# Prefix & Coordinate Compression & Two Pointer & Set

Binglun Wang

University College London

19th Jan, 2023

1. 1D Prefix Sums
2. 2D Prefix Sum
3. Coordinate Compression
4. Two pointer
5. Set & Map & Multiset
6. Q & A

1. 1D Prefix Sums
2. 2D Prefix Sum
3. Coordinate Compression
4. Two pointer
5. Set & Map & Multiset
6. Q & A

# Example

## C. Good Subarrays

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array $a_1, a_2, \ldots, a_n$ consisting of integers from $0$ to $9$. A subarray $a_l, a_{l+1}, a_{l+2}, \ldots, a_{r-1}, a_r$ is good if the sum of elements of this subarray is equal to the length of this subarray ($\sum_{i=l}^{r} a_i = r - l + 1$).

For example, if $a = [1, 2, 0]$, then there are $3$ good subarrays: $a_{1\ldots1} = [1]$, $a_{2\ldots3} = [2, 0]$ and $a_{1\ldots3} = [1, 2, 0]$.

Calculate the number of good subarrays of the array $a$.

### Input

The first line contains one integer $t$ ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains one integer $n$ ($1 \le n \le 10^5$) — the length of the array $a$.

The second line of each test case contains a string consisting of $n$ decimal digits, where the $i$-th digit is equal to the value of $a_i$.

It is guaranteed that the sum of $n$ over all test cases does not exceed $10^5$.

### Output

For each test case print one integer — the number of good subarrays of the array $a$.

图 1: https://codeforces.com/contest/1398/problem/C

# Algorithm

-

$$p_k = \sum_{i}^{k} a_i \tag{1}$$

$$a_i + a_{i+1} + \ldots + a_{j-1} + a_j = p_j - p_{i-1}$$

# Algorithm

- $$p_k = \sum_{i}^{k} a_i \tag{1}$$

$$a_i + a_{i+1} + \ldots + a_{j-1} + a_j = p_j - p_{i-1}$$

- $p$ can be calculated easily in $O(N)$, where $N$ is the length of array.

## Algorithm

- $$p_k = \sum_{i}^{k} a_i \quad (1)$$

  $$a_i + a_{i+1} + \ldots + a_{j-1} + a_j = p_j - p_{i-1}$$

- $p$ can be calculated easily in $O(N)$, where $N$ is the length of array.
- $$p_k = p_{k-1} + a_k \quad (2)$$

# Algorithm

- $$p_k = \sum_{i}^{k} a_i \tag{1}$$
  $$a_i + a_{i+1} + \ldots + a_{j-1} + a_j = p_j - p_{i-1}$$
- $p$ can be calculated easily in $O(N)$, where $N$ is the length of array.
- $$p_k = p_{k-1} + a_k \tag{2}$$
- Explanation from a set perspective

1. 1D Prefix Sums
2. 2D Prefix Sum
3. Coordinate Compression
4. Two pointer
5. Set & Map & Multiset
6. Q & A

# Example

## A - DP on cheessbord

计蒜客 - T2118

如图，$A$ 点有一个过河卒，需要走到目标 $B$ 点。卒行走规则：可以向下、或者向右。同时在棋盘上的任一点有一个对方的马（如上图的 $C$ 点），该马所在的点和所有跳跃一步可达的点称为对方马的控制点。例如上图 $C$ 点上的马可以控制 9 个点（图中的 $P_1, P_2, \cdots P_8$ 和 $C$）。卒不能通过对方马的控制点。

棋盘用坐标表示，$A$ 点 $(0,0)$ 、$B$ 点 $(n,m)$ （$n,m$ 为不超过 20 的整数，并由键盘输入），同样马的位置坐标是需要给出的（约定：$C > A$，同时 $C < B$）。

现在要求你计算出卒从 $A$ 点能够到达 $B$ 点的路径的条数。

### 输入格式

一行四个数据，分别表示 $B$ 点坐标和马的坐标。

### 输出格式

一个数据，表示所有的路径条数。

**Sample 1**

| Input copy | Output copy |
| --- | --- |
| 6 6 3 3 | 6 |

图 2: [NOIP2002] 过河卒

## Algorithm

-

$$p_{a,b} = \sum_{i}^{a} \sum_{j}^{b} A_{i,j}$$
$$\sum_{i=a}^{c} \sum_{i=b}^{d} A_{i,j} = p_{c,d} - p_{a-1,d} - p_{c,b-1} + p_{a-1}b - 1 \tag{3}$$

## Algorithm

- $$p_{a,b} = \sum_{i}^{a} \sum_{j}^{b} A_{i,j}$$
$$\sum_{i=a}^{c} \sum_{i=b}^{d} A_{i,j} = p_{c,d} - p_{a-1,d} - p_{c,b-1} + p_{a-1}b - 1 \quad (3)$$

- $p$ can be calculated easily in $O(NM)$, where $N, M$ is the size of 2D array.

## Algorithm

- $$p_{a,b} = \sum_{i}^{a} \sum_{j}^{b} A_{i,j}$$
  $$\sum_{i=a}^{c} \sum_{i=b}^{d} A_{i,j} = p_{c,d} - p_{a-1,d} - p_{c,b-1} + p_{a-1} b - 1 \tag{3}$$
- $p$ can be calculated easily in $O(NM)$, where $N, M$ is the size of 2D array.
- $$p_{i,j} = p_{i-1,j} + p_{i,j-1} - p_{i-1,j-1} + A_{i,j} \tag{4}$$

## Algorithm

- $$p_{a,b} = \sum_{i}^{a} \sum_{j}^{b} A_{i,j}$$
  $$\sum_{i=a}^{c} \sum_{i=b}^{d} A_{i,j} = p_{c,d} - p_{a-1,d} - p_{c,b-1} + p_{a-1}b - 1 \tag{3}$$
- $p$ can be calculated easily in $O(NM)$, where $N, M$ is the size of 2D array.
- $$p_{i,j} = p_{i-1,j} + p_{i,j-1} - p_{i-1,j-1} + A_{i,j} \tag{4}$$
- Explanation from a set perspective

# Example2

Farmer John's largest pasture can be regarded as a large 2D grid of square "cells" (picture a huge chess board). Currently, there are $N$ cows occupying some of these cells ($1 \leq N \leq 2500$).

Farmer John wants to build a fence that will enclose a rectangular region of cells; the rectangle must be oriented so its sides are parallel with the $x$ and $y$ axes, and it could be as small as a single cell. Please help him count the number of distinct subsets of cows that he can enclose in such a region. Note that the empty subset should be counted as one of these.

**INPUT FORMAT (input arrives from the terminal / stdin):**

The first line contains a single integer $N$. Each of the next $N$ lines Each of the next $N$ lines contains two space-separated integers, indicating the $(x, y)$ coordinates of a cow's cell. All $x$ coordinates are distinct from each-other, and all $y$ coordinates are distinct from each-other. All $x$ and $y$ values lie in the range $0 \ldots 10^9$.

**OUTPUT FORMAT (print output to the terminal / stdout):**

The number of subsets of cows that FJ can fence off. It can be shown that this quantity fits within a signed 64-bit integer (e.g., a "long long" in C/C++).

**SAMPLE INPUT:**

```
4
0 2
1 0
2 3
3 5
```

**SAMPLE OUTPUT:**

```
13
```

There are $2^4$ subsets in total. FJ cannot create a fence enclosing only cows 1, 2, and 4, or only cows 2 and 4, or only cows 1 and 4, so the answer is $2^4 - 3 = 16 - 3 = 13$.

**SCORING:**

- Test cases 2-3 satisfy $N \leq 20$.
- Test cases 4-6 satisfy $N \leq 100$.
- Test cases 7-12 satisfy $N \leq 500$.
- Test cases 13-20 satisfy no additional constraints.

图 3: http://www.usaco.org/index.php?page=viewproblem2&cpid=1063

1. 1D Prefix Sums
2. 2D Prefix Sum
3. Coordinate Compression
4. Two pointer
5. Set & Map & Multiset
6. Q & A

## Example

- Query Q times, each time asking how many times a number appears in the array.

# Example

- Query Q times, each time asking how many times a number appears in the array.
- The maximum value is $1e18$

# Algorithm

- Map values to their rank.

## Algorithm

- Map values to their rank.
- Get rank: $O(N \log N)$ sort algorithm, hash map could be faster. (we will study in the future. One insert and query expected time complexity $O(\frac{N}{M})$, Where $N$ is the hash map size, $M$ is parameter with particular constrain)

## Algorithm

- Map values to their rank.
- Get rank: $O(N \log N)$ sort algorithm, hash map could be faster. (we will study in the future. One insert and query expected time complexity $O(\frac{N}{M})$, Where $N$ is the hash map size, $M$ is parameter with particular constrain)
- Example $[100, 999, 23]$ -> $[2, 3, 1]$

1 1D Prefix Sums

2 2D Prefix Sum

3 Coordinate Compression

4 Two pointer

5 Set & Map & Multiset

6 Q & A

# Example

### C. Hard Process

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array $a$ with $n$ elements. Each element of $a$ is either $0$ or $1$.

Let's denote the length of the longest subsegment of consecutive elements in $a$, consisting of only numbers one, as $f(a)$. You can change no more than $k$ zeroes to ones to maximize $f(a)$.

**Input**

The first line contains two integers $n$ and $k$ $(1 \le n \le 3 \cdot 10^5, 0 \le k \le n)$ — the number of elements in $a$ and the parameter $k$.

The second line contains $n$ integers $a_i$ $(0 \le a_i \le 1)$ — the elements of $a$.

**Output**

On the first line print a non-negative integer $z$ — the maximal value of $f(a)$ after no more than $k$ changes of zeroes to ones.

On the second line print $n$ integers $a_j$ — the elements of the array $a$ after the changes.

- If there are multiple answers, you can print any one of them.

图 4: https://codeforces.com/problemset/problem/660/C

# Algorithm

- Denote $l, r$ are two pointers the beginning and end of the segment in an array.

# Algorithm

- Denote $l, r$ are two pointers the beginning and end of the segment in an array.
- Two pointers with monotonic variation over iterations of the algorithm.

# Example2

## C. An impassioned circulation of affection

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Nadeko's birthday is approaching! As she decorated the room for the party, a long garland of Dianthus-shaped paper pieces was placed on a prominent part of the wall. Brother Koyomi will like it!

Still unsatisfied with the garland, Nadeko decided to polish it again. The garland has $n$ pieces numbered from $1$ to $n$ from left to right, and the $i$-th piece has a colour $s_i$, denoted by a lowercase English letter. Nadeko will repaint **at most** $m$ of the pieces to give each of them an arbitrary new colour (still denoted by a lowercase English letter). After this work, she finds out all subsegments of the garland containing pieces of only colour $c$ — Brother Koyomi's favourite one, and takes the length of the longest among them to be the Koyomity of the garland.

For instance, let's say the garland is represented by "kooomo", and Brother Koyomi's favourite colour is "o". Among all subsegments containing pieces of "o" only, "ooo" is the longest, with a length of $3$. Thus the Koyomity of this garland equals $3$.

But problem arises as Nadeko is unsure about Brother Koyomi's favourite colour, and has swaying ideas on the amount of work to do. She has $q$ plans on this, each of which can be expressed as a pair of an integer $m_i$ and a lowercase letter $c_i$, meanings of which are explained above. You are to find out the maximum Koyomity achievable after repainting the garland according to each plan.

**Input**

The first line of input contains a positive integer $n$ $(1 \le n \le 1\,500)$ — the length of the garland.

The second line contains $n$ lowercase English letters $s_1 s_2 \dots s_n$ as a string — the initial colours of paper pieces on the garland.

The third line contains a positive integer $q$ $(1 \le q \le 200\,000)$ — the number of plans Nadeko has.

The next $q$ lines describe one plan each: the $i$-th among them contains an integer $m_i$ $(1 \le m_i \le n)$ — the maximum amount of pieces to repaint, followed by a space, then by a lowercase English letter $c_i$ — Koyomi's possible favourite colour.

**Output**

Output $q$ lines: for each work plan, output one line containing an integer — the largest Koyomity achievable after repainting the garland according to it.

图 5: https://codeforces.com/problemset/problem/814/C

1. 1D Prefix Sums
2. 2D Prefix Sum
3. Coordinate Compression
4. Two pointer
5. Set & Map & Multiset
6. Q & A

## Overview in C++

- Demo

1 1D Prefix Sums

2 2D Prefix Sum

3 Coordinate Compression

4 Two pointer

5 Set & Map & Multiset

6 Q & A
- wormhole sort
- Where's Bessie?

1. 1D Prefix Sums
2. 2D Prefix Sum
3. Coordinate Compression
4. Two pointer
5. Set & Map & Multiset
6. Q & A
   - wormhole sort
   - Where's Bessie?

1. 1D Prefix Sums
2. 2D Prefix Sum
3. Coordinate Compression
4. Two pointer
5. Set & Map & Multiset
6. Q & A
   - wormhole sort
   - Where's Bessie?