# Алгоритмы и модели вычислений.
# Задание 12: Алгоритмы на графах I

Сергей Володин, 272 гр.

задано 2014.05.08

## Задача 1

Вход: матрица $A\colon N \times N$ смежности неориентированного графа $G = (V, E)$.

1. Алгоритм:

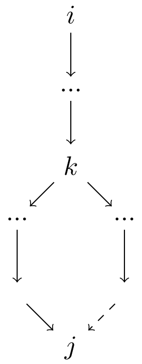
```c
int A[NMAX][NMAX];
int color[NMAX];
int visited[NMAX];

#define nextColor(x) ((x + 1) % 2)

int N;

int dfs(int v, int startColor)
{
  if(visited[v])
    return(startColor != color[v]);

  visited[v] = 1;
  color[v] = startColor;

  for(int i = 0; i < N; i++)
    if(i != v && A[i][v])
    {
      if(dfs(i, nextColor(startColor)))
        return(1);
    }

  return(0);
}

int check()
{
  for(int i = 0; i < N; i++)
    visited[i] = 0;

  bool ans = true;

  for(int i = 0; i < N; i++)
    if(!visited[i])
    {
      if(dfs(i, 0)) ans = false;
      break;
    }

  return(ans);
}
```

2. Время работы. В функции `check()` выполняется не более $N = |V|$ поисков в глубину. Каждый выполняется за $O(|V| + |E|)$, поэтому $T(G) = O(|V|^2 + |V||E|) = O(|V|^2 + |V|^3)$. Описание графа $I(G) = c|V|^2$, поэтому $T(I) = O(I^{\frac{3}{2}})$, поэтому алгоритм эффективный.

3. Идея: выполняем обход из каждой непосещенной вершины, для них определяем долю как 0. Если $u$ — в доле $m$, и ребро $(u, v) \in E$ рассматривается сейчас при обходе, то $v$ в доле $(1 + i) \mod 2$ (в другой). Если возникает противоречие, то не граф двудольный, иначе двудольный, причем найдены доли.

4. Докажем корректность. $P_1(G) =$ [граф $G$ — двудольный], $P_2(G) = [\text{check}()\,|_{G \text{ на входе}} = 1]$. $\urcorner P_2 \Rightarrow \text{check}() = 0$, так как алгоритм всегда останавливается и выдает 0 либо 1. Поэтому нужно доказать $P_1(G) \Leftrightarrow P_2(G)$

(a) Пусть $P_1(G)$, т.е. граф двудольный. Пусть $\neg P_2(G)$. Тогда один из вызовов $\mathrm{dfs}(i, 0)$ вернул 1. Значит, для некоторой вершины $j$ $\mathrm{visited}[j] = 1$, и $\mathrm{startColor} \neq \mathrm{color}[i]$. Цвет вершины (доля) меняется на противоположный при переходе по ребру, значит, существуют два пути $i \to j$ четной и нечетной длины. Значит, существует простой цикл $j \to k \to j$ нечетной длины. Действительно, пусть $T \subseteq G$ — дерево поиска в глубину (пунктирного ребра в нем нет, но оно из $E$). Длина цикла $i \to k \to j \to k \to i$ нечетна, если убрать часть $i \to k \to i$ (четной длины), получится простой цикл $j \to k \to j$ нечетной длины. Значит, граф не двудольный (цикл нечетной длины, доли вершин чередуются, получаем ребро между двумя вершинами из одной компоненты связности — противоречие)




Значит, $P_2(G)$ ■

(b) Пусть $P_2(G)$. Найдены множества $V \supseteq L = \{v \in V \big| \mathrm{color}(v) = 0\}$, $R = \{v \in V \big| \mathrm{color}(v) = 1\}$. Пусть $(L^2 \cup R^2) \cap E \neq \varnothing$. Без ограничения общности, $L^2 \cap E \neq \varnothing$. Пусть $(l_1, l_2) \in E$. У $l_1$ и $l_2$ одинаковые цвета 0. Без ограничени общности $l_2$ была найдена первой при поиске. Тогда при поиске в ширину из $l_1$ был вызов $\mathrm{dfs}(l_2, 1)$, который привел бы к конфликту — противоречие. Значит, это пересечение пусто, и найдены доли графа $\Rightarrow P_1(G)$ ■

## Задача 2

Вход: матрица $A$: $n \times n$ смежности неориентированного графа $G = (V, E)$.

1. Идея: выполним обход из каждой вершины, запоминаем посещенные. Те, которые посещены только при текущем обходе — в новой компоненте связности.

2. Алгоритм: (каждая компонента связности печатается на отдельной строке)

```
void dfs(int v)
{
  if(visited[v])
    return;

  visited[v] = 1;
  cout << v << " ";

  for(int i = 0; i < N; i++)
    if(i != v && arr[i][v])
      dfs(i);
}

void find()
{
  for(int i = 0; i < N; i++)
    visited[i] = 0;

  for(int i = 0; i < N; i++)
    if(!visited[i])
    {
      dfs(i);
      cout << endl;
    }
}
```

3. Время работы (аналогично задаче 1) $T(I) = O(I^{3/2})$.

4. Корректность. Поиск в ширину находит все вершины, достижимые из $v$, и только их, т.е. $C(v)$ — класс эквивалентности $C \in V/\sim$: $C \ni v$ (доказано на семинаре). Последующие вызовы dfs производятся только для непосещенных вершин, т.е. для вершин из других компонент связности.

## Задача 3

1. Пример:

$$s \qquad u \xrightarrow{-1} v$$

Из $s$ нет путей, поэтому все $\mathrm{d}[i] = \infty$ — правильный ответ.

2. Пример:

$$\begin{tikzcd} s \arrow[r, bend left, "1"] & u \arrow[l, bend left, "-2"] \end{tikzcd}$$

На первой итерации будет найдено $\mathrm{d}[s] = 0$, $\mathrm{d}[u] = \infty$. На второй (непомеченная вершина с минимальным $\mathrm{d}$ — $s$) $\mathrm{d}[s] = 0$, $\mathrm{d}[u] = 1$, на третьей (непомеченная вершина с минимальным $\mathrm{d}$ — $u$) $\mathrm{d}[s] = -1$, $\mathrm{d}[u] = 1$, и алгоритм остановится (все вершины помечены). Для $s$ ответ неверный: $0 \neq 1$

## (каноническое) Задача 51.1, 51.2

1. Идея: модифицируем алгоритм Беллмана-Форда (релаксации).

2. Надежность пути $u \to v_1 \to ... \to v_k \to v$ по определению $r(u, v_1, v_2, ..., v_k, v) \overset{\text{def}}{=} r(u, v_1) \cdot r(v_1, v_2) \cdot ... \cdot r(v_k, v)$

3. Алгоритм (печатается путь в обратном порядке и максимальные надежности $s \to v_i$):

```
void init(int s)
{
  for(int i = 0; i < N; i++)
  {
    p[i] = -1;
    r[i] = 0;
  }

  r[s] = 1;
}

void solve(int s, int t)
{
  init(s);

  for(int i = 0; i < N; i++)
    for(int u = 0; u < N; u++)
      for(int v = 0; v < N; v++)
      {
        double *r0 = &(r[v]);
        double r1 = r[u] * arr[u][v];
        if(*r0 < r1)
        {
          *r0 = r1;
          p[v] = u;
        }
      }

  for(int k = t; k != -1; k = p[k])
    cout << k << " ";
  cout << endl;
}
```

4. Это алгоритм Беллмана-Форда с восстановлением пути (массив предков) с другой релаксацией

$$(u, v) \in E \Rightarrow r(v) = \max(r_{k-1}(v), r_{k-1}(u) \cdot r(u, v))$$

5. Время работы равно времени работы алгоритма Беллмана-Форда. Рассмотрим одну релаксацию. Пусть числа по $m$ бит. Надежность сети имеет $nm$ бит, поэтому перемножение двух таких чисел займет $cn^2m^2$ тактов. Всего $O(|V|^4m^2)$ операций — полином от входа (эффективен на всех сетях?).

6. Корректность. Рассмотрим сеть $(G, r')$, $r'(u, v) = -\ln r(u, v)$. Тогда (индукция по $k$) релаксация запишется как $r_k(v) = \max(r_{k-1}(v), r_{k-1}(u) \cdot r(u, v)) = \max(e^{-r'_{k-1}(v)}, e^{-r'_{k-1}(u)} e^{-r'(u,v)}) = e^{-\min(r'_{k-1}(v), r'_{k-1}(u) \cdot r'(u,v))}$, т.е. $r'_k(v) = -\ln r_k(v) = \min(r'_{k-1}(v), r'_{k-1}(v) \cdot r'(u, v))$, т.е. выполняется обычный алгоритм Беллмана-Форда в сети $(G, r')$. Он найдет путь $s \to v$ с минимальным значением $r'$, т.е. (монотонность $\ln$) максимальным $r$.

7. Выполним алгоритм на графе из условия. Файл test1, запускать:

```
$ g++ main.cpp && cat test1 | ./a.out
```

   Ответ: путь 03142578 (нумерация вершин слева направо и сверху вниз).

## (каноническое) Задача 51.3

1. Заметим, что задача сводится к поиску минимального остовного дерева в ориентированном графе с вершиной $s$. Будут найдены ребра, такие что граф $T \subseteq G$ связен, и имеет минимальный вес. Сеть $(G, r')$: $r'(u, v) = -\ln r(u, v)$. Будет найдено дерево, причем $r'(T) = -\ln r(u_1, u_2) - \ln r(u_2, u_3) ... = -\ln(r(T)) \to \min \Rightarrow r(T) \to \max$.

2. Для данного случая: минимальное количество ребер $n-1=9-1=8$. Рассмотрим ребра с наибольшими надежностями $(0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.12)$. Произведение монотонно по каждому из сомножителей, поэтому большей надежности получить нельзя. Подграф $(V, E')$ с выделенными ребрами

$$\begin{tikzcd}
s \arrow[r, "0.03"] \arrow[d, "0.12"'] & v2 \arrow[r, "0.12"] \arrow[d, "0.5"'] & v3 \arrow[d, "0.5"] \\
v4 \arrow[ur, "0.5"] \arrow[r, "0.06"] \arrow[d, "0.5"] & v5 \arrow[ur, "0.5"] \arrow[r, "0.12"] \arrow[d, "0.06"'] & v6 \arrow[dl, "0.5"'] \arrow[d, "0.11"] \\
v7 \arrow[r, "0.03"] & v8 \arrow[r, "0.5"] & t
\end{tikzcd}$$

связен: $s, v_4, v_2, v_5, v_3, v_6, v_8, t$ достижимы, так как существует такой путь, $(v_4, v_7) \in E'$, поэтому $v_7$ достижима.

3. Ответ:

$$\begin{tikzcd}
s \arrow[d, "0.12"'] & v2 \arrow[d, "0.5"'] & v3 \arrow[d, "0.5"] \\
v4 \arrow[ur, "0.5"] \arrow[d, "0.5"] & v5 \arrow[ur, "0.5"] & v6 \arrow[dl, "0.5"'] \\
v7 & v8 \arrow[r, "0.5"] & t
\end{tikzcd}$$