可持久化数据结构

丁思韬 2021.4

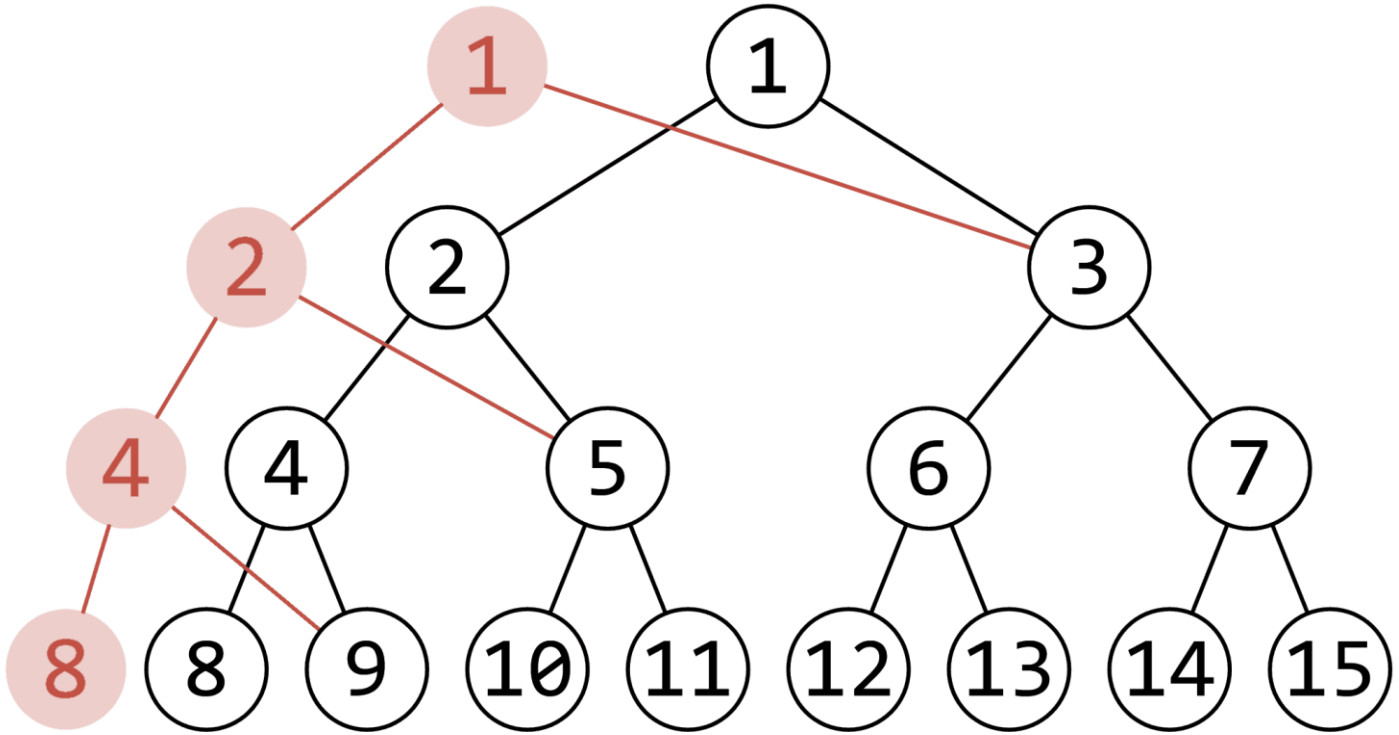
简介

- 本文围绕可持久化数据结构(Persistent data structure)展开，着重介绍可持久化线段树。
- 可持久化数据结构，是指一类可以保留每一个历史版本（初始及每次修改后的数据结构称为版本），并且支持操作的不可变特性的数据结构，分为部分可持久化和完全可持久化两类。
- 部分可持久化：所有版本都可以访问，但是只有最新版本可以修改。
- 完全可持久化：所有版本都既可以访问又可以修改。
- 在阅读可持久化线段树前，请先掌握线段树。
- 在阅读可持久化字典树前，请先掌握字典树。

可持久化线段树

- 我们先来考虑可持久化线段树的实现。
- 最简单的方法是每次修改新建一棵线段树，但会耗费大量时空。
- 我们注意到，每次修改操作最多会修改$\log$个元素，于是新建的线段树可以借用一些以前的节点。
- 对于每一次调用$update$函数，都新建一个节点，并在最后返回节点编号。不妨设接下来要进入左子树修改，那么就置新建的线段树的当前节点的左儿子为继续递归$update$后返回的新节点，右儿子为原树的右儿子。
- 对于第k次询问，记录$update$函数返回的节点编号ver_k作为根。在具体实现中，可以使用动态开点线段树。
- 总空间理论开$(\lceil n\log_2 n\rceil+3)n$，一般开$32n$。

可持久化线段树-图例



- 如图所示，以上是一个修改节点8的例子。黑树表示原树，红树表示新树。图中节点3,5,9的子树都被保留，并为新树所用。

可持久化线段树-代码

```
int upd(int pre,int L,int R,int pos,int val){
        int ind=++tot;
        if(L==R){
                tr[ind].val=val;
                return ind;
        }
        tr[ind].ls=tr[pre].ls;
        tr[ind].rs=tr[pre].rs;
        int mid=(L+R)>>1;
        if(pos<=mid)
                tr[ind].ls=upd(tr[pre].ls,L,mid,pos,val);
        else
                tr[ind].rs=upd(tr[pre].rs,mid+1,R,pos,val);
        return ind;
}
```

可持久化线段树-例题1

- Luogu P3919 [模板]可持久化线段树1（可持久化数组）
- 题意：你需要维护这样一个数组，支持如下几种操作：
- 在某个历史版本上修改某一个位置上的值
- 访问某个历史版本上的某一位置的值
- $n, m \leq 10^6$。
- 这是一道模板题，主要是提供一份模板。

可持久化线段树-代码1

```
#include<iostream>
#include<cstdio>
using namespace std;
struct Node{
	int ls,rs,val;
}tr[20000005];
int n,m,a[1000005],rt[1000005],tot;
int build(int L,int R){
	int ind=++tot;
	if(L==R){
		tr[ind].val=a[L];
		return ind;
	}
	int mid=(L+R)>>1;
	tr[ind].ls=build(L,mid);
	tr[ind].rs=build(mid+1,R);
	return ind;
}
int upd(int pre,int L,int R,int pos,int val){
	int ind=++tot;
	if(L==R){
		tr[ind].val=val;
		return ind;
	}
	tr[ind].ls=tr[pre].ls;
	tr[ind].rs=tr[pre].rs;
	int mid=(L+R)>>1;
	if(pos<=mid)
```

可持久化线段树-代码1

```
        tr[ind].ls=upd(tr[pre].ls,L,m
id,pos,val);
        else
        tr[ind].rs=upd(tr[pre].rs,mid
+1,R,pos,val);
        return ind;
}
int query(int ind,int L,int R,int
pos){
        if(L==R)
                return tr[ind].val;
        int mid=(L+R)>>1;
        if(pos<=mid)
                return
query(tr[ind].ls,L,mid,pos);
        else
                return
query(tr[ind].rs,mid+1,R,pos);
}
int main(){
        int i,j,ver,opt,x,y;
        scanf("%d%d",&n,&m);
        for(i=1;i<=n;i++)
                scanf("%d",&a[i]);
        rt[0]=1;
        build(1,n);
        for(i=1;i<=m;i++){
                scanf("%d%d%d",&ver,&op
t,&x);
```

可持久化线段树-代码1

```
        if(opt==1){
            scanf("%d",&y);
            rt[i]=upd(rt[ver],1,n,x,y);
        }
        else{
            rt[i]=rt[ver];
            printf("%d\n",query(rt[ver],1,n,x));
        }
    }
    return 0;
}
```

可持久化线段树-例题2-主席树

- Luogu P3834[模板]可持久化线段树2（主席树）
- 题意：多次询问区间内第k小值（不带修）
- $n, m \leq 2 \cdot 10^5$。
- 区间第k小值要用权值线段树解决，但此题还要维护区间$[l, r]$。
- 如何表达这个**二维的信息**？
- 考虑建立可持久化权值线段树（即主席树），将线段树中依次加入$a_1 - a_n$，并记录每个版本$ver_0 - ver_n$。
- 那么对于询问$[l, r]$，我们可以巧妙地对ver_r和ver_{l-1}的每个节点作差，并询问。

可持久化线段树-例题2-实现

- 如果朴素实现对两个线段树作差得到新的线段树，显然会超时。我们可以考虑同时遍历两个版本。在$query$函数中，传入ver_{l-1}和ver_r两个当前子树的根节点。
- 从上面的例子中可以看出，**主席树可以静态地维护二维的信息**。
- 由于主席树是以$a_1 - a_n$的加入作为修改，所以如果要修改序列中的一个节点k，会导致$ver_k - ver_n$的改变。
- 因此，主席树**不能处理动态的问题**。

可持久化线段树-代码2

```
int query(int ind,int pre,int L,int R,int k){
	if(L==R)
		return L;
	int mid=(L+R)>>1,tmp=tr[tr[ind].ls].s-tr[tr[pre].ls].s;
	if(tmp>=k)
		return query(tr[ind].ls,tr[pre].ls,L,mid,k);
	return query(tr[ind].rs,tr[pre].rs,mid+1,R,k-tmp);
}
```

可持久化线段树-例题3

- BZOJ#3585 mex（改）
- 题意：多次询问一个区间内最小没有出现过的自然数。强制在线
- $n, m \leq 2 \cdot 10^5$。
- 提示：离线做法只需要用到权值线段树，不用主席树。

可持久化线段树-例题3

- 权值线段树每个节点($[L,R]$)考虑维护$L-R$这$R-L+1$个数在序列中出现位置的最小值（若未出现则为0）。
- 在ver_r上询问。对于某个节点，若结果小于l，说明存在一个数未在序列中出现或只在l前出现，也即存在没有出现过的自然数。
- 那么在主席树上二分最大的满足所有自然数都出现过的区间$[1,r]$即可，$r+1$即为答案。
- 代码略。
- 由此可见，**主席树可以将一类只能离线的问题在线解决**。

可持久化线段树-练习题

- [POI2014] couriers
- [CQOI2015] 任务查询系统
- [SDOI2009] HH的项链（此题当作强制在线做）
- [CTSC2018] 混合果汁
- The Preliminary Contest for ICPC Asia Xuzhou 2019 I. query
- [湖南集训] 谈笑风生
- [SCOI2016] 美味（先来复习一下0/1trie）

可持久化字典树

- 类似于可持久化线段树。
- 对于每一次调用$update$函数，不妨设接下来要进入左子树修改，那么就置新建的字典树的当前节点的左儿子为新节点，右儿子为原树的右儿子。
- 在应用中，o/1 trie较为常用。

可持久化字典树-例题1

- 对一个长度为n的序列a维护以下操作：
- 在序列末尾新建一个元素
- 给定l, r, k，求$\max_{i=l}^{r}(k \text{ xor } a_i)$
- $n \leq 3 \cdot 10^5$。

- 类似于主席树，考虑建立可持久化权值字典树，将字典树中依次加入$a_1 - a_n$，并记录每个版本$ver_0 - ver_n$。
- 那么对于询问$[l, r]$，我们可以巧妙地对ver_r和ver_{l-1}的每个节点作差，并询问。

可持久化字典树-练习题

- Luogu P4735 最大异或和
- [TJOI2018] 异或
- CF916D Jamie and To-do List
- [十二省联考2019] 异或粽子