NHẬP MÔN TRÍ TUỆ NHÂN TẠO

CHƯƠNG 2

GIẢI QUYẾT VẤN ĐỀ BẰNG TÌM KIẾM

Nội dung

- Biểu diễn bài toán trong KGTT
- Tìm kiếm mù (uninformed search)
- Tìm kiếm heuristic (informed search)
- Cây trò chơi, cắt tỉa alpha –beta
- Bài toán thoả mãn ràng buộc

TÌM KIẾM MÙ
(UNINFORMED BLIND SEARCH)

Tìm kiếm mù

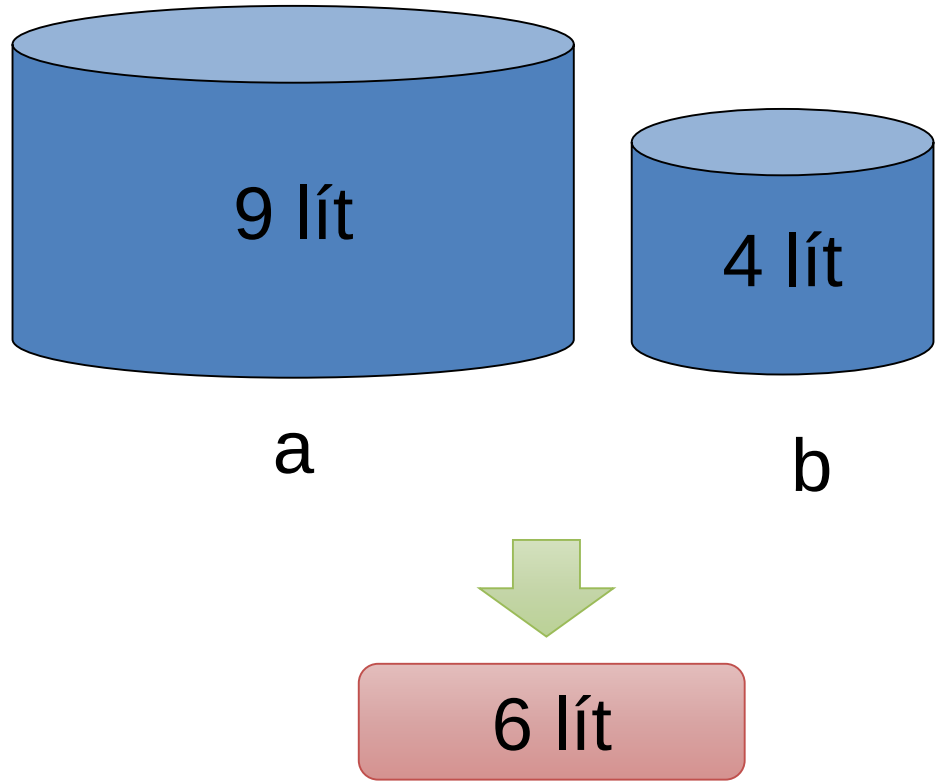
- **Tìm kiếm rộng (breath-first search)**
- **Tìm kiếm sâu (depth-first search)**
- Tìm kiếm theo độ sâu có giới hạn (depth-limited search)
- Tìm kiếm theo chiều sâu lặp (iterative deepening depth-first search)
- Tìm kiếm giá thành đồng nhất

Tìm kiếm rộng (breath-first search - BFS)

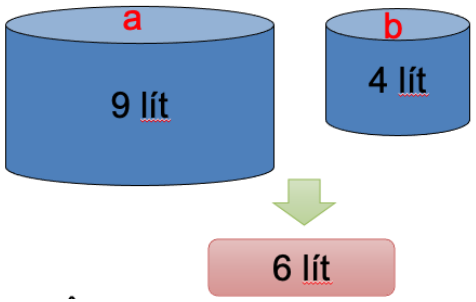
```
Procedure breadth-first-search;
    Begin                    % khởi đầu
        Open:= [start];                          Closed:= [ ];
        While open ≠ [ ] do      % còn các trạng thái chưa khảo sát
        Begin
        Loại bỏ trạng thái ngoài cùng bên trái khỏi open, gọi nó là X;
        If X là một đích then trả lời kết quả (thành công)      % tìm thấy đích
      else begin
        Phát sinh các con của X;
        Đưa X vào closed;
        Loại các con của X đã tồn tại trong open hoặc closed; % kiểm tra vòng lặp
        Đưa các con còn lại của X vào đầu bên phải của open;      % hàng đợi
      end;
        End;
        Trả lời kết quả (thất bại);       % không còn trạng thái nào
    End;
```

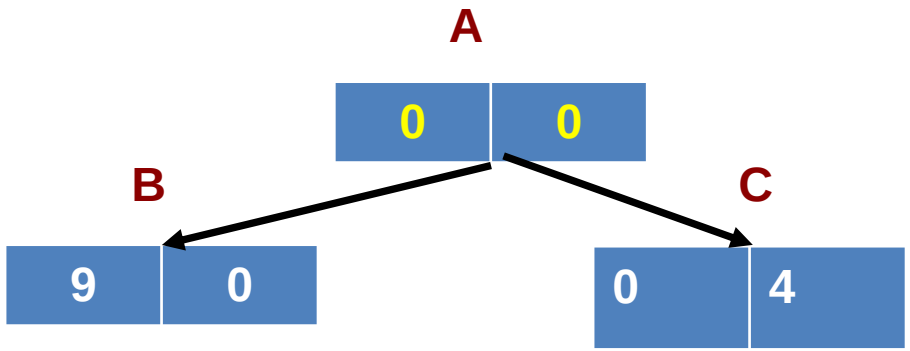
Ví dụ: Bài toán đong nước

	a	b
Start	0	0
1		



Ví dụ: Bài toán đong nước






```
Procedure breadth-first-search;
  Begin                  % khởi đầu
     Open:= [start];                    Closed:= [ ];
     While open ≠ [ ] do     % còn các trạng thái chưa khảo sát
     Begin
     Loại bỏ trạng thái ngoài cùng bên trái khỏi open, gọi nó là X;
     If X là một đích then trả lời kết quả (thành công)      % tìm thấy đích
   else begin
     Phát sinh các con của X;
     Đưa X vào closed;
     Loại các con của X đã tồn tại trong open hoặc closed; % kiểm tra vòng lặp
     Đưa các con còn lại của X vào đầu bên phải của open;      % hàng đợi
   end;
     End;
     Trả lời kết quả (thất bại);       % không còn trạng thái nào
End;
```

30

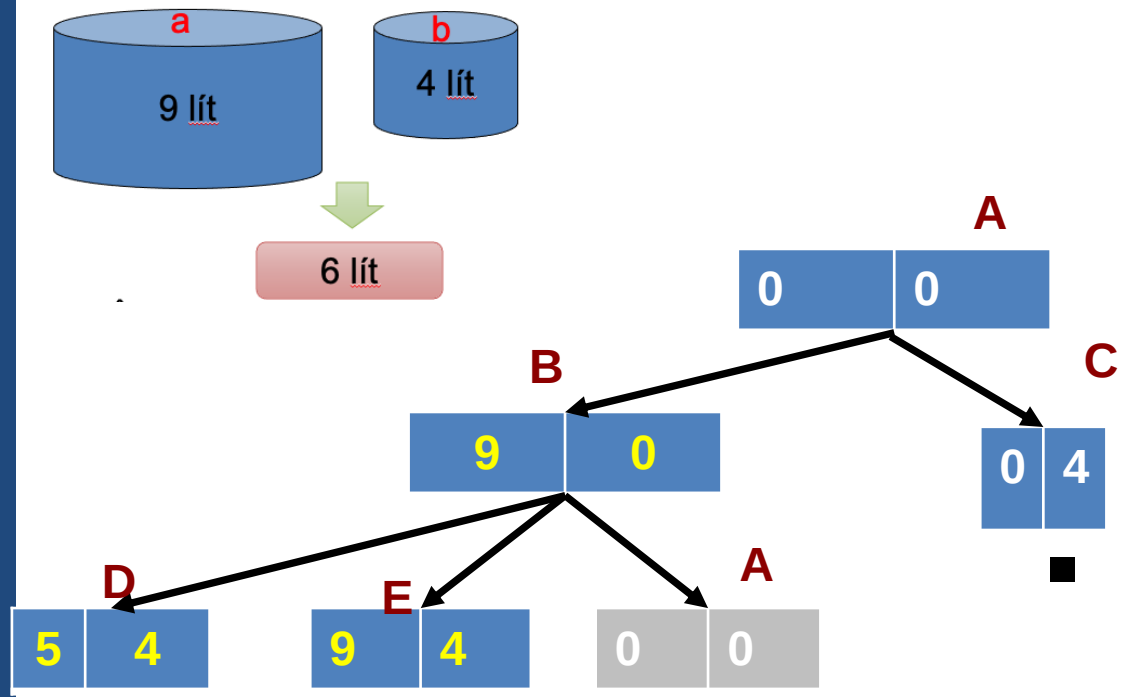
- Các bước thực hiện tìm kiếm rộng:

  Bước 1: *Open = [A: 0-0]; closed = []*

  - *Xét A, A ko phải trạng thái đích*
  - *Đưa A vào closed: closed=[A:0-0]*
  - *Xét các con của A là B:9-0 và C:0-4*
  - *Các con của A ko tồn tại trong open-closed*
  - *Đưa các con của A vào bên phải open*

  open [B: 9-0,C: 0-4];

Ví dụ: Bài toán đong nước




```
Procedure breadth-first-search;
  Begin                    % khởi đầu
     Open:= [start];                    Closed:= [ ];
     While open ≠ [ ] do    % còn các trạng thái chưa khảo sát
     Begin
     Loại bỏ trạng thái ngoài cùng bên trái khỏi open, gọi nó là X;
     If X là một đích then trả lời kết quả (thành công)     % tìm thấy đích
   else begin
     Phát sinh các con của X;
     Đưa X vào closed;
     Loại các con của X đã tồn tại trong open hoặc closed; % kiểm tra vòng lặp
     Đưa các con còn lại của X vào đầu bên phải của open;      % hàng đợi
   end;
     End;
     Trả lời kết quả (thất bại);       % không còn trạng thái nào
  End;
```

30

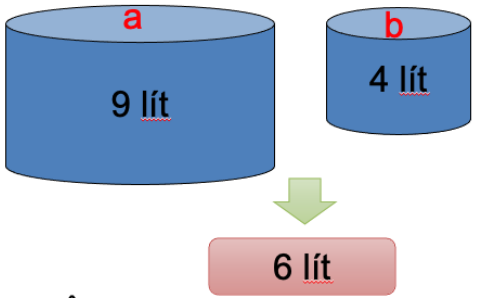
- Các bước thực hiện tìm kiếm rộng:

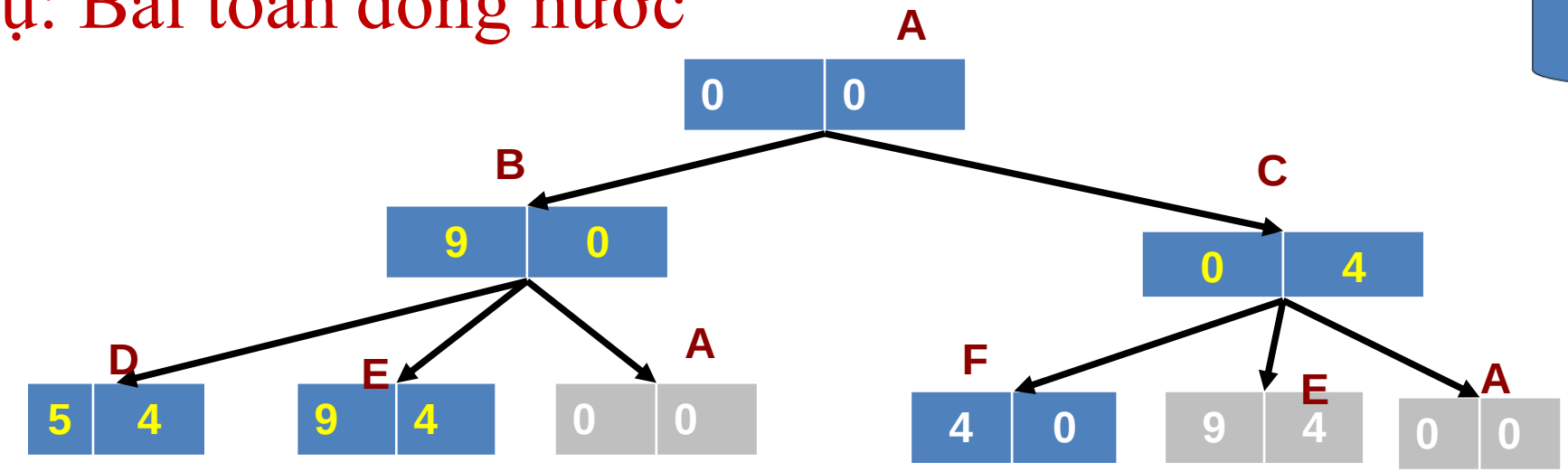
Bước 2: *Open = [B: 9-0,C: 0-4]; closed = [A: 0-0]*

- *Xét B:9-0, B không là trạng thái đích*
- *Đưa B vào Closed: closed[A,B]*
- *Xét các con của B: E:9-4,D:5-4, A:0-0*
- *Loại các con đã tồn tại trong open và closed – loại A:0-0*
- *Thêm các con còn lại vào open*

Open = [C:0-4,D:5-4,E:9-4,]; closed = [A,B]

Ví dụ: Bài toán đong nước





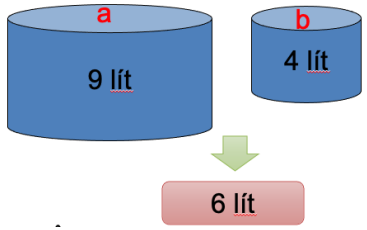
- Các bước thực hiện tìm kiếm rộng:

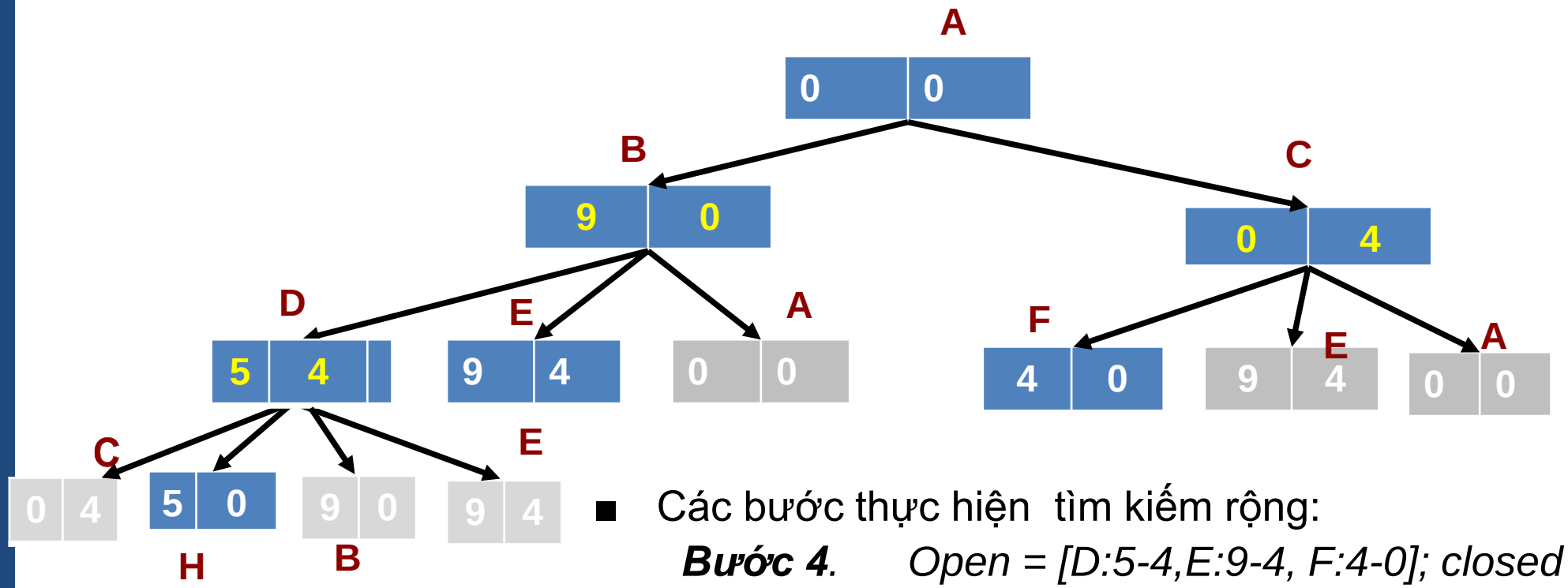
Bước 3. *Open = [C:0-4,D:5-4,E:9-4,]; closed = [B,A]*

- *Xét C, ko phải trạng thái đích*
- *Đưa C vào Closed*
- *Các con của C: F: 4-0, E:9-4, A:0-0*
- *Loại các con đã tồn tại trong open và closed – loại A:0-0 và E:9-4*
- *Thêm các con còn lại vào open*

Open = [D:5-4,E:9-4, F:4-0]; closed = [A,B,C]

Ví dụ: Bài toán đong nước





- Các bước thực hiện tìm kiếm rộng:

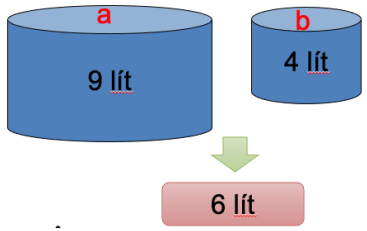
  Bước 4. *Open = [D:5-4,E:9-4, F:4-0]; closed = [A,B,C]*

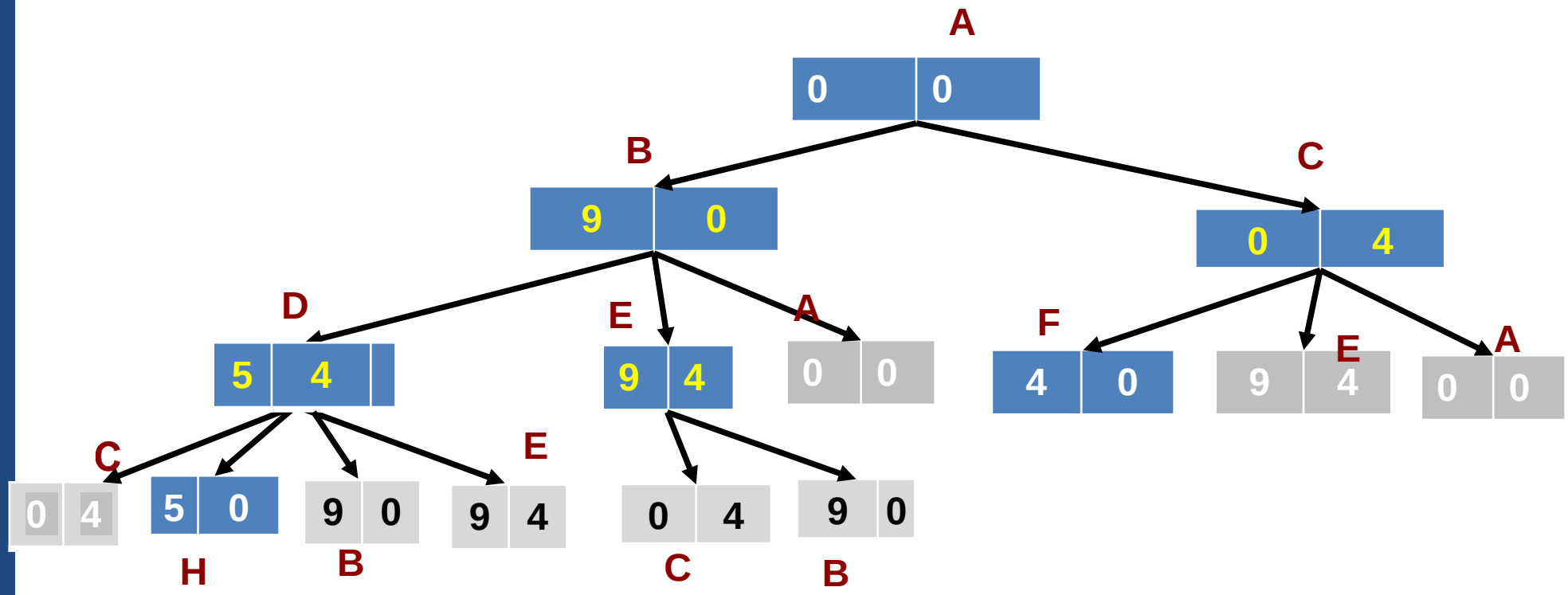
  Xét D, ko phải trạng thái đích

  - *Đưa D vào Closed: closed = [A,B,C,D]*
  - *Các con của D: C: 0-4, H:5-0, B:9-0, E:9-4*
  - *Loại các con đã tồn tại trong open và closed –*
  - *loại C: 0-4, B:9-0, E:9-4*
  - *Thêm các con còn lại vào open*

    Open = [E:9-4, F:4-0 , H:5-0,]; closed = [A,B,C,D]

Ví dụ: Bài toán đong nước



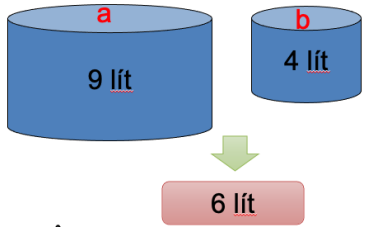


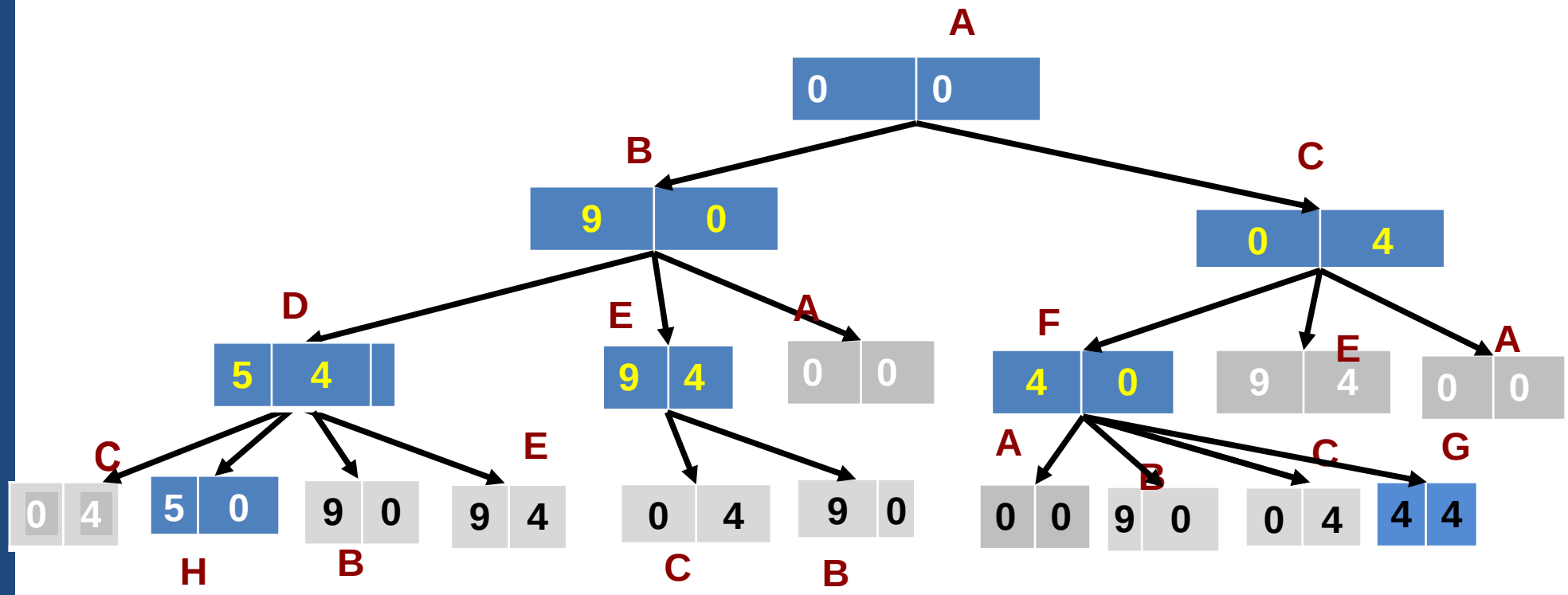
Bước 5: *Open = [E:9-4, F:4-0 , H:5-0,]; closed = [A,B,C,D]*

Xét E, ko phải trạng thái đích

- *Đưa E vào Closed: closed = [A,B,C,D,E]*
- *Các con của E: C: 0-4, B:9-0*
- *Loại các con đã tồn tại trong open và closed – loại : C: 0-4, B:9-0*
- *Open = [F:4-0, H:5-0,]; closed = [A,B,C,D,E]*

Ví dụ: Bài toán đong nước



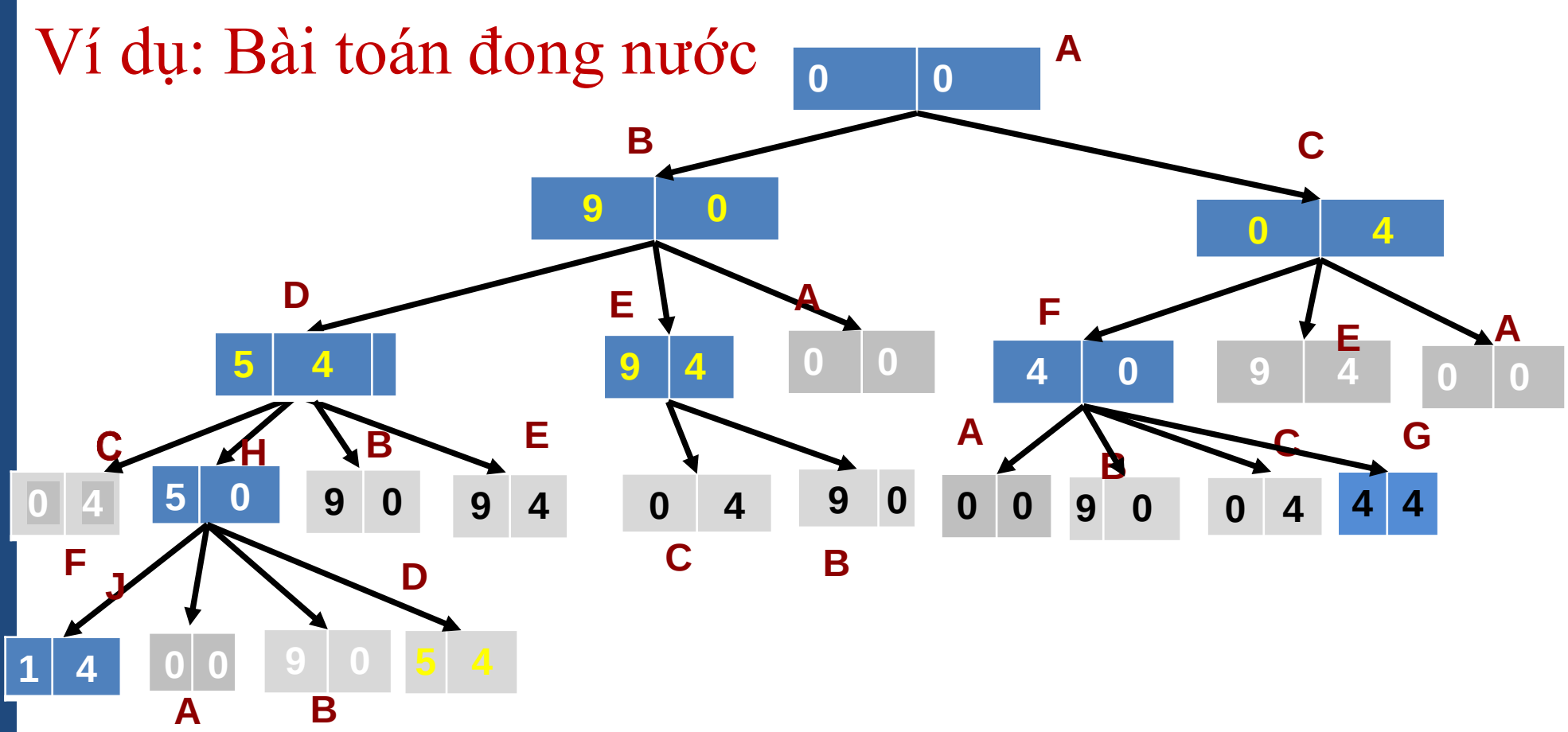


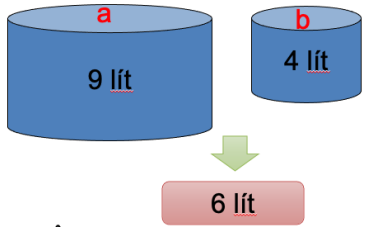
Bước 6: *[F:4-0, H:5-0,]; closed = [A,B,C,D,E]*

Xét F, ko phải trạng thái đích

- *Đưa F vào Closed: closed = [A,B,C,D,E,F]*
- *Các con của F: A:0-0,B:9-0, C: 0-4, G:4-4*
- *Loại các con đã tồn tại trong open và closed – loại : A:0-0,B:9-0, C: 0-4,*
- *Open = [H:5-0, G:4,4]; closed = [A,B,C,D,E,F]*

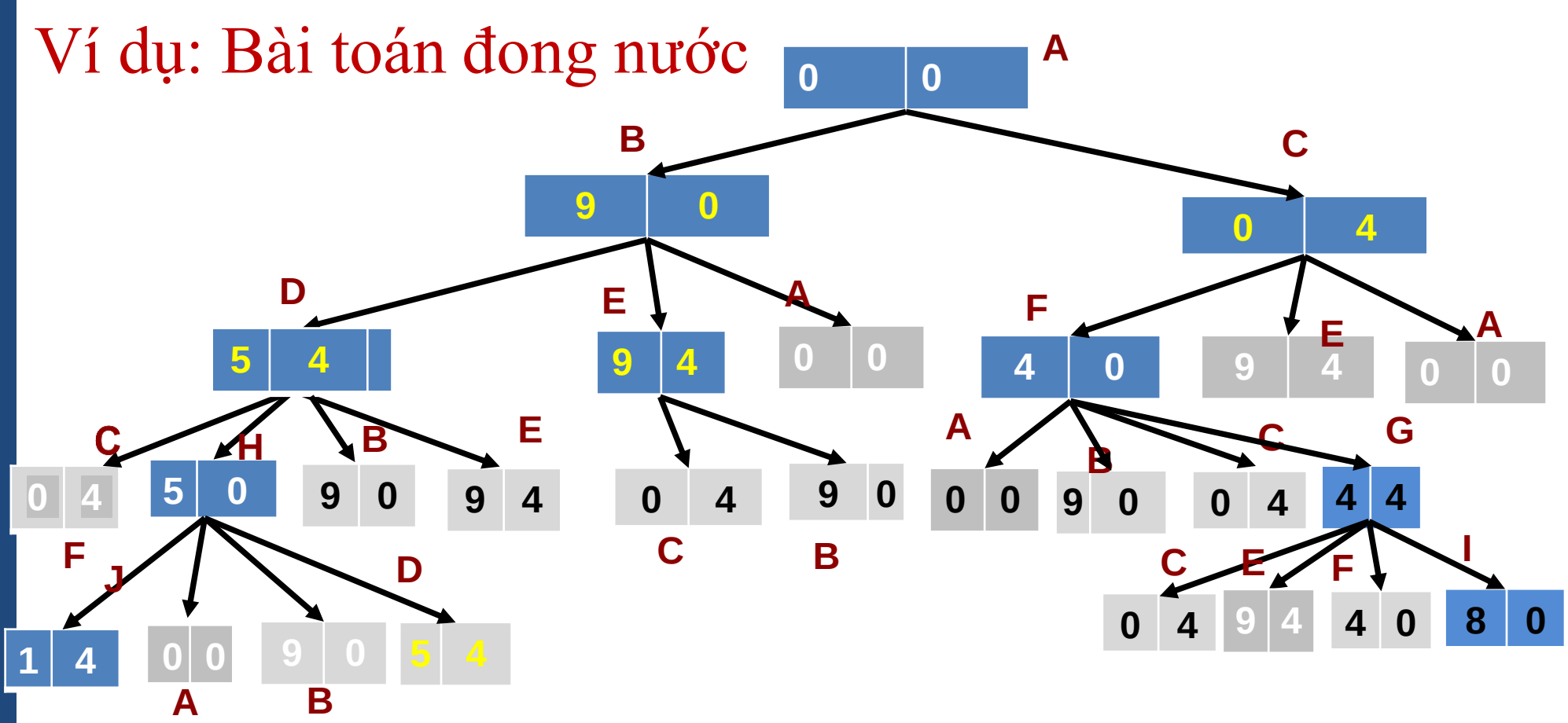
Ví dụ: Bài toán đong nước

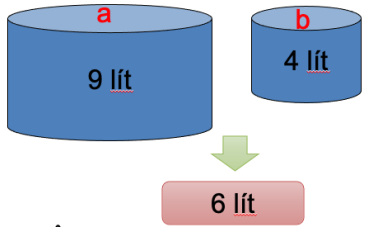




- ***Bước 7:*** *Open = [H:5-0, G:4,4]; closed = [A,B,C,D,E,F]*
  Xét H, ko phải trạng thái đích
- *Đưa H vào Closed: closed = [A,B,C,D,E,F,H]*
- *Các con của H: A: 0-0, B:9-0, D:5-4, J:1-4*
- *Loại các con đã tồn tại trong open và closed – loại : A: 0-0, B:9-0, D:5-4,*
- *Thêm J:1-4 vào Open = [G:4-4,J:1-4]; closed = [A,B,C,D,E,F,H]*

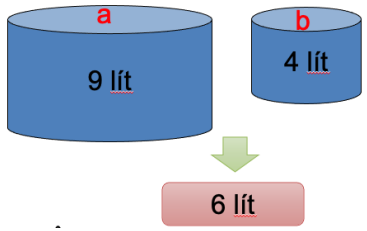
Ví dụ: Bài toán đong nước

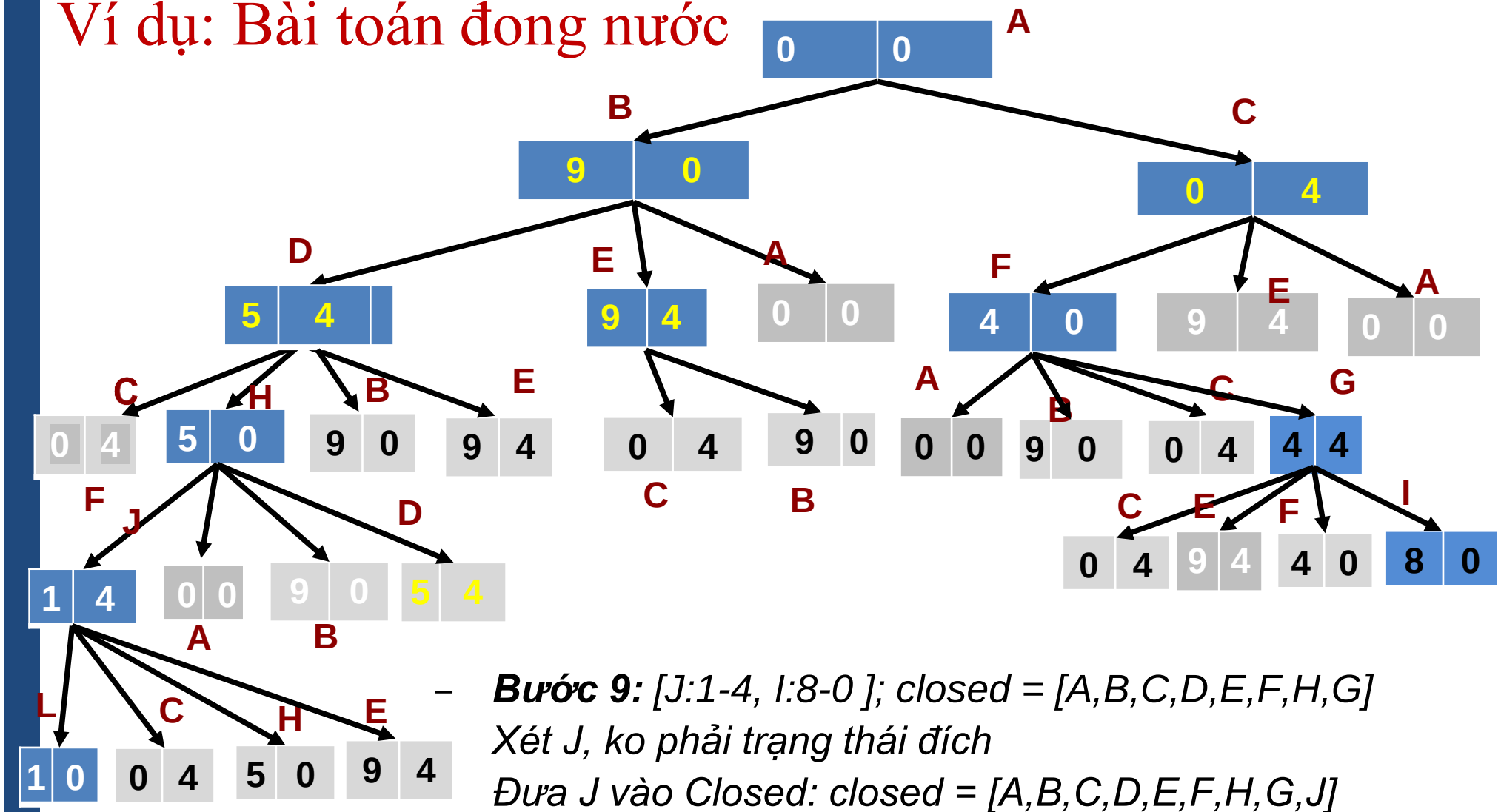




- ***Bước 8:*** *[G:4-4,J:1-4]; closed = [A,B,C,D,E,F,H*
- *Xét G, ko phải trạng thái đích*
- *Đưa G vào Closed: closed = [A,B,C,D,E,F,H,G]*
- *Các con của G: C: 0-4, E:9-4, F:4-0, I:8-0*
- *Loại các con đã tồn tại trong open và closed – loại : C: 0-4, E:9-4, F:4-0*
- *Thêm I:8-0 vào Open = [J:1-4, I:8-0]; closed = [A,B,C,D,E,F,H,G]*

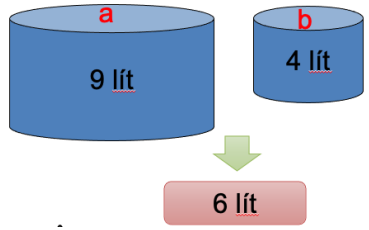
Ví dụ: Bài toán đong nước

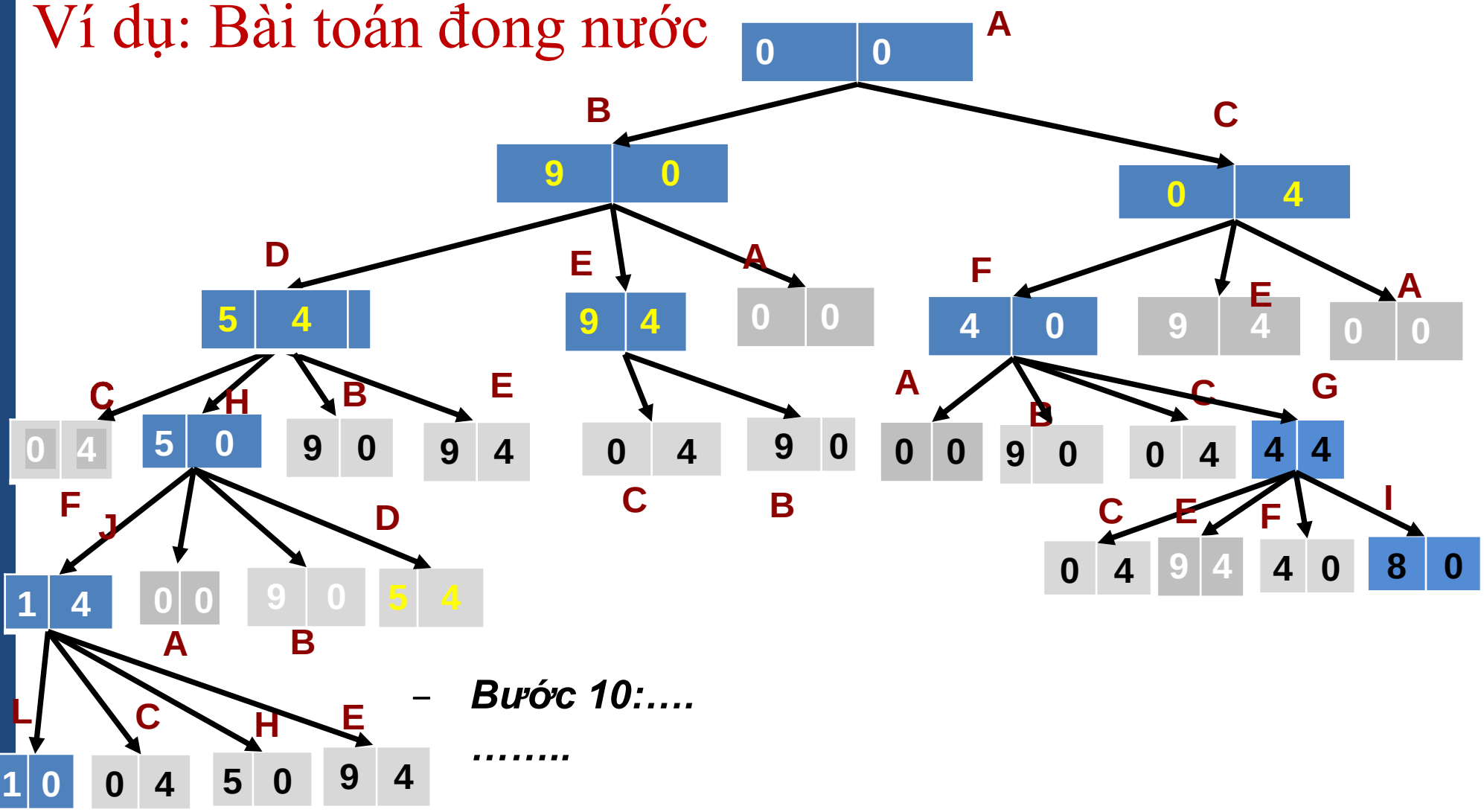




- ***Bước 9:*** *[J:1-4, I:8-0]; closed = [A,B,C,D,E,F,H,G]*
  Xét J, ko phải trạng thái đích
  Đưa J vào Closed: closed = [A,B,C,D,E,F,H,G,J]
- *Các con của J: L: 1-0, C:0-4, H:5-0, E:9-4*
- *Loại các con đã tồn tại trong open và closed – loại C:0-4, H:5-0, E:9-4*
- *Thêm L: 1-0, vào Open = [I:8-0, L: 1-0,]; closed = [A,B,C,D,E,F,H,G,J]*

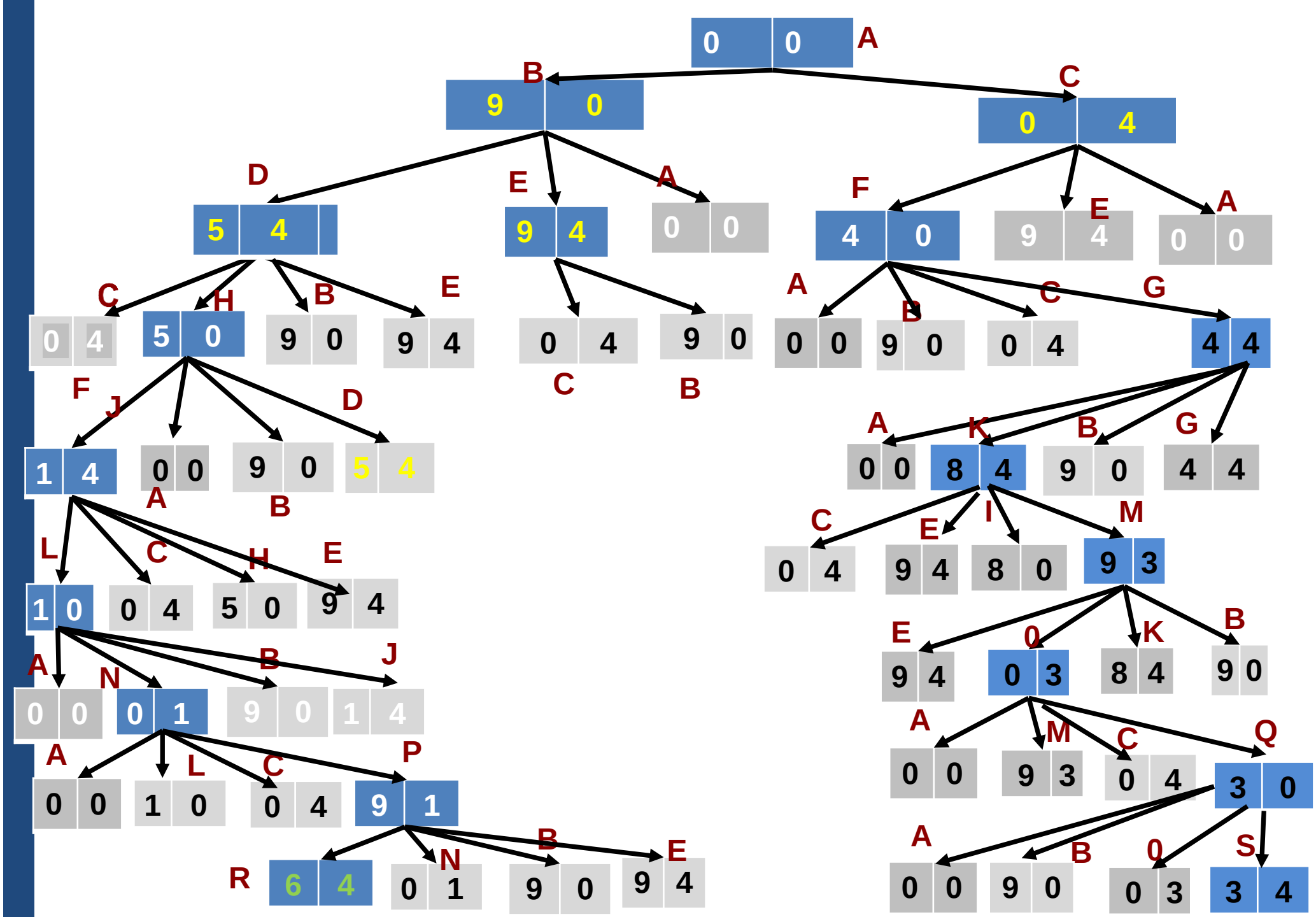
Ví dụ: Bài toán đong nước





- ***Bước 10:….***

……..

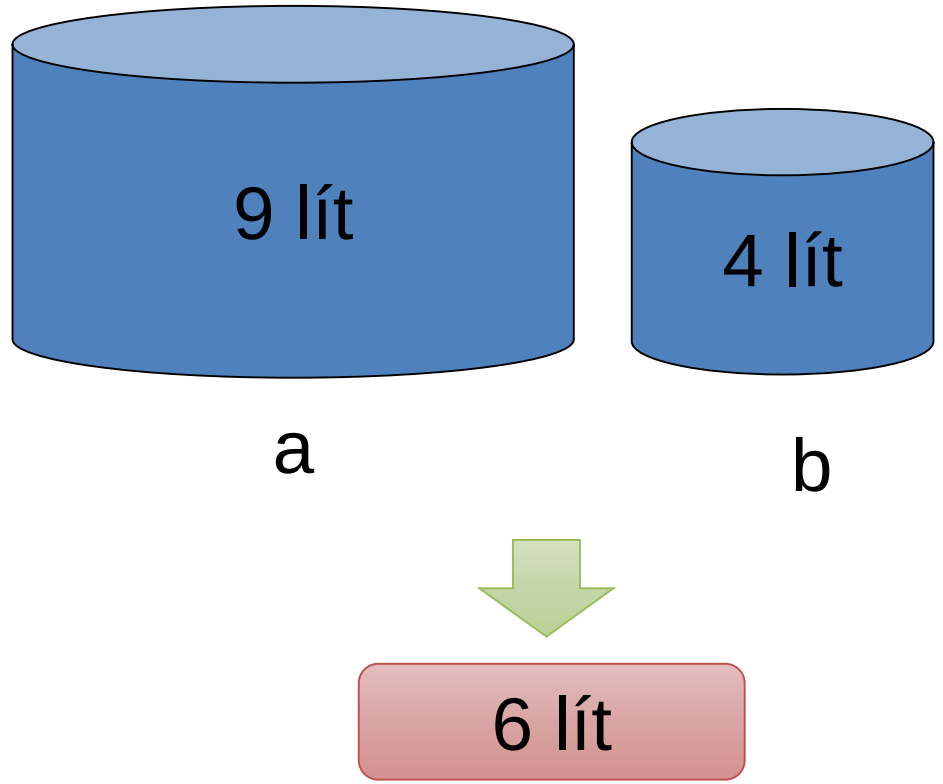

0 0
A
B
9 0
C
0 4
D
5 4
E
9 4
A
0 0
F
4 0
E
9 4
A
0 0
C
H
B
E
0 4
5 0
9 0
9 4
0 4
9 0
C
B
A
B
C
G
0 0
9 0
0 4
4 4
F
J
D
1 4
0 0
9 0
5 4
A
B
A
K
B
G
0 0
8 4
9 0
4 4
L
C
H
E
C
E
I
M
1 0
0 4
5 0
9 4
0 4
9 4
8 0
9 3
A
N
B
J
E
0
K
B
0 0
0 1
9 0
1 4
9 4
0 3
8 4
9 0
A
L
C
P
A
M
C
Q
0 0
1 0
0 4
9 1
0 0
9 3
0 4
3 0
R
N
B
E
A
B
0
S
6 4
0 1
9 0
9 4
0 0
9 0
0 3
3 4

Tìm kiếm sâu (depth-first search)

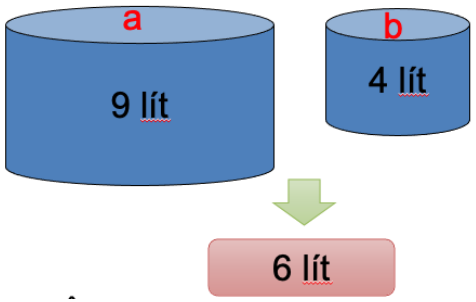
```
Procedure depth – first –search;
    Begin                    % khởi đầu
        Open:= [start];          Closed:= [ ];
        While open ≠ [ ] do    % còn các trạng thái chưa khảo sát
                Begin
                    Loại bỏ trạng thái ngoài cùng khỏi open, gọi nó là X;
                    If X là một đích then trả lời kết quả (thành công)% tìm thấy đích
                    else begin
                            Phát sinh các con của X;
                                Đưa X vào closed;
                             Loại các con của X trong open hoặc closed;
                  Đưa các con còn lại vào trên cùng của open; %ngăn xếp
                             end;
                  End;
                 Trả lời kết quả (thất bại);       % không còn trạng thái nào
               End;
```

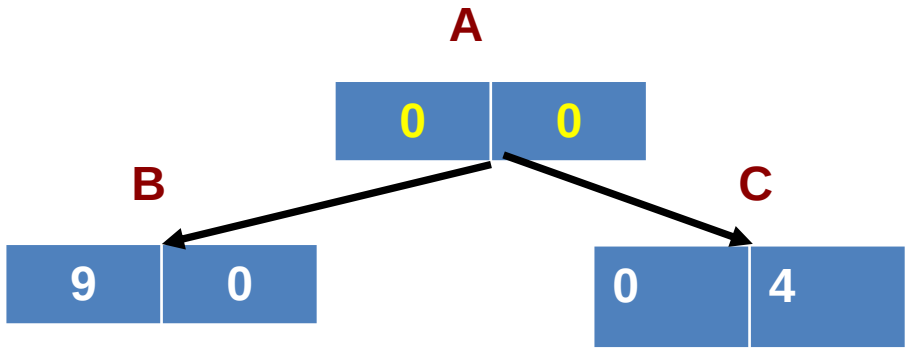
Ví dụ: Bài toán đong nước

	a	b
Start	0	0
1		



Ví dụ: Bài toán đong nước





Tìm kiếm sâu (depth-first search)

```
Procedure depth – first –search;
  Begin                  % khởi đầu
    Open:= [start];      Closed:= [ ];
    While open ≠ [ ] do  % còn các trang thái chưa khảo sát
        Begin
            Loại bỏ trang thái ngoài cùng bên trái khỏi open, gọi nó là X;
            If X là một đích then trả lời kết quả (thành công)% tìm thấy đích
            else begin
                    Phát sinh các con của X;
                      Đưa X vào closed;
                    Loại các con của X trong open hoặc closed;
          Đưa các con còn lai vào đầu bên trái của open; %ngăn xếp
                    end;
          End;
        Trả lời kết quả (thất bai);      % không còn trang thái nào
      End;
```

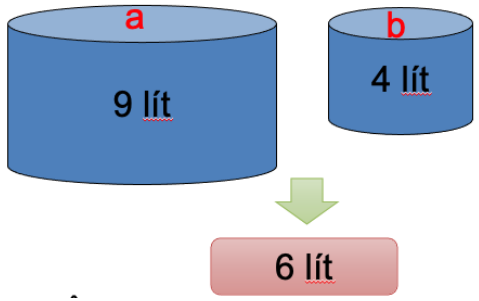
- Các bước thực hiện tìm kiếm sâu:

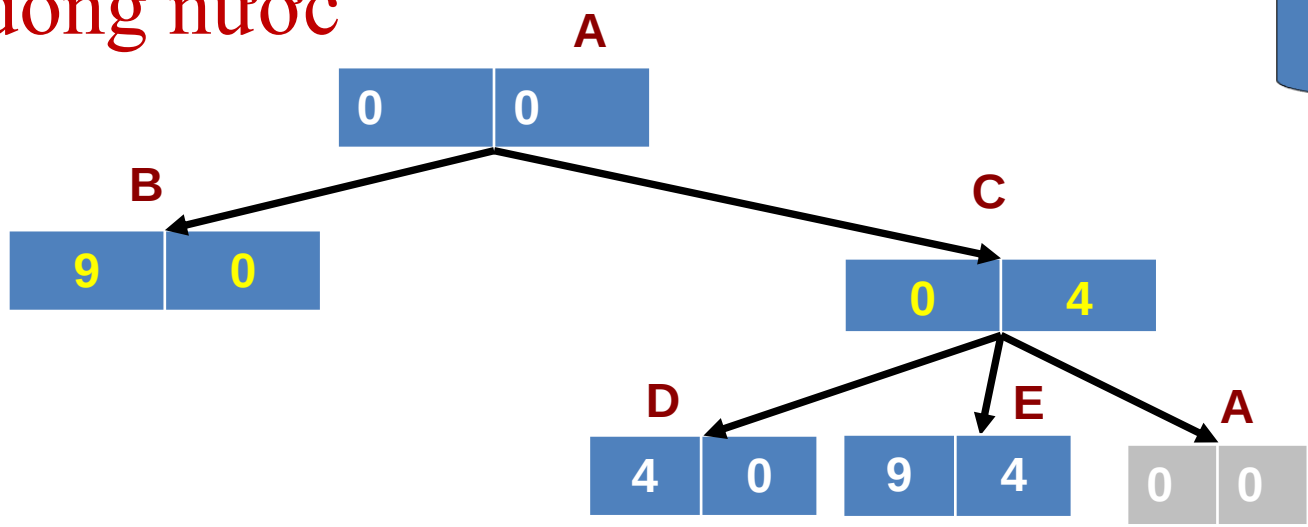
  Bước 1: *Open = [A: 0-0]; closed = []*

  - *Xét A, A ko phải trạng thái đích*
  - *Đưa A vào closed: closed=[A:0-0]*
  - *Xét các con của A là B:9-0 và C:0-4*
  - *Các con của A ko tồn tại trong open-closed*
  - *Đưa các con của A vào trên cùng open*

  open [C: 0-4, B: 9-0];

Ví dụ: Bài toán đong nước





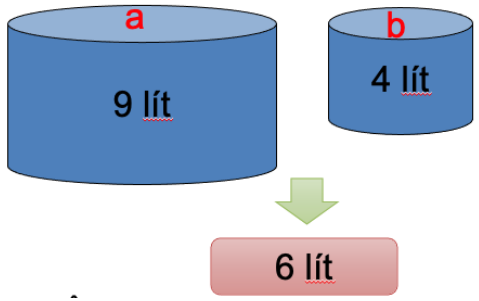
- Các bước thực hiện tìm kiếm sâu:

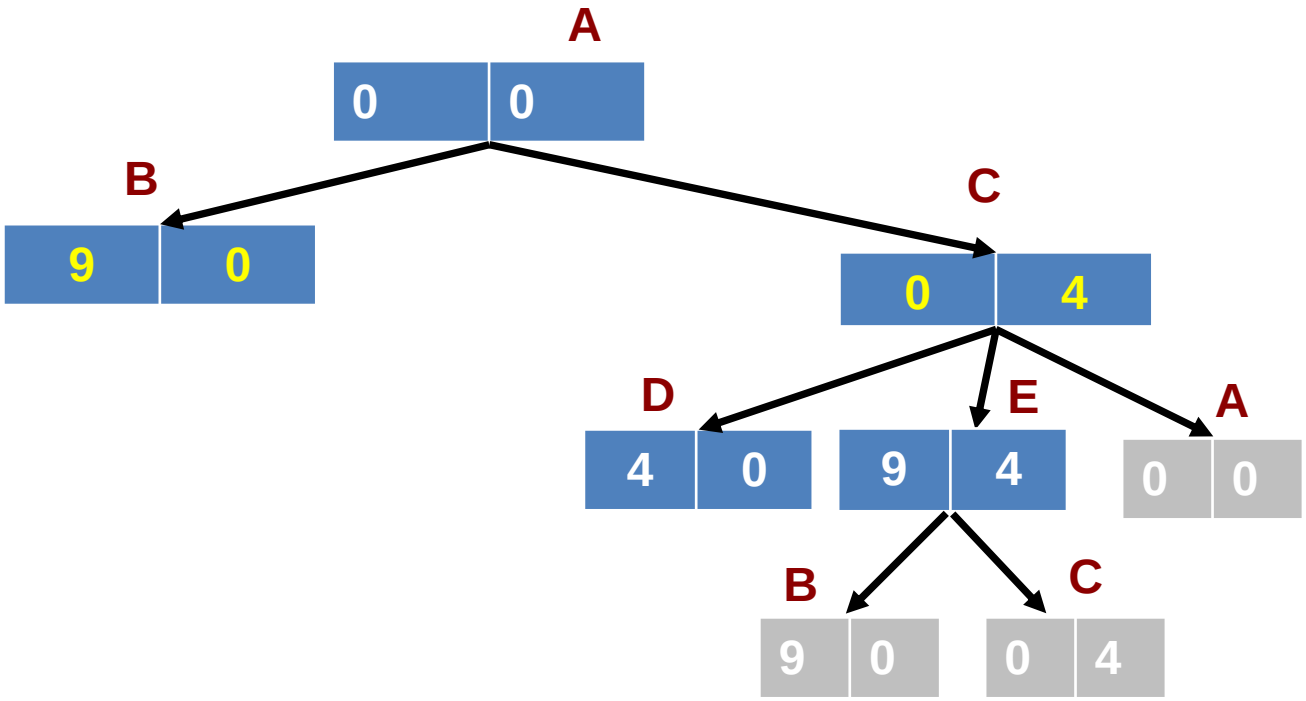
Bước 2. *open [C: 0-4, B: 9-0]; closed = [A]*

- *Xét C, ko phải trạng thái đích*
- *Đưa C vào Closed*
- *Các con của C: D: 4-0, E:9-4, A:0-0*
- *Loại các con đã tồn tại trong open và closed – loại A:0-0*
- *Thêm các con còn lại vào open*

Open = [E:9-4, D:4-0, B: 9-0]; closed = [C, A]

Ví dụ: Bài toán đong nước





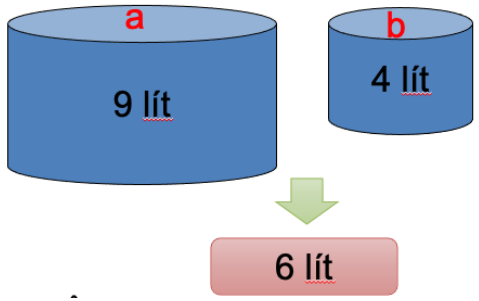
- Các bước thực hiện tìm kiếm sâu:

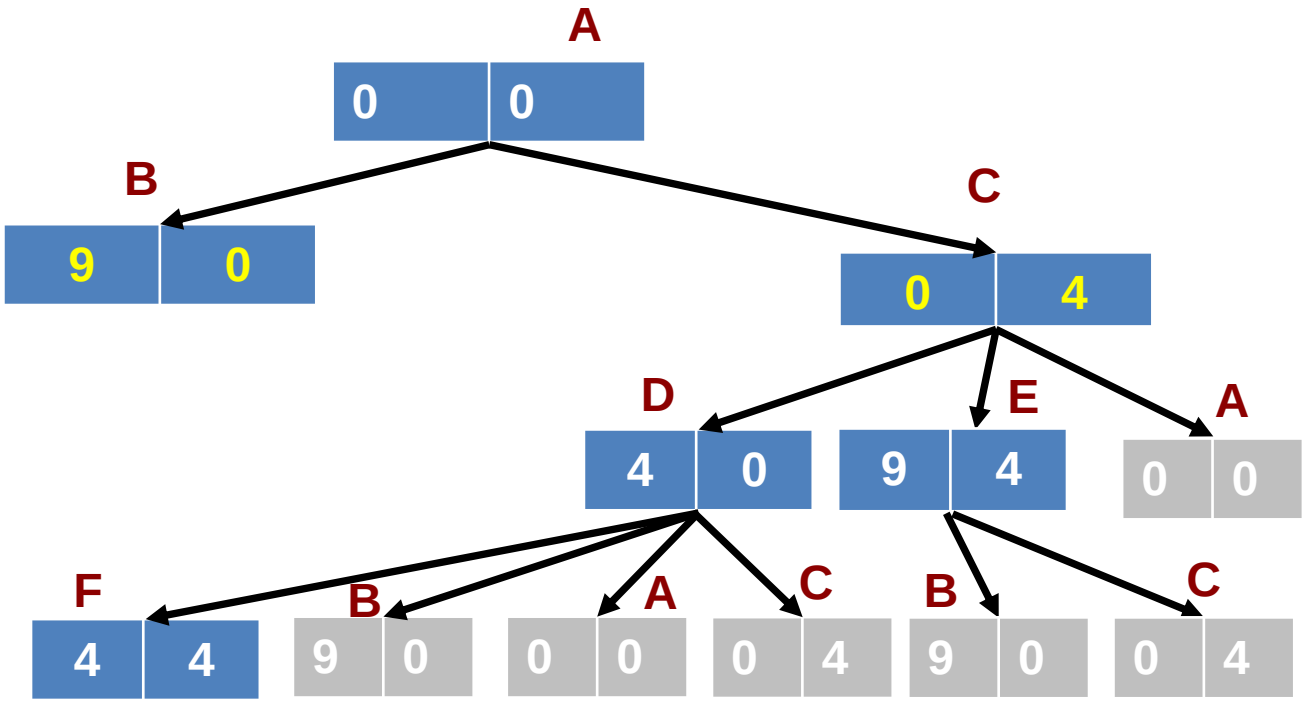
  Bước 3. *Open = [E:9-4, D:4-0, B: 9-0]; closed = [C, A]*

  - *Xét E, ko phải trạng thái đích*
  - *Đưa E vào Closed*
  - *Các con của E: B: 9-0, C:0-4*
  - *Loại các con đã tồn tại trong open và closed – loại B, C*
  - *Thêm các con còn lại vào open (không thêm con mới)*

    Open = [D:4-0, B: 9-0]; closed = [E, C, A]

Ví dụ: Bài toán đong nước





- ***Bước 5:….***
- ……..

- Các bước thực hiện tìm kiếm sâu:

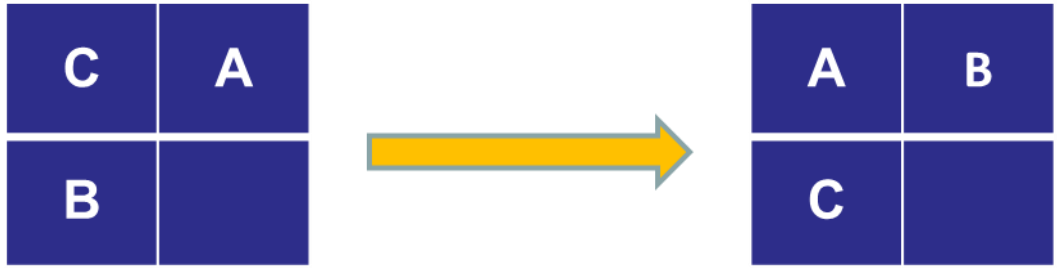
Bước 4. *Open = [D:4-0, B: 9-0]; closed = [E, C, A]*

- *Xét D, ko phải trạng thái đích*
- *Đưa D vào Closed*
- *Các con của D: B: 9-0, C:0-4, A: 0-0, F: 4-4*
- *Loại các con đã tồn tại trong open và closed – loại B, C, A*
- *Thêm các con còn lại vào open Open = [F:4-4, B: 9-0]; closed = [D, E, C, A]*

Bài tập 01

Cho bài toán sau:

Bài toán gồm một bảng 2×2 với các ô chữ được hiển thị như hình và một ô trống. Ở trạng thái bắt đầu (Initial State), các ô được sắp đặt như hình bên dưới, và nhiệm vụ của người giải là tìm cách đưa chúng về đúng thứ tự như minh họa dưới (Goal State).



a. Hãy xác định không gian trạng thái của bài toán (+1)

b. Vẽ cây tìm kiếm BFS, DFS cho bài toán trên theo không gian trạng thái đã định nghĩa. (+2)

Bài tập 02

Cho bài toán sau:

Một bài toán được phát biểu như sau: "Có 2 bình nước không chia độ: Bình thứ nhất dung tích 5 lít, bình thứ hai dung tích 3 lít. Cả hai bình ban đầu đều trống, sử dụng vòi bơm nước và không dùng thêm dụng cụ chứa nước nào khác hãy đong làm sao được 4 lít nước".

a. Hãy xác định không gian trạng thái của bài toán. (+1)

b. Vẽ cây tìm kiếm BFS, DFS cho bài toán trên theo không gian trạng thái đã định nghĩa. (+2)

Bài tập 03

Cho bài toán sau:

Cho 2 bình có dung tích lần lượt là m và n (lít). Với nguồn nước không hạn chế, dùng 2 bình trên để đong k lít nước. Không mất tính tổng quát có thể giả thiết k <= min(m,n). Tại mỗi thời điểm xác định, lượng nước hiện có trong mỗi bình phản ánh bản chất hình trạng của bài toán ở thời điểm đó. Gọi x là lượng nước hiện có trong bình dung tích m và y là lượng nước hiện có trong bình dung tích n. Như vậy bộ có thứ tự (x, y) có thể xem là trạng thái của bài toán. Với cách mô tả như vậy, hãy trả lời các câu hỏi sau:

a. **Trạng thái đầu và cuối bài toán là gì?**

b. **Các thao tác đong nước có thể là: đong đầy bình, làm rỗng bình hoặc đổ nước từ bình này sang bình khác. Giả sử trạng thái đang xét đang là (x,y) thì các trạng thái kế tiếp chuyển đến thông qua các thao tác đong nước có thể là những trạng thái nào? Với 0 <= x <= m và 0 <= y <= n**

Tìm kiếm mù

- Tìm kiếm rộng (breath-first search)
- Tìm kiếm sâu (depth-first search)
- **Tìm kiếm theo độ sâu có giới hạn (depth-limited search)**
- **Tìm kiếm theo chiều sâu lặp (iterative deepening depth-first search)**
- **Tìm kiếm giá thành đồng nhất**

Tìm kiếm theo độ sâu có giới hạn (depth-limited search)

- Tương tự như tìm kiếm theo độ sâu, tuy nhiên chỉ tìm đến một độ sâu **d** nhất định nào đó

Tìm kiếm theo chiều sâu lặp (iterative deepening depth-first search)

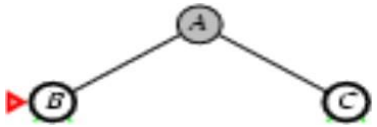
- Tương tự như tìm kiếm theo độ sâu, tuy nhiên lặp lại việc tìm kiếm với độ sâu giới hạn được tăng dần cho đến khi tìm được trạng thái đích
- *Vd: lần thứ nhất giới hạn độ sâu = 1, nếu tìm không thấy tăng độ sâu giới hạn lên 2, ...*

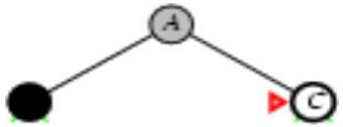
Tìm kiếm sâu lặp, l = 0

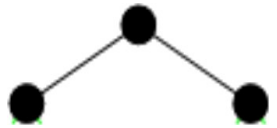
Limit = 0

Tìm kiếm sâu lặp, l = 1

Limit = 1

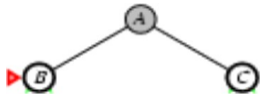


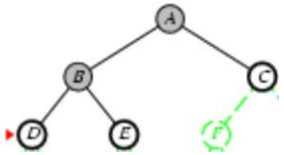


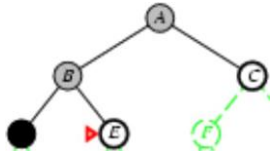


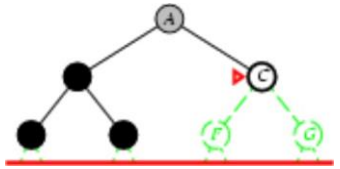
Tìm kiếm sâu lặp, l = 2

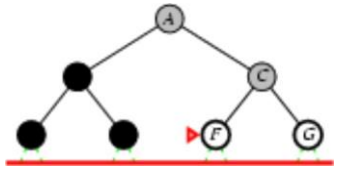
- Limit = 2

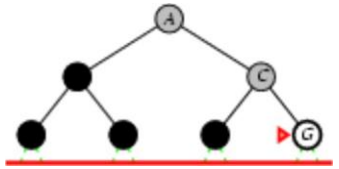


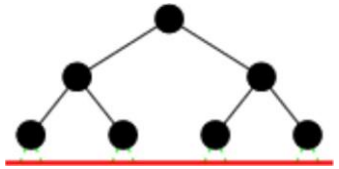




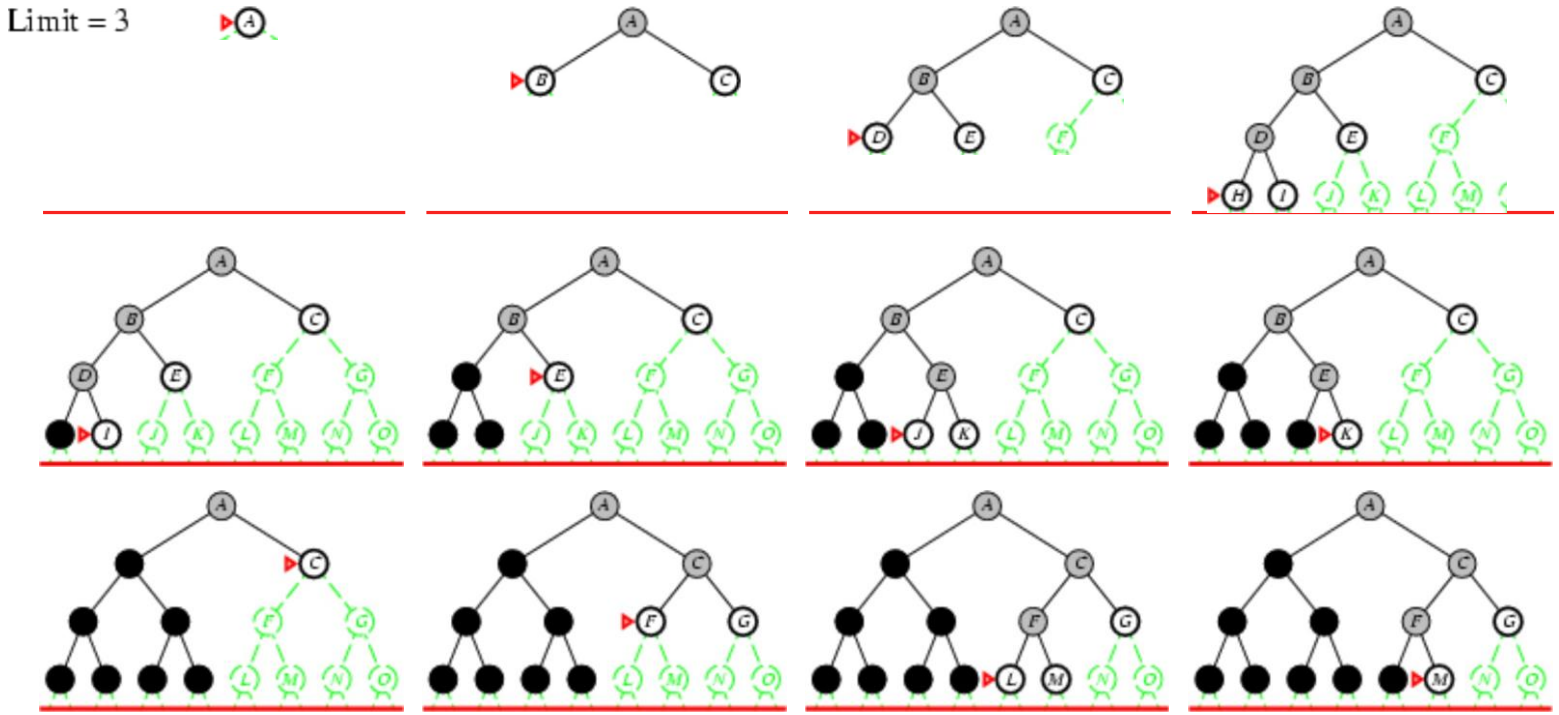








Tìm kiếm sâu lặp, l = 3



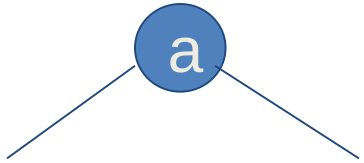
Tìm kiếm giá thành đồng nhất (Uniform-cost search)

Procedure Uniform-Cost–Search;
Begin % khởi đầu
PQ:= [start]; Closed:= [ ]; (PQ: Priority Queue)
While PQ ≠ [ ] do % còn các trạng thái chưa khảo sát
Begin
Lấy một trạng thái n (có chi phí thấp nhất) ra khỏi PQ. Đưa n vào CLOSE.
If n là Goal then trả lời kết quả (thành công) % Giải thuật dừng
else begin
Phát sinh các con n' của n;
Tính chi phí cho từng trạng thái con theo công thức:
g(n') = g(n) + cost(n', n)
If các con n' không có tồn tại trong PQ hoặc Closed then:
Thêm n' vào PQ (kèm theo chi phí)
else:
if node trong PQ có chi phí lớn hơn node đang xét:
Thay thế node trong PQ bởi node đang xét
End;
Trả lời kết quả (thất bại); % không còn trạng thái nào
End;

Tìm kiếm giá thành đồng nhất (Uniform-cost search)

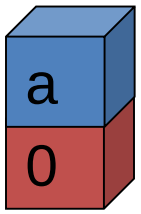
- Tìm kiếm theo chiều rộng
  - *Đảm bảo tìm ra giải pháp cho bài toán*
  - *Không chắc tìm ra **đường đi chi phí thấp nhất***
- Tìm kiếm giá thành đồng nhất (rất giống GT Dijsktra)
  - *Chọn nút có **giá thành đường đi** đến nó thấp nhất mà triển khai*
  - *Đảm bảo tìm được giải pháp với chi phí thấp nhất nếu biết rằng chi phí tăng khi chiều dài đường đi tăng*
  - *Tối ưu và đầy đủ*
  - *Nhưng có thể rất chậm*

Ví dụ Uniform Cost Search

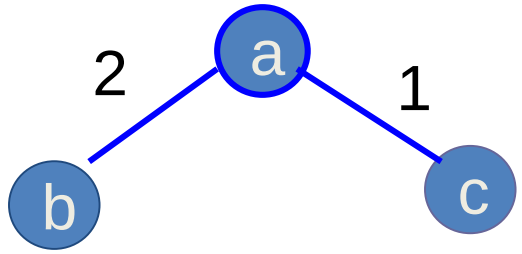


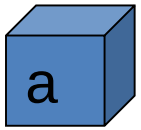
Closed list:

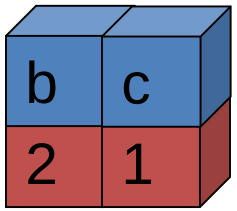
Open list:



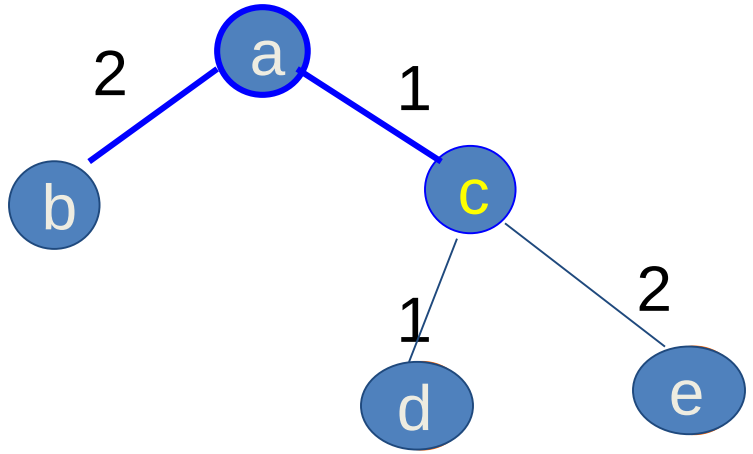
Ví dụ Uniform Cost Search



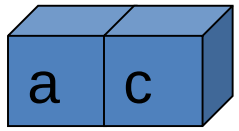
Closed list: 

Open list: 

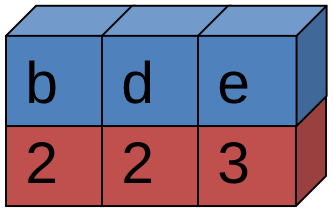
Ví dụ Uniform Cost Search



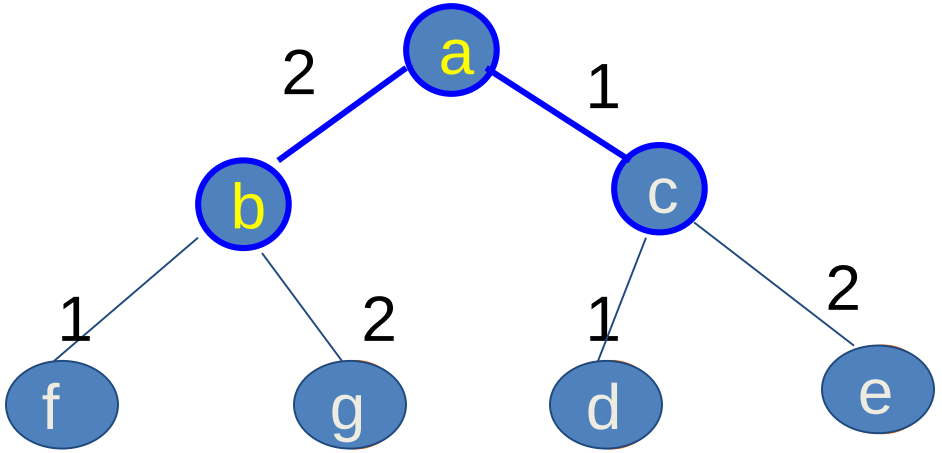
Closed list:



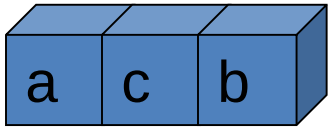
Open list:



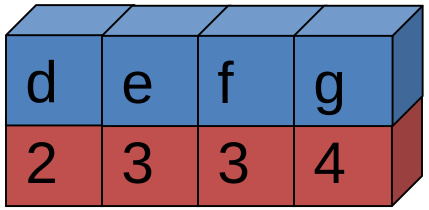
Ví dụ Uniform Cost Search



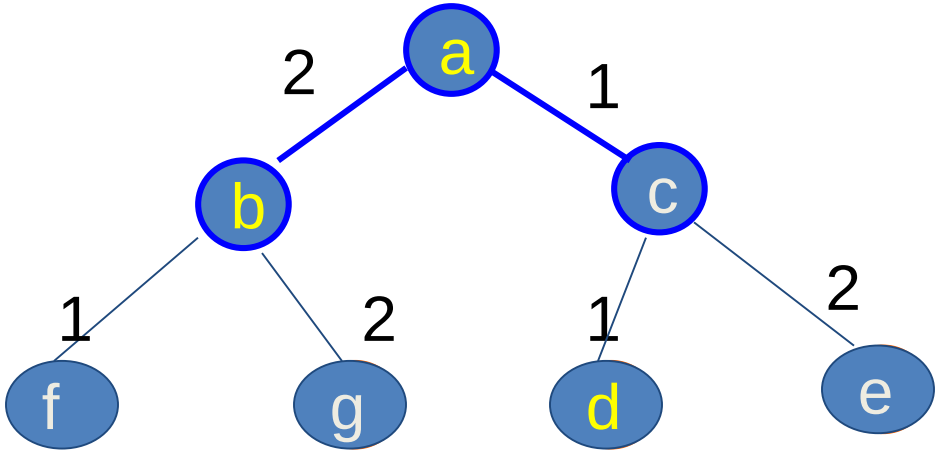
Closed list:



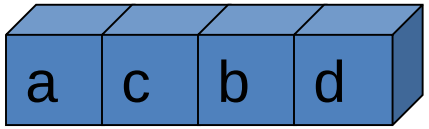
Open list:



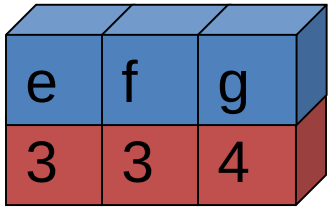
Ví dụ Uniform Cost Search



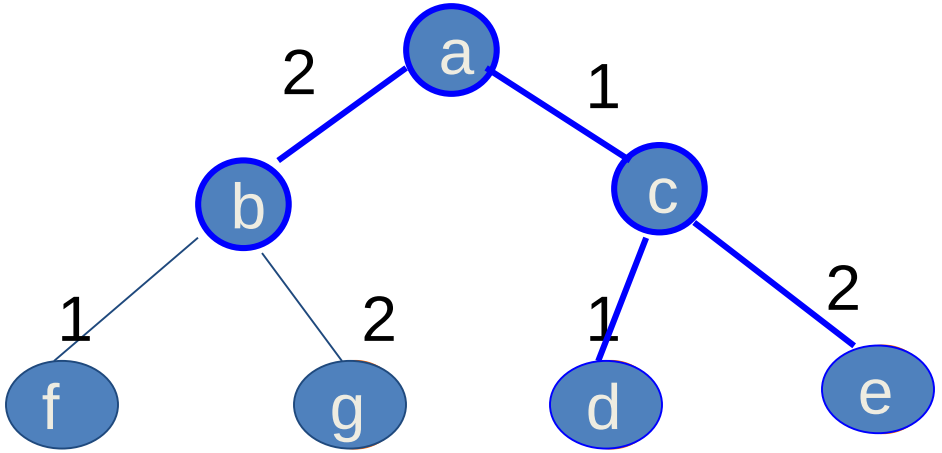
Closed list:



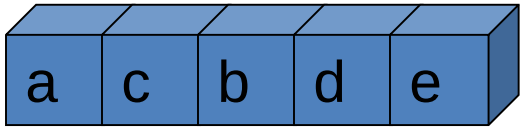
Open list:



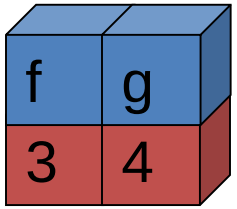
Ví dụ Uniform Cost Search



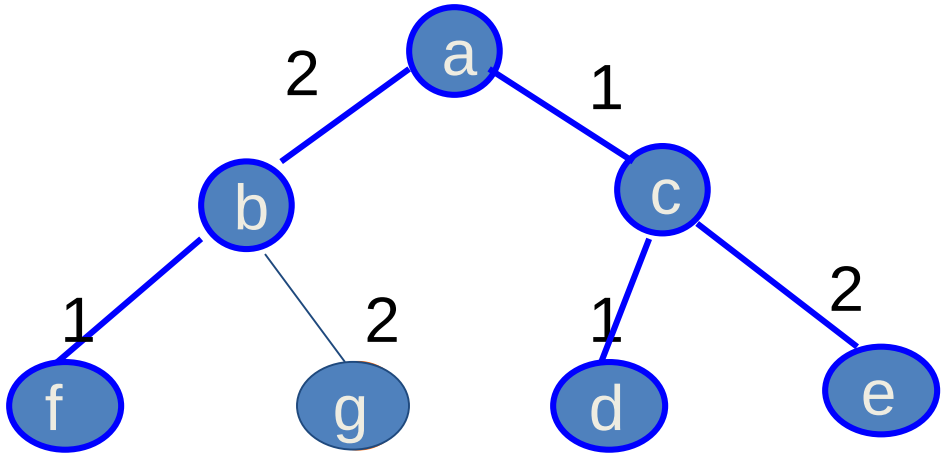
Closed list:



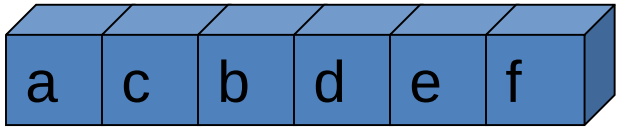
Open list:



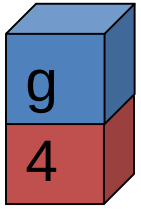
Ví dụ Uniform Cost Search



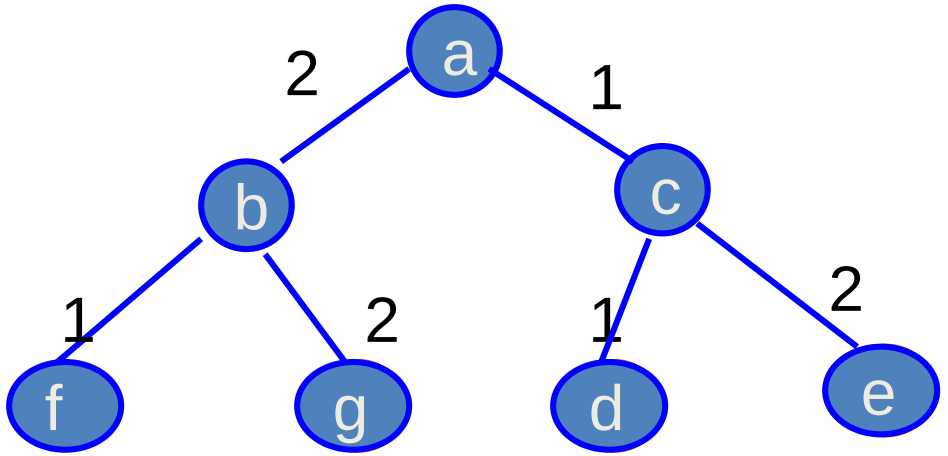
Closed list:

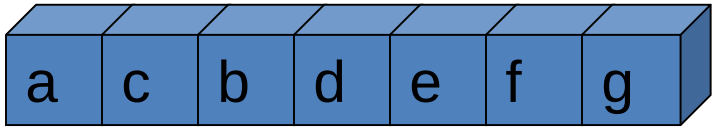


Open list:



Ví dụ Uniform Cost Search



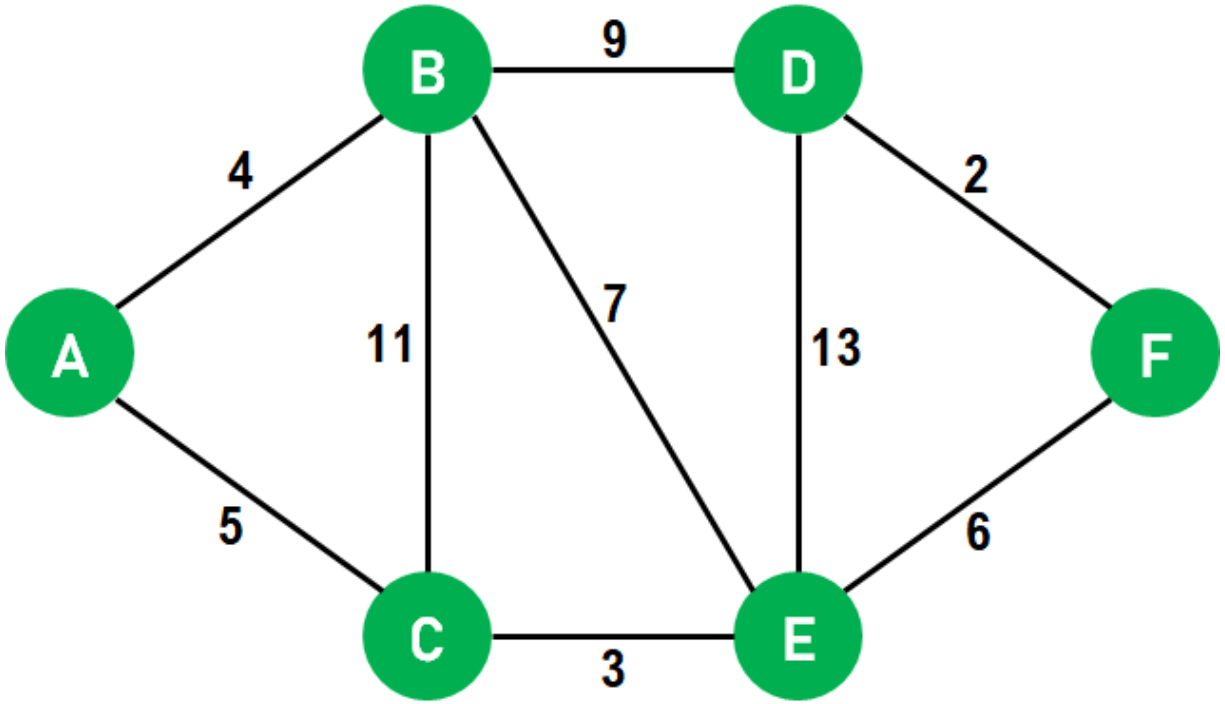
Closed list: 

Open list:

Bài tập 01

Cho bài toán sau: Bài toán tìm đường

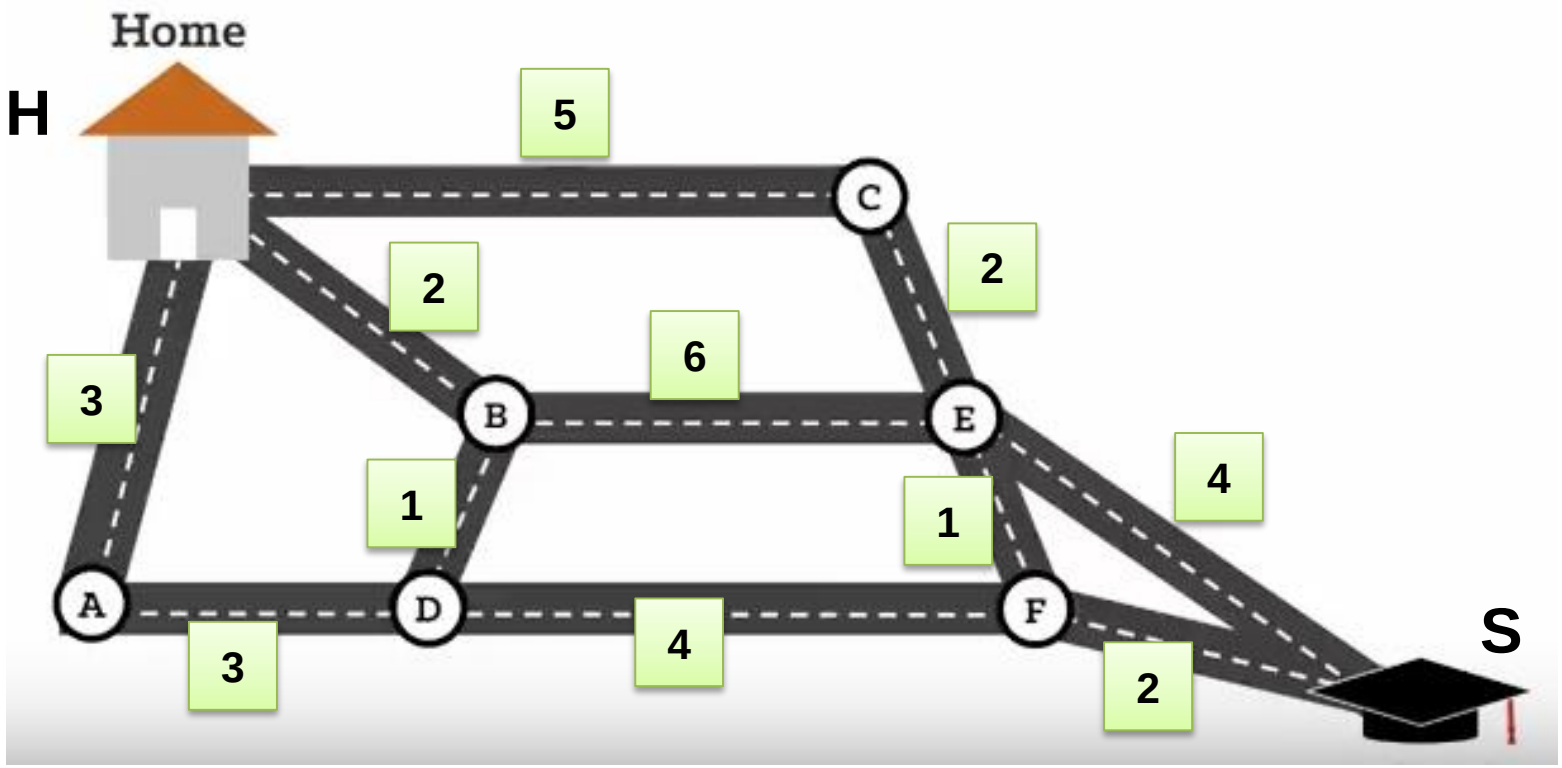
Cho bản đồ đường đi như hình bên dưới, các điểm A, B, C,… là các địa điểm với điểm xuất phát là A và điểm kết thúc là F. Biết giá trị ghi trên các cạnh của đồ thị là chi phí di chuyển giữa 2 địa điểm. Hãy áp dụng giải thuật UCS để tìm đường đi từ A đến F. Vẽ cây trạng thái bài toán.



Bài tập 02

Cho bài toán sau: Bài toán tìm đường từ nhà đến trường

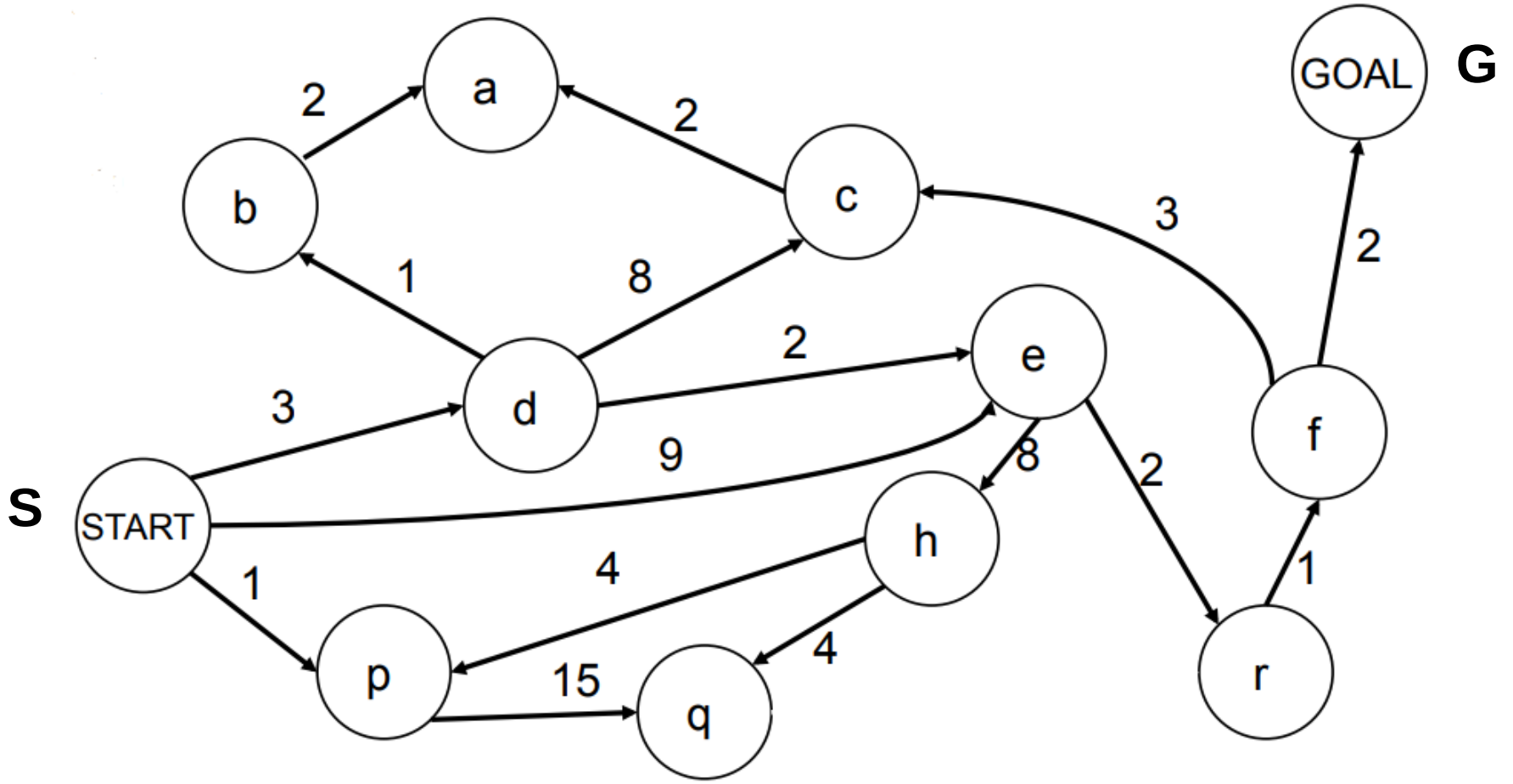
Cho bản đồ như hình bên dưới, các điểm A, B, C,… là các địa điểm đi qua với điểm xuất phát là tại nhà (Home) và điểm kết thúc là tại trường (School). Biết giá trị ghi kế các con đường là chi phí di chuyển giữa 2 địa điểm. Hãy áp dụng giải thuật UCS để tìm đường đi từ nhà đến trường. Vẽ cây trạng thái bài toán.



Bài tập 03

Cho bài toán sau: Bài toán tìm đường

Cho bản đồ đường đi như hình bên dưới, các điểm a, b, c,… là các địa điểm với điểm xuất phát là Start và điểm kết thúc là Goal. Biết giá trị ghi trên các cạnh của đồ thị là chi phí di chuyển giữa 2 địa điểm. Hãy áp dụng giải thuật UCS để tìm đường đi từ Start đến Goal. Vẽ cây trạng thái bài toán.




The End