Ворожцов Артём Викторович
Кандидат физико-математических наук,
преподаватель кафедры информатики
Московского физико-технического института (МФТИ),
тренер сборной команды МФТИ по программированию.

Задача «Стабильный коллектив»

Здесь приведён разбор задачи с Международной студенческой олимпиады по программированию (ACM ICPC 2006, четверть финала, МГУ, Москва). Эта задача была одной из сложных и интересных задач этой олимпиады. У неё достаточно простое решение, но догадаться до него самостоятельно не так просто. Попробуйте сами догадаться до метода решения, проследите за ходом своих мыслей и обратите внимание на ключевые шаги. Попробовать свои решения можно на сайте http://acm.mipt.ru/judge (задача 106).

Примечание: Предполагается, что читатель знаком с языком программирования C++ и стандартной библиотекой шаблонов STL.

Условие задачи

Максимальное время работы программы на одном тесте: 2 секунды.

В фирме ООО «Новый Софт» работа совсем не идёт, фирма несёт убытки, а всё из-за того, что рабочий коллектив не дружен, а правильные отношения не налажены.

Директор фирмы Пётр Новиков

решил принять радикальные меры, а именно – уволить часть сотрудников, чтобы остался максимально стабильный коллектив.

Для начала необходимо выяснить, кто с кем дружит (известно, что дружба всегда взаимна). Затем на основании этих данных найти коллектив (непустое подмножество сотрудников) с максимальной стабильностью.

Стабильность коллектива A определяется слабым звеном. Слабое звено коллектива A – это человек, у которого меньше всего друзей в коллективе. Число друзей у слабого звена и есть стабильность коллектива.

Да, и ещё – Петр Новиков хотел бы остаться работать в фирме и себя он увольнять не собирается, хотя, в принципе, он может быть слабым звеном. Известно, что у Петра Новикова есть друзья среди своих сотрудников.

Помогите Петру и напишите программу, которая среди коллективов, в которых он есть, находит коллектив с максимальной стабильностью. Среди возможных коллективов, удовлетворяющих этому свойству, найдите тот, у которого максимальный размер. Если таких коллективов несколько, то найдите один из возможных.

Вход. В первой строке дано количество M дружных пар, $0 < M < 40000$. Затем идёт M строчек, в каждой из которых даны два разных имени, разделённых пробелом. Имена сотрудников записаны латинскими символами и имеют длину менее 32 символов. Имя Петра Новикова записано как Petr. Общее число сотрудников не превосходит 7000. Первая буква любого имени заглавная, а остальные маленькие.

Выход. В первой строке укажите размер самого стабильного коллектива, в котором есть Пётр, а затем, через пробел, его стабильность K. Во второй строке приведите список имён людей, которые вошли в этот коллектив, в лексикографическом порядке.

Математическая формализация

Многие уже, наверно, догадались, что это задача связана с графами.

Граф – это множество вершин и множество рёбер. Ребро – это неупорядоченная пара вершин. Записывают это так: $e = (u,v)$.

Про две вершины u и v говорят, что они соединены ребром e. Также говорят, что вершины u и v смежны (adjacent). Число вершин, смежных с некоторой вершиной u, называют степенью вершины.

В нашей задаче вершины графа – это сотрудники фирмы, а рёбра – отношение дружбы.

Степень вершины – это

Пример входа/выхода

Вход#1	Выход#1
8 Petr Vasya Petr Kolya Sergey Max Petr Lena Lena Vasya Lena Kolya Vasya Kolya Sergey Lena	4 3 Kolya Lena Petr Vasya

число друзей соответствующего сотрудника.

По сути, в задаче требуется найти такое подмножество вершин, что если выкинуть все остальные вершины (и связанные с ними рёбра), то останется граф, в котором вершина с минимальной степенью имеет как можно бóльшую степень.

Идея решения

Предположим, что нам известно значение K стабильности искомого коллектива (степень вершины с минимальной степенью).

Тогда мы можем безбоязненно удалить из графа вершины со степенью меньше K, причём в любом порядке. Удалим сначала висячие вершины, затем вершины, у которых две смежных вершины и т.д., пока не останутся вершины со степенью больше либо равно K.

Из этих простых рассуждений немедленно следует решение задачи:

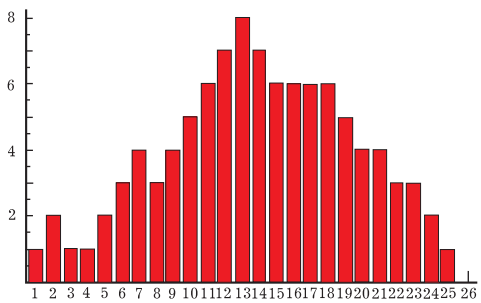

Рис. 1

нужно просто пошагово убирать из графа вершину с минимальной степенью (слабое звено). В конце концов, будут убраны все вершины из графа.

Понятно, что на каком-то шаге будут удалены все вершины со степенью K (которые мы пока не знаем).

На рис. 1 для некоторого графа показана зависимость степени извлекаемой вершины от шага. Всего в этом графе было 26 вершин, одна из них – висячая, то есть имеет только одну смежную вершину. Именно она была извлечена первой. Затем пошла вершина со степенью 2. В результате граф изменился, и у него снова появились висячие вершины.

На 12 шаге была извлечена вершина со степенью 8. Все остальные вершины имели на этом шаге степень не меньше 8. К концу начинается уменьшение степени извлекаемой вершины, и на последнем шаге удаляется одинокая вершина, у которой нет смежных.

Несложно понять, что ответом является $K = 8$, а искомое множество вершин – это вершины, из которых состоит граф после 11-го шага. На первых 11 шагах извлекались вершины степени менее 8.

Решение

Задача решается жадным алгоритмом. Помещаем все вершины в очередь. На каждом шаге удаляем «самое слабое звено» – вершину с минимальной степенью – и по ходу вычисляем максимальную степень извлекаемых вершин. Максимальная степень – это и есть искомое число K. На втором проходе извлечём все вершины степени меньше K, а оставшиеся вернём как ответ.

Описание основных переменных

Переменная **`adj`** – это матрица смежности. Она содержит информацию о том, какие вершины соединены ребром. А именно, условие **`adj[i][j] == 1`** означает, что вершина **`j`** смежна вершине **`i`**. У нас

граф неориентированный (если A дружит с B, то и B дружит с A), то есть **adj[i][j] == adj[j][i].**
Тип матрицы смежности может быть равен одному из следующих типов:
vector< vector<int > > adj
vector< map<int,int > > adj
map< int, map<int,int > > adj
Все три типа поддерживают оператор квадратные скобки и позволяют писать выражение вида **adj[i][j]**. Переменная **q** используется для хранения очереди вершин, а точнее множества пар (степень вершины, идентификатор вершины). Её тип **set< pair<int,int> >**, то есть множество пар элементов, каждый из которых имеет тип **int**.

Начало кода

```
#include <cstdio>
#include <string>
#include <set>
#include <algorithm>
#include <map>
#include <vector>
using namespace std;

int debug = 0; // установите в 1 для вывода отладочной
               // информации
#define M 20002
#define MP make_pair

vector< map<int,int > > adj;  /* adj - "матрица смежности" */
vector< map<int,int > > adj2; /* Копия adj */

map<string,int> id;    /* Отображение name->id */
map<int,string> name; /* Отображение id->name */

/* Множество пар (степень вершины, вершина)
 * set используется как очередь с приоритетами
 * Ключ - степень вершины, значение - идентификатор вершины
 */
set< pair<int,int> > q;
set< pair<int,int> > q2; // копия q для второго прохода

int m, n = 0;
int petr = 0;
string petr_name("Petr");

/* Возвращает целочисленный идентификатор вершины по её имени.
 * Если вершина новая, то добавляет её в словари name и id.
 */
int getid(string s) {
   if( id.find(s) == id.end() ) {
         // если имени нет в словаре, то добавим его туда,
         // назначив ему следующий порядковый идентификатор
```

Продолжение кода на следующей странице

Продолжение кода

```
        id[s] = n;
        name[n] = s;
        return n++;
    } else {
        // иначе вернём идентификатор человека с таким именем
        return id[s];
    }
}
int main( ) {
    scanf("%d", &m); // считываем число рёбер

    adj.resize(2 * m); // число вершин не более чем 2 * m
    for ( int i = 0 ; i < m ; i++ ) {
        char s[50],d[50]; // имена вершин (source,destination)
        string ss, dd;   // их string-овый вариант
        int s_id, d_id;  // и их целочисленные идентификаторы
        scanf("%s%s", s, d);
      ss = s;   // В этих строчках происходит копирование
                // содержания строк, а не их адресов.
      dd = d;   // Дело в том, что оператор присвоения (=)
                // переопределён для случая,
                // когда справа стоит char*, а слева string.
        s_id = getid(ss);
        d_id = getid(dd);

        if ( adj[s_id].find(d_id) != adj[s_id].end() ) {
            fprintf(stderr, "Duplicate: %s %s\n", s, d);
        }
        if ( s_id == d_id ) {
            fprintf(stderr, "Equal      : %s %s\n", s, d);
        }
        adj[d_id][s_id] = 1;
        adj[s_id][d_id] = 1;
    }

    petr = getid(petr_name);

    for ( int i = 0 ; i < n ; i++ ) {
        q.insert( MP(adj[i].size(), i) );
    }

    q2 = q;      // Создадим копии для второго прохода
    adj2 = adj; //

    int max_degree = (*(q.begin())).first;
```

Продолжение кода на следующей странице

Продолжение кода

```
int degree, a;
while ( q.size() ) {
     /* извлекаем  вершину с минимальной степенью ("слабое
    звено" - работник, у которого меньше всего друзей) */
     degree = (*(q.begin())).first;
     a      = (*(q.begin())).second;
     /* Дело в том, что set.first() возвращает _наименьшую_
    пару. Порядок на парах совпадает с порядком на первом
    элементе. Первый элемент нашей пары - степень вершины.
    Таким образом, set может использоваться для реализации
    функциональности очереди с приоритетом.*/
     if ( debug )
          printf("Max degree=%d\nExtracting %s with %d\n\n",
                    max_degree, name[a].c_str(), degree);

     if ( max_degree < degree ) {
          max_degree = degree;
     }
     if ( a == petr ) break;

     // удаляем "слабое звено"
     q.erase(q.begin());

     // и "подчищаем" все его "дружбы"
     map<int, int> friends = adj[a];
     map<int,int>::iterator it;
     for(it = friends.begin();it != friends.end();it++) {
          int b = (*it).first;
          q.erase( MP(adj[b].size(), b ) );
          adj[b].erase(a);
          q.insert( MP(adj[b].size(), b) );
     }
}
if (debug) {printf("Final max degree=%d\n", max_degree);}
/*Итак, мы узнали  максимальное значение "стабильности".
  Теперь повторим всё сначала, уже печатая ответ.  */
while ( q2.size() ) {
     degree = (*(q2.begin())).first;
     a      = (*(q2.begin())).second;
     if ( degree >= max_degree ) {
          break;
     }
     q2.erase( q2.begin() );
     if ( debug )
     printf("Max degree=%d\nExtracting %s with %d\n\n",
            max_degree, name[a].c_str(), degree);
```

Продолжение кода на следующей странице

Окончание кода

```
        map<int,int> friends = adj2[a];
        map<int,int>::iterator it;
        for (it = friends.begin();it != friends.end();it++) {
            int b = (*it).first;
            q2.erase( MP(adj2[b].size(), b ) );
            adj2[b].erase(a);
            q2.insert( MP(adj2[b].size(), b) );
        }
    }
    // Выводим размер коллектива и его стабильность
    printf("%d %d\n", q2.size(), max_degree);
    set<string> answer; // шаблон set можно использовать
                        // для сортировки;
    {
        set< pair<int,int> >::iterator j;
        for ( j = q2.begin(); j != q2.end() ; j++ ) {
            answer.insert( name[((*j).second)]  );
        }
    }
    {
        set<string>::iterator j;
        for ( j = answer.begin(); j != answer.end() ; j++ ) {
            printf("%s ", (*j).c_str() );
        }
    }
    printf("\n");
    return 0;
}
```

Юмор Юмор Юмор Юмор Юмор Юмор

- Пусть эпсилон будет меньше 0...
- Если яблоку негде упасть, это не значит, что под деревом сидит толпа Ньютонов.
- - Сколько раз 7 можно вычесть из 83-х, и что получится в остатке?

  - 7 можно вычитать из 83-х столько раз, сколько захочешь. А в остатке всегда будет 76.