# TÌM KIẾM MÙ (UNINFORMED BLIND SEARCH)

29

## 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

30

30

# Tìm kiếm rộng (breath-first search - BFS)

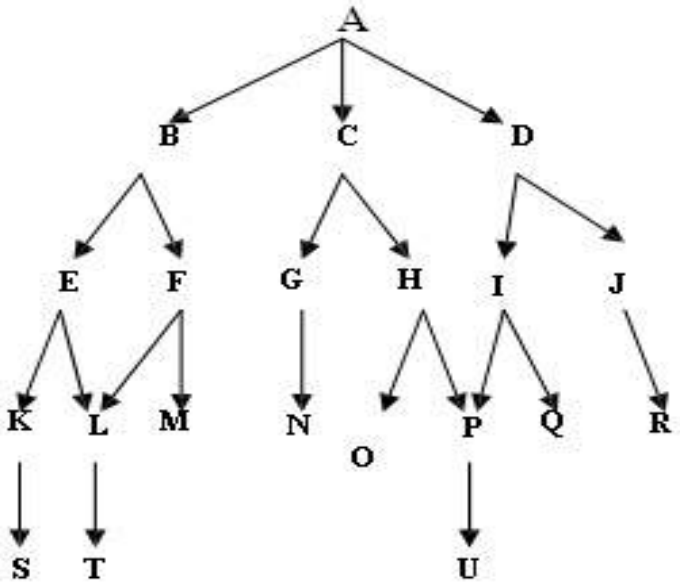
```
Procedure breadth-first-search;
  Begin                  % khởi đầu
     Open:= [start];                    Closed:= [ ];
     While open ≠ [ ] do     % còn các trạng thái chưa khảo sát
     Begin
     Loại bỏ trạng thái ngoài cùng bên trái khỏi open, gọi nó là X;
     If X là một đích then trả lời 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;
```

31

31

# Tìm kiếm mù

- Xét KGTT sau:



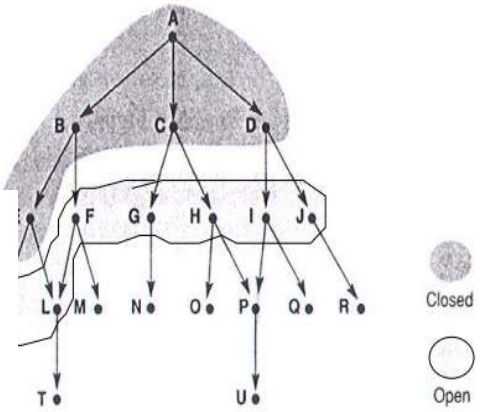

32

32

# Tìm kiếm rộng (breath-first search - BFS)

- Tìm kiếm rộng:
- Các bước thực hiện tìm kiếm rộng:
  1. *Open = [A]; closed = []*
  2. *Open = [B,C,D]; closed = [A]*
  3.

```
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;
```

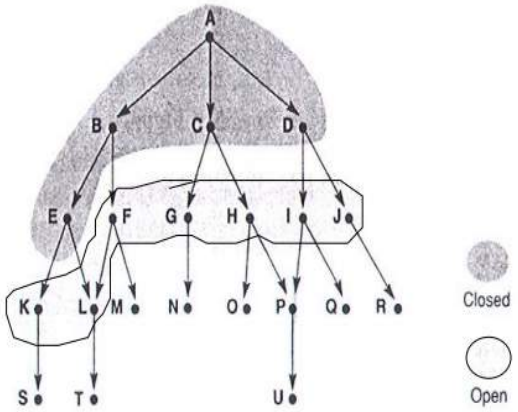



30 33

33

# Tìm kiếm rộng (breath-first search - BFS)

- Tìm kiếm rộng:
- Các bước thực hiện tìm kiếm rộng:
  1. *Open = [A]; closed = []*
  2. *Open = [B,C,D]; closed = [A]*
  3. *Open = [C,D,E,F];closed = [B,A]*
  4. *Open = [D,E,F,G,H];*
     *closed = [C,B,A]*
  5. *Open = [E,F,G,H,I,J];*
     *closed = [D,C,B,A]*
  6. *Open = [F,G,H,I,J,K,L];*
     *closed = [E,D,C,B,A]*
  7. *Open = [G,H,I,J,K,L,M];*
     *(vì L đã có trong open);*
     *closed = [F,E,D,C,B,A]*
  8. *…*

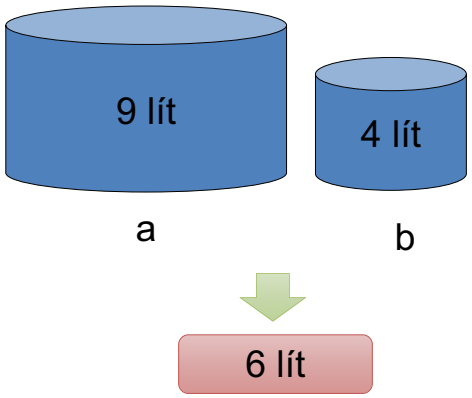



34

34

# Ví dụ: Bài toán đong nước

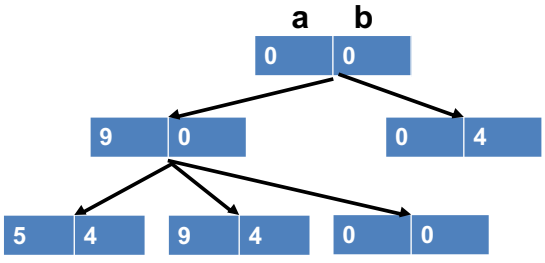
| | a | b |
|---|---|---|
| **Start** | 0 | 0 |
| 1 | | |

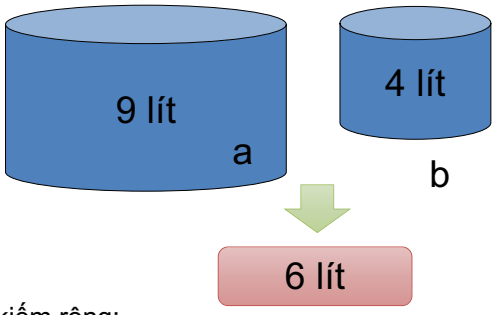



35

35

# Ví dụ: Bài toán đong nước



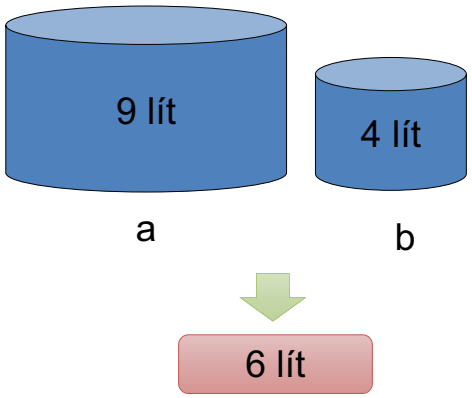

- Tìm kiếm rộng:
- Các bước thực hiện tìm kiếm rộng:
  1. *Open = [A: 0-0]; closed = []*
  2. *Open = [B: 9-0,C: 0-4]; closed = [A: 0-0]*
  3. *Open = [C:0-4,D:5-4,E:9-4, F: 0-0]; closed = [B,A]*
  4. *…..*

36

36

# Ví dụ: Bài toán đong nước

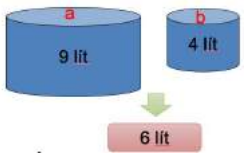
| | a | b |
|---|---|---|
| **Start** | 0 | 0 |
| 1 | | |

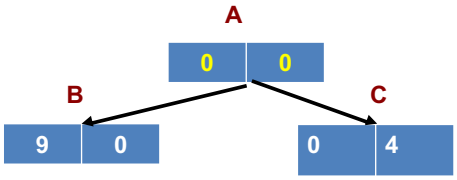



35

35

# 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

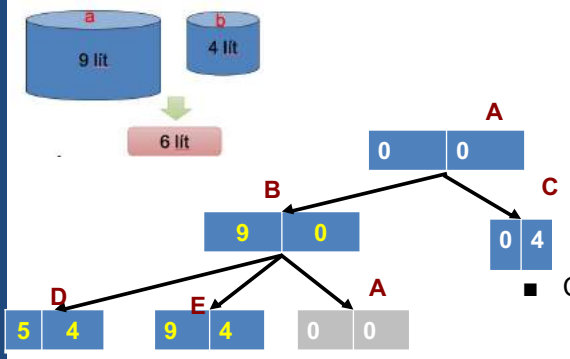
- Các bước thực hiện tìm kiếm rộng:
  - ***Bước 1:*** *Open = [A: 0-0]; closed = []*
  - *Xét A, A ko phải trạng thái đích*
  - *Đưa A vào closed: closed=[A:0-0]*
  - *Xét các con của A là B:9-0 và C:0-4*
  - *Các con của A ko tồn tại trong open-closed*
  - *Đưa các con của A và bên phải open*

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

36

36

# 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]*

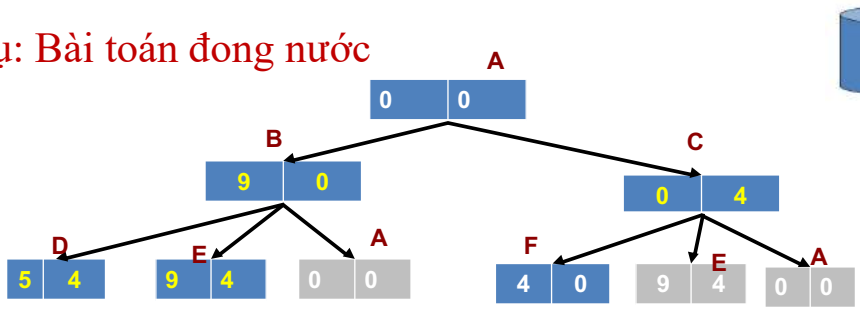
- *Xét B:9-0, B không là trạng thái đích*
- *Đưa B vào Closed: closed[A,B]*
- *Xét các con của B: C:0-4,D:5-4, A:0-0*
- *Loại các con đã tồn tại trong open và closed – loại A:0-0*
- *Thêm các con còn lại vào open*

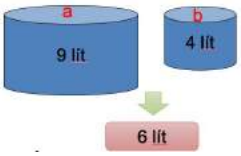
*Open = [C:0-4,D:5-4,E:9-4,]; closed = [A,B]*

37

37

# 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]*

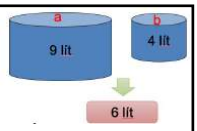
- *Xét C, ko phải trạng thái đích*
- *Đưa C vào Closed*
- *Các con của C: F: 4-0, E:9-4, A:0-0*
- *Loại các con đã tồn tại trong open và closed – loại A:0-0 và E:9-4*
- *Thêm các con còn lại vào open*

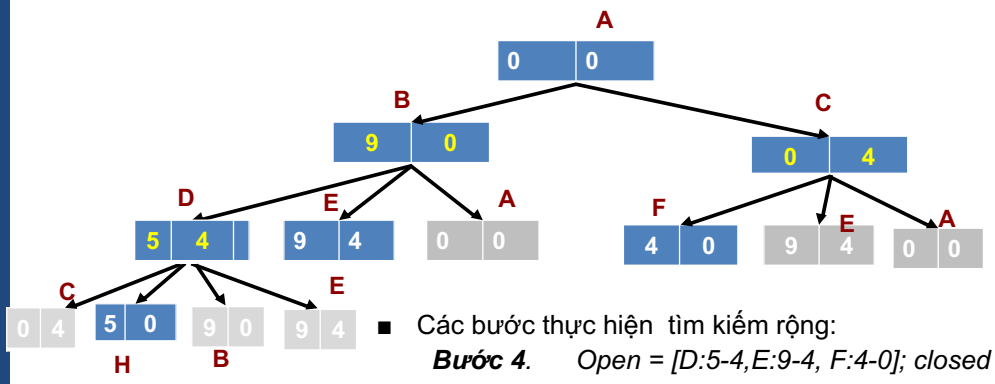
*Open = [D:5-4,E:9-4, F:4-0]; closed = [A,B,C]*

38

38

# 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*

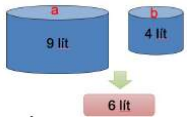
- *Đưa D vào Closed: closed = [A,B,C,D]*
- *Các con của D: C: 0-4, H:5-0, B:9-0, E:9-4*
- *Loại các con đã tồn tại trong open và closed –*
- *loại C: 0-4, B:9-0, E:9-4*
- *Thêm các con còn lại vào open*

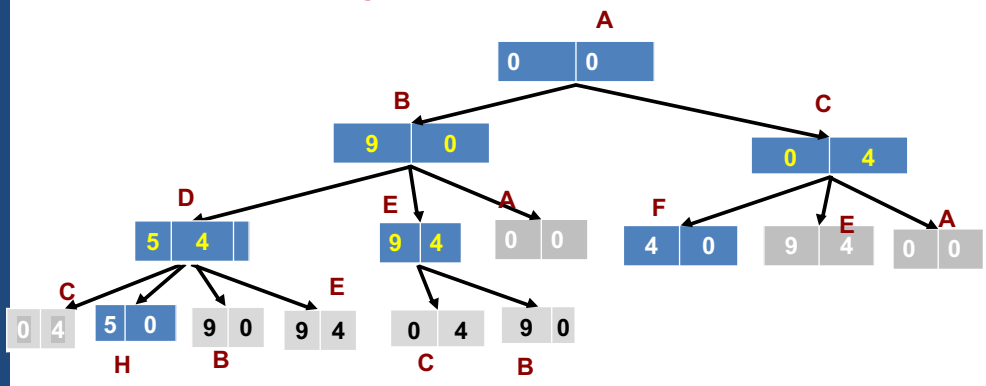
*Open = [E:9-4, F:4-0 , H:5-0, ]; closed = [A,B,C,D]*

39

39

# Ví dụ: Bài toán đong nước




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

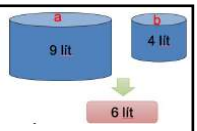
*Xét E, ko phải trạng thái đích*

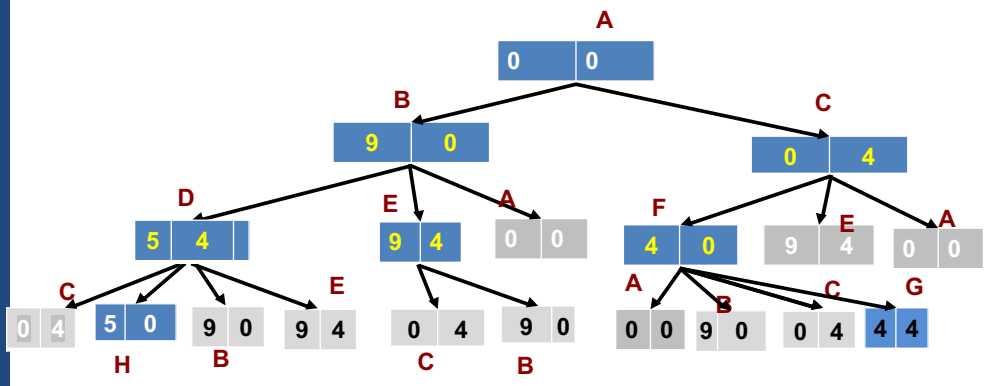
- *Đưa E vào Closed: closed = [A,B,C,D,E]*
- *Các con của E: C: 0-4, B:9-0*
- *Loại các con đã tồn tại trong open và closed – loại : C: 0-4, B:9-0*
- *Open = [F:4-0, H:5-0, ]; closed = [A,B,C,D,E]*

40

40

## Ví dụ: Bài toán đong nước



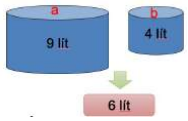

***Bước 6:*** *[F:4-0, H:5-0, ]; closed = [A,B,C,D,E]*
*Xét F, ko phải trạng thái đích*

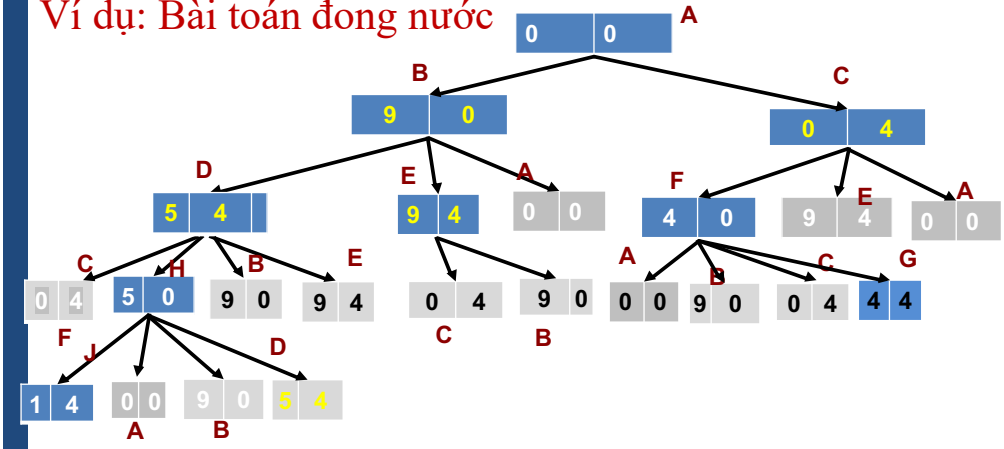
- *Đưa F vào Closed: closed = [A,B,C,D,E,F]*
- *Các con của F: A:0-0,B:9-0, C: 0-4, G:4-4*
- *Loại các con đã tồn tại trong open và closed – loại : A:0-0,B:9-0, C: 0-4,*
- *Open = [H:5-0, G:4,4 ]; closed = [A,B,C,D,E,F]*

41

41

## Ví dụ: Bài toán đong nước



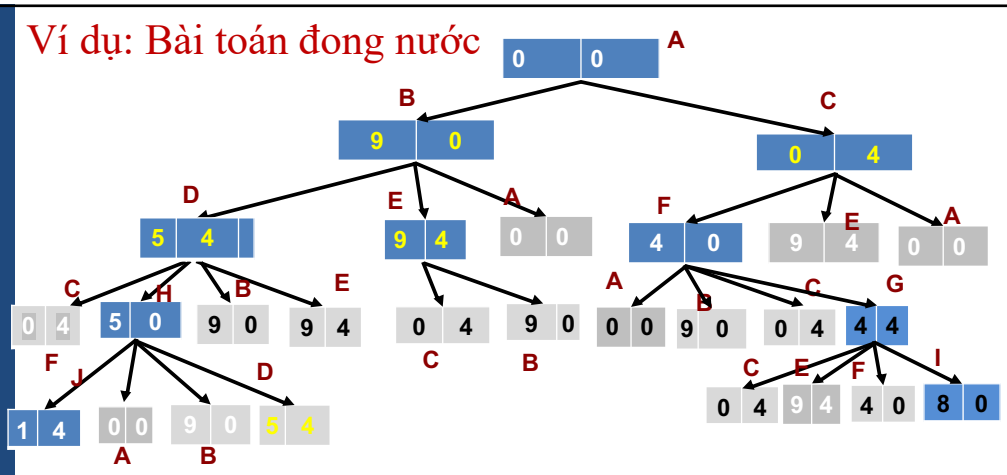



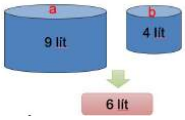

- ***Bước 7:*** *Open = [H:5-0, G:4,4 ]; closed = [A,B,C,D,E,F]*
  *Xét H, ko phải trạng thái đích*
- *Đưa H vào Closed: closed = [A,B,C,D,E,F,H]*
- *Các con của H: A: 0-0, B:9-0, D:5-4, J:1-4*
- *Loại các con đã tồn tại trong open và closed – loại : A: 0-0, B:9-0, D:5-4,*
- *Thêm J:1-4 vào Open = [G:4-4,J:1-4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

42

42

## Ví dụ: Bài toán đong nước



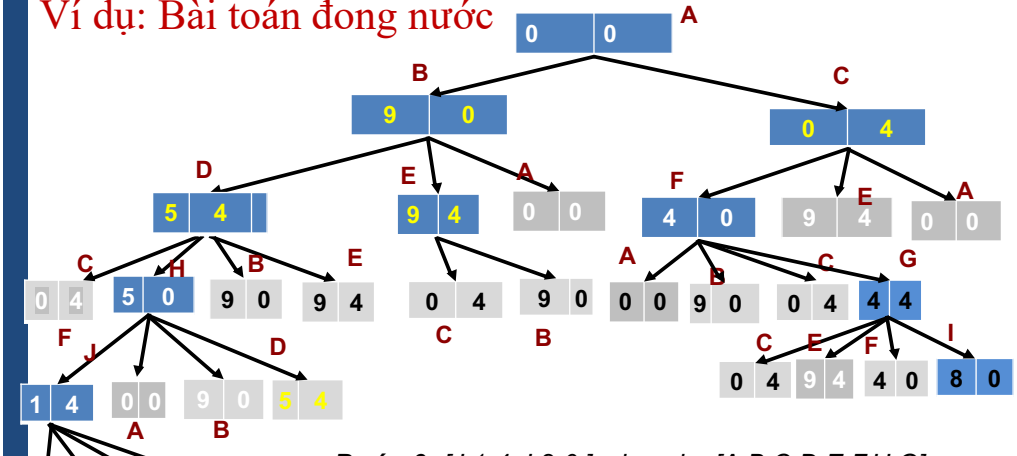



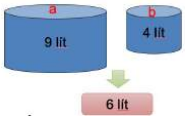

- ***Bước 8:*** *[G:4-4,J:1-4 ]; closed = [A,B,C,D,E,F,H*
- *Xét G, ko phải trạng thái đích*
- *Đưa G vào Closed: closed = [A,B,C,D,E,F,H,G]*
- *Các con của G: C: 0-4, E:9-4, F:4-0, I:8-0*
- *Loại các con đã tồn tại trong open và closed – loại : C: 0-4, E:9-4, F:4-0*
- *Thêm I:8-0 vào Open = [J:1-4,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

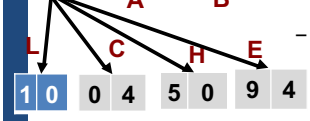
43

43

## Ví dụ: Bài toán đong nước



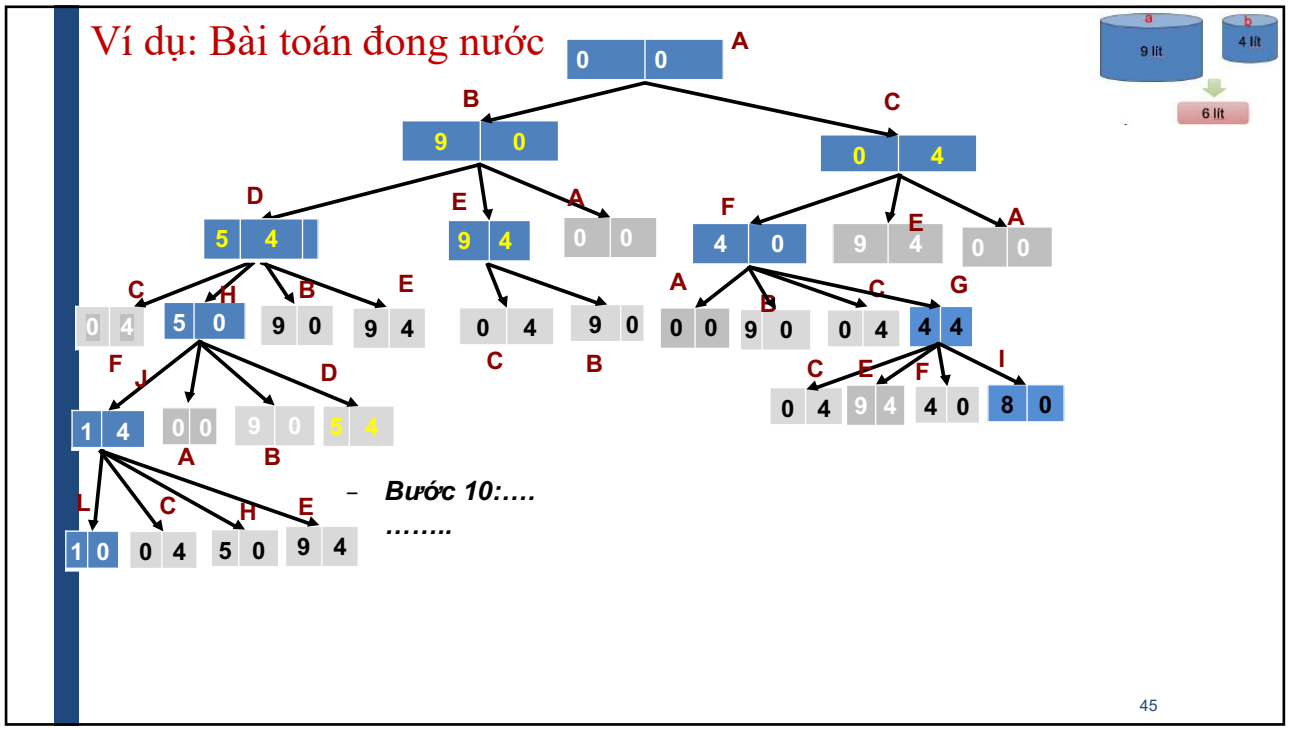



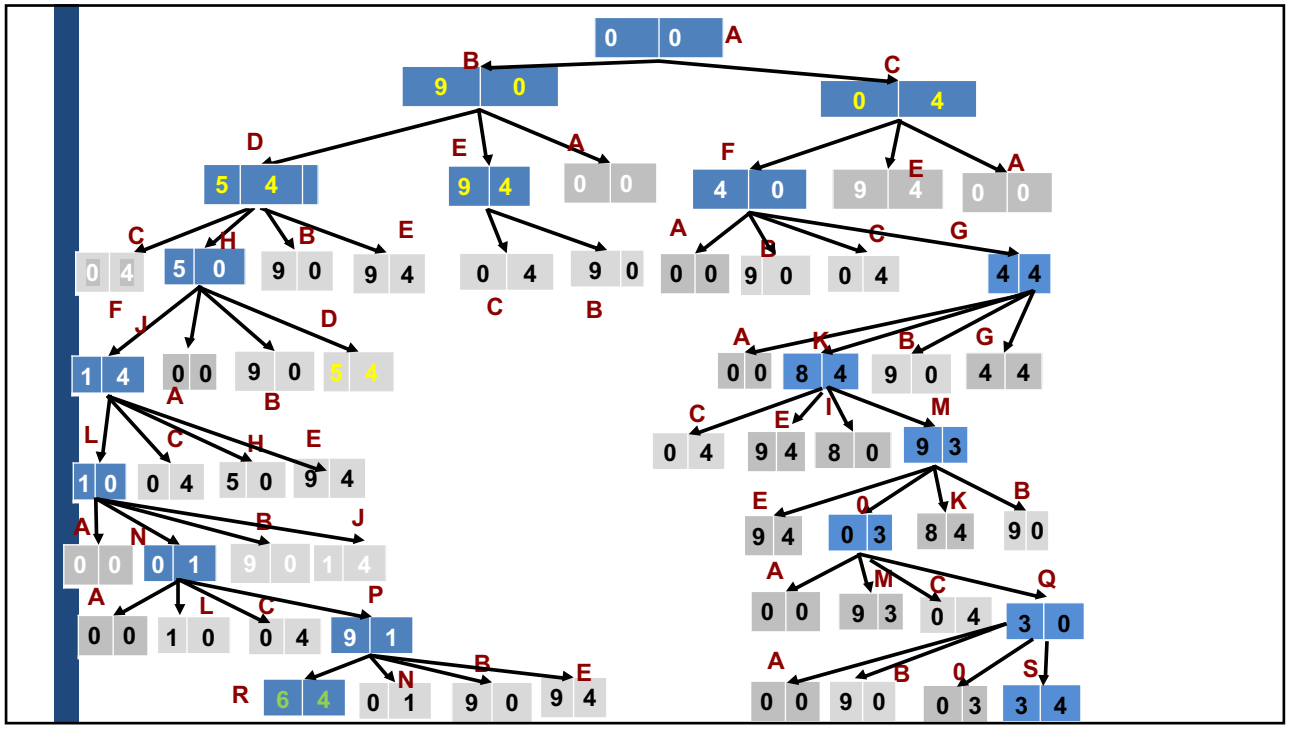

- ***Bước 9:*** *[J:1-4, I:8-0 ]; closed = [A,B,C,D,E,F,H,G]*
  *Xét J, ko phải trạng thái đích*
  *Đưa J vào Closed: closed = [A,B,C,D,E,F,H,G,J]*
- *Các con của J: L: 1-0, C:0-4, H:5-0, E:9-4*
- *Loại các con đã tồn tại trong open và closed – loại C:0-4, H:5-0, E:9-4*
- *Thêm L: 1-0, vào Open = [I:8-0,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

44

44




45



46

# Tìm kiếm sâu (depth-first search)

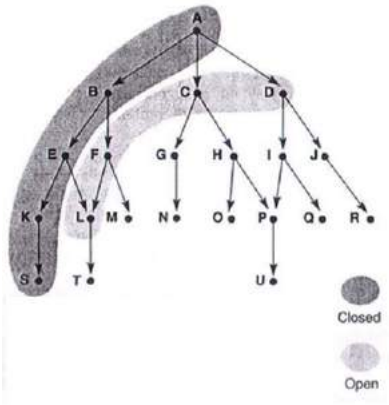
```
Procedure depth – first –search;
  Begin                  % khởi đầu
    Open:= [start];      Closed:= [ ];
    While open ≠ [ ] do  % còn các trạng thái chưa khảo sát
        Begin
          Loại bỏ trạng thái ngoài cùng bên trái khỏi open, gọi nó là X;
          If X là một đích then trả lời kết 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 lại vào đầu bên trái của open; %ngăn xếp
                 end;
          End;
        Trả lời kết quả (thất bại);      % không còn trạng thái nào
       End;
```

47



# Tìm kiếm sâu (depth-first search - DFS)

- Tìm kiếm sâu:
- Các bước thực hiện tìm kiếm sâu:
  1. Open = [A]; closed = []
  2. Open = [B,C,D]; closed = [A]
  3. Open = [E,F,C,D];closed = [B,A]
  4. Open = [K,L,F,C,D];
     closed = [E,B,A]
  5. Open = [S,L,F,C,D];
     closed = [K,E,B,A]
  6. Open = [L,F,C,D];
     closed = [S,K,E,B,A]
  7. Open = [T,F,C,D];
     closed = [L,S,K,E,B,A]
  8. Open = [F,C,D];
     closed = [T,L,S,K,E,B,A]




48

# Sử dụng BFS, DFS để tìm trạng thái đích

| | |
|---|---|
| 3 | 1 |
| 2 | |

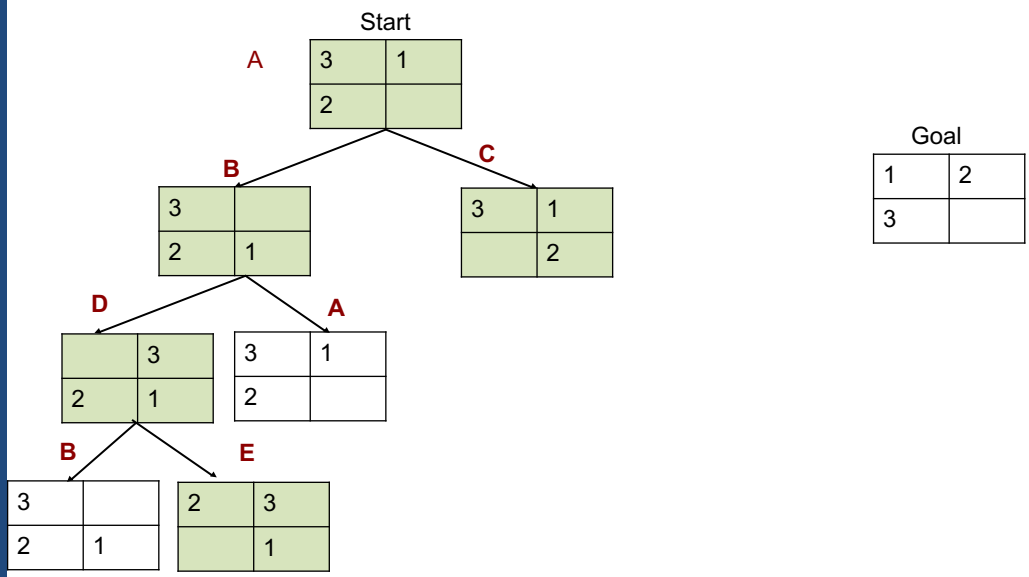
Start

| | |
|---|---|
| 1 | 2 |
| 3 | |

Goal

49

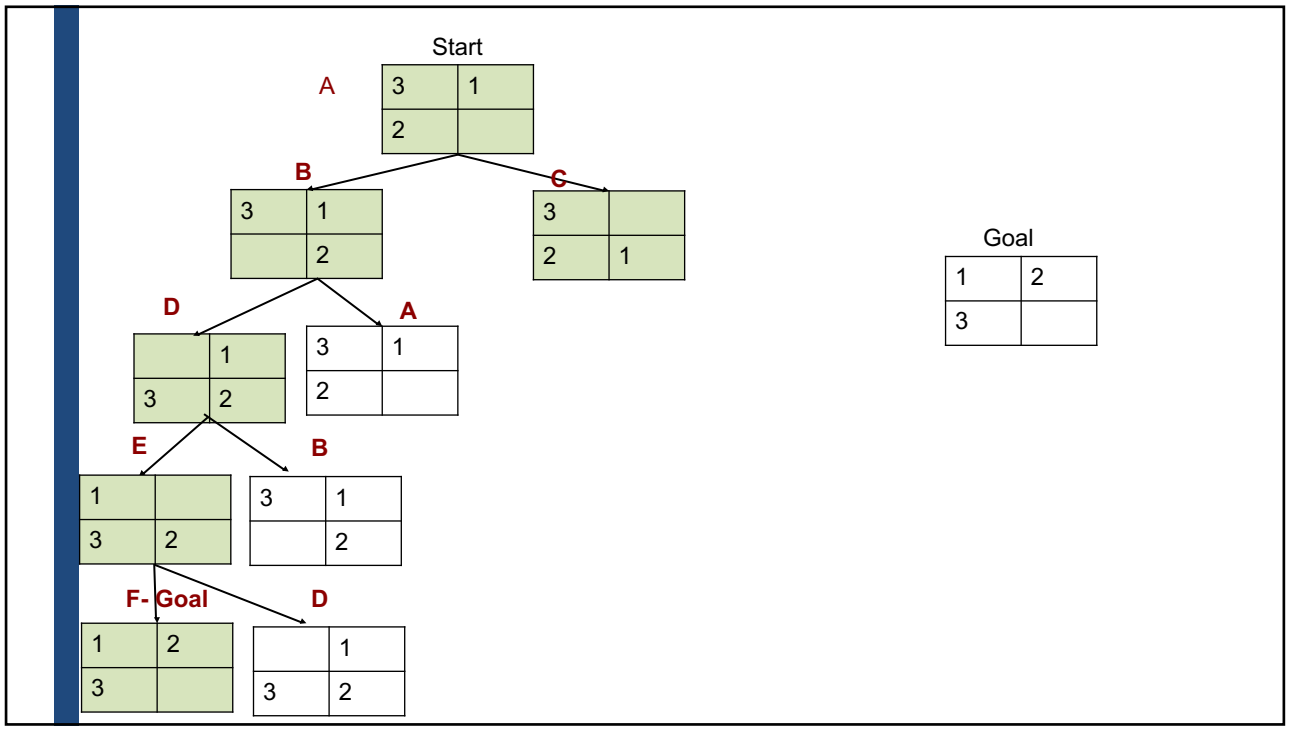
# Sử dụng BFS, DFS để tìm trạng thái đích




Goal

| | |
|---|---|
| 1 | 2 |
| 3 | |

50




51

# Sử dụng BFS, DFS để tìm trạng thái đích

Start

| | |
|---|---|
| 3 | 1 |
| 2 | |

| | |
|---|---|
| 1 | 2 |
| 3 | |

Goal

```
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

52

# Sử dụng BFS, DFS để tìm trạng thái đích

Start

| | |
|---|---|
| 3 | 1 |
| 2 | |

| | |
|---|---|
| 1 | 2 |
| 3 | |

Goal

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;
```

53

# 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

54

## 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, ...*

55

55

## Tìm kiếm sâu lặp, l = 0

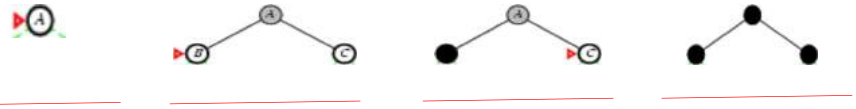
Limit = 0


56

56

# Tìm kiếm sâu lặp, l = 1

Limit = 1

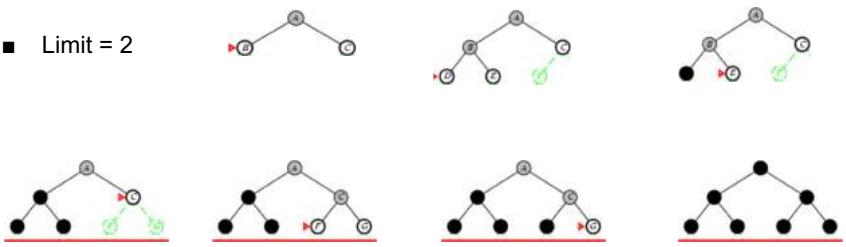


57

57

# Tìm kiếm sâu lặp, l = 2

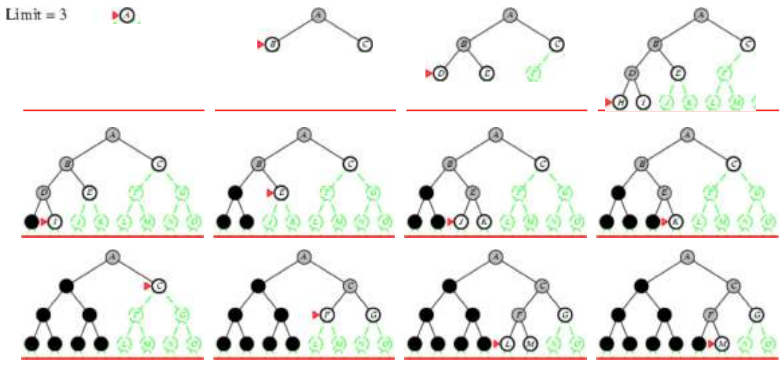
- Limit = 2



58

58

# Tìm kiếm sâu lặp, l = 3




59



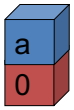
# Tìm kiếm giá thành đồng nhất (Uniform-cost search)

- 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*

60

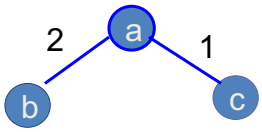
# Ví dụ Uniform Cost Search


***Closed list:***

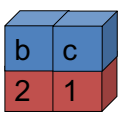
***Open list:*** 


61

# Ví dụ Uniform Cost Search

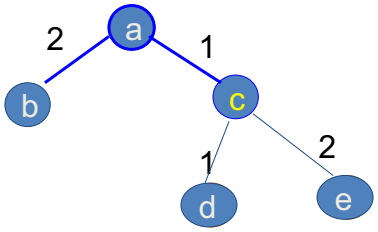



***Closed list:*** 

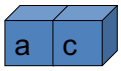

***Open list:*** 


62

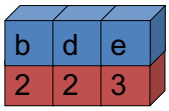
# Ví dụ Uniform Cost Search




***Closed list:***

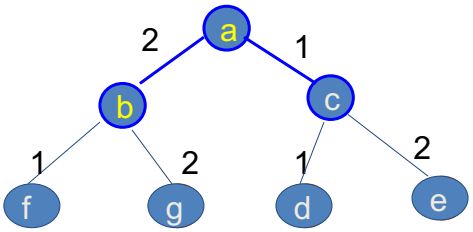



***Open list:***

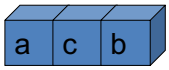



63

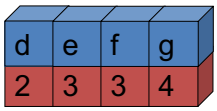
# Ví dụ Uniform Cost Search




***Closed list:***

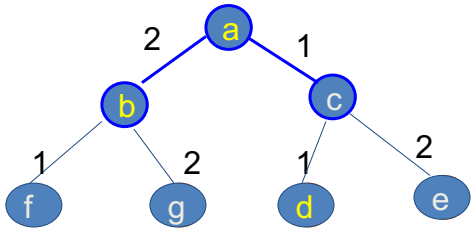



***Open list:***




64

# Ví dụ Uniform Cost Search



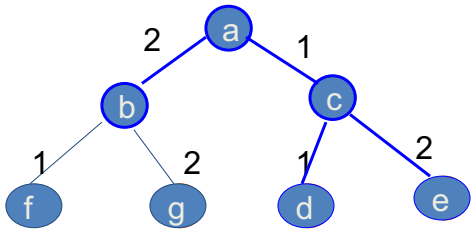

***Closed list:*** a c b d

***Open list:***

| e | f | g |
|---|---|---|
| 3 | 3 | 4 |

65

# Ví dụ Uniform Cost Search



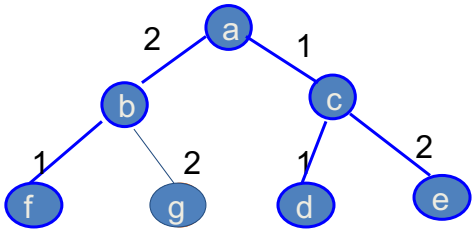

***Closed list:*** a c b d e

***Open list:***

| f | g |
|---|---|
| 3 | 4 |

66

# Ví dụ Uniform Cost Search

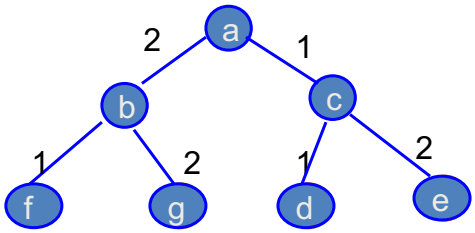



***Closed list:*** a c b d e f

***Open list:*** g 4

67

# Ví dụ Uniform Cost Search



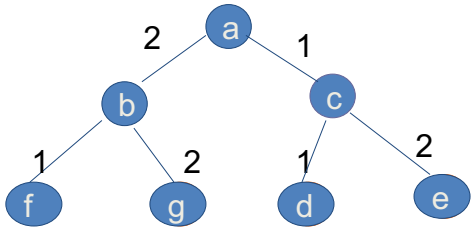

***Closed list:*** a c b d e f g

***Open list:***

68

# Ví dụ Uniform Cost Search

- Cho cây tìm kiếm với chi phí như sau:




Phân biệt:

- nút được sinh ra

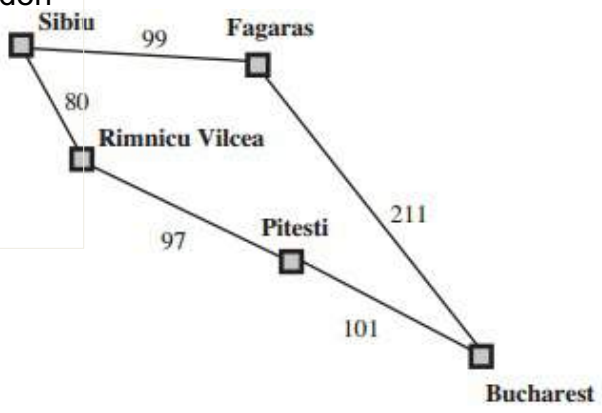
  nút được mở rộng (duyệt qua)

69

# Ví dụ Uniform Cost Search

Tìm đường đi ngắn nhất từ Sibiu đến Bucharest?

B1:
Open list: [Sibiu]
Closed list: []



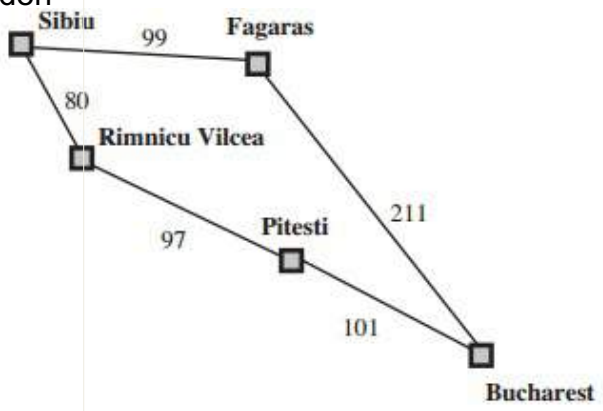

70

70

# Ví dụ Uniform Cost Search

Tìm đường đi ngắn nhất từ Sibiu đến Bucharest?

B1:
Open list: [Sibiu]
Closed list: []
B2:
Open list: [80 R, 99 F]
Closed list: [Sibiu]
B3:
Open list: [99 F, 177 P]
Closed list: [Sibiu, R]
B4:
Open list: [177P, 310B]
Closed list: [Sibiu, R, F]
B5:
Open list: [278B, 310B]
Closed list: [Sibiu, R, F, P]




71

71