adamant's blog

C++ STL: Policy based data structures

By adamant, 8 years ago, translation,

Hi everyone! After a relatively long lull, I decided that ~~my contribution growing too slowly~~ the hour has come to please you with another article in the blog *:)*

2 months ago user **Perlik** wrote an article, in which he described a very interesting STL implemented data structure that allows you to quickly perform various operations with substrings. Some time after I tested it on various tasks and, unfortunately, tend to get a negative result — rope was too slow, especially when it came to working with individual elements.

For some time, I forgot about that article. Increasingly, however, I was faced with problems in which it was necessary to implement set with the ability to know ordinal number of item and also to get item by its ordinal number (ie, order statistic in the set). And then I remembered that in the comments to that article, someone mentioned about the mysterious data structure order statistics tree, which supports these two operations and which is implemented in STL (unfortunately only for the GNU C++). And here begins my fascinating acquaintance with policy based data structures, and I want to tell you about them *:)*

Let's get started. In this article I will talk about IMO the most interesting of the implemented structures — tree. We need to include the following headers:

```
#include <ext/pb_ds/assoc_container.hpp> // Common file
#include <ext/pb_ds/tree_policy.hpp> // Including tree_order_statistics_node_update
```

After closer inspection you may find that the last two files contained in the library

```
#include <ext/pb_ds/detail/standard_policies.hpp>
```

Namespace, which we will have to work in newer versions of C++ is called `__gnu_pbds;`, earlier it was called `pb_ds;`

Now let's look at the concrete structure.

The tree-based container has the following declaration:

```
	template<
	typename Key, // Key type
	typename Mapped, // Mapped-policy
	typename Cmp_Fn = std::less<Key>, // Key comparison functor
	typename Tag = rb_tree_tag, // Specifies which underlying data structure to use
	template<
	typename Const_Node_Iterator,
	typename Node_Iterator,
	typename Cmp_Fn_,
	typename Allocator_>
	class Node_Update = null_node_update, // A policy for updating node invariants
	typename Allocator = std::allocator<char> > // An allocator type
	class tree;
```

→ Pay attention

Before contest
Codeforces Round #764 (Div. 3)
5 days

Like 147 people like this. Be the first of your

→ Mumu

Rating: **979**
Contribution: **0**

Mumu

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

→ Top rated

#	User	Rating
1	tourist	3911
2	maroonrk	3606
3	MiracleFaFa	3604
4	Benq	3583
5	Radewoosh	3545
6	Miracle03	3486
7	ecnerwala	3457
8	peehs_moorhsum	3430
9	sunset	3338
10	ksun48	3334

Countries | Cities | Organizations View all →

→ Top contributors

#	User	Contrib.
1	YouKn0wWho	213
2	Monogon	202
3	Um_nik	187
4	awoo	183
5	-is-this-fft-	179
5	Errichto	179
7	sus	177
8	tourist	176
9	antontrygubO_o	172
10	maroonrk	169

View all →

→ Favourite groups

#	Name
1	Shaazzz

View all →

Experienced participants may have already noticed that if initialize the template only the first two types, we obtain almost exact copy of the container `map` . Just say, that this container can be `set` , for this you just need to specify the second argument template type as `null_type` (in older versions it is `null_mapped_type`).

By the way `Tag` and `Node_Update` are missing in `map` . Let us examine them in more detail.

`Tag` — class denoting a tree structure, which we will use. There are three base-classes provided in STL for this, it is `rb_tree_tag` (red-black tree), `splay_tree_tag` (splay tree) and `ov_tree_tag` (ordered-vector tree). Sadly, at competitions we can use only red-black trees for this because splay tree and OV-tree using linear-timed split operation that prevents us to use them.

`Node_Update` — class denoting policy for updating node invariants. By default it is set to `null_node_update` , ie, additional information not stored in the vertices. In addition, C++ implemented an update policy `tree_order_statistics_node_update` , which, in fact, carries the necessary operations. Consider them. Most likely, the best way to set the tree is as follows:

```
typedef tree<
int,
null_type,
less<int>,
rb_tree_tag,
tree_order_statistics_node_update>
ordered_set;
```

If we want to get map but not the set, as the second argument type must be used mapped type. Apparently, the tree supports the same operations as the `set` (at least I haven't any problems with them before), but also there are two new features — it is `find_by_order()` and `order_of_key()` . The first returns an iterator to the k-th largest element (counting from zero), the second — the number of items in a set that are strictly smaller than our item. Example of use:

```
    ordered_set X;
    X.insert(1);
    X.insert(2);
    X.insert(4);
    X.insert(8);
    X.insert(16);

    cout<<*X.find_by_order(1)<<endl; // 2
    cout<<*X.find_by_order(2)<<endl; // 4
    cout<<*X.find_by_order(4)<<endl; // 16
    cout<<(end(X)==X.find_by_order(6))<<endl; // true

    cout<<X.order_of_key(-5)<<endl;  // 0
    cout<<X.order_of_key(1)<<endl;   // 0
    cout<<X.order_of_key(3)<<endl;   // 2
    cout<<X.order_of_key(4)<<endl;   // 2
    cout<<X.order_of_key(400)<<endl; // 5
```

Finally I would like to say about the performance of order_statistics_tree in STL. For this, I provide the following table.

Solution\Problem	1028	1090	1521	1439
order_statistics_tree, STL	0.062	0.218	0.296	0.468
Segment tree	0.031	0.078	0.171	0.078 0.859*
Binary Indexed Tree	0.031	0.062	0.062	-

→ Find user

Handle:

Find

→ Recent actions

oussamabenhassen → IDE

RAD → Codeforces Round #161 Tutorial

Hudayar → Let's grow!

Utkarsh.25dec → Invitation to CodeChef Starters 21 (Rated for Div 2 & 3) — 5th January

rgoewedky → Prefix Sum 1-D and 2-D

acraider → [Video Tutorial] Burnside Lemma

ivan100sic → Let's solve some hard problems!

xiaowuc1 → USACO 2021-2022 First Contest

MohamedAboOkail → Codeforces Extensions

awoo → Educational Codeforces Round 116 Editorial

realMatrixCascade → I will reach IGM in one year

pritishn → Share your big brain moments from back when you were low rated.

kpw29 → New Year wishes from participants — "Cyan Rebellion 2022" survey results

deltixlab → Codeforces Deltix Round Autumn 2021 [Div.1 + Div.2]

markos → All division questions, helper for virtual participations and questions from the Ultimate Topic List

YouKn0wWho → The Ultimate Topic List (with Resources, Problems and Templates)

IgorI → Hello 2022 Editorial

maroonrk → XXI Opencup GP of Tokyo

L_ryusaki_912 → From newbie to newbie :(

SlavicG → Petition To Change Catalog's Position

dzy493941464 → Codeforces Round #FF(255) Editorial

Hisham_hates_graphs → Is there a way to improve this solution ?

IgorI → Hello 2022

Stepavly → Codeforces Round #697 (Div. 3) Editorial

perseverance_01 → [Tutorial] Bit Manipulation for Competitive Programming

Detailed →

* The final task requires direct access to the nodes of the tree for the implementation of solutions for O (mlogn). Without it, the solution works in O (mlogn*logn).

As you can see from all this , order_statistics_tree relatively little behind handwritten structures, and at times ahead of them in execution time. At the same time the code size is reduced considerably. Hence we can conclude is that order_statistics_tree — it is good and it can be used in contests.

Besides tree, I also wanted to describe here trie. However , I was confused by some aspects of its implementation, greatly limiting its usefulness in programming olympiads, so I decided not to talk about it. If anyone want he is encouraged to try to learn more about this structure by himself.

Useful links:
— Documentation of pb_ds
— Testing of pb_ds
— Using of pb_ds
— Demonstration of order_statistics_tree
— Demonstration of trie with prefix search
— Operations with intervals with handwritten update policy class
— More examples from that site

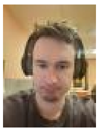
P.S. Sorry for my poor English *:)*

stl, k-th order statistic, set, map

+120 ☆ adamant 8 years ago 132

Comments (132)

Write comment?


adamant

8 years ago, # | ☆ +12

Example of trie with search of prefix range.
Problem: 1414
Solution: http://ideone.com/6VFNZl
→ Reply

low_kii_savage

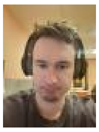
3 years ago, # ^ | ☆ ← Rev. 2 0

Is there a way of counting number of strings in the trie with a certain prefix without iterating through them all?
→ Reply

daksh_

12 months ago, # ^ | ☆ 0

You augment the trie node to also contain a number. Update this number everytime you insert a string into the trie. To get the number of strings which share the prefix, Just traverse the prefix and output the num in the ending node.
→ Reply


adamant

8 years ago, # | ☆ +17

Возможно, вам покажутся слегка нетривиальными решения деревом отрезков и деревом Фенвика, особенно, задач 1521 и 1439. Скорее всего, позже я также предоставлю статью, в которой опишу некоторые интересные способы использования этих структур, которые редко встречаются.

===

You may be wondered about how I use segment tree and binary indexed tree in my solutions, especially for problems 1521 and 1439. Most likely, later I'll provide an entry about some interesting ways of using this structures, which are quite rare.
→ Reply

8 years ago, # ^ | ☆ 0

Here it is :)

→ Reply