CODEFORCES β
Sponsored by Telegram

| JacobianDet | Logout

HOME CONTESTS GYM PROBLEMSET GROUPS RATING API VK CUP CALENDAR 8 YEARS!

ATSTNG BLOG TEAMS SUBMISSIONS GROUPS CONTESTS

ATSTNG's blog

Fenwick tree: initialization in O(N)

By **ATSTNG**, history, 9 hours ago, translation,

Hi, CodeForces.

Recently I decided to take a closer look at data structure called Fenwick tree or binary indexed tree.

In this blogpost I will only consider Fenwick tree for sum, but everything stated there can be easily generalized for any associative, commutative and inversable operation.

Some implementation

After reading articles about it on e-maxx.ru and neerc.ifmo.ru and asking friends I've got interested by this moment. Both articles (and my friends) use this algorithm to initialize Fenwick tree on given array:

1. Fenwick tree for array of zeros is array of zeros, we will take it as a basis
2. We will use tree queries to replace all elements of array with elements we need

```
void init_vector_classic (vector<int> a) {
    init_empty(a.size());

    for (int i = 0; i < a.size(); i++) inc(i, a[i]);
}
```

This will work in $O(N * logN)$ where N is length of given array.

Can we do it faster? Definitely can!

By definition of the Fenwick tree each tree element is the sum of continious segment of initial array. Considering that we have no queries to change elements in initialization process, we can use prefix sums to calculate elements of our Fenwick tree. This will require $O(N)$ preprocessing, but will allow to calculate tree elements in $O(1)$. So initialization asymptotic improves to $O(N) + O(1) * O(N) = O(N)$.

```
void init_vector_linear_prefix (vector<int> a) {
    init_empty(a.size());

    vector<int> prefix_sum;
    prefix_sum.assign(a.size(), 0);
    prefix_sum[0] = a[0];
    for (int i = 1; i < a.size(); i++) prefix_sum[i] = prefix_sum[i-1] +
a[i];

    for (int i = 0; i < a.size(); i++) {
        int lower_i = (i & (i+1)) - 1;
        fenwick[i] = prefix_sum[i] - (lower_i >= 0 ? prefix_sum[lower_i]
: 0);
    }
}
```

But in this implementation we get additional $O(N)$ memory to store additional array.

Can we avoid this? Yes, we can!

→ Pay attention

Before contest
Codeforces Round #480 (Div. 2)
32:36:32

Like 78 people like this. Sign Up to see what your friends like.

→ JacobianDet

Rating: **1321**
Contribution: **+6**

- Settings
- Blog
- Favourites
- Teams
- Submissions
- Groups
- Talks
- Contests

JacobianDet

→ Top rated

#	User	Rating
1	Petr	3353
2	Um_nik	3278
3	Syloviaely	3274
4	V--o_o--V	3236
5	tourist	3235
6	O0000000O000000...O	3201
7	Radewoosh	3197
8	fateice	3172
9	mnbvmar	3096
10	dotorya	3086

Countries | Cities | Organizations View all →

→ Top contributors

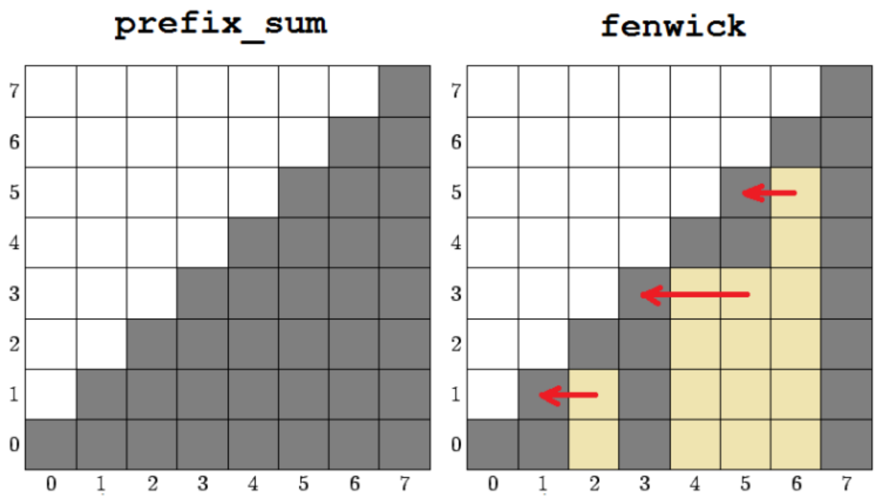
#	User	Contrib.
1	tourist	178
2	rng_58	169
3	csacademy	159
3	Petr	159
5	lewin	151
6	Swistakk	149
7	matthew99	141
7	BledDest	141
9	Errichto	139
9	adamant	139

View all →

→ Favourite groups

#	Name
1	ACM-OI

We can notice that to initialize $fenwick[i]$ we only require such elements of $prefix\_sum[j]$ where $j \leq i$. This allows to use only one array and, starting from the end, change each element from prefix sum to Fenwick tree element.



```
void init_vector_linear (vector<int> a) {
    init_empty(a.size());

    fenwick[0] = a[0];
    for (int i = 1; i < a.size(); i++) fenwick[i] = fenwick[i-1] + a[i];

    for (int i = a.size()-1; i > 0; i--) {
        int lower_i = (i & (i+1)) - 1;
        if (lower_i >= 0) fenwick[i] -= fenwick[lower_i];
    }
}
```

This way we can improve initialization time of Fenwick tree to $O(N)$ using $O(1)$ additional memory.

Of course, in most of the problems amount of queries is not much less than the size of Fenwick tree, so improving initialization time will not improve final asymptotic of your solution. But this optimization gives as ability to improve constant of the part of your solution, that works with Fenwick tree (or trees) with just a few lines of code.

What do you think about this and how do you initialize your Fenwick trees?

+46 ☆ ATSTNG 9 hours ago 1

Comments (1)

Write comment?

Jakube

7 hours ago, # | ☆ +3

I'm also using a linear time construction for the Fenwick tree (although I haven't used a Fenwick tree in ages). Learned the method from Is it possible to build a Fenwick tree in O(n)?. This method approaches the construction of the array from the exact opposite direction.

→ Reply

View all →

→ Find user

Handle:

Find

→ Recent actions

Vovuh → Codeforces Round #479 (Div. 3)

Vovuh → Codeforces Round #479 (Div. 3) Editorial

redocyz → Google Code Jam Round 1C

I_LOVE_THU_HUYEN → Did Codefights remove 1-on-1 challenge?

.o. → Invitation to CodeChef May Long Challenge 2018!

FieryPhoenix → Requested feature for contest proposals

mostafa.saad.fci → Online Free Problem-Solving Coaching

vlecomte → Original Gym contest: Geometry Special 2018

MikeMirzayanov → Codeforces: Updates in rating and rounds

Rollo → Codeforces Round #478 (Div. 2), Editorial

XMORE → 280 Div1-C problem help

Al.Cash → Efficient and easy segment trees

ATSTNG → Fenwick tree: initialization in O(N)

competitiveprogramming → Test failure at 325th Line - No clue on how to debug it

jinnatul → My programming problem solving blog

Roman1 → DrMemory in "hack" mode

bakrocker → Need Information

nikizakr → Looking for Probability Library CPP

AminAnvari → LCA problems

frankenstein → UVALive - 7887 (Back to the Future)

Nickolas → Upcoming TCO18 Marathon Round 1

walnutwaldo20 → Bug With Rating Changes in Contests Tab

DarthPrince → Algorithm Gym :: Data structures

tunyash → Codeforces Round #213 Editorial

E869120 → AtCoder Beginner Contest 096

Detailed →

The only programming contests Web 2.0 platform
Server time: May/07/2018 11:58:25$^{\text{UTC+5.5}}$ (d1).
Desktop version, switch to mobile version.