acm.mipt.ru

олимпиады по программированию на Физтехе

Поиск минимального контролирующего множества в двудольном графе: C++

```
/*
 * Поиск минимального контролирующего множества вершин в двудольном
 * Даниил Швед, 2008. МФТИ.
 * danshved [no-spam] gmail.com
 */
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <algorithm>
using namespace std;

typedef vector<int> VInt;
typedef vector<VInt> VVInt;
typedef VInt::iterator VIter;
typedef vector<bool> VBool;

VVInt graph;
int leftCount, rightCount;
VInt match;
VBool visited;
VBool isRight;

// Обход в глубину для поиска увеличивающего чередующегося пути
bool matchVisit(int u) {
   visited[u] = true;
   for(VIter it = graph[u].begin(); it != graph[u].end(); it++)
      if(match[*it] == -1 || (!visited[match[*it]] && matchVisit(match[*it]))) {
         match[*it] = u;
         return true;
      }
   return false;
}

// Обход в глубину для переключения ребер паросочетания
void controlVisit(int u) {
   visited[u] = true;
   for(VIter it = graph[u].begin(); it != graph[u].end(); it++)
      if(!isRight[*it] && !visited[match[*it]]) {
         controlVisit(match[*it]);
         isRight[*it] = true;
      }
}

// Поиск наименьшего контролирующего множества
pair<VInt, VInt> control() {
   int i;

   // Найдем наибольшее паросочетание
   match.assign(rightCount, -1);
   visited.assign(leftCount, false);

   for(i = 0; i < leftCount; i++) {
      visited.assign(leftCount, false);
      matchVisit(i);
   }
```

Поиск
Поиск
Раздел «Алгоритмы»
Главная
Форум
Ссылки
El Judge
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

```
    // Найдем свободные вершины в левой доле
    VBool isFree(leftCount, true);
    for(i = 0; i < rightCount; i++)
        if(match[i] != -1)
            isFree[match[i]] = false;

    // Запустим поиск в глубину из свободных вершин, в процессе поиска
    // переключим некоторые ребра из "левого" состояния в "правое".
    isRight.assign(rightCount, false);
    visited.assign(leftCount, false);
    for(i = 0; i < leftCount; i++)
        if(isFree[i])
            controlVisit(i);

    // Вернем ответ в виде пары массивов (контролирующие вершины в каждой из долей)
    pair<VInt, VInt> result;
    for(i = 0; i < rightCount; i++)
        if(match[i] != -1) {
            if(isRight[i])
                result.second.push_back(i);
            else
                result.first.push_back(match[i]);
        }
    return result;
}

// Демонстрация: считаем граф и выведем контролирующее множество
int main() {
    int edgeCount;

    scanf("%d%d%d", &leftCount, &rightCount, &edgeCount);
    graph.resize(leftCount);
    while(edgeCount--) {
        int from, to;
        scanf("%d%d", &from, &to);
        graph[from-1].push_back(to-1);
    }

    pair<VInt, VInt> answer = control();

    printf("In the left part: ");
    for(VIter it = answer.first.begin(); it != answer.first.end(); it++)
        printf("%d ", *it + 1);
    printf("\nIn the right part: ");
    for(VIter it = answer.second.begin(); it != answer.second.end(); it++)
        printf("%d ", *it + 1);
    printf("\n");
}
```

-- DanielShved - 23 Mar 2008