CHƯƠNG 2
Không gian trạng thái và Phương pháp tìm kiếm

TS. Nguyễn Đình Hiển

Nội dung

- ***Vấn đề, giải quyết vấn đề***
- ***Khái niệm về thuật toán, thuật giải***
- ***Các nguyên lý của thuật giải heuristic***
- ***Các chiến lược tìm kiếm và Thuật giải A****

Vấn đề?

Những vướng mắc khó khăn cần giải quyết

Một yêu cầu tìm kiếm xử lý trong một ngữ cảnh cụ thể

Bao gồm:

- các sự kiện
- các thông tin
- những ràng buộc nhất định.

vấn đề = bài toán

Mô hình vấn đề

A ➔ B

A: giả thiết, điều kiện ban đầu

B: kết luận cần đạt đến

➔: suy luận hay giải pháp cần xác định = một số hữu hạn bước

Phân loại vấn đề

- Xác định rõ
  - A, B đều rõ
- Chưa rõ
  - A rõ, B chưa rõ
  - A chưa rõ, B rõ
  - A, B đều chưa rõ

Thuật toán

- Thuật toán là giải pháp viết dưới dạng thủ tục và thỏa 3 tiêu chuẩn

  Xác định : không mập mờ và có thể thực thi được

  Hữu hạn

  Đúng

➔ Thuật toán là một dãy hữu hạn các bước không mập mờ và có thể thực thi được, quá trình hành động theo các bước này phải dừng và cho kết quả mong muốn.

Thuật toán

- Tính tổng các số nguyên dương lẻ từ 1→n
  - B1: S=0;
  - B2: i=1
  - B3: Nếu **i>n** thì sang bước 7, ngược lại sang bước 4
  - B4: S=S+i
  - B5: i=i+2
  - B6: Quay lại 3
  - B7: Tổng cần tìm là S

Thuật toán

Thuật toán có thể được thể hiện qua:

- Ngôn ngữ tự nhiên
- Lưu đồ
- Mã giả
- NN lập trình

Ngoài ra thuật toán còn phải đạt hiệu quả cao hay có độ phức tạp thấp

Thuật toán

$O(\log_2 n)$
$O(n)$
$O(n\log_2 n)$
$O(n^2)$
$O(n^k)$

} polyminimal complexity $\rightarrow$ acceptable

$O(2^n)$
$n!$

} high (Not plolyminimal) complexity $\rightarrow$ unacceptable

Một số ví dụ về bài toán có độ phức tạp cao

- Bài toán phân công công việc

  Một đề án gồm n công việc và các việc sẽ được thực hiên bởi m máy như nhau.

  Giả sử biết thời gian để 1 máy thực hiện viêc thứ j là tj

  Yêu cầu: Tìm phương án phân công sao cho thời gian hoàn thành toàn bộ công việc là thấp nhất.

  Mẫu số liệu

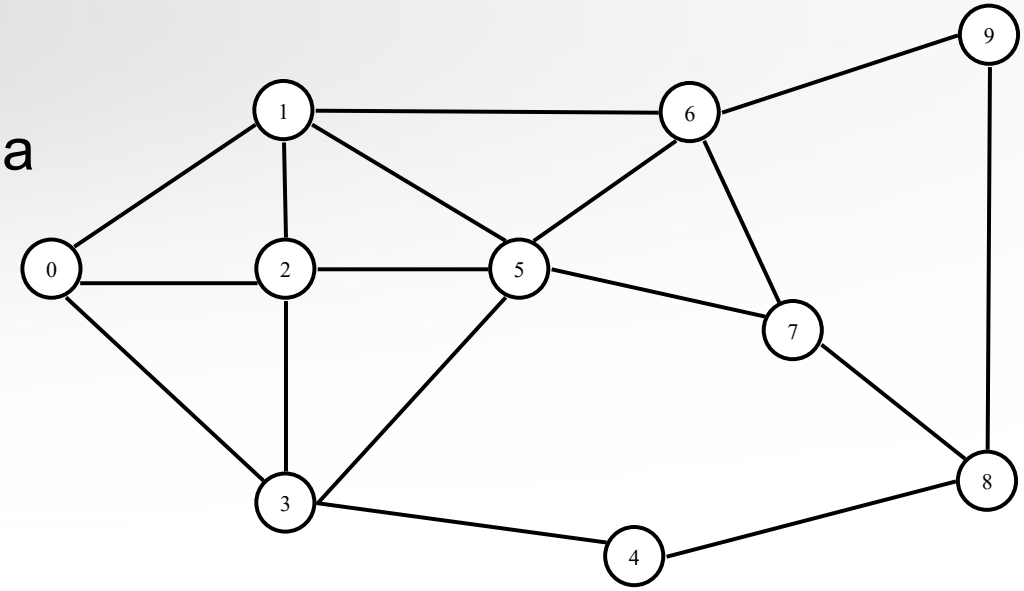
  n=10, m=3

  tj = 4 9 5 2 7 6 10 8 7 5

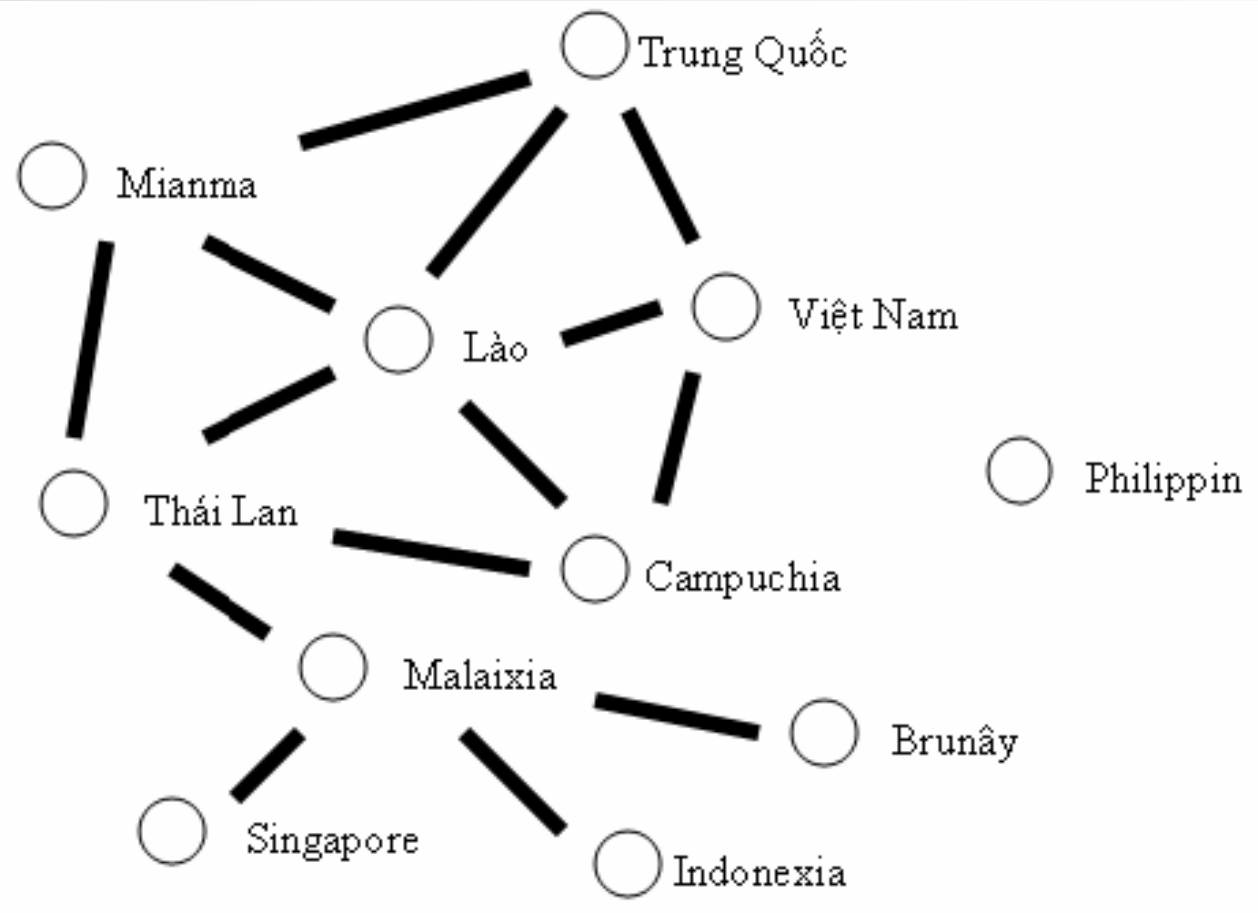
Một số ví dụ về bài toán có độ phức tạp cao

- Bài toán tô màu

  Giả sử ta có bản đồ các quốc gia trên thế giới, ta muốn tô màu các quốc gia này sao cho các nước khác nhau được tô khác màu.



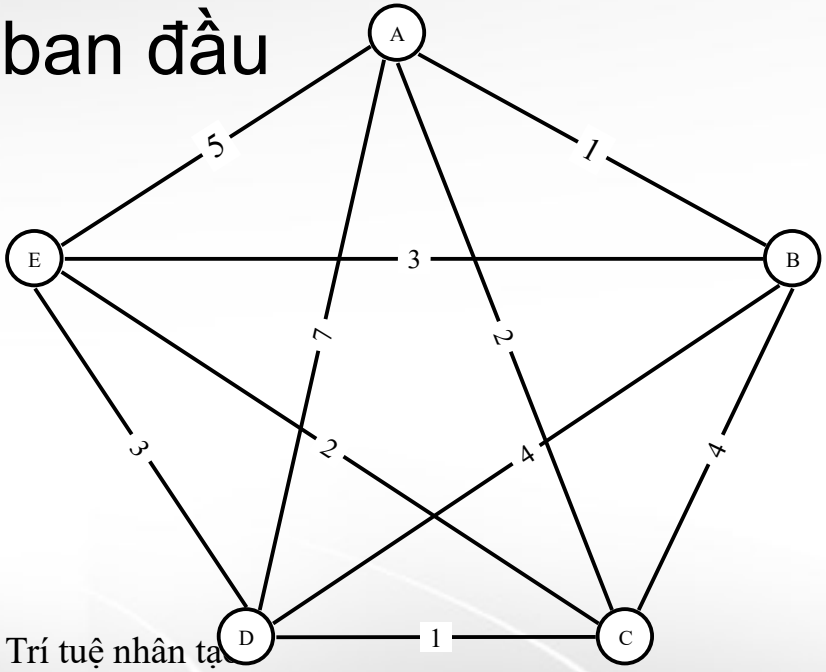
Yêu cầu tìm cách tô sao cho số màu sử dụng là ít nhất.


Trung Quốc
Mianma
Việt Nam
Lào
Philippin
Thái Lan
Campuchia
Malaixia
Brunây
Singapore
Indonexia

Một số ví dụ về bài toán có độ phức tạp cao

Bài toán người đưa thư

- Giả sử có một đồ thị có trọng số dương, tìm đường đi ngắn nhất qua tất cả các đỉnh của đồ thị rồi trở về đỉnh ban đầu



Thuật giải

- Thuật giải: giải pháp được viết dưới dạng thủ tục tương tự như thuật toán nhưng không đòi hỏi các tiêu chuẩn như thuật toán.

  Tính đúng: chấp nhận các thuật giải đơn giản có thể cho kết quả đúng hay gần đúng nhưng có khả năng thành công cao hơn.

Thuật giải heuristic

Để có thể được chấp nhận thuật giải phải thể hiện một giải pháp hợp lý nhất có thể trong tình huống hiện tại bằng cách:

Tận dụng mọi thông tin hữu ích

Sử dụng tri thức, kinh nghiệm trực giác của con người

Tự nhiên đơn giản nhưng cho kết quả chấp nhận được

➔ Thuật giải Heuristic

Thuật giải heuristic

Đặc tính

- Thường tìm được lời giải tốt, mặc dù không phải là tốt nhất
- Thực hiện dễ dàng nhanh chóng so với thuật giải tối ưu
- Khá tự nhiên gần gũi với cách giải của con người

Các nguyên lý của thuật giải heuristic

- **Vét cạn thông minh**
- **Nguyên lý thứ tự**
- **Nguyên lý tham lam**
- **Hàm heuristic**

Các nguyên lý của thuật giải heuristic

Vét cạn thông minh

Hạn chế vùng không gian tìm kiếm và có sự định hướng để nhanh chóng tìm đến mục tiêu.

Tạo miền D’ rất nhỏ so với D

Vét cạn trên D’

Các nguyên lý của thuật giải heuristic

- **Nguyên lý thứ tự**

  Trong quá trình hành đông để thực hiện việc chọn lọc các cách làm các trạng thái ta có thể dựa trên một thứ tự hợp lý để giải pháp đạt tính hiệu quả cao

Các nguyên lý của thuật giải heuristic

- **Nguyên lý tham lam**

  Trong nhiều vấn đề cần phải đạt đến một 1 mục tiêu tối ưu toàn cục mà không nhìn thấy được toàn bộ quá trình hành động, hơn nữa trong từng bước ta phải lựa chọn hành động dựa trên những thông tin cục bộ. Khi đó trong từng bước của quá trình hành động người ta dựa trên mục tiêu tối ưu toàn cục để định ra mục tiêu cục bộ và dựa theo đó chọn lựa hành động

Các nguyên lý của thuật giải heuristic

- **Hàm heuristic**

  Là hàm ứng với mỗi trạng thái hay mỗi sự chọn lựa một giá trị có ý nghĩa đối với vấn đề để dựa vào giá trị hàm này ta chọn lựa hành động.

Ví dụ 1

- Giải xấp xỉ nghiệm của phương trình f(x)=0 lên tục trên [a,b] với f(a)f(b)<0.

- Với điều kiện f(a)f(b)<0 và f liên tục trên a,b ta biết f(x) có nghiệm thuộc [a,b].

  phương pháp vét cạn như sau:

  - Gọi c là trung điểm của [a,b]
  - Nếu f(a)f(c)<0 thì b=c, ngược lại a=c

Ví dụ 2

Thuật giải cho bài toán phân công đơn giản

Chọn việc J chưa phân công có thời gian thực hiện cao nhất phân công cho máy có thời gian làm việc thấp nhất

for(k=0;k<n;k++)

{

Chọn việc J chưa phân công có thời gian thực hiện cao nhất.

Chọn máy M có thời gian làm việc thấp nhất

Bố trí việc J cho máy M.

}

Ví dụ 2

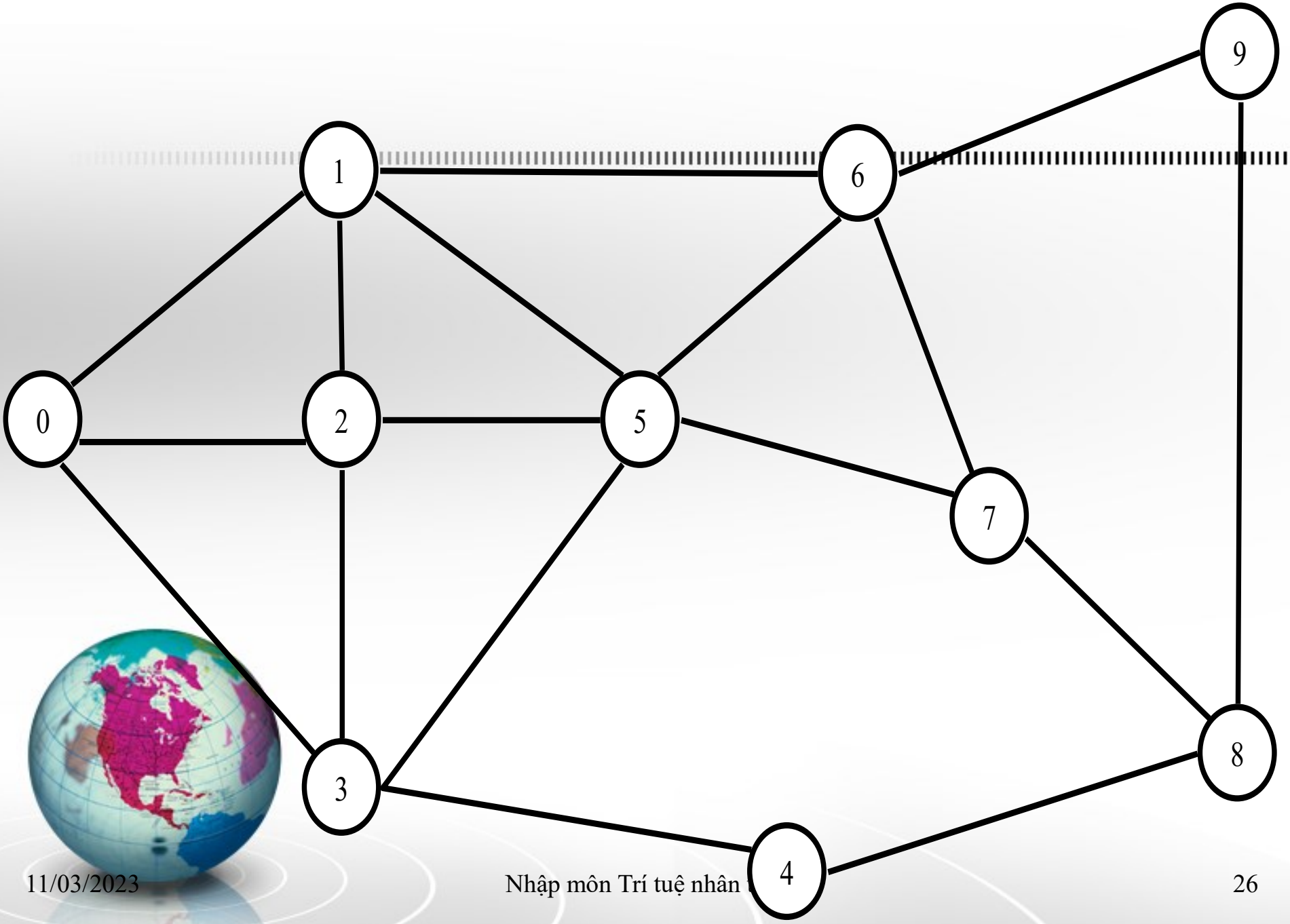
n=10, m=3

tj = 4 9 5 2 7 6 10 8 7 5

2 máy, 5 công việc (3,3,2,2,2)

Ví dụ 3

- Bài toán tô màu
  - QT1: Chọn đỉnh có số đỉnh chưa tô ở cạnh nó là lớn nhất (bậc lớn nhất)
  - QT2: Chọn màu:Với một đỉnh N đang xét, trước tiên thử tô bằng những màu đã tô, nếu không được thì sử dụng màu mới
  - Sau khi tô màu cho đỉnh N thì ta xóa các cạnh có nối đến N và đánh dấu các đỉnh kế bên không được tô màu vừa tô cho N


9
1
6
0
2
5
7
3
8
4

Ví dụ 3

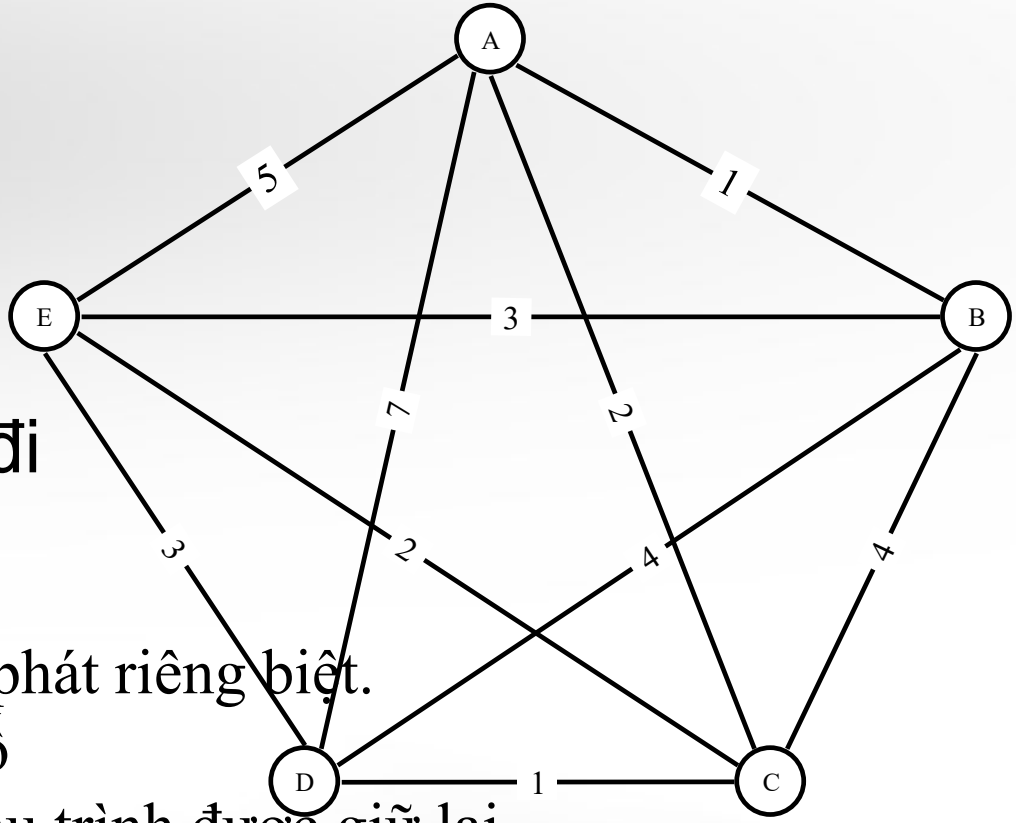
- Có một cuộc hội thảo khoa học với 9 chủ đề khác nhau: A, B, C…
- Các chủ đề sau không được đồng thời; AE, BC, ED, ABD, AHI, BHI, DFI, DHI, FGH.
- Xây dựng lịch sao cho số buổi tổ chức là ít nhất

Ví dụ 4

Bài toán người đưa thư

- thuật giải GST(Greedy Traveling Saleman):

Ở mỗi bước chọn cạnh ngắn nhất tiếp theo để đi

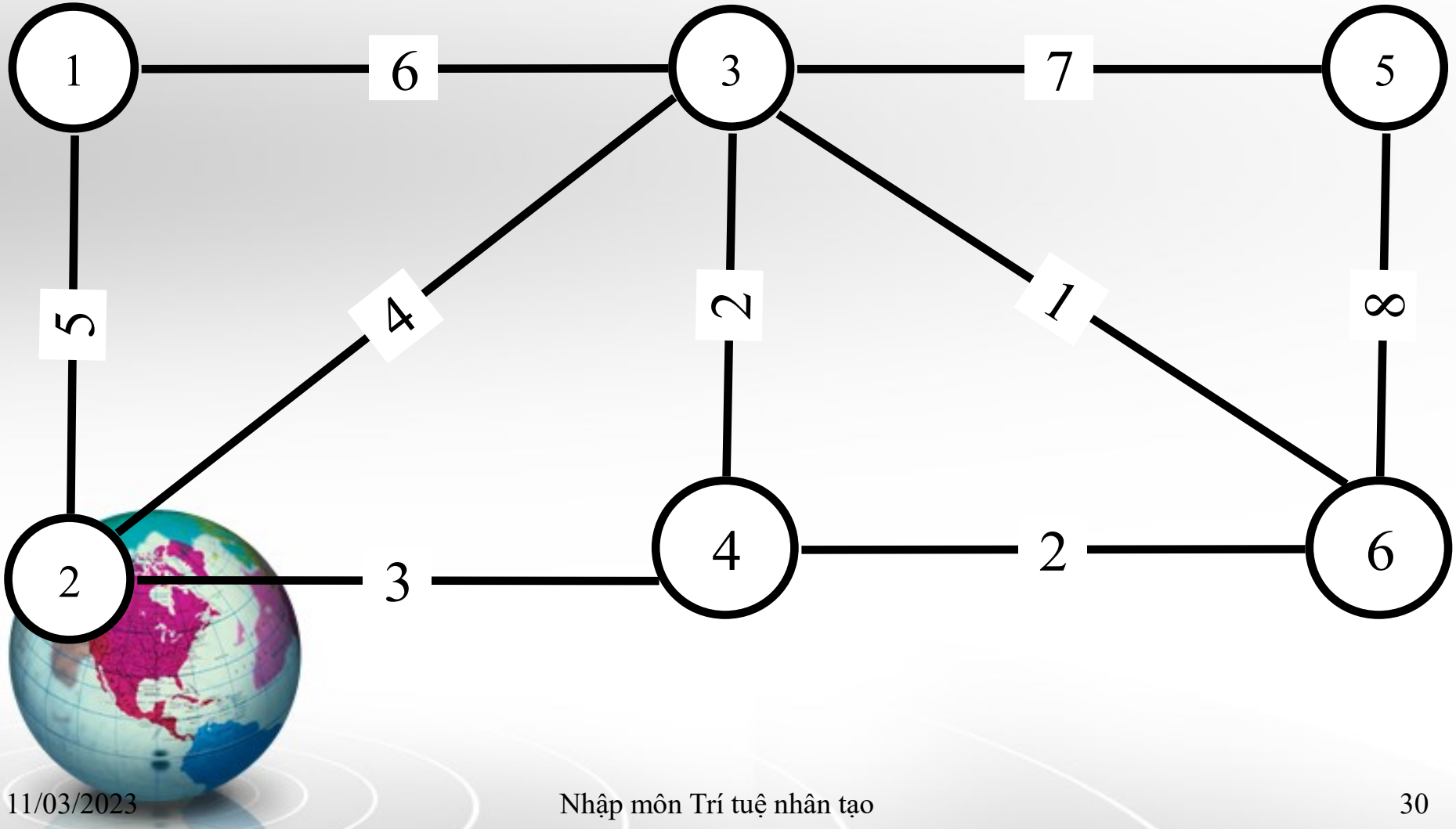


Tạo ra lịch từ p thành phố xuất phát riêng biệt.
Tìm p chu trình qua n thành phố
Chỉ chu trình tốt nhất trong p chu trình được giữ lại.

Ví dụ 5

- Có 1 trạm cấp nước và N điểm dân cư. Hãy xây dựng chương trình thiết kế tuyến đường ống nước cung cấp đến mọi nhà sao cho tổng chiều dài đường ống phải dùng là ít nhất. Giả sử rằng các đường ống chỉ được nối giữa 2 điểm dân cư hoặc giữa trạm cấp nước với điểm dân cư.

Ví dụ 5



Ví dụ 5

- Sắp xếp các cạnh theo thứ tự tăng của trọng số
- for(i=0;i<n;i++)

  lấy cạnh e_j nhỏ nhất thêm vào T sao cho T không có chu trình
- Kết thúc ta có T là cây khung nhỏ nhất

Bài toán tìm kiếm

Vấn đề = Tìm kiếm mục tiêu

Không gian trạng thái

- Không gian trạng thái: tập tất cả các trạng thái **có thể có** của bài toán.

Các ví dụ

2	8	3
1	6	4
7		5

1	2	3
8		4
7	6	5

- Bài toán đổ nước
- Bài toán giải tam giác
- Bài toán người quỷ qua sông

Không gian trạng thái

- Giữa các trạng thái có sự liên hệ gọi là chuyển trạng thái hay toán tử chuyển trạng thái.

Ví dụ

2	8	3
1	6	4
7		5

1	2	3
8		4
7	6	5

- Trạng thái đầu và cuối của bài toán 8 số
- Các toán tử: qua trái, phải, lên trên, xuống dưới

- Biểu diễn trạng thái bằng **(a, b, k)**

  Với a, b là số người và quỷ ở bên bờ phải

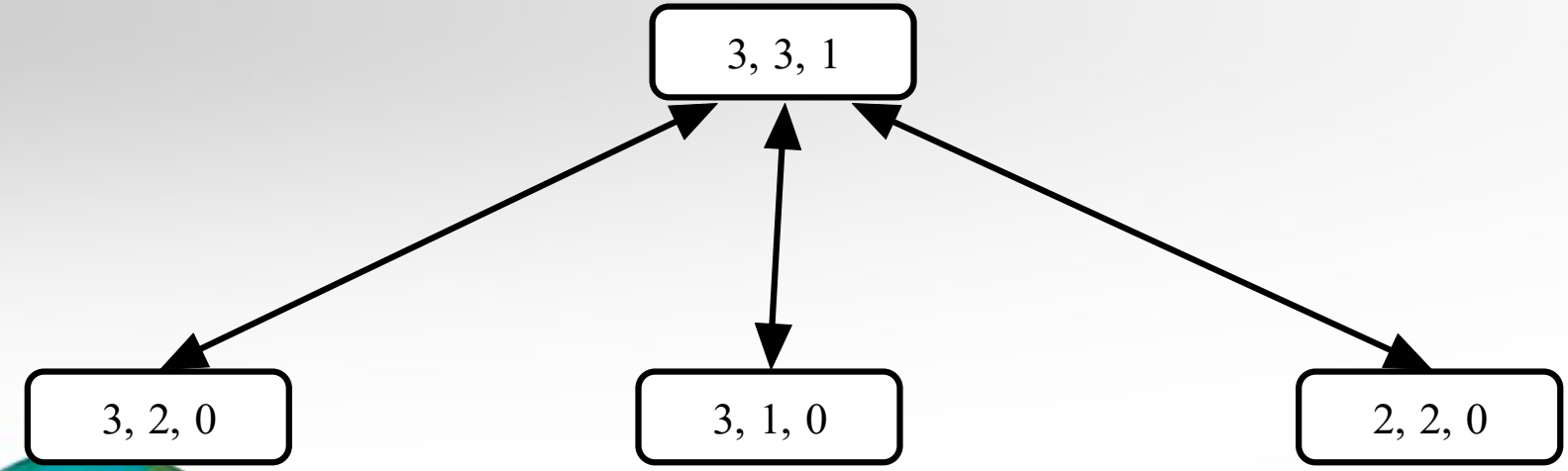
  k=1 là thuyền ở bờ phải, k=0 thuyền ở bờ trái

Không gian trạng thái

  Trạng thái ban đầu (3, 3, 1)

  Các toán tử: chở 1 người, 1 quỷ, 2 người, 2 quỷ, 1người và 1 quỷ

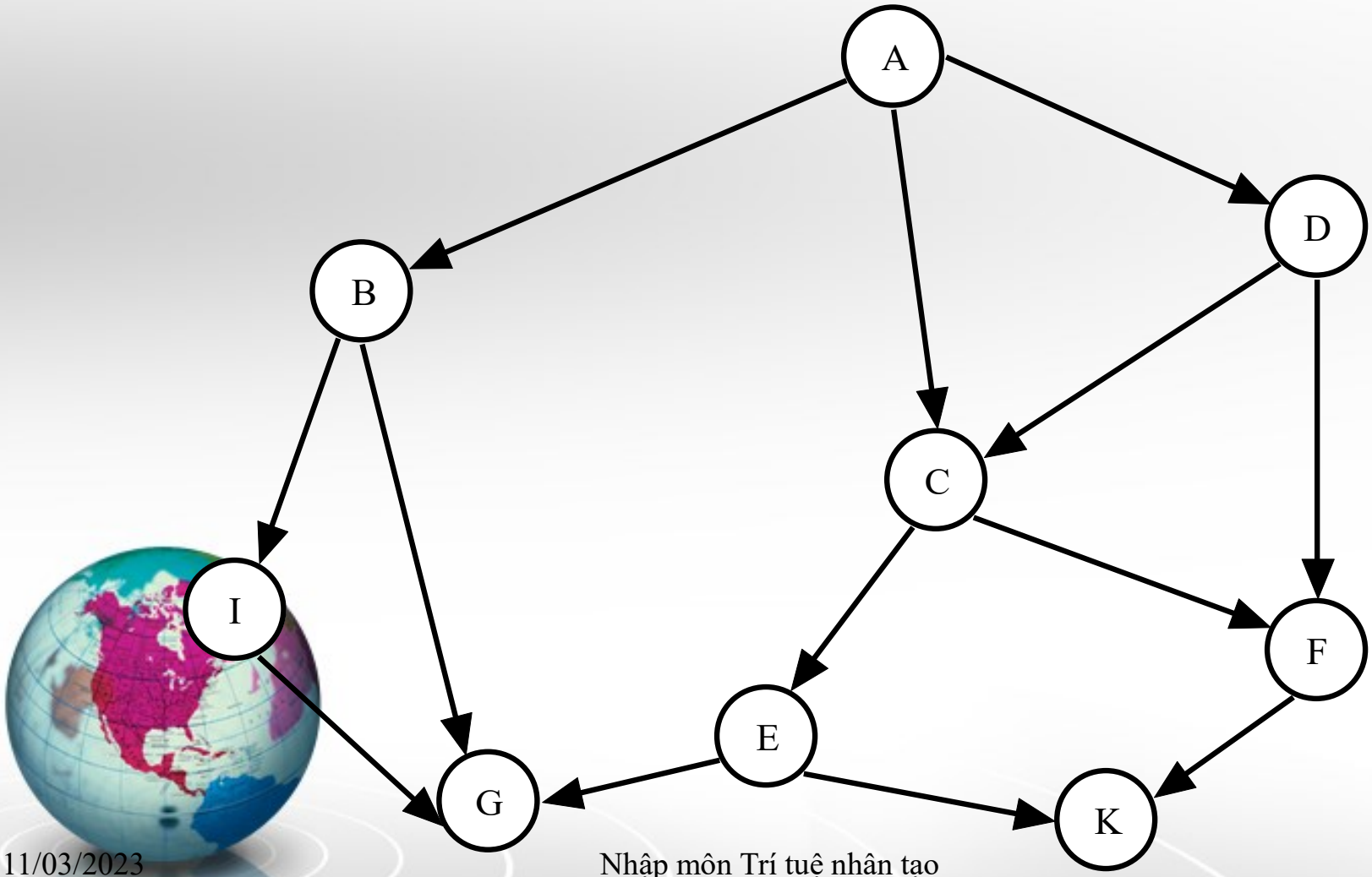
  Trạng thái kết thúc (0, 0, 0)


3, 3, 1
3, 2, 0
3, 1, 0
2, 2, 0

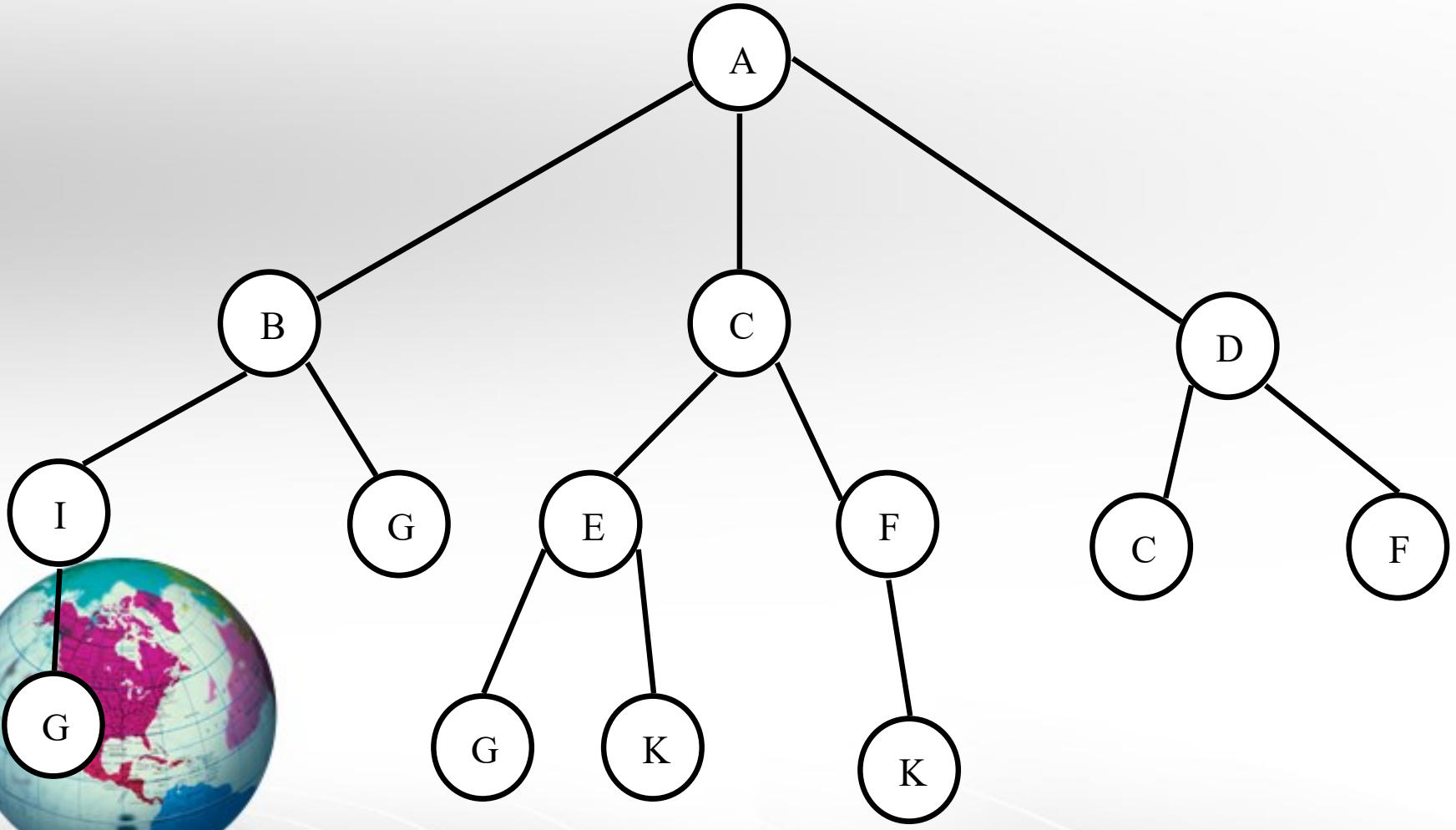
Không gian trạng thái

- Không gian trạng thái này có thể được biểu biễn bằng đồ thị
- Đồ thị trang thái xác định khi:
  - Biết trạng thái đầu
  - Biết hàm next(S)(tập hợp toán tử) trả về các trạng thái kế của S.
  - Biết trạng thái kết thúc (tập trạng thái kết thúc)

Cây tìm kiếm



Cây tìm kiếm



Phát biểu bài toán

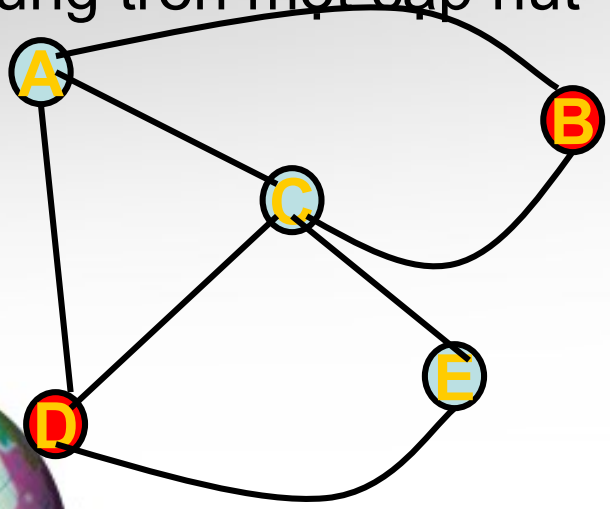
- Trên một đồ thị có trọng số dương ta muốn tìm đường đi ngắn nhất từ một đỉnh xuất phát S0 đến một đỉnh mục tiêu G.
- Giả sử ta có thêm thông tin như sau: tại mỗi đỉnh S ta biết ước tính quãng đường từ S đến mục tiêu là h(S).

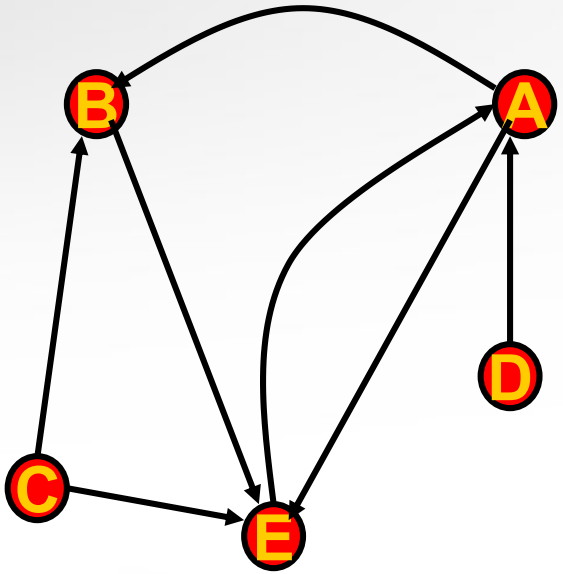
Các vấn đề trong thiết kế các CT tìm kiếm

- Xác định hướng tìm (forward hay backward reasoning).
- Cách lựa chọn luật để áp dụng (matching)
- Cách biểu diễn NODE của quá trình tìm (knowledge representation problem, frame problem).
- Các NODE trong đồ thị có thể được phát sinh nhiều lần, và có thể đã được xem xét trước đó trong quá trình duyệt → cần loại bỏ những NODE lặp lại. → cần lưu lại các NODE đã xét.

Lý thuyết đồ thị - **Review**

- Đồ thị: là một cấu trúc bao gồm:
  - Tập các nút N1, N2,… Nn,.. Không hạn chế
  - Tập các cung nối các cặp nút, có thể có nhiều cung trên một cặp nút





Nút: {A,B,C,D,E}

Cung: {(a,d), (a,b), (a,c), (b,c), (c,d), (c,e), (d,e) }

Đặc tính đồ thị

- **Đồ thị có hướng**: là đồ thị với các cung có định hướng, nghĩa là cặp nút có quan hệ thứ tự trước sau **theo từng cung.** Cung (Ni,Nj) có hướng từ Ni đến Nj, Khi đó Ni là nút cha và Nj là nút con.
- **Nút lá**: là nút không có nút con.
- **Path**: là chuỗi có thứ tự các nút mà 2 nút kế tiếp nhau tồn tại một cung.

Đặc tính đồ thị

- **Đồ thị có gốc**: Trên đồ thị tồn tại nút X sao cho tất cả các path đều đi qua nút đó. X là gốc - Root
- **Vòng** : là một path đi qua nút nhiều hơn một lần
- **Cây**: là graph mà không có path vòng
- **Hai nút nối nhau** :nếu có một path đi qua 2 nút đó

Các chiến lược tìm kiếm

- Tìm kiếm mù
- Tìm kiếm tốt nhất
- Tìm kiếm heuristic

➔ Mục tiêu: tìm ra một solution path
và/hay solution path tốt nhất

Tìm kiếm mù

- Tìm kiếm theo chiều sâu
- Tìm kiếm theo chiều rộng
- Tìm kiếm sâu dần

Tìm kiếm theo chiều sâu

- Tìm kiếm sâu trong không gian bài toán được bắt đầu từ một nút rồi tiếp tục cho đến khi hoặc đến ngõ cụt hoặc đến đích
- Tại mỗi nút có luật trọng tài

Tìm kiếm theo chiều sâu

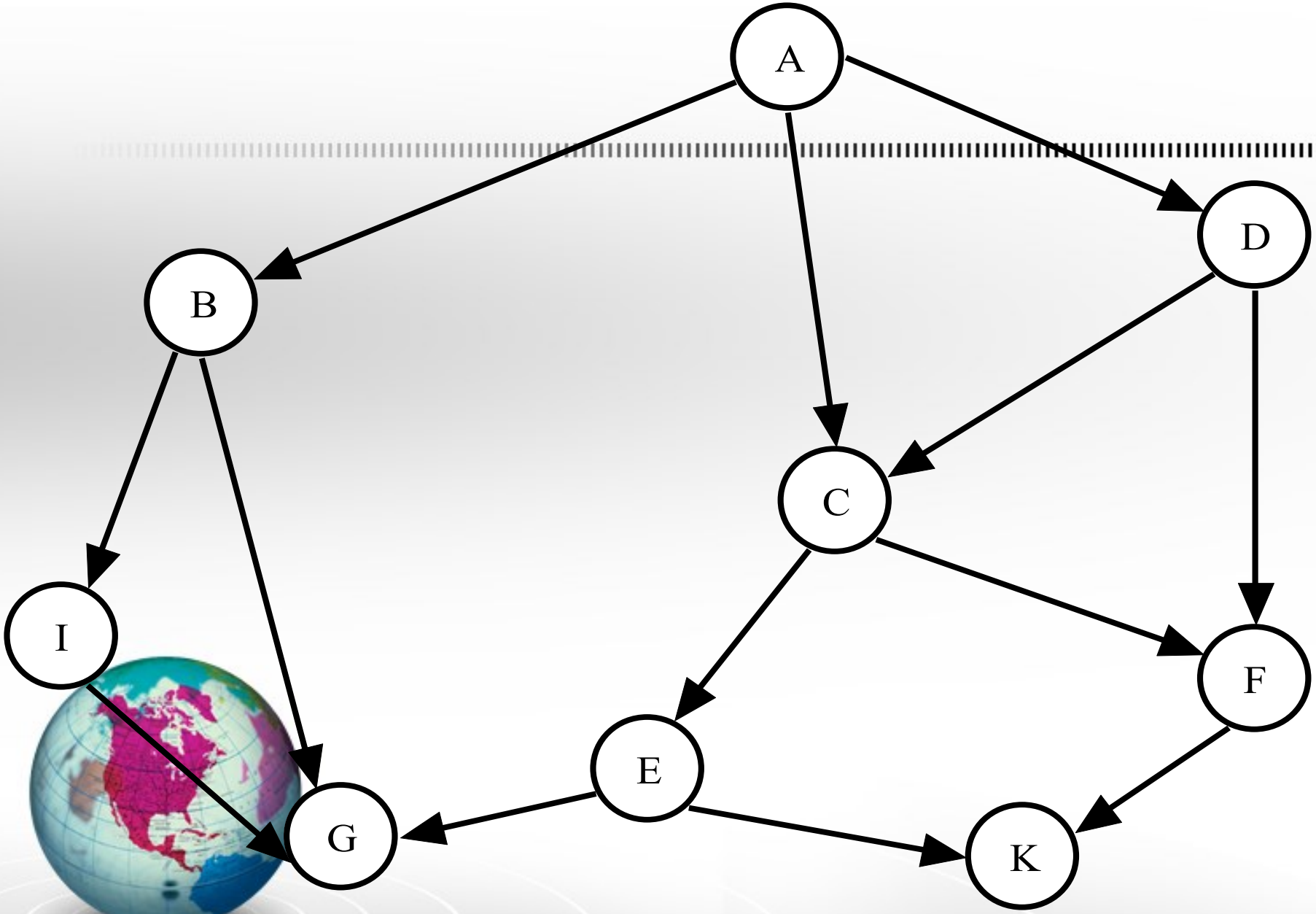
- O = [S]; C={};
- While O khác rỗng
- {

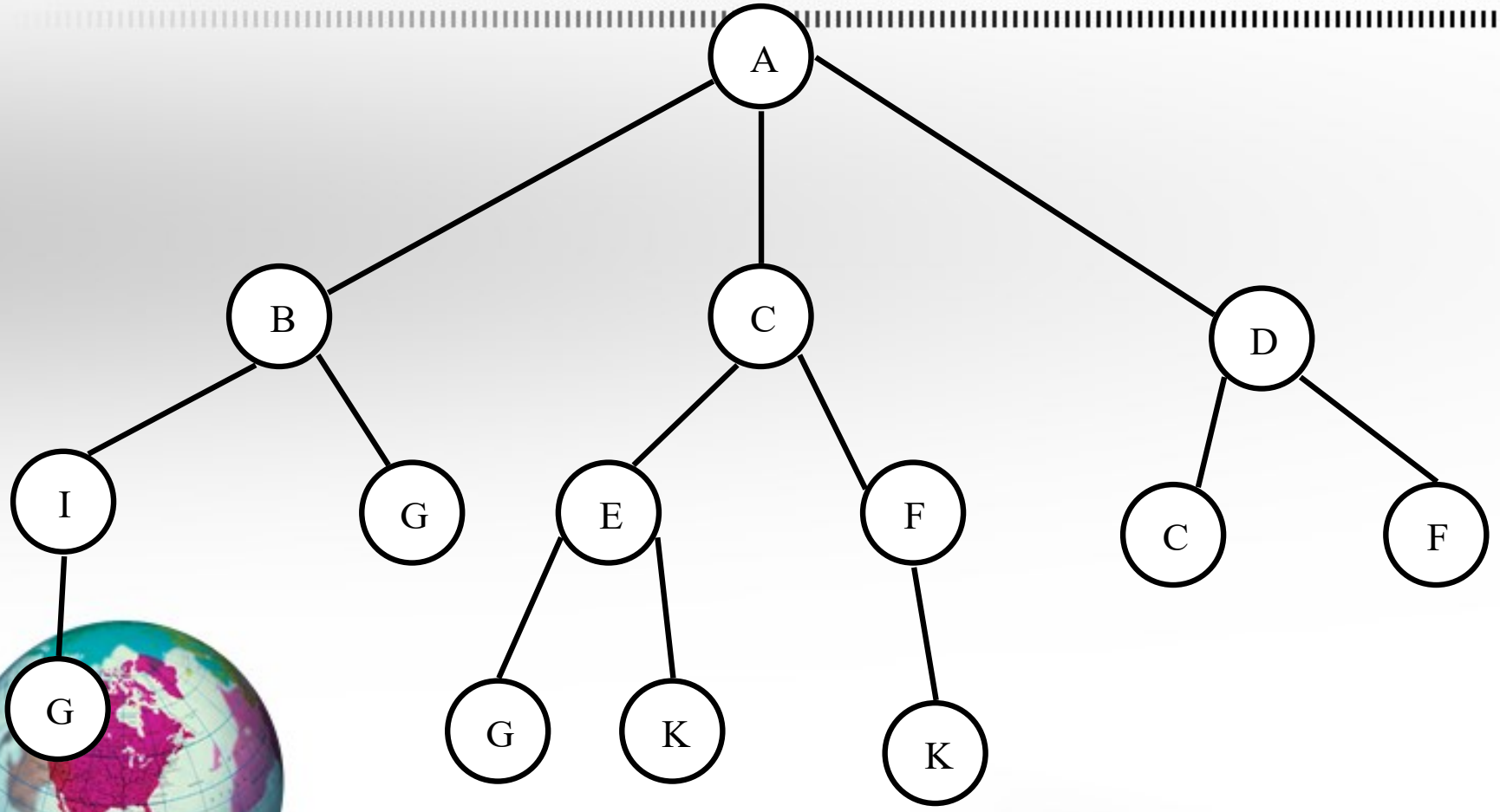
  Lấy p từ đầu O

  Duyệt p

  Bỏ p vào C

  Với mỗi q kề p, q không thuộc C: bỏ q vào đầu O
- }


A
B
D
C
I
F
E
G
K


A
B
C
D
I
G
E
F
C
F
G
G
K
K

Tìm kiếm theo chiều rộng

- Tìm kiếm trên tất cả các nút của một mức trong không gian bài toán trước khi chuyển sang các nút của mức tiếp theo

Tìm kiếm theo chiều rộng

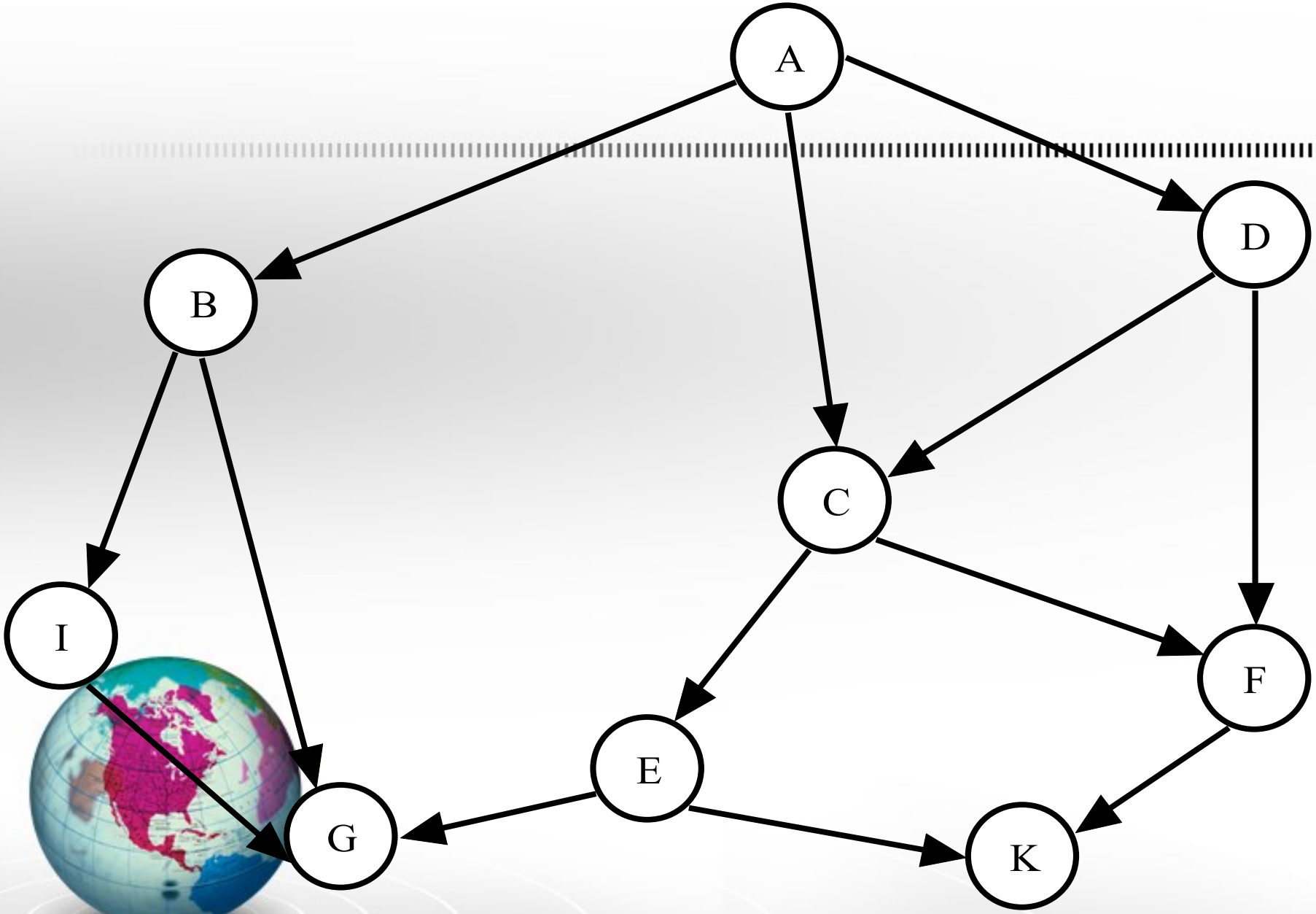
- O = [S]; C={};
- While O khác rỗng
- {

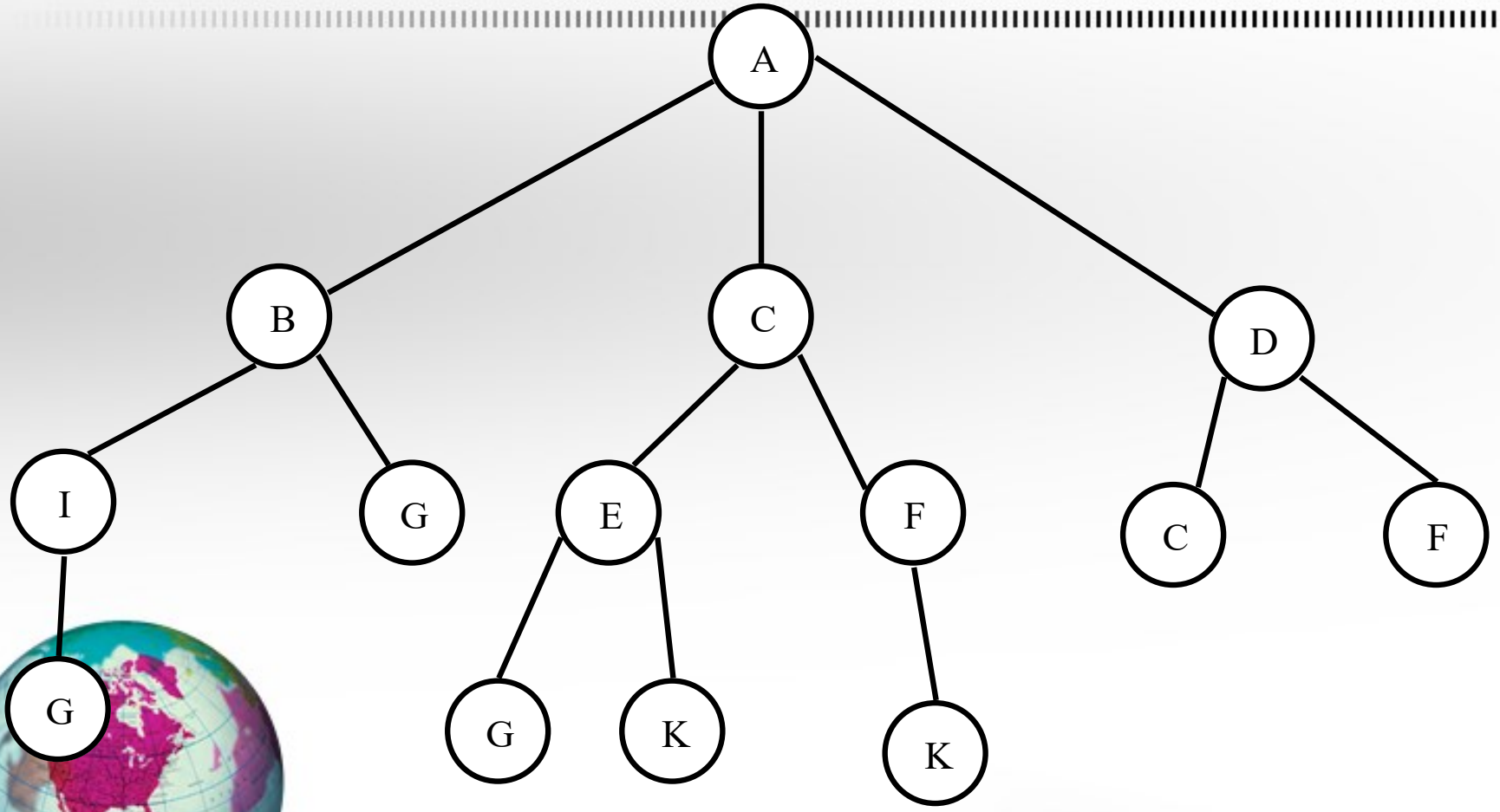
  Lấy p từ đầu O

  Duyệt p

  Bỏ p vào C

  Với mỗi q kề p, q không thuộc C: bỏ q vào cuối O
- }


A
B
D
C
I
F
E
G
K


A
B
C
D
I
G
E
F
C
F
G
G
K
K

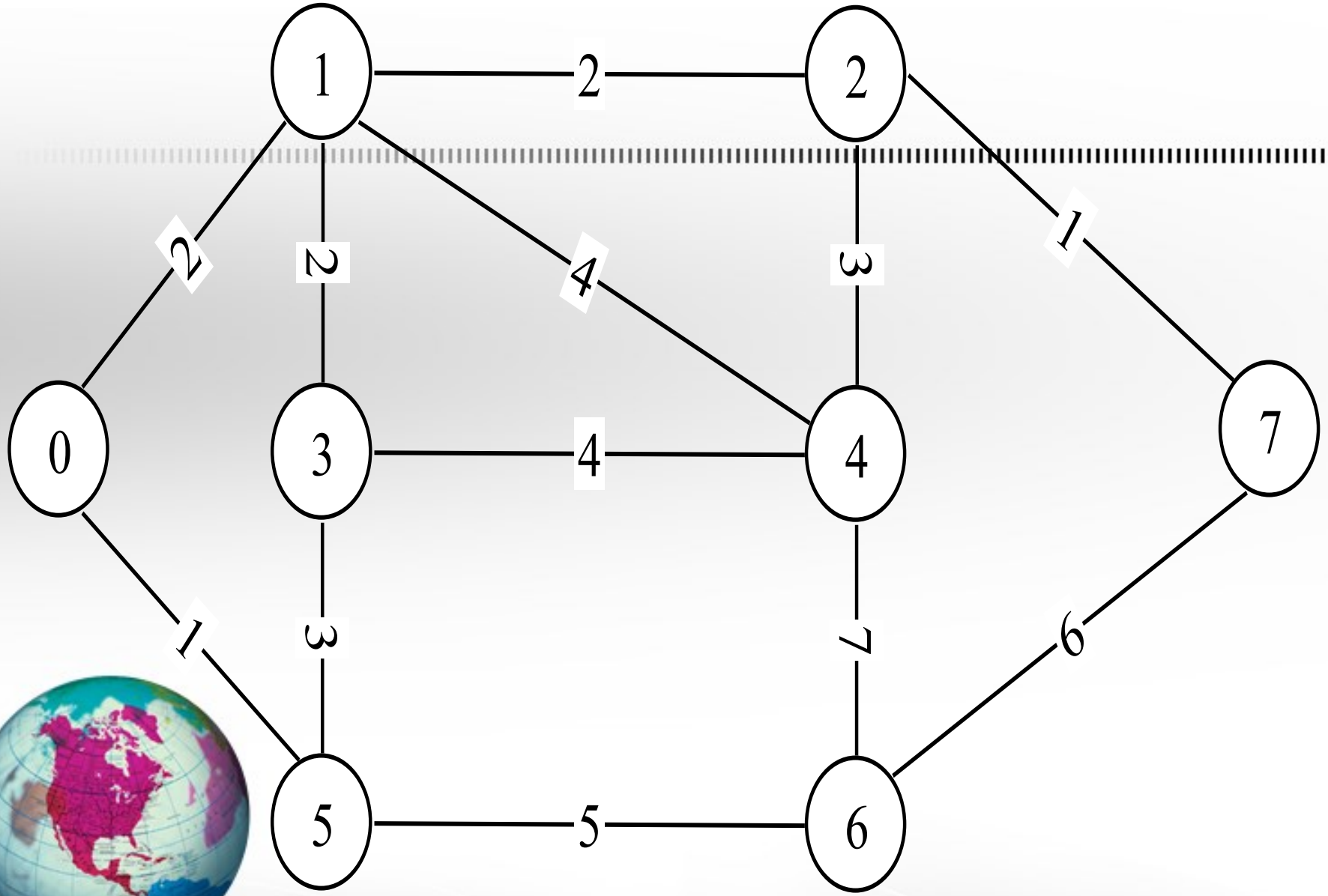
- **Bài tập 3**. Đại dương được xem như là một mặt phẳng toạ độ trên đó có n hòn đảo với toạ độ lần lượt là (x1, y1), (x2, y2), …, (xn, yn). Một chiếc ca nô xuất phát từ đảo d1 muốn tuần tra đến đảo d2. bình xăng của ca nô chỉ chứa đủ xăng để đi được một quãng đường dài không quá m (km). Trên đường đi ca nô có thể ghé một số đảo nào đó để tiếp thêm xăng, lúc này ca nô được tiếp thêm xăng đầy bình chứa. Hãy chỉ ra một đường đi từ đảo d1 đến đảo d2 sao cho số lần ghé đảo trung gian để tiếp thêm xăng là ít nhất.

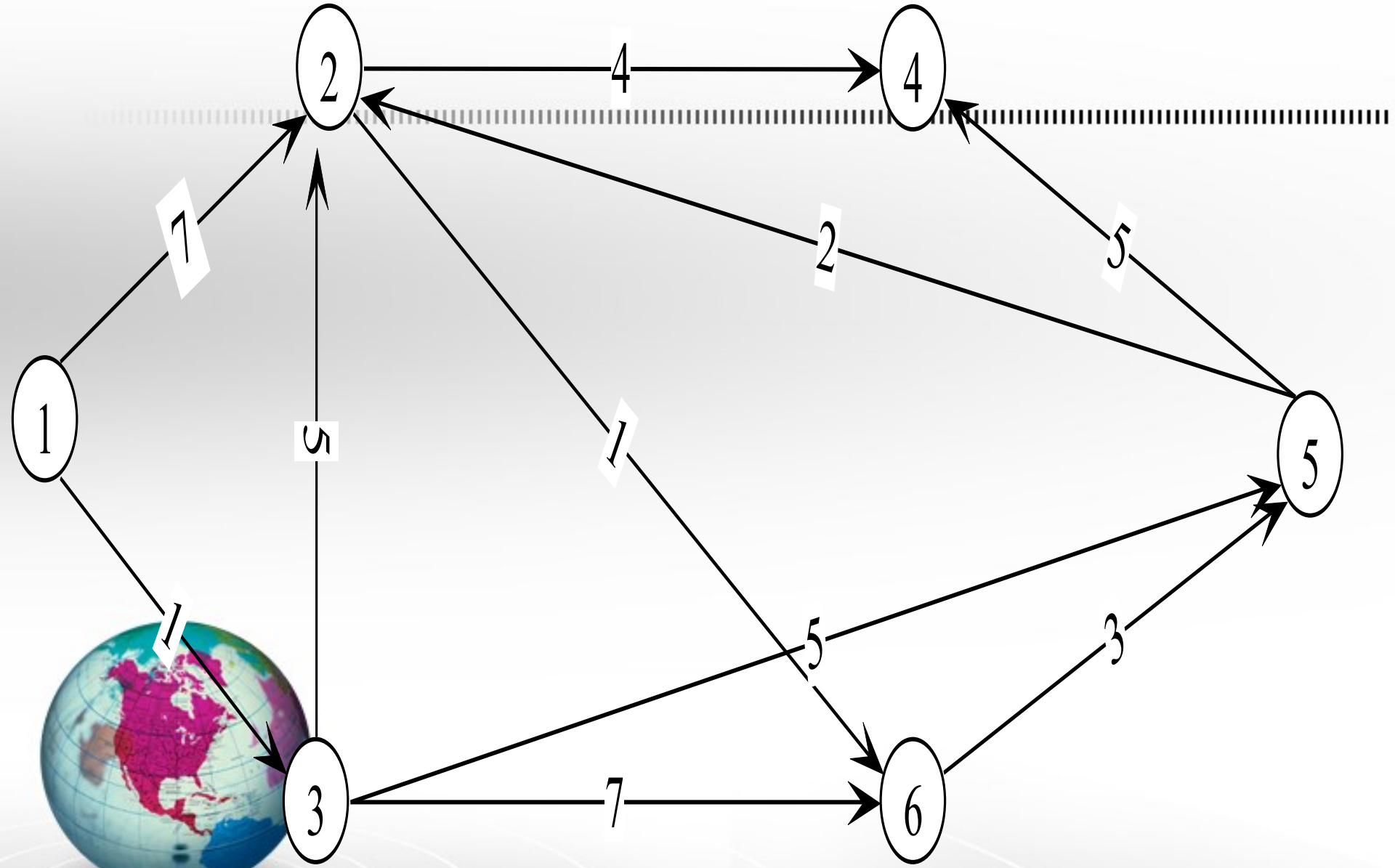
Tìm kiếm sâu dần

- Kỹ thuật tìm kiếm sâu dần là thực hiện việc tìm kiếm với độ sâu ở mức giới hạn d nào đó. Nếu không tìm ra nghiệm ta tăng độ sâu lên d+1 và lại tìm kiếm theo độ sâu tới mức d+1. Quá trình trên được lặp lại với d lần lượt là 1, 2,...đến độ sâu max nào đó

Tìm kiếm tốt nhất

- Dùng tri thức về bài toán để hướng dẫn
- Tại mỗi nút được xem xét: tìm kiếm tiếp tục theo nhánh nào tin tưởng sẽ dẫn đến lời giải.


1
2
2
2
2
4
3
1
0
3
4
4
7
1
3
7
6
5
5
6


2
4
4
7
2
5
1
5
1
5
1
5
3
7
6
3

Tìm kiếm đường đi có giá thành cực tiểu
Thuật toán Dijkstra (1)

Mỗi đỉnh tương ứng với 1 số g(n): giá thành đi từ đỉnh ban đầu tới n

C : đỉnh đóng

O : đỉnh mở

Bước 1: O= {S}

C={}

g(S)=0

Bước 2: While (**O≠{}**)

2.1 Chọn N thuộc O có g(N) nhỏ nhất

N: mục tiêu ➔ dừng, thông báo kết quả

Nếu không tồn tại N ➔ dừng

2.2 Chuyển N qua C, và mở các Q sau N

Bước 3: Không có kết quả.

Tìm kiếm đường đi có giá thành cực tiểu
Thuật toán Dijkstra (2)

- **2.2 Chuyển N qua C, và mở các Q sau N**

  2.2.1 Nếu Q đã có trong O

  nếu g(Q)> g(N)+cost(N,Q)

  g(Q) = g(N)+cost(N,Q)
  prev(Q)=N

  2.2.2 Nếu Q chưa có trong O

  g(Q) = g(N)+cost(N,Q)

  prev(Q)=N

Tìm kiếm cực tiểu sử dụng hàm đánh giá - Thuật toán A*

Cụ thể trong quá trình lựa chọn đỉnh để duyệt xét các đỉnh kế tiếp thì thuật giải A* dựa vào giá trị sau:

$$f(N) = g(N) + h(N)$$

với g(N) số đo lộ trình từ S tới N

f(N) ước tính độ dài từ S đến N

Tìm kiếm cực tiểu sử dụng hàm đánh giá - Thuật toán A*

Bước 1:

C = {};

O = {S};

g(S) = 0;

f(S) = h(S);

Tìm kiếm cực tiểu sử dụng hàm đánh giá - Thuật toán A*

Bước 2:

while(O≠{})

{

2.1 Chọn N ∈ O có f(N) nhỏ nhất

2.2 Lấy N từ O cho vào C

2.3 if(N≡G)

Dừng. Kết luận: tìm được

Tìm kiếm cực tiểu sử dụng hàm đánh giá - Thuật toán A*

2.4 Xét các đỉnh kế Q của N

TH1: **Q∈O**

```
if(g(Q)>g(N)+w(N,Q))
        g(Q)= g(N)+w(N,Q);
        f(Q)=g(Q)+h(Q);
        prev(Q)=N
```

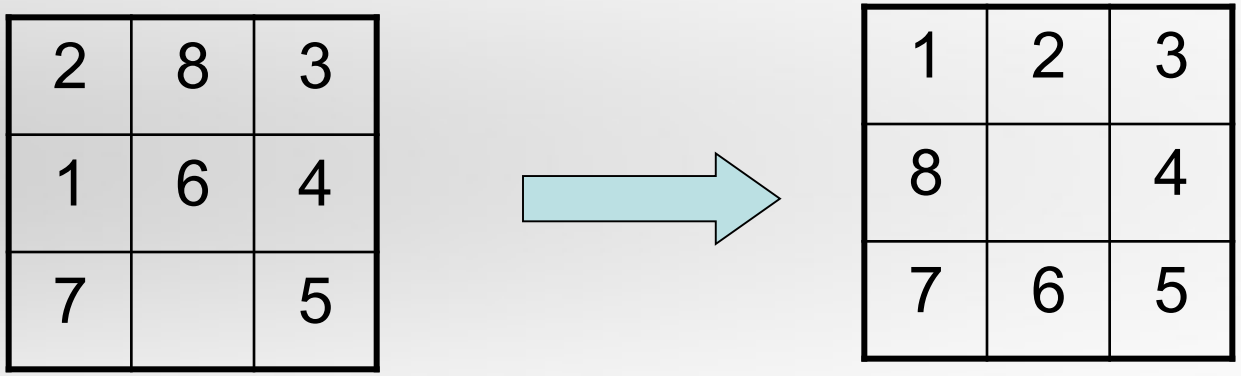
TH2: **Q∉O**

```
g(Q)= g(N)+w(N,Q);
f(Q)=g(Q)+h(Q);
prev(Q)=N
```

}

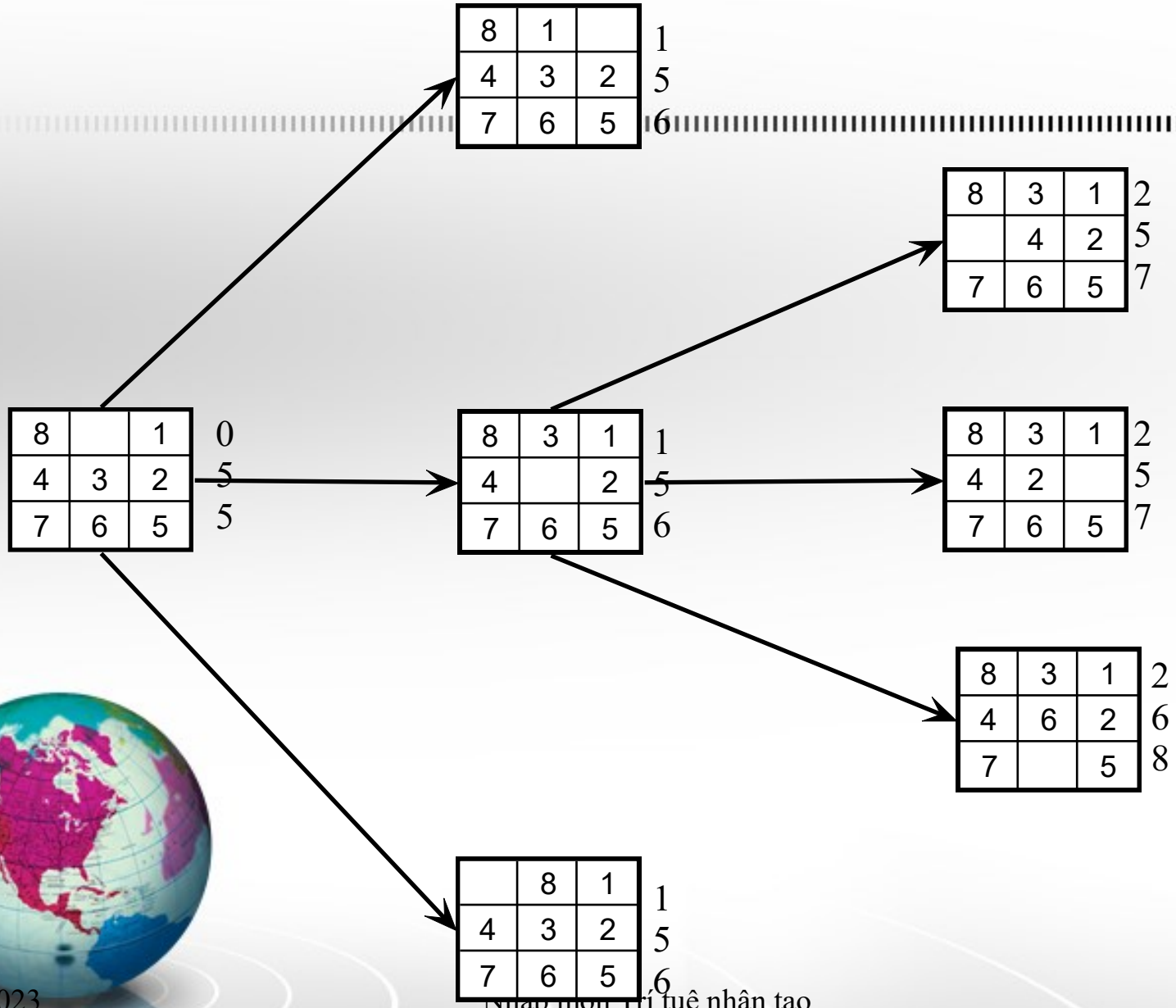
Bước 3: Kết luận…

A* - Ví dụ



- Trạng thái đầu và cuối của bài toán 8 số
- Các toán tử: qua trái, phải, lên trên, xuống dưới

$h(S_i)$ = số nút đúng của S_i so với G


8 1
4 3 2
7 6 5
1 5 6
8 3 1
4 2
7 6 5
2 5 7
8 1
4 3 2
7 6 5
0 5 5
8 3 1
4 2
7 6 5
1 5 6
8 3 1
4 2
7 6 5
2 5 7
8 3 1
4 6 2
7 5
2 6 8
8 1
4 3 2
7 6 5
1 5 6

Puzzle – Cài đặt

```
#define UP 1
#define DN 2
#define LT 3
#define RT 4
#define NO 0
```

Puzzle – Cài đặt

```
typedef char TAB[3][3];
struct info{
    TAB S;
    unsigned g,h,f;
    int d;
};
```

Puzzle – Cài đặt

```
define SIZE 500
typedef struct List{
    int n;
    info e[SIZE];
    List(){n=0;}
    void them(info X);
    info timfnho();
    void xuatlist();
    int vitri(info X)
};
```

Puzzle – Cài đặt

```
List O,C,KQ;
TAB S0, G;
void main()
{
    nhap();
    if(Astart())
    {
        timkq();
        KQ.xuatlist();
    }
}
```

Puzzle – Cài đặt

```
void nhap();
unsigned count(TAB S); //h
int sobang(TAB S1, TAB S2);
void ganbang(TAB S1, TAB S2); //gán S2 cho S1
void timotrong(TAB S, int &k, int &l)
void bangke(TAB N, int d, TAB S,int k, int l) //S là bảng kề của N
int Astart();
void xulyke(info X, int d, TAB S);
void xuat(TAB S);
void timkq();
```

Puzzle – Cài đặt

```
int Astart()
{
    info X;
    ganbang(X.S,S0);
    X.g=0;
    X.h=count(S0);
    X.f=X.h;
    X.d=NO;
    O.them(X);
    while(O.n>0)
    {
        X=O.timfnho();
        C.them(X);
```

Puzzle – Cài đặt

```
while(O.n>0)
   {
        X=O.timfnho();
        C.them(X);
        if(sobang(X.S,G)) return 1;
        timotrong(X.S, k, l);
        if(k>0)
        {
              d=UP;
              bangke(X.S,d,BK,k,l);
              xulyke(X,d,BK);
        }
```

Puzzle – Cài đặt

```
        if(k<2){…}
        if(l>0){…}
        if(l<2){…}
        }
    return 0;
}
```

Puzzle – Cài đặt

```
void xulyke(info X, int d, TAB S)
{
    info Y;
    int k;
    if(C.vitri(S)==-1) return;
    if(k=O.vitri(S)==-1)
    {
        ganbang(Y.S,S);
        Y.g=X.g+1;
        Y.h=count(S);
        Y.f=Y.g+Y.h;
        Y.d=d;
        O.them(Y);
    }
```

Puzzle – Cài đặt

```
}
    else if(X.g+1<O.e[k].g)
    {
        O.e[k].g=X.g+1;
        O.e[k].f= O.e[k].g+ O.e[k].h;
        O.e[k].d=d;
    }
}
```