



Algorithm challenge

Solved UVA problems

[Home](#)

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 position
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.second);
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.second);
47     blocks[d.first].push_back(a);

```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#)

Close and accept

```

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

```

```

55         blocks[d.first].push_back(blocks[c.first][c.seco
56         blocks[c.first].erase(blocks[c.first].begin() +
57     }
58 }
59
60 void pile_over(pair<int, int> c, pair<int, int> d) {
61     while(c.second != blocks[c.first].size()) {
62         blocks[d.first].push_back(blocks[c.first][c.seco
63         blocks[c.first].erase(blocks[c.first].begin() +
64     }
65 }
66
67 int main() {
68     scanf("%d\n", &n);
69     blocks.resize(n);
70     for(int i = 0; i < n; i++)
71         blocks[i].push_back(i);
72     for(;;) {
73         cin >> instr1;
74         if(instr1 == "quit") break;
75         cin >> a >> instr2 >> b;
76
77         // if it's an illegal instruction keep going
78         if(a == b) continue;
79         pair<int, int> c = column(a);
80         pair<int, int> d = column(b);
81         if(c.first == d.first) continue;
82
83         if(instr1 == "move" && instr2 == "onto")
84             move_onto(c, d);
85         if(instr1 == "move" && instr2 == "over")
86             move_over(c, d);
87         if(instr1 == "pile" && instr2 == "onto")
88             pile_onto(c, d);
89         if(instr1 == "pile" && instr2 == "over")
90             pile_over(c, d);
91     }
92
93     for(int i = 0; i < blocks.size(); i++) {
94         printf("%d:", i);
95         for(int j = 0; j < blocks[i].size(); j++) {
96             printf(" %d", blocks[i][j]);
97         }
98         printf("\n");
99     }
100 }

```

input

```

1 10
2 move 9 onto 1
3 move 4 onto 9
4 move 5 onto 9
5 move 6 onto 9
6 move 5 onto 6
7 move 3 onto 7
8 move 7 over 1
9 move 8 over 4
10 pile 6 onto 4
11 pile 6 over 1
12 pile 1 over 1
13 pile 6 over 4
14 quit

```

output

```

1 0: 0
2 1: 1 9

```

9 | 8: 8
10 | 9:

Advertisements

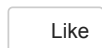
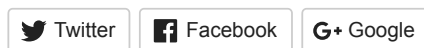


REPORT THIS AD



REPORT THIS AD

Share this:



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 →](#)

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#).

Close and accept

Enter your comment here...

[Create a free website or blog at WordPress.com.](#)



Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: [Cookie Policy](#).

Close and accept