Phân tích và Thiết kế THUẬT TOÁN

Hà Đại Dương

duonghd@mta.edu.vn

Web: fit.mta.edu.vn/~duonghd

Bài 12 - Back tracking method (tiếp)

PHÂN TÍCH VÀ THIẾT KẾ THUẬ TOÁN

NỘI DUNG

I. Giới thiệu
II. Lược đồ chung
III. Bài toán áp dụng
IV. Bài tập

III. Bài toán áp dụng

4. Bài toán liệt kê các hoán vị

1. Phát biểu bài toán

Liệt kê các hoán vị của n số nguyên dương đầu tiên.

III. Bài toán áp dụng

4. Bài toán liệt kê các hoán vị

2. Thiết kế thuật toán

Ta biểu diễn các hoán vị dưới dạng $a_1 \dots a_n$; $a_i \in \{1,..,n\}$ và $a_i \neq a_j$ nếu $i \neq j$. Với mọi i, a_i chấp nhận giá trị j nếu j chưa được sử dụng, và vì vậy ta cần ghi nhớ j đã được sử dụng hay chưa khi quay lui. Để làm điều này ta dùng một dãy các biến logic b_j với quy ước :

$$\forall j = \overline{1,n} : b_j = \begin{cases} 1; \textit{nếu } j \textit{ chưa sử dụng} \\ 0; \textit{nếu ngược lại.} \end{cases}$$

III. Bài toán áp dụng

4. Bài toán liệt kê các hoán vị

2. Thiết kế thuật toán

Sau khi gán j cho a_i , ta cần ghi nhớ cho b_j ($b_j = 0$) và phải trả lại trạng thái cũ cho b_j (b_j = True) khi thực hiện việc in xong một hoán vị.

Ta chú ý rằng dãy các biến b_j sẽ được khởi động bằng 1

III. Bài toán áp dụng

4. Bài toán liệt kê các hoán vị

Cài đặt

```
Try( i)≡
{
        for ( j = 1; j <= n; j++)
            if ( b[j])
            {
                a[i] = j;
                b[j] = 0;          // Ghi nhận trạng thái mới
                if (i < n)
                    Try(i+1);
                else
                    Xuất();
                b[j] = True;     // Trả lại trạng thái cũ
            }
}
```

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

1. Phát biểu bài toán

G = (V, U) là đơn đồ thị (có hướng hoặc vô hướng). V = {1,. ., n} là tập các đỉnh, U là tập cạnh (cung). Với s, t ∈ V, tìm tất cả các đường đi từ s đến t.

Các thuật toán tìm kiếm cơ bản :

Thuật toán DFS : Tìm kiếm theo chiều sâu.

Thuật toán BFS : Tìm kiếm theo chiều rộng.

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

2. Thuật toán DFS (Depth First Search)

a) Ý tưởng

Thuật toán DFS tiến hành tìm kiếm trong đồ thị theo chiều sâu. Thuật toán thực hiện việc thăm tất cả các đỉnh có thể đạt được cho tới đỉnh t từ đỉnh s cho trước. Đỉnh được thăm càng muộn sẽ càng sớm được duyệt xong (cơ chế LIFO – Vào Sau Ra Trước). Nên thuật toán có thể tổ chức bằng một thủ tục đệ quy quay lui.

III. Bà toá

5. Bài toán tìm kiếm đường đ

b) Mô tả thuật toán

```
Input  G = (V,U), s, t
Output  Tất cả các đường đi từ s đến t (nếu có).
DFS ( int  s) ≡
        for ( u = 1; u <= n; u++)
        {
                if  (chấp nhận được)
                {
                        Ghi nhận nó;
                        if  (u ≠ t)
                                DFS(u);
                        else
                                In đường đi;
                        bỏ việc ghi nhận;
                }
        }
```

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

c) Cài đặt

Ta cần mô tả dữ liệu đồ thị và các mệnh đề được phát biểu trong mô hình.

Ma trận sẽ được biểu diễn bằng ma trận kề :

$$a_{ij} = \begin{cases} 1; (i,j) \in U \\ 0; (i,j) \notin U \end{cases}$$

Ghi nhận đỉnh được thăm để tránh trùng lặp khi quay lui bằng cách đánh dấu. Ta sữ dụng một mảng một chiều Daxet[] với qui ước :

Daxet[u] = 1 , u đã được thăm.

Daxet[u] = 0 , u chưa được thăm.

Mảng Daxet[] lúc đầu khởi động bằng 0 tất cả.

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

c) Cài đặt

Điều kiện chấp nhận được cho đỉnh u chính là u kề với v ($a_{vu} = 1$) và u chưa được thăm (Daxet[u] = 0).

Để ghi nhận các đỉnh trong đường đi, ta dùng một mảng một chiều Truoc[ ], với qui ước :

Truoc[u] = v $\Leftrightarrow$ v là đỉnh đứng trước đỉnh u, và u kề với v

Ta khởi động mảng Truoc[ ] bằng 0 tất cả.

5. Bài toán tìm

c) Cài đặt

Input $G = (a_{ij})_{nxn}$, s, t

Output Tất cả các đường đi từ s đến t (nếu có).

void DFS(s) ≡ toá

```
int u;
daxet[s] = 1;
for( u = 1;u <= n; u++)
{
        if( a[s][u] && !daxet[u])
        {
                Truoc[u] = s;
                if ( u == t )
                        Xuat_duongdi();
                else
                        DFS(u);
                daxet[u] = 0;
        }
}
```

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

c) Cài đặt

Mảng truoc[ ] lưu trử các đỉnh trên đường đi,

Nếu kết thúc thuật toán, Daxet[t] = 0 (Truoc[t] = 0) thì không có đường đi từ s đến t.

Trong trường hợp tồn tại đường đi, xuất đường đi chính là xuất mảng Truoc[ ]. Thao tác này có thể viết như sau :

```
Xuat_duongdi()≡
        cout<<t<<"<--";
        j = t;
        while ( truoc[j] != s)
        {
                cout<<truoc[j]<<"<--";
                j = truoc[j];
        }
        cout<<s<<endl;
```

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

d) Kết quả

Với đồ thị có hướng cho bởi ma trận kề :

7

0 0 0 1 0 1 1
0 0 1 1 0 0 0
0 1 0 1 0 1 0
1 0 1 0 0 0 0
0 0 0 0 1 1 0
1 0 0 0 1 0 0
1 0 0 1 0 0 0

s = 1, t = 4
4 ← 1
4 ← 7 ← 1

s = 2, t = 5
5 ← 6 ← 1 ← 4 ← 3 ← 2
5 ← 6 ← 3 ← 2
5 ← 6 ← 1 ← 4 ← 2
5 ← 6 ← 3 ← 4 ← 2

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

3. Thuật toán BFS (Breadth First Search)

a) Ý tưởng

Thuật toán BFS tiến hành tìm kiếm trên đồ thị theo chiều rộng. Thuật toán thực hiện việc thăm tất cả các đỉnh có thể đạt được cho tới đỉnh t từ đỉnh s cho trước theo từng mức kề. Đỉnh được thăm càng sớm thì sẽ càng sớm được duyệt xong (cơ chế FIFO – Vào Trước Ra Trước).

5. Bài toán tìm toá

b) Mô tả thuật toán

Input $G = (V,E)$;

$s, t \in V$;

Output

Đường đi từ s đến t.

Mô tả :

- Bước 0 : $A_0 = \{s\}$.
- Bước 1 : $A_1 = \{x \in V \setminus A_0 : (s,x) \in E\}$.
- Bước 2 : $A_2 = \{x \in V \setminus [A_0 \cup A_1] : \exists y \in A_1, (y,x) \in E\}$.
- . . .
- Bước i : $A_i = \{x \in V \setminus \bigcup_{k=0}^{i-1} A_k : \exists y \in A_{i-1}, (y,x) \in E\}$.
- …

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

Thuật toán có không quá n bước lặp; một trong hai trường hợp sau xảy ra :

- Nếu với mọi i, $t \notin A_i$: không có đường đi từ s đến t;
- Ngược lại, $t \in A(m)$ với m nào đó. Khi đó tồn tại đường đi từ s tới t, và đó là một đường đi ngắn nhất từ s đến t.

Trong trường hợp này, ta xác định được các đỉnh trên đường đi bằng cách quay ngược lại từ t đến các đỉnh trước t trong từng các tập trước cho đến khi gặp s.

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

c) Cài đặt

Trong thuật toán BFS, đỉnh được thăm càng sớm sẽ càng sớm trở thành duyệt xong, nên các đỉnh được thăm sẽ được lưu trử trong hàng đợi queue. Một đỉnh sẽ trở thành duyệt xong ngay sau khi ta xét xong tất cả các đỉnh kề của nó .

Ta dùng một mảng logic Daxet[ ] để đánh dấu các đỉnh được thăm, mảng này được khởi động bằng 0 tất cả để chỉ rằng lúc đầu chưa đỉnh nào được thăm.

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

c) Cài đặt

Một mảng truoc[ ] để lưu trữ các đỉnh nằm trên đường đi ngắn nhất cần tìm (nếu có), với ý nghĩa Truoc[i] là đỉnh đứng trước đỉnh i trong đường đi. Mảng Truoc[ ] được khởi động bằng 0 tất cả để chỉ rằng lúc đầu chưa có đỉnh nào.

Đồ thị G được biểu diễn bằng ma trận kề $a = (a_{uv})_{nxn}$

trong đó : $a_{uv} = \begin{cases} 1; (u,v) \in E; \\ 0; (u,v) \notin E; \end{cases}$

5. Bài toán tìm k

c) Cài đặt

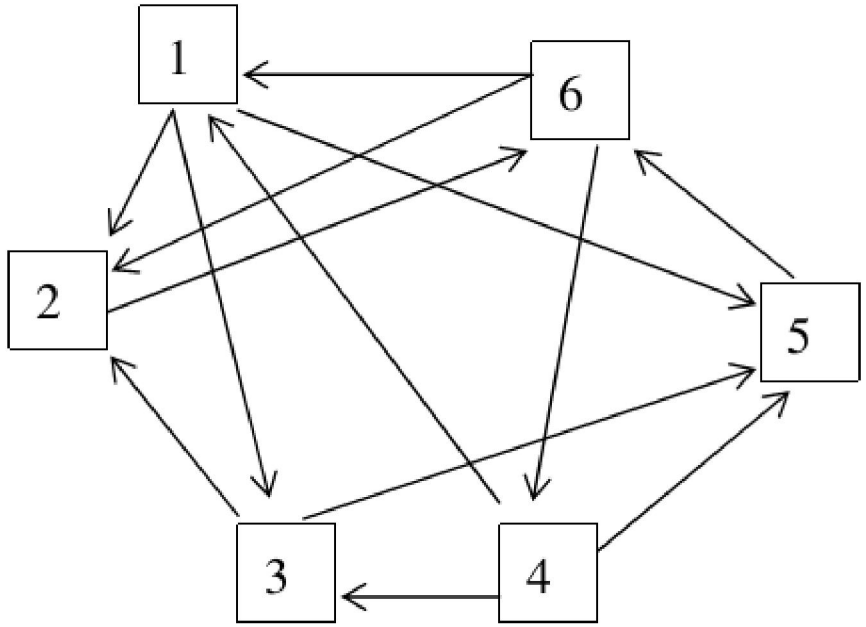
```
BFS(s) ≡
        int u, j, dauQ = 1, cuoiQ = 1;
        queue[cuoiQ] = s;
        Daxet[s] = 1;
        while ( dauQ <= cuoiQ)
        {
                u = queue[dauQ];
                dauQ++;
                for ( j = 1; j <= n; j++)
                        if( a[u][j] == 1 && !Daxet[j] )
                        {
                                cuoiQ++;
                                queue[cuoiQ] = j;
                                Daxet[j] = 1;
                                Truoc[j] = u;
                        }
        }
```

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

Minh hoạ :

Cho đơn đồ thị có hướng :



Tìm đường đi từ đỉnh (1) đến đỉnh (4) :

$A(0) = \{1\};$

$A(1) = \{2,3,5\}$

$A(2) = \{6\}$

$A(3) = \{4\}$

Đường đi ngắn nhất tìm được là

$4 \leftarrow 6 \leftarrow 5 \leftarrow 1$, có chiều dài là 3.

III. Bài toán áp dụng

5. Bài toán tìm kiếm đường đi trên đồ thị

Nhận xét

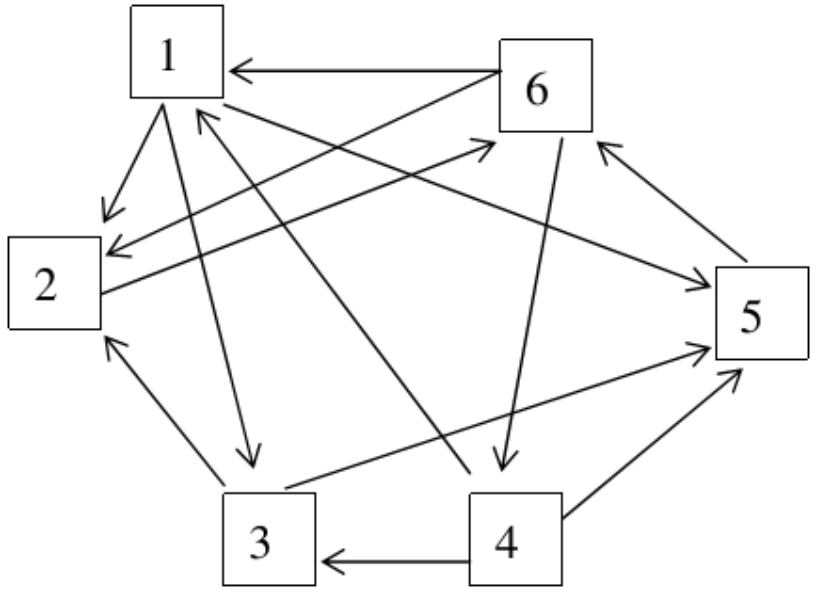
Ta có thể thấy mỗi lần gọi DFS(s), BFS(s) thì mọi đỉnh cùng thành phần liên thông với s sẽ được thăm, nên sau khi thực hiện hàm trên thì :

- Truoc[t] == 0 : có nghỉa là không tồn tại đường đi từ s đến t,
- Ngược lại, có đường đi từ s đến t. Khi đó lời giải được cho bởi :

$$t \leftarrow p1 = Truoc[t] \leftarrow p2 = Truoc[p1] \leftarrow \ldots \leftarrow s .$$

IV. Bài tập

Cho đồ thị có hướng



1. Thực hiện từng bước thuật toán DFS trên đồ thị tìm đường đi từ đỉnh 1 đến đỉnh 4.
2. Thực hiện từng bước thuật toán BFS trên đồ thị tìm đường đi từ đỉnh 2 đến đỉnh 5.

IV. Bài tập

3. Cài đặt thuật toán giải bài toán người du lịch (dựa trên thuật toán liệt kê các hoán vị) theo phương pháp quay lui. Đánh giá độ phức tạp thuật toán bằng lý thuyết, bằng thực nghiệm và so sánh.
4. Cài đặt thuật toán tìm đường đi trên đồ thị, thuật toán DFS, theo phương pháp quay lui . Đánh giá độ phức tạp thuật toán bằng lý thuyết, bằng thực nghiệm và so sánh.
5. Cài đặt thuật toán tìm đường đi trên đồ thị, thuật toán BFS, theo phương pháp quay lui . Đánh giá độ phức tạp thuật toán bằng lý thuyết, bằng thực nghiệm và so sánh.

NỘI DUNG

I. Giới thiệu
II. Lược đồ chung
III. Bài toán áp dụng
IV. Bài tập