Grafos

Depth-First Search: problemas resolvidos

Prof. Edson Alves - UnB/FGA

2019

Sumário

Codeforces Round #503 (by SIS, Div. 2) – Problem B: Badge

Problema

In Summer Informatics School, if a student doesn't behave well, teachers make a hole in his badge. And today one of the teachers caught a group of n students doing yet another trick.

Let's assume that all these students are numbered from 1 to n. The teacher came to student a and put a hole in his badge. The student, however, claimed that the main culprit is some other student p_a.

After that, the teacher came to student p_a and made a hole in his badge as well. The student in reply said that the main culprit was student p_{pa}.

This process went on for a while, but, since the number of students was finite, eventually the teacher came to the student, who already had a hole in his badge.

Problema

After that, the teacher put a second hole in the student's badge and decided that he is done with this process, and went to the sauna.

You don't know the first student who was caught by the teacher. However, you know all the numbers p_i. Your task is to find out for every student a, who would be the student with two holes in the badge if the first caught student was a.

Entrada e saída

Input

The first line of the input contains the only integer n $(1 \leq n \leq 1000)$ – the number of the naughty students.

The second line contains n integers $p_1, \ldots, p_n$ $(1 \leq p_i \leq n)$, where p_i indicates the student who was reported to the teacher by student i.

Output

For every student a from 1 to n print which student would receive two holes in the badge, if a was the first student caught by the teacher.

Exemplo de entradas e saídas

Sample Input	Sample Output
3 2 3 2	2 2 3
3 1 2 3	1 2 3

Solução com complexidade $O(N^2)$

- Como a entrada é pequena $N \leq 10^3$, é possível resolver este problema com um algoritmo quadrático
- Considere que cada estudante seja um vértice do grafo G, e que uma aresta (u, v) indique que o estudante u indicou o estudante v como principal responsável
- Logo $|V| = |E| = N$, de modo que cada travessia tem complexidade $O(N)$
- Assim, basta realizar N travessias por profundidade, interrompendo a mesma caso ela indifique um vértice que já foi encontrado anteriormente na travessia

Solução AC com complexidade $O(N^2)$

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

using namespace std;

const int MAX { 1010 };

bitset<MAX> found;

int dfs(int u, const vector<int>& ps)
{
    if (found[u])
        return u;

    found[u] = true;

    return dfs(ps[u], ps);
}

vector<int> solve(int N, const vector<int>& ps)
{
    vector<int> ans(N + 1);
```

Solução AC com complexidade $O(N^2)$

```
22
23      for (int u = 1; u <= N; ++u)
24      {
25          found.reset();
26          ans[u] = dfs(u, ps);
27      }
28
29      return ans;
30  }
31
32  int main()
33  {
34      ios::sync_with_stdio(false);
35
36      int N;
37      cin >> N;
38
39      vector<int> ps(N + 1);
40
41      for (int i = 1; i <= N; ++i)
42          cin >> ps[i];
```

Solução AC com complexidade $O(N^2)$

```
43
44      auto ans = solve(N, ps);
45
46      for (int i = 1; i <= N; ++i)
47          cout << ans[i] << (i == N ? '\n' : ' ');
48
49      return 0;
50  }
```

UVA 10113 – Exchange Rates

Problema

Using money to pay for goods and services usually makes life easier, but sometimes people prefer to trade items directly without any money changing hands. In order to ensure a consistent "price", traders set an exchange rate between items. The exchange rate between two items A and B is expressed as two positive integers m and n, and says that m of item A is worth n of item B. For example, 2 stoves might be worth 3 refrigerators. (Mathematically, 1 stove is worth 1.5 refrigerators, but since it's hard to find half a refrigerator, exchange rates are always expressed using integers.)

Your job is to write a program that, given a list of exchange rates, calculates the exchange rate between any two items.

Entrada e saída

Input

The input file contains one or more commands, followed by a line beginning with a period that signals the end of the file. Each command is on a line by itself and is either an assertion or a query. An assertion begins with an exclamation point and has the format

$$! \; m \; itema = n \; itemb$$

where $itema$ and $itemb$ are distinct item names and m and n are both positive integers less than 100. This command says that m of $itema$ are worth n of $itemb$. A query begins with a question mark, is of the form

$$? \; itema = itemb$$

and asks for the exchange rate between $itema$ and $itemb$, where $itema$ and $itemb$ are distinct items that have both appeared in previous assertions (although not necessarily the same assertion).

Entrada e saída

Output

For each query, output the exchange rate between $itema$ and $itemb$ based on all the assertions made up to that point. Exchange rates must be in integers and must be reduced to lowest terms. If no exchange rate can be determined at that point, use question marks instead of integers. Format all output exactly as shown in the example.

Entrada e saída

Note:

- Item names will have length at most 20 and will contain only lowercase letters.
- Only the singular form of an item name will be used (no plurals).
- There will be at most 60 distinct items.
- There will be at most one assertion for any pair of distinct items.
- There will be no contradictory assertions. For example, "2 pig = 1 cow", "2 cow = 1 horse", and "2 horse = 3 pig" are contradictory.
- Assertions are not necessarily in lowest terms, but output must be.
- Although assertions use numbers less than 100, queries may result in larger numbers that will not exceed 10000 when reduced to lowest terms

Exemplo de entradas e saídas

Sample Input

```
! 6 shirt = 15 sock
! 47 underwear = 9 pant
? sock = shirt
? shirt = pant
! 2 sock = 1 underwear
? pant = shirt
.
```

Sample Output

```
5 sock = 2 shirt
? shirt = ? pant
45 pant = 188 shirt
```

Solução com complexidade $O(QN^2)$

- Para simplificar a solução, pode-se associar um número inteiro distinto para cada item
- A cada *query* do tipo '!' deve se associar o identificador inteiro a cada item ainda não encontrado e registrar, em uma matriz de adjacências, o par (m, n) que corresponde à taxa de conversão entre ambos itens
- Esta taxa deve ser convertida a um par de elementos co-primos através da divisão de ambos pelo maior divisor comum entre eles
- Para cada *query* do tipo '?' deve se determinar os identificadores u e w de cada item e reiniciar a tabela de vetores encontrados DFS
- Para cada item v ainda não encontrado pela DFS, computa-se a taxa de conversão entre u e v
- Esta taxa pode já estar registrada na tabela de conversões ou deve ser estabelecida se a busca já visitou anteriormente outro nó
- Se $v = w$ a busca retorna esta taxa; caso contrário, a busca continua com $u = v$

Solução com complexidade $O(QN^2)$

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

using namespace std;
using ll = long long;
using ii = pair<long, long>;

const int MAX { 70 };

ii adj[MAX][MAX];
bitset<MAX> found;
map<string, int> label;

ll gcd(ll a, ll b)
{
    return b ? gcd(b, a % b) : a;
}

ii dfs(int u, int w, const ii& prev = ii(1, 1))
{
    found[u] = true;

```

Solução com complexidade $O(QN^2)$

```
22      auto ans = ii(-1, -1);
23      int N = label.size();
24
25      for (int v = 1; v <= N; ++v)
26      {
27          auto p = adj[u][v];
28
29          if (found[v] or p == ii(0, 0))
30              continue;
31
32          // Taxa de conversão de v para w
33          auto m = prev.first * p.first;
34          auto n = prev.second * p.second;
35          auto d = gcd(m, n);
36
37          m /= d;
38          n /= d;
39
40          if (v == w)
41              return ii(m, n);
42
```

Solução com complexidade $O(QN^2)$

```
43          ans = dfs(v, w, ii(m, n));
44
45          if (ans != ii(-1, -1))
46              return ans;
47      }
48
49      return ans;
50  }
51
52  int main()
53  {
54      ios::sync_with_stdio(false);
55
56      string cmd, from, to, equals;
57      int idx = 0;
58
59      while (cin >> cmd)
60      {
61          switch (cmd[0]) {
62          case '!':
63              {
```

Solução com complexidade $O(QN^2)$

```
64                  ll m, n;
65
66                  cin >> m >> from >> equals >> n >> to;
67
68                  if (label.count(from) == 0)
69                      label[from] = ++idx;
70
71                  if (label.count(to) == 0)
72                      label[to] = ++idx;
73
74                  auto d = gcd(m, n);
75                  auto u = label[from], v = label[to];
76
77                  adj[u][v] = ii(m/d, n/d);
78                  adj[v][u] = ii(n/d, m/d);
79              }
80
81          break;
82
83          case '?':
84              {
```

Solução com complexidade $O(QN^2)$

```
85                  cin >> from >> equals >> to;
86
87                  auto u = label[from], v = label[to];
88
89                  found.reset();
90                  auto p = dfs(u, v);
91
92                  if (p == ii(-1, -1))
93                  {
94                      cout << "? " << from << " = ? " << to << '\n';
95                      break;
96                  }
97
98                  cout << p.first << " " << from << " = " << p.second
99                      << " " << to << '\n';
100             }
101         }
102     }
103
104     return 0;
105 }
```

Referências

1. Codeforces Round #503 (by SIS, Div. 2) – Problem B: Badge
2. UVA 10113 – Exchange Rates