HỌC VIỆN KỸ THUẬT QUÂN SỰ

KHOA CÔNG NGHỆ THÔNG TIN

****

**BÁO CÁO MÔN HỌC**

**PHÂN TÍCH VÀ THIẾT KẾ GIẢI THUẬT**

***Đề 21:***

***Thiết kế thuật toán tìm đường đi trên đồ thị***

***theo phương pháp quay lui, thuật toán BFS***

*Giáo viên hướng dẫn: Hà Đại Dương*

*Sinh viên thực hiện: Bùi Thị Mai Hương*

*Lớp: TH12B*

Hà Nội, 6/2016

**MỤC LỤC**

[**I.** **GIỚI THIỆU CHUNG VỀ PHƯƠNG PHÁP QUAY LUI** 2](#_Toc453053187)

[**1.** **Ý tưởng** 2](#_Toc453053188)

[**2.** **Mô hình** 3](#_Toc453053189)

[**II.** **THUẬT TOÁN BFS – DUYỆT ĐỒ THỊ THEO CHIỀU RỘNG** 3](#_Toc453053190)

[**1.** **Bài toán** 3](#_Toc453053191)

[**2.** **Mô tả chi tiết thuật toán** 3](#_Toc453053192)

[**3.** **Cài đặt chương trình sử dụng C, C++** 5](#_Toc453053193)

[**4.** **Thực hiện từng bước thuật toán trên 5 bộ dữ liệu** 6](#_Toc453053194)

[**5.** **Đánh giá độ phức tạp của thuật toán theo lý thuyết và bằng thực nghiệm, so sánh kết quả** 17](#_Toc453053195)

1. **GIỚI THIỆU CHUNG VỀ PHƯƠNG PHÁP QUAY LUI**
2. **Ý tưởng**

Nét đặc trưng của phương pháp quay lui là các bước hướng tới lời giải cuối cùng của bài toán được làm thử.

Tại mỗi bước, nếu có một lựa chọn được chấp nhận thì ghi nhận lại lựa chọn này và tiến hành các bước thử tiếp theo. Còn ngược lại không có lựa chọn thích hợp thì làm lại bước trước, xóa bỏ sự ghi nhận và quay về chu trình thử các lựa chọn còn lại. Hành động này được gọi là quay lui, thuật toán thể hiện phương pháp này gọi là quay lui.

Điểm quan trọng của thuật toán là phải ghi nhớ tại mỗi bước đi qua để tránh trùng lặp khi quay lui. Dễ thấy là các thông tin này cần được lưu trữ vào một ngăn xếp, nên thuật toán thể hiện ý thiết kế một cách đệ quy.

Lời giải của bài toán thường được biểu hiện bằng một véc tơ gồm n thành phần x = () phải thỏa mãn các điều kiện nào đó. Để chỉ ra lời giải x, ta phải xây dựng các thành phần lời giải .

Tại mỗi bước i:

* Đã xây dựng xong các thành phần
* Xây dựng thành phần bằng cách lần lượt thử tất cả các khả năng mà có thể chọn:
  + Nếu một khả năng j nào đó phù hợp cho thì xác định theo khả năng j. Thường phải có thêm thao tác ghi nhận trạng thái mới của bài toán để hỗ trợ cho bước quay lui. Nếu I = n thì ta có được một lời giải, ngược lại thì tiến hành bước i + 1 để xác định .
  + Nếu không có một khả năng nào chấp nhận được cho thì ta lùi lại bước trước (bước i - 1) để xác định lại thành phần .

Thuật toán quay lui thường được áp dụng khi giải các bài toán: bài toán 8 hậu, mã đi tuần, liệt kê các hoán vị, duyệt đồ thị.

1. **Mô hình**

Mô hình của phương pháp quay lui có thể viết bằng thủ tục sau, với n là số bước cần phải thực hiện, k là số khả năng mà có thể chọn lựa.

Try(i) ≡

for (j = 1 k)

if( chấp nhận được khả năng j)

{

Xác định theo khả năng j;

Ghi nhận trạng thái mới;

if(i < n)

Try(i + 1);

else

Ghi nhận nghiệm;

Trả lại trạng thái cũ cho bài toán;

}

1. **THUẬT TOÁN BFS – DUYỆT ĐỒ THỊ THEO CHIỀU RỘNG**
2. **Bài toán**

Xét một đơn đồ thị có hướng hoặc vô hướng G= (V, E).

V = {1,2,…..,n} là tập các đỉnh, E là tập cạnh (cung). Với s, t V, tìm tất cả các đường đi từ s đến t.

Sử dụng thuật toán BFS: tìm kiếm theo chiều rộng.

1. **Mô tả chi tiết thuật toán**
2. 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 Sau).

1. Mô tả chi tiết thuật toán:

Input: G≡ (V, E)

s, t V;

Output: Đường đi từ s đến t.

Mô tả:

* Bước 0: ={s}.
* Bước 1: .
* Bước 2: .
* …
* Bước i: .
* …

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

* Nếu với mọi i, t : không có đường đi nào từ s đến t;
* Ngược lại, t 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.

1. 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 trữ 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.

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=

Trong đó:

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;

}

}

Ta có thể thấy mỗi lần gọi 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 nghĩa 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 pl = Truoc[t] p2= Truoc[p1] …. s.

1. **Cài đặt chương trình sử dụng C, C++**

void BFS(int s, int t){

cout<<"So dinh cua do thi n = "<<endl;

cin>>n;

cout<<"nhap ma tran ke"<<endl;

for(int i=1; i<=n;i++){

for(int j=1; j<=n;j++){

cin>>G[i][j];

}

}

for(int i=1; i<=n;i++){

Daxet[i]=0;

Truoc[i]=0;

}

cout<< "nhap dinh bat dau: ";

cin>>s;

cout<<"nhap dinh ket thuc: ";

cin>>t;

int u, dauQ, cuoiQ;

dauQ = 1;

cuoiQ = 1;

QUEUE[cuoiQ] = s;

Daxet[s] = TRUE;

while(dauQ <= cuoiQ){

u = QUEUE[dauQ];

cout<<"duyet dinh: "<<u<<endl;

dauQ=dauQ+1;

for(int j=1; j<=n;j++){

if(G[u][j]==1 && !Daxet[j] ){

cuoiQ=cuoiQ+1;

QUEUE[cuoiQ]=j;

Daxet[j]=TRUE;

Truoc[j]=u;

}

}

}

}

1. **Thực hiện từng bước thuật toán trên 5 bộ dữ liệu**

***Bộ dữ liệu số 1: tìm đường đi từ đỉnh 1 đến đỉnh 5***

2

6

1

5

4

3

Các bước của thuật toán:

* Bước 1: Khởi tạo

n = 6;

Ma trận kề:

Daxet[] = 0;

Truoc[]= 0;

* Bước 2: (Bước lặp)

dauQ= 1;

cuoiQ = 1;

QUEUE[1]=1;

Daxet[1]=1;

// bắt đầu vòng lặp while

* B1: u=QUEUE[1]=1;

dauQ=2;

a[1][1]=a[1][2]= 0

a[1][3]= 1 và Daxet[3]=0

cuoiQ= 2;

QUEUE[2]= 3;

Daxet[3]=1;

Truoc[3]=1;

a[1][4]=1 và Daxet[4]=0

cuoiQ= 3;

QUEUE[3]= 4;

Daxet[4]=1;

Truoc[4]=1;

* B2: u=QUEUE[2]=3;

dauQ=3;

a[3][4]= 1 và Daxet[4]=1

* B3: u=QUEUE[3]=4;

dauQ=4;

a[4][2]= 1 và Daxet[2]=0

cuoiQ= 4;

QUEUE[4]= 2;

Daxet[2]=1;

Truoc[2]=4;

a[4][5]= 1 và Daxet[5]=0

cuoiQ= 5;

QUEUE[5]= 5;

Daxet[5]=1;

Truoc[5]=4;

a[4][6]= 1 và Daxet[6]=0

cuoiQ= 6;

QUEUE[6]= 6;

Daxet[6]=1;

Truoc[6]=4;

* B4: u=QUEUE[4]=2;

dauQ=5;

a[2][1]=1 và Daxet[1]=0

* B5: u=QUEUE[5]=5;

dauQ=6; // đã xét hết các đỉnh

* B6: u=QUEUE[6]=6;

dauQ=7; // dauQ> cuoiQ: kết thúc vòng lặp while

* Vậy đường đi từ đỉnh 1 đến đỉnh 5 là :

1 🡪 4 🡪 5

* Tóm tắt:

= {1};

= {3, 4};

= {2, 5, 6};

***Bộ dữ liệu số 2: tìm đường đi từ đỉnh 2 đến đỉnh 5***

6

1

5

2

4

3

Các bước của thuật toán:

* Bước 1: Khởi tạo

n = 6;

Ma trận kề:

Daxet[] = 0;

Truoc[]= 0;

* Bước 2: (Bước lặp)

dauQ= 1;

cuoiQ = 1;

QUEUE[1]=2;

Daxet[2]=1;

// bắt đầu vòng lặp while

* B1: u=QUEUE[1]=2;

dauQ=2;

a[2][1]=…=a[2][5]=0

a[2][6]= 1 và Daxet[6]=0

cuoiQ= 2;

QUEUE[2]= 6;

Daxet[6]=1;

Truoc[6]=2;

* B2: u=QUEUE[2]=6;

dauQ=3;

a[6][1]= 1 và Daxet[1]=0

cuoiQ= 3;

QUEUE[3]= 1;

Daxet[1]=1;

Truoc[1]=6;

a[6][4]= 1 và Daxet[4]=0

cuoiQ= 4;

QUEUE[4]= 4;

Daxet[4]=1;

Truoc[4]=6;

* B3: u=QUEUE[3]=1;

dauQ=4;

a[1][3]= 1 và Daxet[3]=0

cuoiQ= 5;

QUEUE[5]= 3;

Daxet[3]=1;

Truoc[3]=1;

a[1][5]= 1 và Daxet[5]=0

cuoiQ= 6;

QUEUE[6]= 5;

Daxet[5]=1;

Truoc[5]=1;

* B4: u=QUEUE[4]=4;

dauQ=5; // đã duyệt hết các đỉnh

* B5: u=QUEUE[5]=3;

dauQ=6;

* B6: u=QUEUE[6]=5;

dauQ=7; // dauQ> cuoiQ: kết thúc vòng lặp

* Vậy đường đi từ đỉnh 2 đến đỉnh 5 là :

2 🡪 6 🡪 1 🡪 5

* Tóm tắt :

= {2};

= {6};

= {1, 4};

= {3, 5};

***Bộ dữ liệu số 3: tìm đường đi từ đỉnh 2 đến đỉnh 7***

2

1

6

3

7

4

5

Các bước của thuật toán:

* Bước 1: Khởi tạo

n = 7;

Ma trận kề:

Daxet[] = 0;

Truoc[]= 0;

* Bước 2: (Bước lặp)

dauQ= 1;

cuoiQ = 1;

QUEUE[1]=2;

Daxet[2]=1;

// bắt đầu vòng lặp while

* B1: u=QUEUE[1]=2;

dauQ=2;

a[2][1]= 1 và Daxet[1]=0

cuoiQ= 2;

QUEUE[2]= 1;

Daxet[1]=1;

Truoc[1]=2;

a[2][2]=…=a[2][7]=0

* B2: u=QUEUE[2]=1;

dauQ=3;

a[1][3]= 1 và Daxet[3]=0

cuoiQ= 3;

QUEUE[3]= 3;

Daxet[3]=1;

Truoc[3]=1;

* B3: u=QUEUE[3]=3;

dauQ=4;

a[3][4]= 1 và Daxet[4]=0

cuoiQ= 4;

QUEUE[4]= 4;

Daxet[4]=1;

Truoc[4]=3;

a[3][5]= 1 và Daxet[5]=0

cuoiQ= 5;

QUEUE[5]= 5;

Daxet[5]=1;

Truoc[5]=3;

* B4: u=QUEUE[4]=4;

dauQ=5;

a[4][1]= 1 và Daxet[1]=0

a[4][2]= 1 và Daxet[2]=0

* B5: u=QUEUE[5]=5;

dauQ=6;

a[5][6]= 1 và Daxet[6]=0

cuoiQ= 6;

QUEUE[6]= 5;

Daxet[6]=1;

Truoc[6]=5;

* B6: u=QUEUE[6]=6;

dauQ=7;

a[6][7]= 1 và Daxet[7]=0

cuoiQ= 7;

QUEUE[7]= 7;

Daxet[7]=1;

Truoc[7]=6;

* B7: u=QUEUE[7]=7;

dauQ=8;

// đã xét hết các đỉnh

// dauQ > cuoiQ: kết thúc vòng lặp while

* Vậy đường đi từ đỉnh 2 đến đỉnh 7 là:

2 🡪 1 🡪 3 🡪 5 🡪 6 🡪 7

* Tóm tắt :

= {2};

= {1};

= {3};

= {4, 5};

= {6};

= {7};

***Bộ dữ liệu số 4: tìm đường đi từ đỉnh 3 đến 8***

1

5

7

2

8

4

3

6

Các bước của thuật toán:

* Bước 1: Khởi tạo

n = 8;

Ma trận kề:

Daxet[] = 0;

Truoc[]= 0;

* Bước 2: (Bước lặp)

dauQ= 1;

cuoiQ = 1;

QUEUE[1]=3;

Daxet[3]=1;

// bắt đầu vòng lặp while

* B1: u=QUEUE[1]=3;

dauQ=2;

a[3][1]= 1 và Daxet[1]=0

cuoiQ= 2;

QUEUE[2]= 1;

Daxet[1]=1;

Truoc[1]=3;

a[3][4]= 1 và Daxet[4]=0

cuoiQ= 3;

QUEUE[3]= 4;

Daxet[4]=1;

Truoc[4]=3;

a[3][6]= 1 và Daxet[6]=0

cuoiQ= 4;

QUEUE[4]= 6;

Daxet[6]=1;

Truoc[6]=3;

* B2: u=QUEUE[2]=1;

dauQ=3;

a[1][2]= 1 và Daxet[2]=0

cuoiQ= 5;

QUEUE[5]= 2;

Daxet[2]=1;

Truoc[2]=1;

* B3: u=QUEUE[3]=4;

dauQ=4;

a[4][5]= 1 và Daxet[5]=0

cuoiQ= 6;

QUEUE[6]= 5;

Daxet[5]=1;

Truoc[5]=4;

* B4: u=QUEUE[4]=6;

dauQ=5;

a[6][7]= 1 và Daxet[7]=0

cuoiQ= 7;

QUEUE[7]= 7;

Daxet[7]=1;

Truoc[7]=6;

* B5: u=QUEUE[5]=2;

dauQ=6;

// 2 chỉ kề với 3, 4 đã xét

* B6: u=QUEUE[6]=5;

dauQ=7; // 5 không kề với đỉnh nào

* B7: u=QUEUE[7]=7;

dauQ=8;

a[7][8]= 1 và Daxet[8]=0

cuoiQ= 8;

QUEUE[8]= 8;

Daxet[8]=1;

Truoc[8]=7;

* B8: u=QUEUE[8]=8;

dauQ=9;

// kết thúc vòng lặp while

* Vậy đường đi từ đỉnh 3 đến đỉnh 8 là:

3 🡪 6 🡪 7 🡪 8

* Tóm tắt:

= {3};

= {1, 4, 6};

= {2, 5};

= {7};

= {8};

***Bộ dữ liệu số 5: tìm đường đi từ đỉnh 2 đến đỉnh 10***

7

6

1

8

10

5

2

9

4

3

Các bước của thuật toán:

* Bước 1: Khởi tạo

n = 10;

Ma trận kề:

Daxet[] = 0;

Truoc[]= 0;

* Bước 2: (Bước lặp)

dauQ= 1;

cuoiQ = 1;

QUEUE[1]=2;

Daxet[3]=1;

* B1: u=QUEUE[1]=2;

dauQ=2;

a[2][3]= 1 và Daxet[3]=0

cuoiQ= 2;

QUEUE[2]= 3;

Daxet[3]=1;

Truoc[3]=2;

a[2][5]= 1 và Daxet[5]=0

cuoiQ= 3;

QUEUE[3]= 5;

Daxet[5]=1;

Truoc[5]=3;

* B2: u=QUEUE[2]=3;

dauQ=3;

a[3][4]= 1 và Daxet[4]=0

cuoiQ= 4;

QUEUE[4]= 4;

Daxet[4]=1;

Truoc[4]=3;

* B3: u=QUEUE[3]=5;

dauQ=4;

a[5][1]= 1 và Daxet[1]=0

cuoiQ= 5;

QUEUE[5]= 1;

Daxet[1]=1;

Truoc[1]=5;

a[5][9]= 1 và Daxet[9]=0

cuoiQ= 6;

QUEUE[6]= 9;

Daxet[9]=1;

Truoc[9]=5;

* B4: u=QUEUE[4]=4;

dauQ = 5;

* B5: u= QUEUE[5]=1;

dauQ=6;

a[1][6]= 1 và Daxet[6]=0

cuoiQ= 7;

QUEUE[7]= 6;

Daxet[6]=1;

Truoc[6]=1;

* B6: u= QUEUE[6]=9;

dauQ=7;

a[9][8]= 1 và Daxet[8]=0

cuoiQ= 8;

QUEUE[8]= 8;

Daxet[8]=1;

Truoc[8]=9;

* B7: u= QUEUE[7]=6;

dauQ=8;

a[6][7]= 1 và Daxet[7]=0

cuoiQ= 9;

QUEUE[9]= 7;

Daxet[7]=1;

Truoc[7]=6;

* B8: u= QUEUE[8]=8;

dauQ=9;

a[8][10]= 1 và Daxet[10]=0

cuoiQ= 10;

QUEUE[10]= 10;

Daxet[10]=1;

Truoc[10]=8;

* B9: u= QUEUE[9]=7;

dauQ=10;

* B10: u= QUEUE[10]=10;

dauQ=11; // kết thúc vòng lặp.

* Vậy đường đi cần tìm là:

2 🡪 5 🡪 9 🡪 8🡪 10

* Tóm tắt:

= {2};

= {3, 5};

= {4, 1, 9};

= {6, 8};

= {7, 10};

1. **Đánh giá độ phức tạp của thuật toán theo lý thuyết và bằng thực nghiệm, so sánh kết quả**

*Độ phức tạp của thuật toán theo lí thuyết:*

* ở vòng lặp for thứ nhất(khởi tạo ma trận kề): có (n)= phép gán
* ở vòng lặp for thứ hai(khởi tạo): có (n)= 2n phép gán
* vòng lặp while:

Ta thấy, thuật toán có không quá n bước lặp, mỗi bước lặp vòng while có tối đa n bước lặp vòng for.

(n) = n \* (2 + n\*(2+4)) = 6 + 2n

* Ta có: f(n) = (n) + (n) +(n)

= + 2n + 6 + 2n = 7+ 4n

Vậy độ phức tạp của thuật toán theo lý thuyết là O()

*Độ phức tạp của thuật toán theo thực tế.*