**摸清一、二维树状数组与线段树 收藏**

先一维：

通过修改点，统计区间.

输入Q 5

输入C 2 6

5在这个区间里了。所以输出1.

再输入Q 3

输入C 2 6

输入当然是2.

view plaincopy to clipboardprint?

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define size 1001

typedef struct

{

int x,y;

}node;

int tree[size];

int n;

inline int lowbit(int x)

{return x&(-x);}

void updata(int x)

{

while(x>0)

{

//printf("%d %d\n",p22.x,p22.y);

tree[x]++;

x-=lowbit(x);

}

}

int read(int x)

{

int re=0;

while(x<=n)

{

re+=tree[x];

x+=lowbit(x);

}

return re;

}

int main()

{

int x,t,a,b,tt;

node p1,p2;

char c[2];

scanf("%d",&x);

while(x--)

{

scanf("%d%d",&n,&t);

memset(tree,0,sizeof(tree));

while(t--)

{

scanf("%s",c);

if(c[0]=='C')

{

scanf("%d%d",&p1.x,&p1.y);

printf("%d",read(p1.x)-read(p1.y+1));

}

else

{

scanf("%d",&a);

updata(a);

}

}if(x)printf("\n");

}

return 0;

}

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define size 1001

typedef struct

{

int x,y;

}node;

int tree[size];

int n;

inline int lowbit(int x)

{return x&(-x);}

void updata(int x)

{

while(x>0)

{

//printf("%d %d\n",p22.x,p22.y);

tree[x]++;

x-=lowbit(x);

}

}

int read(int x)

{

int re=0;

while(x<=n)

{

re+=tree[x];

x+=lowbit(x);

}

return re;

}

int main()

{

int x,t,a,b,tt;

node p1,p2;

char c[2];

scanf("%d",&x);

while(x--)

{

scanf("%d%d",&n,&t);

memset(tree,0,sizeof(tree));

while(t--)

{

scanf("%s",c);

if(c[0]=='C')

{

scanf("%d%d",&p1.x,&p1.y);

printf("%d",read(p1.x)-read(p1.y+1));

}

else

{

scanf("%d",&a);

updata(a);

}

}if(x)printf("\n");

}

return 0;

}

通过修改区间，统计点。

翻纸牌差不多。

There is an array of n cards. Each card is putted face down on table. You have two queries: 1. T i j (turn cards from index i to index j, include i-th and j-th card - card which was face down will be face up; card which was face up will be face down) 2. Q i (answer 0 if i-th card is face down else answer 1)

This has solution for each query (and 1 and 2) has time complexity O(log n).

In array f (of length n + 1) we will store each query T (i , j) - we set f[i]++ and f[j + 1]--.

For each card k between i and j (include i and j) sum f[1] + f[2] + ... + f[k] will be increased for 1, for all others will be same as before (look at the image for clarification).

So our solution will be described sum (which is same as cumulative frequency) module 2.

参考代码：

view plaincopy to clipboardprint?

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define size 1001

typedef struct

{

int x,y;

}node;

int tree[size];

int n;

inline int lowbit(int x)

{return x&(-x);}

void updata(int x)

{

while(x>0)

{

//printf("%d %d\n",p22.x,p22.y);

tree[x]++;

x-=lowbit(x);

}

}

int read(int x)

{

int re=0;

while(x<=n)

{

re+=tree[x];

x+=lowbit(x);

}

return re;

}

int main()

{

int x,t,a,b,tt;

node p1,p2;

char c[2];

scanf("%d",&x);

while(x--)

{

scanf("%d%d",&n,&t);

memset(tree,0,sizeof(tree));

while(t--)

{

scanf("%s",c);

if(c[0]=='C')

{

scanf("%d%d",&p1.x,&p1.y);

updata(p1.x-1);

updata(p1.y);

}

else

{

scanf("%d",&a);

tt = read(a);

if(!(tt%2))

{

printf("0\n");

}

else

printf("1\n");

}

}if(x)printf("\n");

}

return 0;

}

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define size 1001

typedef struct

{

int x,y;

}node;

int tree[size];

int n;

inline int lowbit(int x)

{return x&(-x);}

void updata(int x)

{

while(x>0)

{

//printf("%d %d\n",p22.x,p22.y);

tree[x]++;

x-=lowbit(x);

}

}

int read(int x)

{

int re=0;

while(x<=n)

{

re+=tree[x];

x+=lowbit(x);

}

return re;

}

int main()

{

int x,t,a,b,tt;

node p1,p2;

char c[2];

scanf("%d",&x);

while(x--)

{

scanf("%d%d",&n,&t);

memset(tree,0,sizeof(tree));

while(t--)

{

scanf("%s",c);

if(c[0]=='C')

{

scanf("%d%d",&p1.x,&p1.y);

updata(p1.x-1);

updata(p1.y);

}

else

{

scanf("%d",&a);

tt = read(a);

if(!(tt%2))

{

printf("0\n");

}

else

printf("1\n");

}

}if(x)printf("\n");

}

return 0;

}

上面是两个树状数组一维的例子，一个是修改点，一个是修改区间。

二维的树状数组运用：poj 2155 matrix

翻牌的二维使用：

代码：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define size 1001

typedef struct

{

int x,y;

}node;

int tree[size][size];

int n;

inline int lowbit(int x)

{return x&(-x);}

void updata(int x,int y)

{

int ty = y;

while(x>0)

{

y=ty;

while(y>0)

{

//printf("%d %d\n",p22.x,p22.y);

tree[x][y]++;

y-=lowbit(y);

}

x-=lowbit(x);

}

}

int read(int x,int y)

{

int re=0,tm=y;

while(x<=n)

{

y=tm;

while(y<=n)

{

re+=tree[x][y];

y+=lowbit(y);

}

x+=lowbit(x);

}

return re;

}

int main()

{

int x,t,a,b,tt;

node p1,p2;

char c[2];

scanf("%d",&x);

while(x--)

{

scanf("%d%d",&n,&t);

memset(tree,0,sizeof(tree));

while(t--)

{

scanf("%s",c);

if(c[0]=='C')

{

scanf("%d%d%d%d",&p1.x,&p1.y,&p2.x,&p2.y);

updata(p2.x,p2.y);

updata(p1.x-1,p2.y);

updata(p2.x,p1.y-1);

updata(p1.x-1,p1.y-1);

}

else

{

scanf("%d%d",&a,&b);

tt = read(a,b);

if(!(tt%2))

{

printf("0\n");

}

else

printf("1\n");

}

}if(x)printf("\n");

}

return 0;

}

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define size 1001

typedef struct

{

int x,y;

}node;

int tree[size][size];

int n;

inline int lowbit(int x)

{return x&(-x);}

void updata(int x,int y)

{

int ty = y;

while(x>0)

{

y=ty;

while(y>0)

{

//printf("%d %d\n",p22.x,p22.y);

tree[x][y]++;

y-=lowbit(y);

}

x-=lowbit(x);

}

}

int read(int x,int y)

{

int re=0,tm=y;

while(x<=n)

{

y=tm;

while(y<=n)

{

re+=tree[x][y];

y+=lowbit(y);

}

x+=lowbit(x);

}

return re;

}

int main()

{

int x,t,a,b,tt;

node p1,p2;

char c[2];

scanf("%d",&x);

while(x--)

{

scanf("%d%d",&n,&t);

memset(tree,0,sizeof(tree));

while(t--)

{

scanf("%s",c);

if(c[0]=='C')

{

scanf("%d%d%d%d",&p1.x,&p1.y,&p2.x,&p2.y);

updata(p2.x,p2.y);

updata(p1.x-1,p2.y);

updata(p2.x,p1.y-1);

updata(p1.x-1,p1.y-1);

}

else

{

scanf("%d%d",&a,&b);

tt = read(a,b);

if(!(tt%2))

{

printf("0\n");

}

else

printf("1\n");

}

}if(x)printf("\n");

}

return 0;

}

怎么理解二维树状的含义，而为什么又要

updata(p2.x,p2.y);

updata(p1.x-1,p2.y);

updata(p2.x,p1.y-1);

updata(p1.x-1,p1.y-1);

--------------

二、一维线段树

分析：

struct seg{int l,r,len;};

struct seg{int l,r,len;};

//基本的线段树的构建 框架

void build(int p,int l,int r)

{

tree[p].l=l;

tree[p].r=r;

tree[p].len=r-l+1;

if(l==r)

return ;

int m = (l+r)>>1;

build(p\*2,l,m);

build(p\*2+1,m+1,r);

}

//基本的线段树的构建 框架

void build(int p,int l,int r)

{

tree[p].l=l;

tree[p].r=r;

tree[p].len=r-l+1;

if(l==r)

return ;

int m = (l+r)>>1;

build(p\*2,l,m);

build(p\*2+1,m+1,r);

}

//查询

int query(int p,int l,int r,int n)

{

//tree[p].len--;

int m =(l+r)>>1;

if(l==r)

return l;

else if(tree[p\*2].len>=n)

{

return query(p\*2,l,m,n);

}

else

{return query(p\*2+1,m+1,r,n-tree[p\*2].len);}

}

//查询

int query(int p,int l,int r,int n)

{

//tree[p].len--;

int m =(l+r)>>1;

if(l==r)

return l;

else if(tree[p\*2].len>=n)

{

return query(p\*2,l,m,n);

}

else

{return query(p\*2+1,m+1,r,n-tree[p\*2].len);}

}

view plaincopy to clipboardprint?

//更新

void updata(int p,int l,int r,int x)

{

if(l==r&&l==x)

{

tree[p].len=0;

return ;

}

int m =(l+r)>>1;

if(m>=x){updata(p\*2,l,m,x);}

else

{

updata(p\*2+1,m+1,r,x);

}

tree[p].len--;

}