Algorithm challenge

Solved UVA problems

Home

Search …

UVA 101: The blocks problem

Process

This problem is an exercise in tedium. Be sure to read the directions – if you follow them precisely you will get AC. Keep a vector of vectors and write methods for each operation. Be careful of removing elements in vectors, as using for loops to do so is tricky if you are iterating based on an original size. Be sure to skip cases where the elements are in the same column or a == b. I wrote an O(N^2) method to find the column and row of an element – that should work fine.

```
1  /*
2   * Sai Cheemalapati
3   * UVA 101: The blocks problem
4   *
5   */
6
7  #include<cstdio>
8  #include<iostream>
9  #include<vector>
10
11 using namespace std;
12
13 int n, a, b;
14 string instr1, instr2;
15 vector<vector<int> > blocks;
16
17 // returns the column and row of an element a
18 pair<int, int> column(int a) {
19     for(int i = 0; i < blocks.size(); i++)
20         for(int j = 0; j < blocks[i].size(); j++)
21             if(a == blocks[i][j]) return make_pair(i, j)
22     return make_pair(0, 0);
23 }
24
25 // returns the element at (i, j) to it's initial positio
26 void return_to_initial(int i, int j) {
27     blocks[blocks[i][j]].push_back(blocks[i][j]);
28     blocks[i].erase(blocks[i].begin() + j);
29 }
30
31 void move_onto(pair<int, int> c, pair<int, int> d) {
32     while(d.second + 1 != blocks[d.first].size()) {
33         return_to_initial(d.first, d.second + 1);
34     }
35     while(c.second + 1 != blocks[c.first].size()) {
36         return_to_initial(c.first, c.second + 1);
37     }
38     blocks[c.first].erase(blocks[c.first].begin() + c.se
39     blocks[d.first].push_back(a);
40 }
41
42 void move_over(pair<int, int> c, pair<int, int> d) {
43     while(c.second + 1 != blocks[c.first].size()) {
44         return_to_initial(c.first, c.second + 1);
45     }
46     blocks[c.first].erase(blocks[c.first].begin() + c.se
47     blocks[d.first].push_back(a);
```



```
54     while(c.second != blocks[c.first].size()) {
```

```
        blocks[d.first].push_back(blocks[c.first][c.seco
        blocks[c.first].erase(blocks[c.first].begin() +
    }
}

void pile_over(pair<int, int> c, pair<int, int> d) {
    while(c.second != blocks[c.first].size()) {
        blocks[d.first].push_back(blocks[c.first][c.seco
        blocks[c.first].erase(blocks[c.first].begin() +
    }
}

int main() {
    scanf("%d\n", &n);
    blocks.resize(n);
    for(int i = 0; i < n; i++)
        blocks[i].push_back(i);
    for(;;) {
        cin >> instr1;
        if(instr1 == "quit") break;
        cin >> a >> instr2 >> b;

        // if it's an illegal instruction keep going
        if(a == b) continue;
        pair<int, int> c = column(a);
        pair<int, int> d = column(b);
        if(c.first == d.first) continue;

        if(instr1 == "move" && instr2 == "onto")
            move_onto(c, d);
        if(instr1 == "move" && instr2 == "over")
            move_over(c, d);
        if(instr1 == "pile" && instr2 == "onto")
            pile_onto(c, d);
        if(instr1 == "pile" && instr2 == "over")
            pile_over(c, d);
    }

    for(int i = 0; i < blocks.size(); i++) {
        printf("%d:", i);
        for(int j = 0; j < blocks[i].size(); j++) {
            printf(" %d", blocks[i][j]);
        }
        printf("\n");
    }
}
```

input

```
10
move 9 onto 1
move 4 onto 9
move 5 onto 9
move 6 onto 9
move 5 onto 6
move 3 onto 7
move 7 over 1
move 8 over 4
pile 6 onto 4
pile 6 over 1
pile 1 over 1
pile 6 over 4
quit
```

output

```
0: 0
1: 1 9
```

```
8: 8
9:
```


Share this:

Twitter Facebook Google

Like

Be the first to like this.

Related

UVA 103: Stacking boxes
In "103"

UVa 10131: Is bigger better?
In "10131"

UVA 872: Ordering
In "872"

Posted on October 18, 2013 by saicheems. This entry was tagged 101, algorithm, input, online judge, output, the blocks problem, uva. Bookmark the permalink.

← UVA 10370: Above average

UVA 136: Ugly numbers →

Enter your comment here...