alpc60 ACM / ICPC程序设计

成长的路.....源

C++博客 首页 新随笔 新文章 联系 聚合 管理

< 2008年9月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

常用链接

- 我的随笔
- 我的评论
- 我参与的随笔

留言簿(7)

- 给我留言
- 查看公开留言
- 查看私人留言

随笔分类

- ACM/ICPC(20)

随笔档案

- 2008年9月 (3)
- 2008年8月 (4)
- 2008年7月 (1)
- 2008年6月 (2)
- 2008年5月 (2)
- 2008年4月 (5)
- 2007年9月 (3)

文章档案

树状数组学习心得

POJ 3321 Apple Tree

http://acm.pku.edu.cn/JudgeOnline/problem?id=3321

POJ 2481 Cows

http://acm.pku.edu.cn/JudgeOnline/problem?id=2481

POJ 2155 Matrix

http://acm.pku.edu.cn/JudgeOnline/problem?id=2155

今天在POJ淘了这几道题目，学习了一下树状数组的用法，跟大家分享一下心得。可以把树状数组看成一种数据结构，对于一个数组，如果有多次操作，每次的操作有两种：1、修改数组中某一元素的值，2、求和，求数组元素a[1]+a[2]+...a[num]的和。这是树状数组最基本的应用了，用树状数组可以实现O(log(n))的修改以及O(log(n))的求和。当然用线段树完全可以胜任这些计算，但是线段树写起来代码比较长，并且线段树要占用2*n大小的空间。下面先给出树状数组的三个基本操作的函数：

```
int lowbit(int k)
{
    return k&(-k);
}
```

//lowbit函数是计算k的二进制位最低位不为0的数字的权值。

```
void Modify(int num, int v)
```

2008年4月 (1)

alpcs

alpc02
alpc12
alpc16
alpc40
alpc47
alpc55
alpc62
zzningxp

最新随笔

1. 树状数组学习心得
2. 数学中的爆搞题
3. 最小点覆盖 POJ 2060 Taxi Cab Scheme
4. 从希望走向失望
5. POJ 1112 Team Them Up! 求补图，连通分量，DP
6. 个人比赛时心理素质不好
7. POJ 2047
8. 7月16-7月21荒废的一周！！！
9. POJ 2335 浮点数的gcd
10. Written to alpcs in Normal

搜索

搜索

最新评论 XML

1. re: A*搜索求最短路
“使得由s1生成的每状态s2，都有h(s1)<=h(s2)”应该是“都有h(s1)<=h(s2)+c(s1,s2)”吧
--MJ
2. re: 网络流
看来主要是建模的问题啊
--yyz_max@163.com
3. re: 网络流
很好很强大
--yyz_max@163.com
4. re: 树状数组学习心得
评论内容较长,点击标题查看
--Mr.moon
5. re: A*搜索求最短路
评论内容较长,点击标题查看
--pandy

```
{
    while(num <= n)
    {
        c[num]+=v;
        num+=lowbit(num);
    }
}

//Modify函数是往数组c中修改值，更新整个数组的值，实现了操作1;

int Sum(int num)
{
    int ans=0;
    while(num > 0)
    {
        ans+=c[num];
        num-=lowbit(num);
    }
    return ans;
}

//Sum函数返回数组元素a[1]+a[2]+...a[num]的和，实现操作2;
```

树状数组的巧妙之处在于对于数组下标的二进制的操作，假设a[1...N]为原数组,定义c[1...N]为对应的树状数组:

c[i] = a[i - 2^k + 1] + a[i - 2^k + 2] + ... + a[i]

其中k为i的二进制表示末尾0的个数,所以2^k即为i的二进制表示的最后一个1的权值.

可以把树状数组看作是把数组分成了若干个2^k大小的空间。对于一个下标i，c[i]的值是由i/(lowbit(i))个数组元素的值所组成的，每次步进的单位是k=lowbit(i)，这个有点像二分归并的思想！这样就可以实现O(log(n))的修改和查询。

下面是树状数组的具体应用:

3321 Apple Tree 一棵树上长了苹果，每一个树枝节点上有长苹果和不长苹果两种状态，两种操作，一种操作能够改变树枝上苹果的状态，另一种操作询问某一树枝节点一下的所有的苹果有多少。具体做法是做一次dfs，记下每个节点的开始时间low[i]和结束时间high[i]，那么对于i节点的所有子孙的开始时间和结束时间都应位于low[i]和high[i]之间，另外用一个数组c[i]记录附加在节点i上的苹果的个数，然后用树状数组统计low[i]到high[i]之间的附加苹果总数。这里用树状数组统计区间可以

6. re: POJ 2335 浮点数的gcd
敬爱的……都不更新了
--yumi
7. re: POJ 2335 浮点数的gcd
大牛啊。。最近我都在学习你的blog呢。。。
写的不错啊！！！
--11
8. re: 从希望走向失望
呵呵
--11
9. re: 从希望走向失望
这个这个，玩玩也是不错的，权作娱乐而已，要学还是要自己学的
--alpc40
10. re: POJ 1112 Team Them Up! 求补图，连通分量，DP
不愧是国防科技大学的大牛！！
--ssadwlll

阅读排行榜
1. 动态规划专题(2052)
2. 树状数组学习心得(1884)
3. 迭代加深搜索(1821)
4. A*搜索求最短路(1789)
5. POJ 1112 Team Them Up! 求补图，连通分量，DP(1558)

评论排行榜
1. 从希望走向失望(8)
2. 不用IDE的结果(6)
3. A*搜索求最短路(6)
4. 动态规划专题(4)
5. POJ 2335 浮点数的gcd(3)

用Sum(high[i])-Sum(low[i]-1)来计算。

```
#include <stdio.h>
#include <string.h>
#include <vector>
using namespace std;

//vector<int> g[100005];
struct Node
{
    int v;
    struct Node *next;
}g[100005];
int n,m,cnt,low[100005],high[100005],c[100005],flag[100005];
bool mark[100005];

void dfs(int v)
{
    struct Node *p=g[v].next;
    mark[v]=true;
    cnt++;
    low[v]=cnt;
    while(p)
    {
        if(!mark[p->v])
            dfs(p->v);
        p=p->next;
    }
    high[v]=cnt;
}
int lowbit(int k)
{
    return k&(-k);
}
void Modify(int num, int v)
{
    while(num <= n)
    {
        c[num]+=v;
        num+=lowbit(num);
    }
}
int Sum(int num)
{
    int ans=0;
    while(num > 0)
```

```
45  {
46      ans+=c[num];
47      num-=lowbit(num);
48   }
49   return ans;
50 }
51
52  int main()
53 {
54   int i,j,a,b,ans;
55   char temp[10];
56   struct Node *p;
57   //freopen("in.txt","r",stdin);
58   scanf("%d",&n);
59   memset(g,0,sizeof(g));
60   for(i=1; i<n; i++)
61   {
62      scanf("%d%d",&a,&b);
63      p=new Node;
64      p->next=g[a].next;
65      p->v=b;
66      g[a].next=p;
67      p=new Node;
68      p->next=g[b].next;
69      p->v=a;
70      g[b].next=p;
71   }
72   memset(mark,false,sizeof(mark));
73   memset(c,0,sizeof(c));
74   for(i=1; i<=n; i++)
75      flag[i]=1;
76   cnt=0;
77   dfs(1);
78   scanf("%d",&m);
79   while(m--)
80   {
81      scanf("%s",temp);
82      if(temp[0] == 'Q')
83      {
84         scanf("%d",&a);
85         ans=high[a]-low[a]+1+Sum(high[a])-Sum(low[a]-1);
86         printf("%d\n",ans);
87      }
88      else
89      {
90         scanf("%d",&a);
91        if(flag[a]) Modify(low[a],-1);
92         else Modify(low[a],1);
```

```
            flag[a]^=1;
        }
    }
    return 0;
}
```

2481 Cows 给n个区间[Si,Ei]，区间[Sj,Ej]< [Si,Ei] 有 Si <= Sj and Ej <= Ei and Ei - Si > Ej – Sj。按y坐标从小到达，x坐标从大到小的顺序排序，然后从后往前扫描，记录i之前所有的j区间Sj<Si的个数，这个用树状数组实现。扫描一遍可得出结果。

```
#include <stdio.h>
#include <string.h>
#include <algorithm>
using namespace std;

struct P
{
    int x,y,id;
}p[100005];
int n,a[100005],max_n,b[100005];

int lowbit(int k)
{
    return k&(-k);
}
void Modify(int num, int v)
{
    while(num <= max_n)
    {
        a[num]+=v;
        num+=lowbit(num);
    }
}
int Sum(int num)
{
    int ans=0;
    if(num <= 0) return 0;
    while(num)
    {
        ans+=a[num];
        num-=lowbit(num);
    }
```

```
    return ans;
}
 bool operator <(const P a, const P b)
{
  if(a.y == b.y) return a.x > b.x;
  return a.y < b.y;
}

int main()
{
  int i;
  //freopen("in.txt","r",stdin);
  while(scanf("%d",&n), n)
  {
    max_n=0;
    for(i=0; i<n; i++)
    {
      scanf("%d%d",&p[i].x,&p[i].y);
      p[i].id=i;
      p[i].x++;
      p[i].y++;
      if(p[i].y > max_n) max_n=p[i].y;
    }
    sort(p,p+n);
    memset(a,0,sizeof(a));
    for(i=n-1; i>=0; i--)
    {
      if(i != n-1 && p[i].y == p[i+1].y && p[i].x == p[i+1].x)
        b[p[i].id]=b[p[i+1].id];
      else
        b[p[i].id]=Sum(p[i].x);
      Modify(p[i].x,1);
    }
    for(i=0; i<n; i++)
    {
      if(i) printf(" ");
      printf("%d",b[i]);
    }
    printf("\n");
  }
  return 0;
}
```

2155 Matrix 有n*n的0，1矩阵，两种操作，1、翻转矩形（x1,y1）(x2,y2)的值，2、输出位置为(x,y)矩阵处的值。先考虑一维的情况，设A<B，那么要翻转[A,B]之

间的值，可以分解为两步操作，先翻转[1,A-1]，然后再翻转[1,B]，其中翻转的次数就可以用树状数组来计算。然后再将次操作扩展到二维的情形，只需将x方向与y方向套成一个二重循环即可。从这里我们也可以看到树状数组处理类似问题时比线段树的优越性。从代码的长度，空间消耗上面，树状数组都有明显的优势。

```
#include <stdio.h>
#include <string.h>

int a[1005][1005],n,m;

int lowbit(int k)
{
    return k&(-k);
}
void Modify(int x1, int y1, int x2, int y2)
{
    int i,j;
    for(i=x1-1; i>0; i-=lowbit(i))
    {
        for(j=y1-1; j>0; j-=lowbit(j))
        {
            a[i][j]^=1;
        }
        for(j=y2; j>0; j-=lowbit(j))
        {
            a[i][j]^=1;
        }
    }
    for(i=x2; i>0; i-=lowbit(i))
    {
        for(j=y1-1; j>0; j-=lowbit(j))
        {
            a[i][j]^=1;
        }
        for(j=y2; j>0; j-=lowbit(j))
        {
            a[i][j]^=1;
        }
    }
}
int Sum(int x, int y)
{
    int ans=0,i,j;
    for(i=x; i<=n; i+=lowbit(i))
    {
        for(j=y; j<=n; j+=lowbit(j))
            ans^=a[i][j];
    }
```

```
    return ans;
}

int main()
{
    int i,j,x1,x2,y1,y2,cases,ic=0;
    char temp[10];
    //freopen("in.txt","r",stdin);
    scanf("%d",&cases);
    while(cases--)
    {
      if(ic++) printf("\n");
      scanf("%d%d",&n,&m);
      memset(a,0,sizeof(a));
      while(m--)
      {
        scanf("%s",temp);
       if(temp[0] == 'C')
        {
          scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
          Modify(x1,y1,x2,y2);
        }
        else
        {
          scanf("%d%d",&x1,&y1);
          printf("%d\n",Sum(x1,y1));
        }
      }
  }
  return 0;
}
```

posted on 2008-09-24 16:35 飞飞 阅读(1885) 评论(1) 编辑 收藏 引用 所属分类: ACM/ICPC

FeedBack:

re: 树状数组学习心得 2009-07-04 16:43 | Mr.moon

2481 cows if(p[i].y > max_n) max_n=p[i].y;

好像应该是p[i].x

回复 更多评论

刷新评论列表