小白菜又菜

个人资料

小白菜又菜

访问： 503445次
积分： 14279
等级： BLOG 7
排名： 第489名

原创： 965篇
转载： 0篇
译文： 0篇
评论： 125条

文章搜索

文章分类

杂乱无章 (31)
APOC-UVa (22)
解题报告 (931)
入门题 (234)
初级DS (27)
并查集 (14)
线段树 (7)
图论 (149)
计算几何 (83)
字符串 (96)
动态规划（DP） (168)
数论 (173)
分治 (32)

文章存档

2016年05月 (11)
2016年04月 (16)
2016年03月 (13)
2016年01月 (15)
2015年12月 (16)

UVa 11402 - Ahoy, Pirates!

2012-11-18 02:05 1445人阅读 评论(0) 收藏 举报

分类： 解题报告（930） 线段树（6）

题目：给定一个01串，对串的某些区间进行：变0、变1、取反和查询操作。

分析：线段树、离散化。经典的线段染色的变形。区间大小1000000，查询操作1000，如果直接建立1000000的区间的线段树，在上面操作会TLE。因为查询只有1000个，那么利用查询的端点将区间划分成最多4001段（每个端点各是一段、相邻端点间的部分各是一段）。对于所有操作，每段都可以看做一个整体。因此，通过离散化可以建立一个4001区间的区间树，树中每个节点代表一个不等长区间。每个节点除了端点值和左右两颗子树指针外还包含：区间中0的个数（sum[0]）、区间中1的个数（sum[1]）以及区间进行的操作（color）。其中区间操作分为：置0（取值0）、置1（取值1）、取反（取值2）和无操作（取值-1）四种。增加这个域是为了避免每次更新要将所有子节点全部更新。有了区间操作域，每次更新到整个节点时（a==Lvalue&&b==Rvalue），把操作到本节点结束，子树不用操作。直到每次更新子区间时，当前区间的状态要转移到两颗子树上（代码中的Deal操作）。每次更新先继承父区间的状态，之后再更新状态保证了操作的先后顺序。Deal具体操作：1.置0，color = 0、sum[0] = 区间大小；2.置1，color = 1、sum[0] = 0、sum[1] = 区间大小；3.取反，swap(sum[0],sum[1])、color = 1-color（如果color == 1则color = 0 、如果color == 0则color = 1、如果color == -1则color = 2、如果color == 2则color = -1都满足上式）。

注意：输入数据中的换行会导致RE，最好用%s读入操作对应的字母。

[cpp] view plain copy print ?

```cpp
#include <iostream>
#include <cstdlib>
#include <cstring>
#include <cstdio>

using namespace std;

//输入数据
char  C[ 1005 ];
int   X[ 1005 ];
int   Y[ 1005 ];

//操作区间端点
int   S[ 2005 ];

//离散化节点数据
int   L[ 4005 ];
int   R[ 4005 ];
int   Sum0[ 4005 ];
int   Sum1[ 4005 ];

//segment_tree__begin
typedef struct tnode
{
```

展开

阅读排行

AC自动机总结 (6147)
UVa 455 - Periodic Strin (3705)
UVa 101 - The Blocks Pr (3491)
UVa 202 - Repeating De (3167)
UVa 145 - Gondwanalar (2551)
UVa 1585 - Score (2357)
UVa 227 - Puzzle (2245)
UVa 10814 - Simplifying (2003)
UVa 10158 - War (1975)
UVa 10301 - Rings and (1965)

评论排行

UVa 227 - Puzzle (20)
大学总结 (12)
AC自动机总结 (8)
UVa 10057 - A mid-sum (7)
UVa 101 - The Blocks Pr (7)
UVa150题O(∩_∩)O~ (6)
UVa 350题记录(☉_☉) (6)
UVa 11384 - Help is nee (5)
UVa 10684 - The jackpo (4)
UVa 350 - Pseudo-Ranc (4)

blogs链接

自己的新网站
WilliamKyle's Home
Jackchess in zzu
祝你好运！
小媛在努力~
小袁在努力~
FOOKWOOD
cgangEE

最新评论

大学总结
sparksnail: 经历好丰富。

大学总结
小白菜又菜: @qq_30368701:你好，O(∩_∩)O~不好意思这几天没怎么上blog，才看见。

大学总结
帝狱大大: 老学长老学长，我是小15

大学总结
帝狱大大: 发现老学长一枚，郑大信工15级

UVa 227 - Puzzle
小白菜又菜: @Oblivion1221:0是字符串的结束标志，在字符的后面赋值为null（数值0）；第34行，用...

UVa 227 - Puzzle
Oblivion1221: 你好 想请教下31行cmd = 0;的意思

UVa 11059 - Maximum Product
小白菜又菜: @qq_21497049:写dp的时候喜欢i从1开始，这样变量和状态好对应。

UVa 11059 - Maximum Product
Kritio: 大神为什么 i 要从 1开始？

```
25.     tnode* Lchild;
26.     tnode* Rchild;
27.     int    Lvalue;
28.     int    Rvalue;
29.     int    Color;//记录节点颜操作：0.置0；1.置1；2.取反；-1.无操作
30.     int    Sum[2];
31. }tnode;
32. tnode Node[ 8008 ];
33. 
34. tnode* Root;
35. int    Count;
36. tnode* buildtree( int a, int b ) {
37.     tnode* np = &Node[ Count ++ ];
38.     np->Lvalue = L[a];
39.     np->Rvalue = R[b];
40.     np->Color  = -1;
41.     if ( a < b ) {
42.         np->Lchild = buildtree( a, (a+b)/2 );
43.         np->Rchild = buildtree( (a+b)/2+1, b );
44.         np->Sum[0] = np->Lchild->Sum[0]+np->Rchild->Sum[0];
45.         np->Sum[1] = np->Lchild->Sum[1]+np->Rchild->Sum[1];
46.     }else {
47.         np->Lchild = NULL;
48.         np->Rchild = NULL;
49.         np->Sum[0] = Sum0[a];
50.         np->Sum[1] = Sum1[a];
51.     }
52.     return np;
53. }
54. void segment_tree( int a, int b ) {
55.     Count = 0;
56.     Root = buildtree( a, b );
57. }
58. void Deal( tnode* r, int v ) {
59.     /* 核心操作，节点状态更新 */
60.     if ( v == 2 ) {
61.         r->Color = 1 - r->Color;//相反的操作编号加和为1
62.         swap( r->Sum[0], r->Sum[1] );
63.     }else {
64.         r->Color = v;
65.         r->Sum[v] = r->Rvalue-r->Lvalue+1;
66.         r->Sum[!v] = 0;
67.     }
68. }
69. void Insert( tnode*r, int a, int b, int v ) {
70.     if ( r->Lvalue == a && r->Rvalue == b ) {
71.         Deal( r, v );return;
72.     }
73.     if ( r->Color != -1 ) {
74.         Deal( r->Lchild, r->Color );
75.         Deal( r->Rchild, r->Color );
76.         r->Color = -1;
77.     }
78.     if ( b <= r->Lchild->Rvalue )
79.         Insert( r->Lchild, a, b, v );
80.     else if ( a >= r->Rchild->Lvalue )
81.         Insert( r->Rchild, a, b, v );
82.     else {
83.         Insert( r->Lchild, a, r->Lchild->Rvalue, v );
84.         Insert( r->Rchild, r->Rchild->Lvalue, b, v );
85.     }
86.     r->Sum[0] = r->Lchild->Sum[0]+r->Rchild->Sum[0];
87.     r->Sum[1] = r->Lchild->Sum[1]+r->Rchild->Sum[1];
88. }
89. int Query( tnode* r, int a, int b ) {
90.     if ( r->Color == 1 || r->Color == 0 )
91.         return (b-a+1)*r->Color;
92.     if ( r->Lvalue == a && r->Rvalue == b )
93.         return r->Sum[1];
94.     int v = 0;
95.     if ( b <= r->Lchild->Rvalue )
96.         v = Query( r->Lchild, a, b );
97.     else if ( a >= r->Rchild->Lvalue )
98.         v = Query( r->Rchild, a, b );
99.     else
100.         v = Query( r->Lchild, a, r->Lchild->Rvalue )+
101.             Query( r->Rchild, r->Rchild->Lvalue, b );
102.     if ( r->Color == -1 ) return v;
103.     else return b-a+1-v;
```

```
104. }
105. void Insert( int a, int b, int v ) {
106.     Insert( Root, a, b ,v );
107. }
108. int Query( int a, int b ) {
109.     return Query( Root, a, b );
110. }
111. //segment_tree__end
112. 
113. char data[ 1024005 ];
114. char save[ 1000 ];
115. int  number;
116. 
117. //统计每个区间中0、1个数
118. int count01( int x, int y, char c )
119. {
120.     int sum = 0;
121.     for ( int i = x ; i <= y ; ++ i )
122.         if ( data[i] == c )
123.             sum ++;
124.     return sum;
125. }
126. 
127. //建立离散化区间
128. int addsegment( int left, int now )
129. {
130.     if ( left < now ) {
131.         L[number] = left;
132.         R[number] = now-1;
133.         number ++;
134.     }
135.     L[number] = R[number] = now;
136.     number ++;
137.     return now+1;
138. }
139. 
140. int cmp( const void *a, const void *b )
141. {
142.     return *((int *)a) - *((int *)b);
143. }
144. 
145. int main()
146. {
147.     int  T,N,M,Q;
148.     while ( scanf("%d",&T) != EOF )
149.     for ( int t = 1 ; t <= T ; ++ t ) {
150.         int count = 0;
151.         scanf("%d",&N);
152.         for ( int i = 0 ; i < N ; ++ i ) {
153.             scanf("%d%s",&M,save);
154.             int len = strlen(save);
155.             for ( int j = 0 ; j < M ; ++ j ) {
156.                 strcpy( &data[count], save );
157.                 count += len;
158.             }
159.         }
160. 
161.         scanf("%d",&Q);
162.         for ( int i = 0 ; i < Q ; ++ i ) {
163.             scanf("%s%d%d",&C[i],&X[i],&Y[i]);
164.             if ( X[i] >= count ) X[i] = count-1;
165.             if ( Y[i] >= count ) Y[i] = count-1;
166.         }
167. 
168.         //离散化
169.         for ( int i = 0 ; i < Q ; ++ i ) {
170.             S[i+0] = X[i];
171.             S[i+Q] = Y[i];
172.         }
173.         qsort( S, 2*Q, sizeof( int ), cmp );
174. 
175.         int s = 0; number = 0;
176.         //最左区间
177.         if ( S[0] != 0 )
178.             s = addsegment( s, S[0] );
179.         else s = addsegment( 0, 0 );
180.         //中间区间
181.         for ( int i = 1 ; i < 2*Q ; ++ i ) {
182.             if ( S[i] == S[i-1] ) continue;
```

```
            s = addsegment( s, S[i] );
        }
        //最右区间
        if ( S[2*Q-1] != count-1 )
            s = addsegment( s, count-1 );

        //统计每个区间中0、1个数
        for ( int i = 0 ; i < number ; ++ i ) {
            Sum0[i] = count01( L[i], R[i], '0' );
            Sum1[i] = count01( L[i], R[i], '1' );
        }

        //建立离散化线段树
        segment_tree( 0, number-1 );

        printf("Case %d:\n",t);
        int query = 1;
        for ( int i = 0 ; i < Q ; ++ i ) {
            if ( C[i] == 'F' ) Insert( X[i], Y[i], 1 );
            if ( C[i] == 'E' ) Insert( X[i], Y[i], 0 );
            if ( C[i] == 'I' ) Insert( X[i], Y[i], 2 );
            if ( C[i] == 'S' ) printf("Q%d: %d\n",query++,Query( X[i], Y[i] ));
        }
    }
    return 0;
}
```

大数据生成程序：

[cpp] view plain copy print ?

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main()
{
    freopen("data.in","w",stdout);
    srand(time(NULL));

    int T = 10;
    printf("%d\n",T);
    for ( int t = 0 ; t < T ; ++ t ) {
        int m = rand()%100+1;
        int sum = 0;
        printf("%d\n",m);
        for ( int i = 0 ; i < m ; ++ i ) {
            int n = rand()%200+1;
            int l = rand()%50+1;
            printf("%d\n",n);
            for ( int j = 0 ; j < l ; ++ j )
                printf("%d",rand()%2);
            printf("\n");
            sum += n*l;
        }
        int q = rand()%1000+1;
        printf("%d\n",q);
        for ( int i = 0 ; i < q ; ++ i ) {
            switch( rand()%4 ) {
                case 0: printf("F ");break;
                case 1: printf("E ");break;
                case 2: printf("I ");break;
                case 3: printf("S ");break;
            }
            int x = rand()%sum;
            int y = rand()%sum;
            if ( x > y ) swap( x, y );
            printf("%d %d\n",x,y);
        }
    }
    return 0;
}
```

上一篇 UVa 100 - The 3n + 1 problem
下一篇 UVa 136 - Ugly Numbers

我的同类文章

解题报告（930） 线段树（6）

- UVa 10400 - Game Show … 2016-05-30 阅读 1
- UVa 321 - The New Villa 2016-05-26 阅读 8
- UVa 10528 - Major Scales 2016-05-24 阅读 4
- UVa 941 - Permutations 2016-05-17 阅读 10
- UVa 10123 - No Tipping 2016-05-16 阅读 9
- UVa 10160 - Servicing Stat… 2016-05-12 阅读 18
- UVa 10959 - The Party, Part I 2016-05-29 阅读 2
- UVa 10279 - Mine Sweeper 2016-05-25 阅读 4
- UVa 310 - L--system 2016-05-21 阅读 13
- UVa 188 - Perfect Hash 2016-05-17 阅读 6
- UVa 11616 - Roman Nume… 2016-05-13 阅读 11

更多文章

参考知识库

Hadoop知识库
1216 关注 | 438 收录

Apache Spark知识库
3209 关注 | 269 收录

猜你在找

- iOS8-Swift开发教程
- 大数据编程语言：Java基础
- 最适合自学的C++基础视频
- Spark 1.x大数据平台
- HTML 5移动开发从入门到精通
- Uva 11402 Ahoy Pirates 线段树成段更新
- uva 11402 ahoy pirates
- UVA 11042 Ahoy Pirates
- UVA - 11402线段树+区间离散
- 博弈学习三 HDU 1538 A Puzzle for Pirates经典的海

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap