2008年11月06日 星期四 20:00

|  |
| --- |
| 二维线段树解法：  对于x轴构建一个二叉树，然后对于x轴的每个区间构造y轴的线段树，此题因为数据修改简单，不需要建树对每个节点赋值初始化，只需要一个二维数组记录每个节点的更改值就行，查询是只要对统计该节点的所有包含它的区间的值，如果和是基数则输出为1，偶数则为0  （此题构造采用第一用方法，给原来线段树中的每个结点都加多一棵线段树  ，另一中方法是将线段树结点中的线段变成矩形，从而变为矩形树。因此矩形树用的是四分的思想，每个矩形分割为4个子矩形，未实现，感觉得动态建立节点（费时费内存），望大牛路过指教  ）  代码如下：  #include<stdio.h>  #include<string.h>  bool a[3005][3005];  inline int leftson(int k)  {  return k<<1;  }  inline int rightson(int k)  {  return (k<<1)+1;  }  void change\_y(int y1,int y2,int tag,int now,int beg,int end)  {  if( y1 == beg && y2 == end )  {  a[tag][now] = !a[tag][now];  return ;  }  int mid = (beg+end)>>1;  if( y2 <= mid ) change\_y(y1,y2,tag,leftson(now),beg,mid);  else if( y1 > mid ) change\_y(y1,y2,tag,rightson(now),mid+1,end);  else  {  change\_y(y1,mid,tag,leftson(now),beg,mid);  change\_y(mid+1,y2,tag,rightson(now),mid+1,end);  }  }  void change\_x(int x1,int x2,int y1,int y2,int now,int beg,int end,int n)  {  if( x1 == beg && x2 == end )  {  change\_y(y1,y2,now,1,1,n);  return ;  }  int mid = (beg+end)>>1;  if( x2 <= mid ) change\_x(x1,x2,y1,y2,leftson(now),beg,mid,n);  else if( x1 > mid ) change\_x(x1,x2,y1,y2,rightson(now),mid+1,end,n);  else  {  change\_x(x1,mid,y1,y2,leftson(now),beg,mid,n);  change\_x(mid+1,x2,y1,y2,rightson(now),mid+1,end,n);  }  }  void query\_Y(int beg,int end,int tag,int now,int y,int&sum)  {  sum += a[tag][now];  if( beg == end ) return ;  int mid = ( beg + end)>>1;  if( y <= mid ) query\_Y(beg,mid,tag,leftson(now),y,sum);  else query\_Y(mid+1,end,tag,rightson(now),y,sum);  }  //对于包含其坐标的任意区间进行查询  void query\_X(int beg,int end,int now,int x,int y,int&sum,int n)  {  query\_Y(1,n,now,1,y,sum);  if( beg == end ) return ;  int mid = (beg + end) >> 1;  if( x <= mid ) query\_X(beg,mid,leftson(now),x,y,sum,n);  else query\_X(mid+1,end,rightson(now),x,y,sum,n);  }  int main()  {  int c,n,t,x1,y1,x2,y2;  char ch;  scanf("%d",&c);  while( c -- )  {  scanf("%d%d",&n,&t);  memset(a,0,sizeof(a));  while( t--)  {  getchar();  scanf("%c",&ch);  if( ch == 'C' )  {  scanf("%d%d%d%d",&x1,&y1,&x2,&y2);  change\_x(x1,x2,y1,y2,1,1,n,n);  }  else  {  scanf("%d%d",&x1,&y1);  int sum = 0;  query\_X(1,n,1,x1,y1,sum,n);  printf("%d\n",sum%2);  }  }  printf("\n");  }  return 0;  }  二维树状数组解法：  此题需要转化，用原先的Getsum修改，用原先的Add进行求解（不太明白，解题思路源于别人，具体思路，网址在下面）。  Up(a) = { a1 = a, a2 = a1+ lowbit(a1), a3 = a2 + lowbit(a2), ... }  Down(a) = { a1 =a, a2 = a1 -lowbit(a1), a3 = a2 - lowbit(a2), ... } 对于任意 a<b; 集合 Up（a） 与集合 down (b) 有且只有一个交集。开一个数组C每个点记录翻转的次数 所以对于待更改的区间 [a, b]. 在数组C中把集合down（b）中的每个元素都加 1，down（a-1）中的每个元素 都 减 1. 这样对于每个查询Q a，up（a）中元素的和 就是在a处翻转的次数。  代码：  #include<stdio.h>  #include<string.h>  #define lowbit(x)(x&(x^(x-1)))  int a[1003][1003],m,n;  void modif(int x,int y,int d)  {  while(x > 0)  { int temp = y;  while(temp > 0)  { a[x][temp] +=d;  temp -= lowbit(temp);  }  x -=lowbit(x);  }  }  int GetSum(int x,int y)  {  int sum = 0;  while( x <= n )  {  int temp = y;  while( temp <= n )  {  sum += a[x][temp];  temp += lowbit(temp);  }  x += lowbit(x);  }  return sum;  }  int main()  {  int i,j,k,c,t,x1,x2,y1,y2;  char ch;  scanf("%d",&c);  while(c -- )  {  scanf("%d%d",&n,&t);  memset(a,0,sizeof(a));  while( t -- )  {  getchar();  scanf("%c",&ch);  if( ch == 'C' )  {  scanf("%d%d%d%d",&x1,&y1,&x2,&y2);  modif(x2,y2,1);  modif(x1-1,y2,1);  modif(x2,y1-1,1);  modif(x1-1,y1-1,1);  }  else  {  scanf("%d%d",&x1,&y1); printf("%d\n",GetSum(x1,y1)%2);  }  }  printf("\n");  }  return 0;  } |