acm.mipt.ru

олимпиады по программированию на Физтехе

Код, производящий выделение сильно связных компонент в графе

Самое поверхностное описание работы алгоритма

Граф задается массивом связей, выходящих из каждой вершины.

Для упрощения ввода, вершины считаются пронумерованными от 0 и далее. Для работы с вершинами, которые задаются, например, *строковыми именами*, следует обратить внимание на хеширование.

```
/*
Формат входных данных:
[Число вершин в графе]
[Число ребер, входящих из вершины (вершина номер 0)] [номер вершины, куда ведет связь] [номер ве                    ]
[Число ребер, входящих из вершины (вершина номер 1)] [номер вершины, куда ведет связь] [номер ве                    ]
...
  Sample intput:
5
1 1
2 2 3
1 0
1 4
1 3
  Sample output:
1 1
2 1
0 1
4 2
3 2
*/
#include <stdio.h>

int N;                  // Количество вершин в графе
#define MAX_NODES 100   // Максимальное количество вершин
#define MAX_EDGES 10    // Максимальное количество ребер, выходящих из одной вершины

int edges[MAX_NODES][MAX_EDGES];  // Граф, в котором ищем сильно связные компоненты
int edges_c[MAX_NODES];

int edgesT[MAX_NODES][MAX_EDGES]; // Транспонированый граф
int edgesT_c[MAX_NODES];

int state[MAX_NODES];             // Ипользуется в поиске для того, чтобы отмечать посещенные вершины

int f[MAX_NODES],last_f=0;        // Список предварительной расстановки вершин

int c=1;  // Номер компоненты (увеличиваем его, когда находим новую)

void dfs(int node){
   state[node]=1;

   for(int i=0; i<edges_c[node]; i++) //  Самый обыкновенный поиск в глубину.
      if (state[edges[node][i]]==0)   //  Проходим по всем непосещенным вершинам,
         dfs(edges[node][i]);         //  заходя в каждую

   f[last_f++]=node;                  //  Предварительная расстановка вершин в списке.
}

void dfsT(int node){
   state[node]=1;

   for(int i=0; i<edgesT_c[node]; i++) //  Самый обыкновенный поиск в глубину в транспонированном графе.
      if (state[edgesT[node][i]]==0)   //  Проходим по всем непосещенным вершинам,
         dfsT(edgesT[node][i]);        //  заходя в каждую

   printf("%d %d\n",node,c);
}

void scc(){  // Strongly Connected Components - функция выделения сильно связных компонент графа
   int i;
   for(i=0; i<N; i++) state[i]=0;

   for(i=0; i<N; i++)                  // 1-ый поиск в глубину
      if (state[i]++==0)               // ...
         dfs(i);                       // Предварительная расстановка вершин.

   for(i=0; i<N; i++) state[i]=0;

   for(last_f--; last_f>=0; last_f--)  // 2-ой поиск в глубину
      if (state[f[last_f]]==0){        // ...
         dfsT(f[last_f]);              // Окончательное выделение сильно связных компонент
         c++;                          // увеличиваем номер следующей компоненты
      }
}

int main(){
   int i;

   scanf("%d",&N);
   for(i=0; i<N; i++) edges_c[i]=edgesT_c[i]=0;
```

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>>

```
   for(i=0; i<N; i++){
      int j;
      scanf("%d",&edges_c[i]);
      for(j=0; j<edges_c[i]; j++){
         int to;
         scanf("%d",&to);
         edges[i][j]=to;  // Построение исходного графа
         edgesT[to][edgesT_c[to]++]=i; // Построение транспонированного графа
      }
   }

   scc(); // Найти и вывести сильно связные компоненты

   return 0;
}
```

-- VladimirSitnikov - 02 Apr 2004

Attachment	Action	Size	Date	Who	Comment
SCC.cpp	manage	2.3 K	02 Apr 2004 - 21:17	VladimirSitnikov	Stongly-Connected-Components.CPP
input.txt	manage	0.1 K	02 Apr 2004 - 21:17	VladimirSitnikov	Sample input