth)

Outline

- ❑ Partial Sum
- ❑ Difference Array
- ❑ Precomputation
- ❑ Sliding Window (Two Pointers)
- ❑ **Batching Step**
- ❑ Discretization

Batching Step

- In some simulation problems, some continuous steps can be performed at once.

Find the date of now + N days

- Instead of iterating over days, iterate over months
- Deduct M_i days from N

J184 Mysterious Area

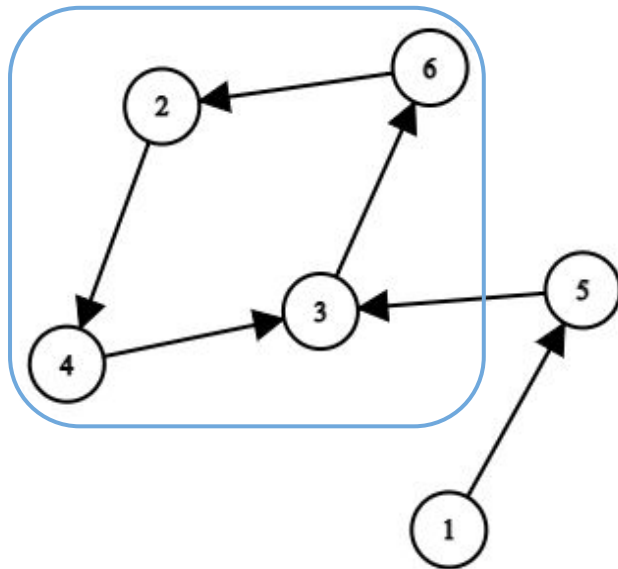
- Instead of moving grid by grid horizontally, directly move to the nearest pillar taller than or equal to current height

Batching Step - Find Cycles

In some simulation problems, we may encounter cycles that are traversed many times, causing TLE.

Walk X ($1 \leq X \leq 10^{18}$) steps starting from node 1, what is the final position?

If we simulate each step ($O(X)$), we will get TLE :<



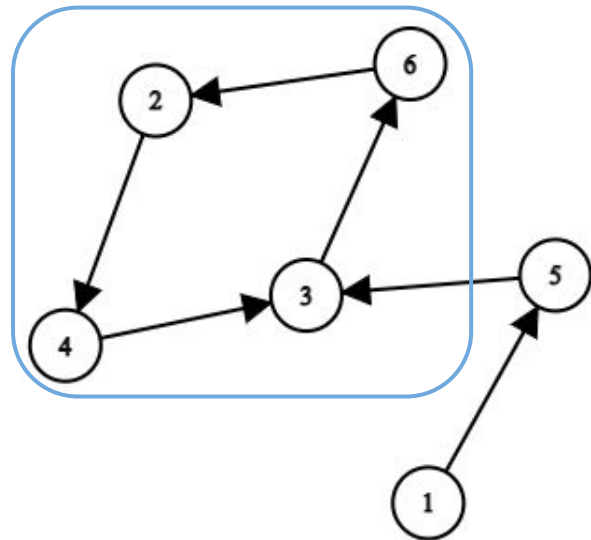
Batching Step - Find Cycles

Instead, we can skip the steps that traverse the whole cycle.
(The results won't be affected by the number of cycles traversed)

The simulation will then become:

2 steps ($1 \rightarrow 5 \rightarrow 3$)
+ $(X - 2) \% 4$ steps

The time complexity is now $O(N)$
(more details in graph sessions)



Batching Step - HKOJ T042 Game of Life IV

A terrorist has started to play a game with you, the best detective in the city, a game with life involved. A timed electronic bomb at the town center is set up to explode in M milliseconds. The bomb is equipped with a numeric display, showing a number between 1 and N inclusive at any moment. The number is 1 when the bomb is set up and it will change every millisecond.

The terrorist has left you a thread of hope. The bomb is equipped with a USB port. You can disarm the bomb, by plugging in a keyboard and typing in the number which would appear at the $(M + 1)^{th}$ millisecond if the bomb does not explode. You don't have enough time to evacuate all people away from the town center, so what you can do is to crack the secret number.

A group of mathematicians have gathered to examine the numbers available on the display, and concluded that the number in the next millisecond only depends on the number appearing just before it. They have computed all such relationships and coded them in a table. You are going to write a program taking this table as input and giving the secret number as output. The bomb is about to explode. Act now!

INPUT

The first line of the input contains 2 space-separated integers, N and M , where $1 \leq N \leq 2^{21} - 1$ and $1 \leq M \leq 2^{31} - 1$. Each of the next N lines contains a single integer between 1 and N inclusive. The integer on the $(i + 1)^{th}$ line indicates the number to which the display will change in the next millisecond if the current number is i .

OUTPUT

The output is a single number that should be typed in to save the lives.

Batching Step - HKOJ T042 Game of Life IV

- Let's simplify the statement first...
- We can view it as a directed graph with N nodes, each node has one outgoing edge.
- We need to go through M edges starting from node 1, where will we finally be at?
- How can we apply batching step to solve it?

Batching Step - HKOJ T042 Game of Life IV

- Note that there are a total of N edges, meaning that it must have a cycle
 - When M is significantly larger than N , we must be stuck in a cycle for long
- ⇒ we can find out the cycle and “skip” it
(again, implementation details in graph sessions)

Outline

- ❑ Partial Sum
- ❑ Difference Array
- ❑ Precomputation
- ❑ Sliding Window (Two Pointers)
- ❑ Batching Step
- ❑ **Discretization**

Problem

Count the number of occurrence of some numbers in array a ($a[i] \leq 10^9$)
whereas N (size of a) ≤ 100000

Naive Solutions

1. using a counting array.

but... we probably cannot create a counting array of $[1..10^9]$:/

2. for each $a[i]$, loop through a to count number of occurrences
obviously, $O(N^2)$ gives TLE for $N \leq 100000$

Discretization - Idea

- Discretization (離散法) is a technique that converts values (not necessarily integers) into integers, while maintaining their relative order
- Example: 7654321, 123456, 934602, 123456789 \rightarrow 3, 1, 2, 4 (or 2, 0, 1, 3)

Discretization - Idea

- Put the values into an array, sort the array 123456, 934602, 7654321, 123456789
- For each value in the original array, find its rank using binary search (*or you can use map for this step, more in C++ STL / data structure*)
- Note: it's better for the same value to be converted into the same number (*by making the sorted array unique*)

Discretization - Solution

Count the number of occurrence of some numbers in array a ($a[i] \leq 10^9$)

We can perform discretization and map the numbers to their rank.

A counting array can be built with size = number of **unique** numbers.

During counting/query, we can use the value's rank to access the counting array to modify/retrieve it.

Discretization - Solution

Count the number of occurrence of some numbers in array a ($a[i] \leq 10^9$)

We can perform discretization and map the numbers to their rank.

A counting array can be built with size = number of **unique** numbers.

During counting/query, we can use the value's rank to access the counting array to modify/retrieve it.

Discretization - Solution

C++ Implementation

```
vector<int> v;  
for (int i = 0; i < n; i++) v.push_back(a[i]);  
sort(v.begin(), v.end());  
v.resize(unique(v.begin(), v.end()) - v.begin());  
for (int i = 0; i < n; i++)  
    a[i] = lower_bound(v.begin(), v.end(), a[i]) - v.begin();
```

More Practice

J144 Fair Santa Claus

M0652 Museum

M1613 Fair Patrol

S072 Partners

S152 Apple Garden

S211 Skyscraperhenge

T042 Game of Life IV



Q&A