Lottery

First AC: Mariusz Trela, Poland (24:13)
#AC = 15

problem author: Juliusz Straszyński

Lottery

Let's focus on one query.

Computing the distance naively is O(L).

There are $O(n^2)$ pairs of intervals.

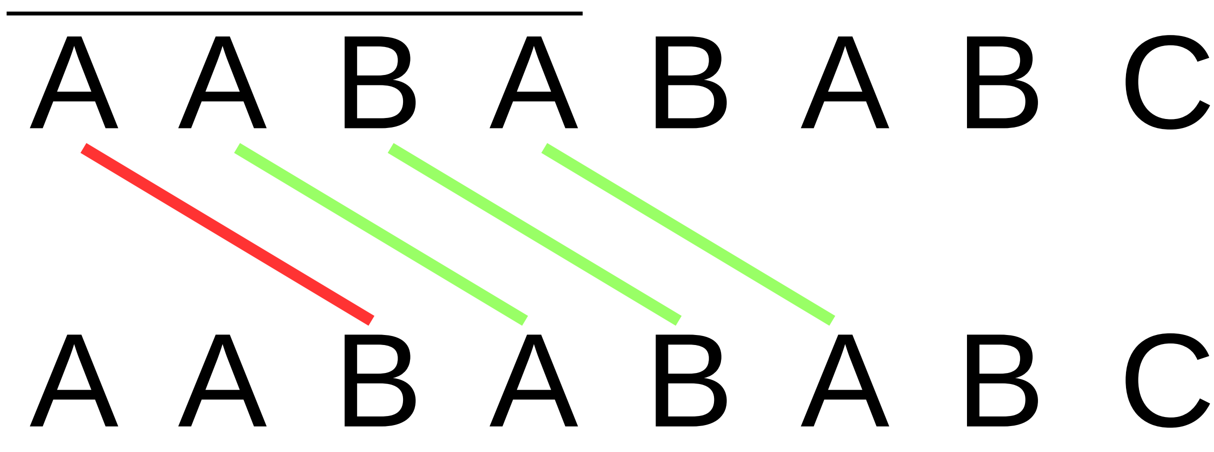
So, the brute force is $O(n^2 \times L) = O(n^3)$.

Lottery

A A B A B A B C

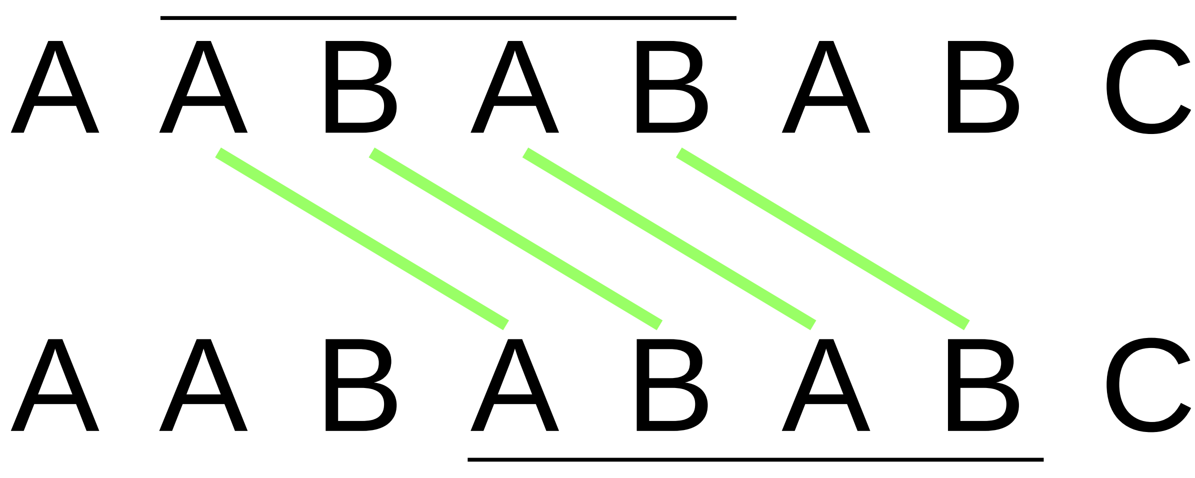
distance(1, 3) = 1

Lottery



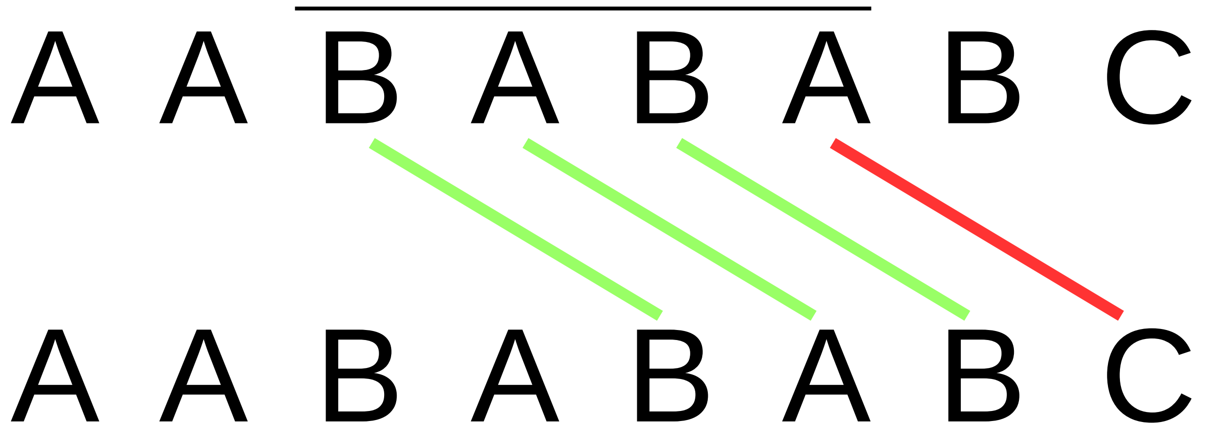
distance(1, 3) = 1

Lottery



distance(2, 4) = 0

Lottery



distance(3, 5) = 1

Lottery

O(n) for every shift.

$O(n^2)$ in total to find all the distances.

How to avoid the q factor in answering queries?

We can't store the $O(n^2)$ array!

Lottery

Queries: 2, 4, 7

0 → 2
1 → 2
2 → 2
3 → 4
4 → 4
5 → 7
6 → 7
7 → 7

Lottery

Time complexity: $O(n^2)$
Memory complexity: $O(nq)$

Cloud Computing

First AC: Costin-Andrei Oncescu, Romania (31:07)
#AC = 16

problem author: Karol Pokorski

Cloud Computing

Easiest possible version

$F_i = 1$, $f_i = 1$

$C_i = 1$, $c_i = 1$

$n = 1$ (one machine)

(consider the most profitable order)

Cloud Computing

Standard version

$F_i = 1$, $f_i = 1$

~~$C_i = 1$, $c_i = 1$~~

$n = 1$ (one machine)

$O(m \times c_1)$

dp[cores] – the largest profit to have so many cores

Cloud Computing

Double version

$F_i = 1, f_i = 1$

~~$C_i = 1, c_i = 1$~~

~~n = 1 (one machine)~~

two knapsacks

$O(n \times (n \times C) + m \times (m \times C))$

Cloud Computing

Double version

$F_i \leq f_i$ ← works too

~~$C_i = 1$, $c_i = 1$~~

~~n = 1 (one machine)~~

two knapsacks

$O(n \times (n \times C) + m \times (m \times C))$

Cloud Computing

One knapsack with modified items, e.g.:

- a task with weight 5 and value 20
- a machine with weight -7 and value -15

We must end with total weight 0 or smaller.

$O((n + m) \times (n \times C))$

Cloud Computing

Sort by f_i, F_i decreasingly.

Then just guarantee that the total weight is 0 or smaller **at every moment of time**.

$O((n + m) \times (n \times C))$

Cloud Computing

The alternative knapsack

$V_i = 1, v_i = 1$

dp[cores] → dp[money]

$O((n + m) \times n)$

Global Warming

First AC: Kacper Kluk, Poland (26:04)
#AC = 27

problem author: Kamil Dębowski

Global Warming

+3

$$0, 3, \overline{1, 5, 2, 4, 6}, 0, 7 \rightarrow \mathbf{0}, \mathbf{3}, \overline{\mathbf{4}, 8, \mathbf{5}, \mathbf{7}, \mathbf{9}}, 0, 7$$

+3

$$0, 3, \overline{1, 5, 2, 4, 6, 0, 7} \rightarrow \mathbf{0}, \mathbf{3}, \overline{\mathbf{4}, 8, \mathbf{5}, \mathbf{7}, \mathbf{9}, 0, \mathbf{10}}$$

suffix can only improve the answer!

Global Warming

$$1, \overset{-2}{\overline{5, 7}}, 6, 9 \rightarrow 1, \overline{3, 5}, 6, 9$$

$$\overset{-2}{\overline{1, 5, 7}}, 6, 9 \rightarrow \overline{-1, 3, 5}, 6, 9$$

$$1, 5, 7, \overset{+2}{\overline{6, 9}} \rightarrow 1, 5, 7, \overline{8, 11}$$

It’s enough to consider suffixes!

$d = x$

Global Warming

Modifying the standard LIS algorithm.

30, 60, 10, | 50, ...

1 – 10
2 – 60

Global Warming

Modifying the standard LIS algorithm.

30, 60, 10, 50, | ...

1 – 10
2 – ~~60~~ 50

Global Warming

The LDS from the right.

..., | 50, 45, 75, 80

1 – 45
2 – 50
3 – 50

Global Warming

The LDS from the right.

30, 60, 10, | **50**, 45, 75, 80

1 – 10
2 – 60

Global Warming

The LDS from the right.

30, 60, 10, | **50**, 45, 75, 80

1 – 10 ← longest ending with
2 – 60 a number 49 or smaller

Global Warming

The LDS from the right.

30, 60, 10, | **50**, 45, 75, 80

1 – 10
2 – 60 ← ok if $x \geq 11$

Global Warming

When first going from left, just before “50” remember the length of LIS ending with a number 50+x-1 or smaller.

30, 60, 10, | 50, ...

1 – 10
2 – 60

Thank you for your attention.

Good luck on Thursday!

Toys

First 100: Dawid Jamka, Poland (14:31)
#AC = 26

problem author: Karol Pokorski

Triangles

First 100: Mariusz Trela, Poland (58:17)
#AC = 18

problem author: Kamil Dębowski

Fibonacci

Max score: 65
First 65: Costin-Andrei Oncescu, Romania (2:19:09)
#65 = 2

problem authors: Dominik Klemba, Kamil Dębowski

Fibonacci

$0100101 = F_2 + F_5 + F_7 = 2 + 8 + 21 = 31$

1) Let’s find $X(F_{a[1]})$

0000001
0000110
0011010
1101010

$X(F_{a[1]}) = (a[1]-1)/2$

Fibonacci

2) Let's find $X(F_{a[1]}+F_{a[2]})$

00000001000001
00000001000110
00000001011010
00000002101010
00000111101010
00011011101010
…

$X(F_{a[1]}+F_{a[2]}) \approx (a_1 / 2) \cdot ((a_2-a_1) / 2)$

Fibonacci

3) General case, values a_i far away from each other.

$X \approx (a_1 / 2) \cdot ((a_2 - a_1) / 2) \cdot ((a_3 - a_2) / 2) \cdot \ldots$

$X \approx (d_1 / 2) \cdot (d_2 / 2) \cdot (d_3 / 2) \cdot (d_4 / 2) \cdot \ldots$

(d_i are distances between sorted a_i)

Fibonacci

The $O(n^2)$ solution, $|a_i - a_j| \geq 2$

For every prefix, sort values $a_1, a_2, \ldots, a_k$, and run O(k) dynamic programming dp[2].

…00010001…
…00010110…
…00021010… ← dp[0] is the number of ways to choose values on the right so that the next 1 on the left must be changed to smaller 1's (pushed further to the left)

dp[1] means: we can leave the next 1 unchanged

Fibonacci

…00010001…
…00010110…
…00021010… ← dp[0] is the number of ways to choose values on the right so that the next 1 on the left must be changed to smaller 1's (pushed further to the left)
dp[1] means: we can leave the next 1 unchanged

dp'[0] = dp[0] + dp[1]   (if distance is even, else 0)

dp'[1] = (dp[0] + dp[1]) · (distance - 1) / 2 + dp[1]

Fibonacci

What if $a_j = a_i + 1$?

…001100…
…000010…

Possible chain effect, amortized O(n) in total

…00010101010…
…00**1**10101010…
…0000**1**101010…
…000000**1**1010…
…

Fibonacci

“a_i are different squares of natural numbers”

“a_i are different even numbers”

No collisions.

Then we already have an $O(n^2)$ solution.

Fibonacci

What if $a_j = a_i$?

…000200…
…010010… because $2 \cdot F_k = F_{k+1} + F_{k-2}$

Doesn't amortize :(

Fibonacci

What if $a_j = a_i$?

Doesn't amortize :(

…0001010101**1**00…

…0001010101**2**00…

…00010101**2**0110…

…000101**2**011110…

…0001**2**01111110…

…010111111110…

…010001010101…

Fibonacci

$O(n^2)$ solution: resolve conflicts in any way in recompute the answer in $O(n)$ each time

Let’s try to avoid recomputing the answer.

Fibonacci

Let’s try to avoid recomputing the answer.

- a segment tree (either off-line or BST)

- matrix 2x2 or 3x3 in every node

- O(log(n)) per change

Fibonacci

Last steps.

way I – distances between consecutive 1’s

way II – maximal intervals of type 1010101

Thank you for your attention.

Good luck on IOI!