acm.mipt.ru

олимпиады по программированию на Физтехе

Поиск мостов и двусвязных компонент: реализация на C++

Код

```
/*
 * Поиск мостов и двусвязных компонент.
 * Даниил Швед, 2008. МФТИ.
 * danshved [no-spam] gmail.com
 */
#include <stdio.h>
#include <vector>
#include <algorithm>
using namespace std;

typedef vector<int> VInt;
typedef vector<VInt> VVInt;
typedef VInt::iterator VIter;
typedef pair<int, int> PInt;
typedef vector<PInt> VPInt;
typedef vector<VPInt> VVPInt;
typedef VPInt::iterator VPIter;

VVInt graph;
VInt colors, parents, enter, leave, low, bcc;
int myTime = 0;
int newIndex = 0;

/*
 * Поиск в глубину, выполняющий вычисление enter, leave и low
 */
void visitLow(int u) {
   colors[u] = 1;
   low[u] = enter[u] = ++myTime;

   for(VIter it = graph[u].begin(); it != graph[u].end(); it++)
      if(colors[*it] == 0) {
         parents[*it] = u;
         visitLow(*it);
         low[u] = min(low[u], low[*it]);
      } else if(colors[*it] == 1 && *it != parents[u]) {
         low[u] = min(low[u], enter[*it]);
      }

   colors[u] = 2;
   leave[u] = ++myTime;
}

/*
 * Второй поиск в глубину, присвающий идентификаторы bcc
 */
void visitBCC(int u) {
   for(VIter it = graph[u].begin(); it != graph[u].end(); it++)
      if(parents[*it] == u) {
         bcc[*it] = (low[*it] < enter[u]) ? bcc[u] :          // Дочернее ребро эквивалентно текущему
                    (low[*it] > enter[u]) ? -1 :              // Дочернее ребро есть мост
                    (newIndex++);                             // Дочернее ребро лежит в новой компоненте
         visitBCC(*it);
      }
}

/*
 * Получение номера BCC, которому принадлежит ребро, или -1, если это мост
 * (u, v) действительно должно быть ребром, иначе возвращенный результат не имеет смысла
 */
 int getBCC(int u, int v) {
    return bcc[(enter[u] > enter[v]) ? u : v];
 }

/*
```

```
 * На входе: числа n и m, затем m описаний ребер
 * На выходе: список мостов, затем списки ребер в каждой компоненте
 */
int main() {
   int n, m, i;

   // Прочитаем граф
   scanf("%d%d", &n, &m);
   graph.resize(n);
   while(m--) {
      int from, to;
      scanf("%d%d", &from, &to);
      graph[from - 1].push_back(to - 1);
      graph[to - 1].push_back(from - 1);
   }

   // Запустим первый поиск (вычислим enter и low)
   colors.assign(n, 0);
   parents.assign(n, -1);
   enter.resize(n);
   leave.resize(n);
   low.resize(n);
   for(i = 0; i < n; i++)
      if(colors[i] == 0)
         visitLow(i);

   // Запустим второй поиск (определение идентификаторов bcc)
   // Второй поиск запускается "по следам" первого,
   // то есть проходит по уже найденному дереву
   bcc.assign(n, -1);
   for(i = 0; i < n; i++)
      if(parents[i] == -1)
         visitBCC(i);

   // Выведем результат
   VPInt bridges;
   VVPInt comps(newIndex);
   for(i = 0; i < n; i++)
      for(VIter it = graph[i].begin(); it != graph[i].end(); it++)
         if(i < *it) {
            int id = getBCC(i, *it);
            ((id == -1) ? bridges : comps[id]).push_back(PInt(i, *it));
         }

   printf("Bridges: ");
   for(VPIter bridge = bridges.begin(); bridge != bridges.end(); bridge++)
      printf("(%d, %d) ", bridge->first + 1, bridge->second + 1);
   printf("\n");

   for(i = 0; i < newIndex; i++) {
      printf("Component %d: ", i);
      for(VPIter edge = comps[i].begin(); edge != comps[i].end(); edge++)
         printf("(%d, %d) ", edge->first + 1, edge->second + 1);
      printf("\n");
   }
}
```

Ссылки

- Мосты, точки раздела и двусвязные компоненты (теория)

-- DanielShved - 27 Mar 2008