УДК 519.1

к.ф.-м.н., доц., Кубайчук О.О.
Європейський університет

ПОШУК НА ГРАФАХ В АНАЛІЗІ ФІНАНСОВОЇ ДІЯЛЬНОСТІ

В середовищі MATHCAD побудовано процедуру DFS_CTRL, яка дозволяє відстежувати зв'язки між дискретними об'єктами (наприклад, банківськими рахунками), використовуючи алгоритм пошуку (обходу) в глибину для орієнтованого графа.

Постановка задачі. Тема боротьби із зловживаннями у фінансовій сфері є актуальною в період бурхливого розвитку економіки. Задача контролю за фінансовими операціями є досить складною, за умови величезного числа таких операцій між господарюючими суб'єктами.

В даній роботі під операцією будемо розуміти факт перерахування коштів з рахунку i на рахунок j, де $1 \le i, j \le N$, $i \ne j$, і N – число рахунків у схемі, що досліджується. Операцію зручно зобразити у вигляді направленого відрізка (стрілки). Тоді, схема – орієнтований граф у якого вершини – рахунки, а ребра – операції. Як показує практика такі графи є розрідженими. Тому, краще використати їх представлення списками суміжності (Хопкрофт (Hopcroft) і Таржан (Tarjan) в [1]).

Для спотворення реального руху коштів застосовують, так звані, «тіньові» схеми. Однією з них є введення значної кількості транзитних рахунків, які утруднюють контроль. В даній роботі пропонується інструмент (процедура DFS_CTRL, розроблена в середовищі MATHCAD), який дає можливість відстежити приховані зв'язки між рахунками. Потрібно однак відмітити, що запропонована методика не дає відповіді про кількісний характер зв'язку, а фіксує тільки його факт.

Логічним є запитання. Чому саме MATHCAD? Звичайно, спеціаліст-практик може використовувати готові процедури для роботи з графами (наприклад [3]), але для цього необхідно мати багатий досвід програмування, зокрема, на C++. MATHCAD є потужною альтернативою, яка дозволяє зосередитись на розв'язанні конкретної задачі за допомогою комп'ютера, без належної фахової підготовки в галузі програмування.

Ключовим моментом у процедурі DFS_CTRL є застосування алгоритму пошуку в глибину, який широко почали застосовувати з кінця 50-х років минулого століття (наприклад, в програмах штучного інтелекту). Пізніше, Таржан і Хопкрофт в [1] показали важливість цього алгоритму для розробки інших ефективних алгоритмів. Рекурсивний варіант пошуку в глибину можна знайти, наприклад, в [2].

Одним із застосувань пошуку в глибину є його використання для класифікації ребер вихідного графа. Нехай, $G_я$ – ліс, отриманий внаслідок пошуку в глибину на графі G. Відповідно визначають чотири типи ребер:

1. Ребра дерев (tree edges, позначають "T") – це ребра графа $G_я$.
2. Обернені ребра (back edges, позначають "B") – це ребра, що з'єднують вершину з її предком у дереві пошуку в глибину. Ребра-цикли, які можуть зустрічатися в орієнтованих графах, розглядаються як обернені ребра.
3. Прямі ребра (forward edges, позначають "F") – це ребра, що не є ребрами дерева і з'єднують вершину з її потомком у дереві пошуку в глибину.
4. Перехресні ребра (cross edges, позначають "C") – всі інші ребра графа.

Процедура DFS_CTRL. Процедура DFS_CTRL використовує алгоритм пошуку в глибину, наведений у [2] з такими модифікаціями:

1. Використовується стек, що дозволяє позбутися рекурсії.
2. В процесі роботи алгоритму ребра класифікуються. Причому процедура не розрізняє прямі та перехресні ребра, а відносить їх в клас "FC".

Орієнтований граф G є ациклічним тоді і тільки тоді, коли пошук в глибину на G не знаходить обернених ребер (Лема 22.11 в [2]). Іншими словами, якщо фіксується обернене ребро, то досліджуваний граф гарантовано містить цикл, що у свою чергу сигналізує про можливі порушення.

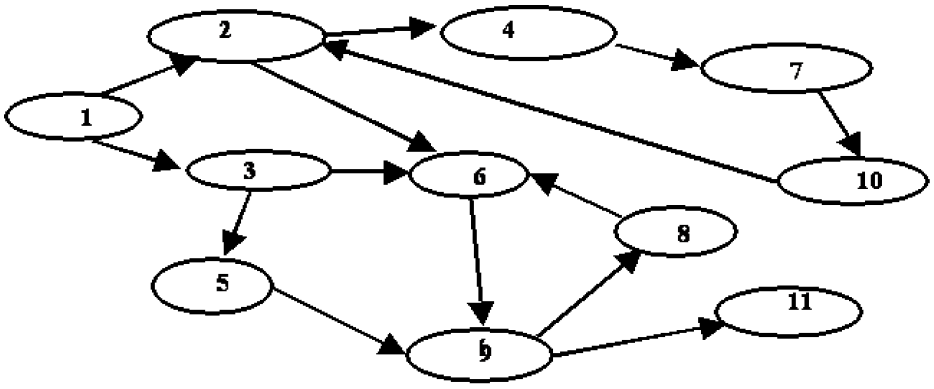

Рис. 1 Система рахунків

Розглянемо роботу DFS_CTRL на конкретному прикладі. Нехай граф на Рис.1 моделює зв'язки, між рахунками.

a) Задамо граф списками суміжності.

$$GRAF := \begin{pmatrix} 1 & T1 \\ 2 & T2 \\ 3 & T3 \\ 4 & T4 \\ 5 & T5 \\ 6 & T6 \\ 7 & T7 \\ 8 & T8 \\ 9 & T9 \\ 10 & T10 \\ 11 & T11 \end{pmatrix}$$

$$T3 := \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

$$T2 := \begin{pmatrix} 4 \\ 6 \end{pmatrix}$$

$$T1 := \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

$T5_1 := 9$

$T6_1 := 9$

$T4_1 := 7$

$T8_1 := 6$

$$T9 := \begin{pmatrix} 8 \\ 11 \end{pmatrix}$$

$T7_1 := 10$

$T11 := 0$

$T10_1 := 2$

$STACK\_EMPTY(S) := S_1 = 0$

$STACK\_OVER(S) := S_1 = S\_max$

b) Операції зі стеком.

```
S_max := 10

INI_STACK := | head ← 0
             | for i ∈ 1 ,S_max
             |   S_i ← 0
             | ⟨head⟩
             |  S
```

```
POP(S) := | error("STACK UNDERFLOW")  if  STACK_EMPTY(S)
          | S_1 ← S_1 - 1
          | [ (S_2)_(S_1+1) ]
          |   S
```

```
PUSH(S ,x) := | error("STACK OVERFLOW")  if  STACK_OVER(S)
              | S_1 ← S_1 + 1
              | (S_2)_(S_1) ← x
              | S
```

c) Пошук непройдених ребер у списку суміжності.

```
FIND_MARK(v,T,B) := | return -1  if  IsScalar(T)
                    | for  i ∈ 1 .. length(T)
                    |     | w ← T_i
                    |     | return w  if  B_(v ,w) = 0
                    | -1
```

d) Допоміжні функції.

```
IS_WHITE(color) := color = "WHITE
```

```
fm(i,j) := 0
```

```
T_num := rows(GRAF)
```

e) Аналіз графа.

```
DFS_CTRL(G) :=
    for s ∈ 1 .. T_num
        color_s ← "WHITE"
        d_s ← 0
        f_s ← 0
        pi_s ← "NIL"
    M ← matrix(T_num, T_num, fm)
    time ← 0
    S ← INI_STACK
    for k ∈ 1 .. T_num
      if IS_WHITE(color_k)
        color_k ← "GRAY"
        d_k ← time ← time + 1
        S ← PUSH(S, k)
        v ← k
        while ¬STACK_EMPTY(S)
            w ← FIND_MARK(v, G_v, 2, M)
            if w ≠ −1
                if color_w ≠ "WHITE"
                    M_v,w ← "B" if color_w = "GRAY"
                    M_v,w ← "FC" otherwise
                otherwise
                    M_v,w ← "T"
                    color_w ← "GRAY"
                    d_w ← time ← time + 1
                    pi_w ← v
                    S ← PUSH(S, w)
                    v ← w
            otherwise
                v ← POP(S)_1
                S ← POP(S)_2
                color_v ← "BLACK"
                f_v ← time ← time + 1
                v ← pi_v
    M
```

DFS_CTRL(GRAF) =

	1	2	3	4	5	6	7	8	9	10	11
1	0	"T"	"T"	0	0	0	0	0	0	0	0
2	0	0	0	"T"	0	"T"	0	0	0	0	0
3	0	0	0	0	"T"	"FC"	0	0	0	0	0
4	0	0	0	0	0	0	"T"	0	0	0	0
5	0	0	0	0	0	0	0	0	"FC"	0	0
6	0	0	0	0	0	0	0	0	"T"	0	0
7	0	0	0	0	0	0	0	0	0	"T"	0
8	0	0	0	0	0	"B"	0	0	0	0	0
9	0	0	0	0	0	0	0	"T"	0	0	"T"
10	0	"B"	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0

Рис. 2 Результат роботи DFS_CTRL.

f) Результат аналізу.

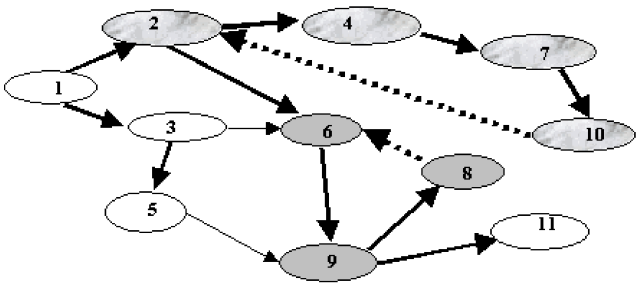

Рис. 3 Виявлені цикли.

На Рис.2 можна побачити результат роботи процедури DFS_CTRL, а на Рис. 3 зображено цикли (пунктирною стрілкою позначені ребра-цикли, самі цикли зафарбовані відповідними кольорами, жирні стрілки – це ребра дерев пошуку в глибину).

Висновки. В теоретико-графовій постановці сформульована задача виявлення зв'язків в певній схемі, яку можуть застосовувати при відмиванні «брудних коштів». Запропоновано методику, в основі якої лежить застосування алгоритму пошуку в глибину, що дозволяє вирішити поставлену задачу. Нарешті, в середовищі MATHCAD розроблено процедуру DFS_CTRL, яка реалізує задумане.

Список літератури

1. *John E. Hopcroft, Robert E. Tarjan*. Efficient Algorithms for Graph Manipulation // Communications of the ACM. – 1973. 16(6). – p. 372-378.
2. *Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн*. Алгоритмы. Построение и анализ. –М., 2005.
3. *Р. Седжвик*. Фундаментальные алгоритмы на С++. Алгоритмы на графах. – СПб., 2002.