Nhóm 4

Thành viên: - Vũ Xuân Trọng

- Nguyễn Đình Giang

- Bùi Huy Hoàng

- Phạm Tú Anh

- Nguyễn Việt Hoàng

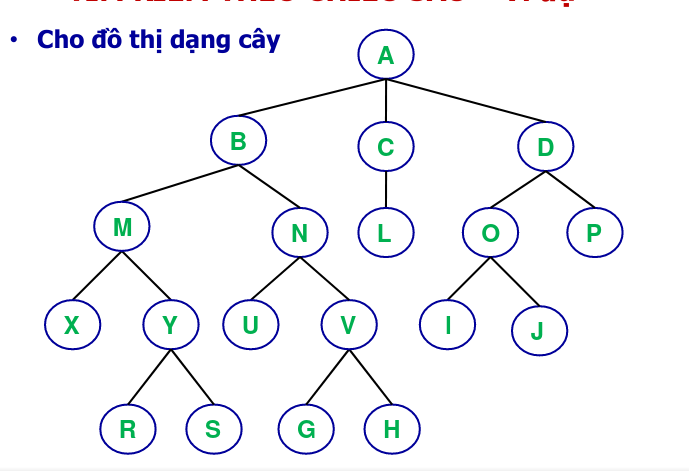
- Đỗ Kế Đạt

- Vũ Ngọc Hải

- Nguyễn Hoàng Hiệp

Chủ đề: ***Tìm hiểu giải thuật và viết chương trình bằng ngôn ngữ Python cho các thuật toán BFS, DFS, IDS, HC***

1, Thuật toán tìm kiếm theo chiều rộng (Breadth First Search)



* Cách thực hiện:

+Từ đỉnh xuất phát duyệt tất cả các đỉnh kề.

+Làm tương tự với các đỉnh vừa được duyệt.

+Quá trình duyệt kết thúc khi tìm thấy đỉnh TG hoặc đã hết đỉnh để duyệt.

* Lưu trữ: Sử dụng hai danh sách DONG và MO hoạt động theo kiểu FIFO (hàng đợi)
* DONG: Chứa các đỉnh đã xét
* MO: Chứa các đỉnh đang xét
* B(n)= {m | (n,m) ∈ E} //tập các đỉnh kề với n
* Thuật toán:

void BFS() {

MO = MO U {To}

while (MO != Ø) {

n = get (MO) *//lấy đỉnh đầu danh sách MO*

if (n==TG)

return true;

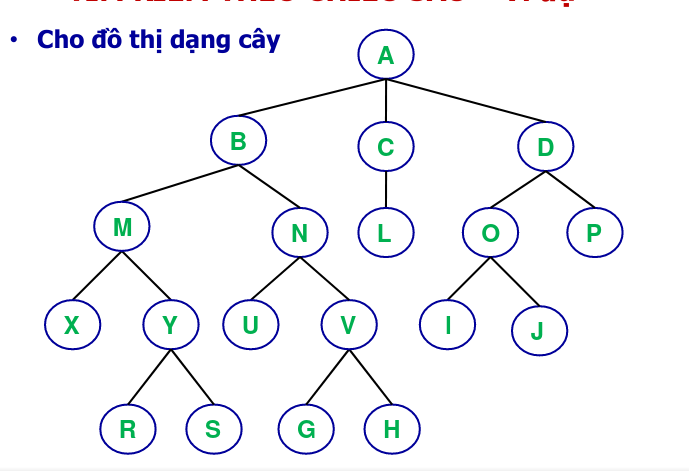
MO = MO U B(n)*//cho B(n)vào cuối DSMO*

DONG = DONG U {n}

}

}

2.Thuật toán tìm kiếm theo chiều sâu (DFS):



* Cách thực hiện :
* Từ đỉnh xuất phát duyệt một đỉnh kề .
* Các đỉnh của đồ thị được duyệt theo các nhánh đến nút lá.
* Nếu chưa tìm thấy đỉnh T(G) thì quay lui tới một đỉnh nào đó để sang nhánh khác .
* Việc tìm kiếm kết thúc khi tìm thấy đỉnh T(G) hoặc hết các đỉnh.
* Lưu trữ: Sử dụng hai danh sách DONG và MO trong đó:

+ DONG: Chứa các đỉnh đã xét, hoạt động theo kiểu FIFO (hàng đợi).

+ MO: chứa các đỉnh đang xét , hoạt động theo kiểu LIFO (ngăn xếp).

+ B(n)={m|(n,m) ∈ E} //Tập các đỉnh kề với n

* Thuật toán:

void DFS()

{ MO = MO hợp {T0 }

while (MO != ) {

n = get(MO) // lấy đỉnh đầu danh sach MO

if (n==TG)

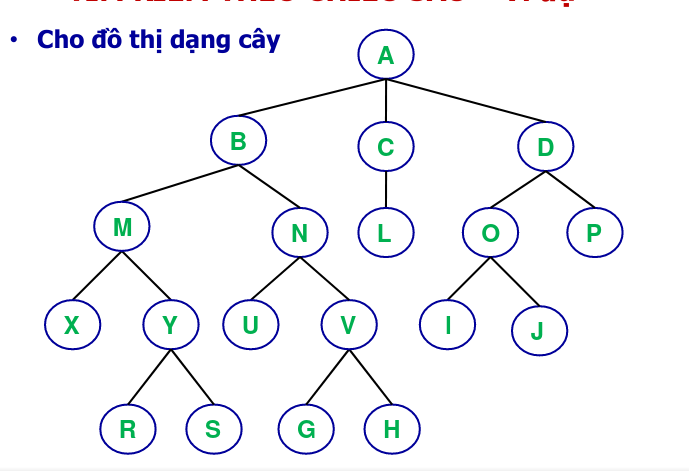
return TRUE MO = MO hợp B(n) /\*cho B(n) vào đầu DS MO \*/

DONG = DONG hợp {n}

}

}

3. Thuật toán tìm kiếm sâu dần (IDS):

* Trong trường hợp đồ thị tồn tại đường đi p từ T0 đến TG ∈ Goal có thể DFS không dừng vì đi theo nhánh vô tận 
* Để khắc phục, người ta đưa vào thủ tục tìm kiếm theo chiều sâu một đại lượng giới hạn độ sâu
* Ký hiệu: d(n) là độ sâu hiện tại của đỉnh n

d(0) =0

d(m) = d(n) + 1 nếu m ∈ B(n)

* Trong trường hợp này thuật toán có thể dừng nhưng có thể cho kết quả không mong muốn.
* Vì đỉnh đích TG nằm ở dưới độ sâu nên để khắc phục nhược điểm này người ta tăng dần độ sâu
* Phương pháp này sử dụng kỹ thuật này gọi là tìm kiếm sâu dần (Iterative Deepening Search)
* Thuật toán:
  + Vào:
    - Đồ thị G=<V, E>
    - Đỉnh đầu T0 và Goal chứa các đỉnh đích
    - Độ sâu ds = k giới hạn độ sâu
  + Ra:
    - Đường đi P: T0 -> TG ∈ Goal
  + Phương pháp:
    - Sử dụng 2 danh sách DONG và MO
    - DONG hoạt động theo nguyên tắc FIFO.
    - MO vừa hoạt động theo nguyên tắc FIFO vừa hoạt động theo nguyên tắc LIFO.

void IDS() {

MO = {T0}; ds = k;

while (MO != 𝜙) {

n = get(MO);

if ( n==TG ) return TRUE;

DONG = DONG ◡ {n};

switch (d(n)){

0...ds-1: Đặt B(n) vào đầu MO

ds: Đặt B(n) vào cuối MO

ds+1: ds = ds + k;

if (k==1) Đặt B(n) vào cuối MO

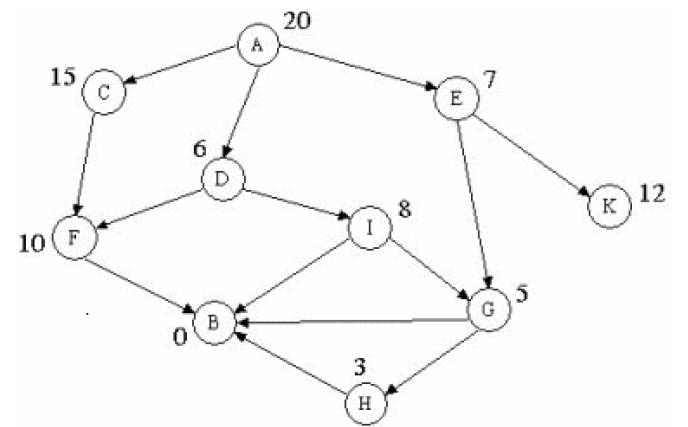
else Đặt B(n) vào đầu MO

}

}

}

4. Thuật toán tìm kiếm Leo đồi (Hill-climbing search)



* Cách thực hiện:
* Bước 1: Ti = T0 ;*(nút khởi đầu*
* Bước 2: Nếu Ti = TG kết thúc *(Thành công)*
* Bước 3: Triển khai Ti ;

.Tính hàm h(Tk ), với Tk là TT kế tiếp của Ti

.Chọn Tk tương ứng với h(Tk )nhỏ nhất và gọi là Tkmax

* Bước 4: Nếu Tkmax không tốt hơn Ti thì dừng *(Không thành công)*
* Bước 5: Ti =Tkmax
* Bước 6: Lặp lại bước 2

* Giải thuật:

void HCS()

{

Ti = T0 ;

Stop = False;

while (Stop==False)

{

if (Ti ==TG ){

<Tìm được kết quả>;

Stop==True;

}

else{

Best==h(Ti );

Tkmax = Ti ;

}

while(∃ TT kế tiếp hợp lệ của Ti )

{

Tk = <1 TT kế tiếp hợp lệ của Ti >;

if (h(Tk ) tốt hơn Best)

{

Best = h(Tk ); Tkmax =Tk ;

}

}

if (Best tốt hơn h(Ti ))

Ti =Tkmax ;

else{

<không có lời giải>;

Stop = True;

}

}