OJ 10051

Tower of Cubes

Prof. Edson Alves – UnB/FGA

Problema

In this problem you are given N colorful cubes each having a distinct weight. Each face of a cube is colored with one color. Your job is to build a tower using the cubes you have subject to the following restrictions:

- Never put a heavier cube on a lighter one.
- The bottom face of every cube (except the bottom cube, which is lying on the floor) must have the same color as the top face of the cube below it.
- Construct the tallest tower possible.

Entrada e saída

Input

The input may contain multiple test cases. The first line of each test case contains an integer N ($1 \leq N \leq 500$) indicating the number of cubes you are given. The ith ($1 \leq i \leq N$) of the next N lines contains the description of the ith cube. A cube is described by giving the colors of its faces in the following order: front, back, left, right, top and bottom face. For your convenience colors are identified by integers in the range 1 to 100. You may assume that cubes are given in the increasing order of their weights, that is, cube 1 is the lightest and cube N is the heaviest.

The input terminates with a value 0 for N.

Entrada e saída

Output

For each test case in the input first print the test case number on a separate line as shown in the sample output. On the next line print the number of cubes in the tallest tower you have built. From the next line describe the cubes in your tower from top to bottom with one description per line. Each description contains an integer (giving the serial number of this cube in the input) followed by a single whitespace character and then the identification string (front, back, left, right, top or bottom) of the top face of the cube in the tower. Note that there may be multiple solutions and any one of them is acceptable.

Print a blank line between two successive test cases.

Exemplo de entradas e saídas

Sample Input

```
3
1 2 2 2 1 2
3 3 3 3 3 3
3 2 1 1 1 1
10
1 5 10 3 6 5
2 6 7 3 6 9
5 7 3 2 1 9
1 3 3 5 8 10
6 6 2 2 4 4
1 2 3 4 5 6
10 9 8 7 6 5
6 1 2 3 4 7
1 2 3 3 2 1
3 2 1 1 2 3
0
```

Sample Output

```
Case #1
2
2 front
3 front

Case #2
8
1 bottom
2 back
3 right
4 left
6 top
8 front
9 front
10 top
```

Solução $O(TN^2)$

- O problema pode ser reduzido a seis problemas de LIS
- Fazendo uma correspondência entre as faces de um cubo e os números de 0 a 5, na mesma ordem dada na entrada, o subproblema $lis(i, s)$ consiste em identificar a maior subsequência crescente válida que tem o i-ésimo elemento como menor elemento da sequência, cujo topo é a face s
- Como $N \leq 500$ é possível utilizar a implementação $O(N^2)$ da LIS
- Com a numeração proposta para as faces, a face oposta de s será $s + 1$, se s for par, ou $s - 1$, se s for ímpar
- Observe que uma sequência só pode ser ampliada caso a base do elemento coincida com o topo do elemento que ele ira sobrepor
- Também é preciso manter o registros dos elementos escolhidos e das faces utilizadas, para que a sequência possa ser reconstruída

Solução $O(TN^2)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ii = pair<int, int>;
using is = pair<int, string>;

vector<string> fs { "front", "back", "left", "right", "top", "bottom" };

vector<is> solve(int N, const vector<vector<int>>& xs)
{
    vector<vector<int>> lis(N, vector<int>(6, 1));
    vector<vector<ii>> ps(N, vector<ii>(6, ii(0, 0)));
    int size = 1;
    auto best = ii(N - 1, 0);

    for (int i = N - 2; i >= 0; --i)
    {
        for (int t = 0; t < 6; ++t)
        {
            auto b = t % 2 ? t - 1 : t + 1;
```

Solução $O(TN^2)$

```
22              for (int j = i + 1; j < N; ++j)
23              {
24                  for (int s = 0; s < 6; ++s)
25                  {
26                      if (xs[i][b] == xs[j][s] and lis[j][s] + 1 > lis[i][t])
27                      {
28                          lis[i][t] = lis[j][s] + 1;
29                          ps[i][t] = ii(j, s);
30
31                          if (lis[i][t] > size)
32                          {
33                              size = lis[i][t];
34                              best = ii(i, t);
35                          }
36                      }
37                  }
38              }
39          }
40      }
```

Solução $O(TN^2)$

```
42      vector<is> ans(size);
43
44      for (int k = 0; k < size; ++k)
45      {
46          auto [i, s] = best;
47          ans[k] = is(i + 1, fs[s]);
48          best = ps[i][s];
49      }
50
51      return ans;
52  }
53
54  int main()
55  {
56      ios::sync_with_stdio(false);
57      int N, test = 0;
58
59      while (cin >> N, N)
60      {
61          vector<vector<int>> xs(N, vector<int>(6));
```

Solução $O(TN^2)$

```
63          for (int i = 0; i < N; ++i)
64              for (int j = 0; j < 6; ++j)
65                  cin >> xs[i][j];
66
67          auto ans = solve(N, xs);
68
69          if (test)
70              cout << '\n';
71
72          cout << "Case #" << ++test << '\n';
73
74          cout << ans.size() << '\n';
75
76          for (auto [i, side] : ans)
77              cout << i << ' ' << side << '\n';
78      }
79
80      return 0;
81  }
```