acm.mipt.ru

олимпиады по программированию на Физтехе

Программа поиска максимального потока методом Форда-Фалкерсона

- "Максимальный поток" — формулировка задачи и теория

Тестовый пример:

```
6
0 5
0 16 0 0 13 0
0 0 12 0 6 0
0 0 0 0 9 20
0 0 7 0 0 4
0 0 0 14 0 0
0 0 0 0 0 0
Результат: 23
```

```
#include <memory.h>
#include <stdio.h>

const int MAX_VERTICES = 40;

int NUM_VERTICES; // число вершин в графе
const int INFINITY = 10000; // условное число обозначающее бесконечность

// f - массив садержащий текушее значение потока
// f[i][j] - поток текущий от вершины i к j
int f[MAX_VERTICES][MAX_VERTICES];
// c - массив содержащий вместимоти ребер,
// т.е. c[i][j] - максимальная величину потока способная течь по ребру (i,j)
int c[MAX_VERTICES][MAX_VERTICES];

// набор вспомогательных переменных используемых функцией FindPath - обхода в ширину
// Flow - значение потока чарез данную вершину на данном шаге поиска
int Flow[MAX_VERTICES];
// Link используется для нахождения собственно пути
// Link[i] хранит номер предыдущей вешины на пути i -> исток
int Link[MAX_VERTICES];
int Queue[MAX_VERTICES]; // очередь
int QP, QC; // QP - указатель начала очереди и QC - число эл-тов в очереди

// поск пути по которому возможно пустить поток алгоритмом обхода графа в ширину
// функция ищет путь из истока в сток по которому еще можно пустить поток,
// считая вместимость ребера (i,j) равной c[i][j] - f[i][j],
// т.е. после каждой итерации(одна итерация - один поик пути) уменьшаем вместимости ребер,
// на величину пущеного потока
int FindPath(int source, int target) // source - исток, target - сток
{
        QP = 0; QC = 1; Queue[0] = source;
        Link[target] = -1; // особая метка для стока
        int i;
        int CurVertex;
        memset(Flow, 0, sizeof(int)*NUM_VERTICES); // в начале из всех вершин кроме истока течет 0
        Flow[source] = INFINITY; // а из истока может вытечь сколько угодно
        while (Link[target] == -1 && QP < QC)
        {
                // смотрим какие вершины могут быть достигнуты из начала очереди
                CurVertex = Queue[QP];
                for (i=0; i<NUM_VERTICES; i++)
                // проверяем можем ли мы пустить поток по ребру (CurVertex,i):
                if ((c[CurVertex][i] - f[CurVertex][i])>0 && Flow[i] == 0)
                {
                        // если можем, то добавляем i в конец очереди
                        Queue[QC] = i; QC++;
                        Link[i] = CurVertex; // указываем, что в i добрались из CurVertex
                        // и находим значение потока текущее через вершину i
                        if (c[CurVertex][i]-f[CurVertex][i] < Flow[CurVertex])
                             Flow[i] = c[CurVertex][i];
                        else
                             Flow[i] = Flow[CurVertex];
                }
            QP++;// прерходим к следующей в очереди вершине
        }
        // закончив поиск пути
        if (Link[target] == -1) return 0; // мы или не находим путь и выходим
```

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>>

```
        // или находим:
        // тогда Flow[target] будет равен потоку который "дотек" по данному пути из истока в сток
        // тогда изменяем значения массива f для  данного пути на величину Flow[target]
        CurVertex = target;
        while (CurVertex != source) // путь из стока в исток мы восстанавливаем с помощбю массива Link
        {
                f[Link[CurVertex]][CurVertex] +=Flow[target];
                CurVertex = Link[CurVertex];
        }
        return Flow[target]; // Возвращаем значение потока которое мы еще смогли "пустить" по графу
}

// основная функция поиска максимального потока
int MaxFlow(int source, int target) // source - исток, target - сток
{
        // инициализируем переменные:
        memset(f, 0, sizeof(int)*MAX_VERTICES*MAX_VERTICES); // по графу ничего не течет
        int MaxFlow = 0; // начальное значение потока
        int AddFlow;
        do
        {
                // каждую итерацию ищем какй-либо простой путь из истока в сток
                // и какой еще поток мажет быть пущен по этому пути
                AddFlow = FindPath(source, target);
                MaxFlow += AddFlow;
        } while (AddFlow >0);// повторяем цикл пока поток увеличивается
        return MaxFlow;
}

int main()
{
   int source, target;
   scanf("%d", &NUM_VERTICES);
   scanf("%d %d", &source, &target);
   int i, j;
   for (i=0; i<NUM_VERTICES; i++)
      for (j=0; j<NUM_VERTICES; j++)
         scanf("%d",&c[i][j]);

   printf("%d", MaxFlow(source, target));
   return 0;
}
```

-- AlexeiKurakin - 08 Apr 2004