# Segment tree 2D, sum of rectangle

I really want to learn and implement segment tree 2D, at last. It's haunting me. I know the 1D case of segment tree, but somehow I can't manage with 2D. The problem is that I have a matrix 1024x1024 (so I use an array [2048][2048] as a tree) and I want to implement two operations:

1. void insert(int x, int y, int val); - which assigns value val to element [x][y] of matrix
2. int query(int x1, int y1, int x2, int y2); - which returns sum of the elements in matrix in rectangle (x1,y1,x2,y2)

So far I wrote this:

```
const int M=1024;
int tree[2*M][2*M];

void insert(int x, int y, int val) {
  int vx=M+x, vy=M+y;
  tree[vx][vy]=val;
  while(vy!=1) {
    vy/=2;
    tree[vx][vy]=tree[vx][2*vy]+tree[vx][2*vy+1];
  }

  while(vx!=1) {
    vy=M+y;
    vx/=2;
    while(vy!=1) {
      vy/=2;
      tree[vx][vy]=tree[2*vx][2*vy]+tree[2*vx+1][2*vy+1];
    }
  }
}
```

```
int query(int x1, int y1, int x2, int y2) {
  int vx1=M+x1, vy1=M+y1, vx2=M+x2, vy2=M+y2;
  int res=tree[vx1][vy1];
  if(vx1!=vx2 || vy1!=vy2) res+=tree[vx2][vy2];
  while(vx1/2 != vx2/2) {
    vy1=M+y1; vy2=M+y2;
    while(vy1/2 != vy2/2) {
      if(vx1%2==0 && vy1%2==0) res+=tree[vx1+1][vy1+1];
      if(vx2%2==1 && vy2%2==1) res+=tree[vx2-1][vy2-1];
      vy1/=2; vy2/=2;
    }
    vx1/=2; vx2/=2;
```

But it doesn't work correctly. Say, for:

insert(5,5,1); query(0,5,1000,5);

It returns 0, instead of 1. I think the problem is in query (I hope insert is OK), that I don't fully understand the idea of this operation. In 1D I had no problems, but this case is difficult to imagine, for me.

Can you please help me implement this correctly? I would be very grateful for help.

**EDIT:** maybe it will be better to show how I can do it in 1D, this code works and I think the idea i simple:

```
const int M=1024;
int tree[2*M];

void insert ( int x ,  int val )  {
  int v = M + x ;
  tree [ v ] = val ;
  while ( v ! = 1 )  {
    v / = 2 ;
    tree [ v ] = tree [ 2 * v ] + tree [ 2 * v + 1 ];
  }
},fi

int query(int a, int b) {
  int va=M+a, vb=M+b;
  int res=tree[va];
  if(va!=vb) res+=tree[vb];
  while(va/2 != vb/2) {
    if(va%2==0) res+=tree[va+1];
    if(vb%2==1) res+=tree[vb-1];
    va/=2; vb/=2;
  }
  return res;
}
```

but unfortunately I can't apply it in 2D..

c++ data-structures tree

edited Jul 18 '12 at 13:36

asked Jul 18 '12 at 12:51
xan
643,ht 1 11

Why are you using a 2-dimentional array instead of a "real" tree? – SingerOfTheFall Jul 18 '12 at 13:02

I would advise you to implement your tree using `nodes`, as a tree is usually implemented, and I would tell you to use recursion instead of iterative approach to traverse the tree ... your `insert` and `query` can be implemented with far fewer lines than you are doing now. – philippe Jul 18 '12 at 13:07

I am using 2D array because it is easier for me to do it first. When I finish this implementation, I will write it using pointers to make it more elegant. – xan Jul 18 '12 at 13:17

@Xan, it's a false sense of easiness. You may think "ok this is easier to implement than a tree". However, if you are implementing a real tree, you can imagine it, draw it on a paper, think of what is happening there, because it is very simple: "ok, I take this node and go left then right, then left, and I'm done". With an array you have to think "ok I take this position inthe array, and, provided the shift for the elements is 15, I need to add 15 to y and X, then add another 15 to Y, then do something else, and I got an index of the desired element!" Such things are very hard to keep in mind. – SingerOfTheFall Jul 18 '12 at 13:33

add comment

## 1 Answer

Well,the reason why your case does return 0 is that only this part of the code is executed:

```
int res=tree[vx1][vy1];
if(vx1!=vx2 || vy1!=vy2)
    res+=tree[vx2][vy2];
```

Here, `res` is 0, because `tree[vx1][vy1]` and `tree[vx2][vy2]` are both zero in your case.

This part doesn't change res because the conditions are never met:

```
if(vx1%2==0 && vy1%2==0)
    res+=tree[vx1+1][vy1+1];
if(vx2%2==1 && vy2%2==1)
    res+=tree[vx2-1][vy2-1];
```

So, the res value will not changed, and will still be 0.

Now, about the whole thing. You are building a segment tree in a very strange way, actually, you are not building any tree at all, and it is a little hard to understand what you are doing with that code. Usually, you would want to implement a binary tree (which a segment tree is) as a linked list with nodes looking something like:

```
struct node
 {
    int data ;
    node * left ;
    node * right ;
};,nl
```

I could suggest you looking here, here, here, here and here for the information and implementations on both binary and interval trees.

edited Jul 18 '12 at 13:27

answered Jul 18 '12 at 13:22
SingerOfTheFall
8,837 2 14 39

add comment

**Not the answer you're looking for? Browse other questions tagged c++ data-structures tree or ask your own question.**