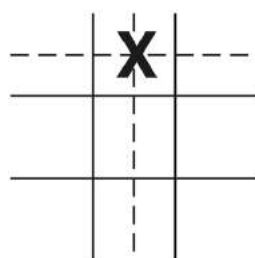
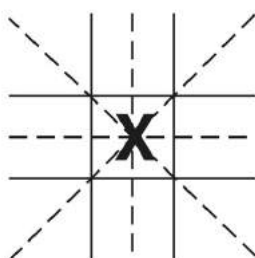
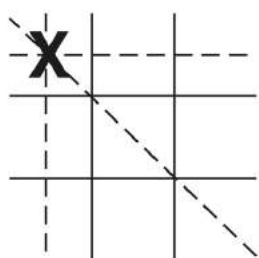


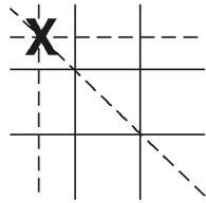
TÌM KIẾM DỰA TRÊN KINH NGHIỆM (INFORMED/ HEURISTIC SEARCH)

75

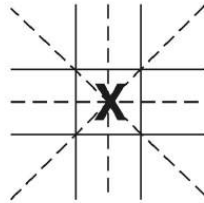


76

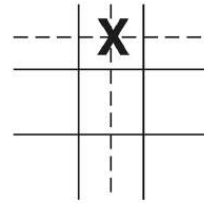
Phép đo heuristic (2)



Three wins through
a corner square



Four wins through
the center square



Two wins through
a side square

Heuristic “Số đường thắng nhiều nhất” áp dụng cho các nút con đầu tiên trong tic-tac-toe.

C 4 – Tìm kiếm Heuristic

TTNT. p.77

77

Heuristic

- Để giải quyết các bài toán lớn, phải cung cấp những kiến thức đặc trưng ở từng lĩnh vực để nâng cao hiệu quả của việc tìm kiếm
- Trong TK KGTT, heuristic là các luật dùng để chọn những nhánh/ lựa chọn nào có nhiều khả năng nhất dẫn đến một giải pháp chấp nhận được

78

78

Heuristic

■ Sử dụng Heuristic trong trường hợp nào?

- Vấn đề có thể có giải pháp chính xác, nhưng **chi phí tính toán** để tìm ra nó không cho phép. VD: cờ vua,...
- Vấn đề có thể **không có giải pháp chính xác** vì sự không rõ ràng trong diễn đạt vấn đề hoặc trong các dữ liệu có sẵn. VD: chẩn đoán y khoa,...

79

79

Heuristic

■ Thuật toán Heuristic gồm hai phần:

1. **Phép đo Heuristic:** thể hiện qua hàm đánh giá heuristic, dùng để đánh giá các đặc điểm của một trạng thái trong KGTT.
2. **Giải thuật tìm kiếm heuristic:**
 - Tìm kiếm tốt nhất đầu tiên (best-first search)
 - Tìm kiếm háu ăn (greedy best-first search)
 - Giải thuật A*
 - Tìm kiếm leo đồi

80

80

Heuristic

▪ Phép đo Heuristic:

- Ước lượng chi phí tối ưu giữa hai hoặc nhiều giải pháp
- Không quá tốn kém để tính toán
- **Được xác định cụ thể trong từng bài toán** (Ví dụ: đối với bài toán TSP, đánh giá khoảng cách giữa các thành phố sao cho chi phí là thấp nhất)

▪ Hàm đánh giá Heuristic tại trạng thái n là $f(n)$:

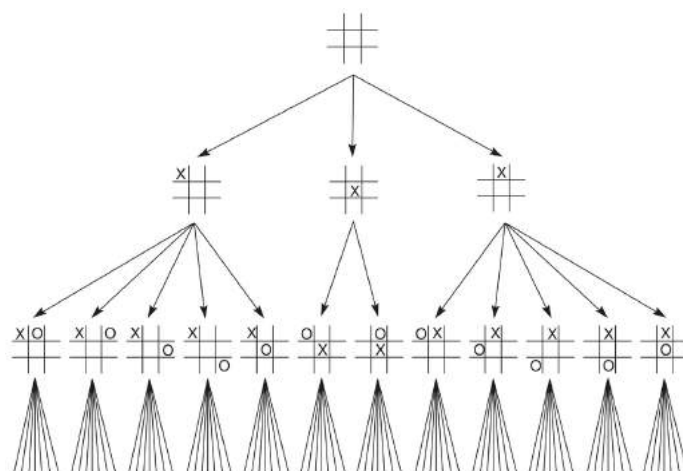
$$f(n) = g(n) + h(n)$$

- $g(n)$ = khoảng cách thực sự từ n đến trạng thái bắt đầu
- $h(n)$ = ước lượng heuristic cho khoảng cách từ trạng thái n đến trạng thái đích

81

81

KGTT của tic-tac-toe được thu nhỏ nhờ tính đối xứng của các trạng thái

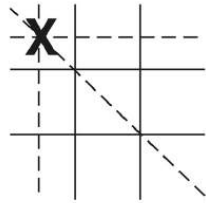


C 4 – Tìm kiếm Heuristic

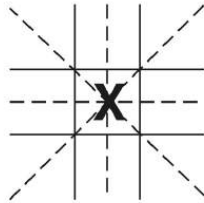
TTNT. p.82

82

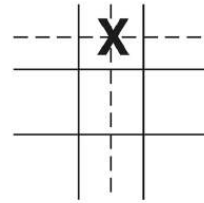
Phép đo heuristic (2)



Three wins through
a corner square



Four wins through
the center square



Two wins through
a side square

Heuristic “Số đường thắng nhiều nhất” áp dụng cho các nút con đầu tiên trong tic-tac-toe.

C 4 – Tìm kiếm Heuristic

TTNT. p.83

83

Cài đặt hàm đánh giá Heuristic

- Xét trò chơi 8-puzzle. Hàm đánh giá Heuristic tại trạng thái n là $f(n)$:

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

- $g(n)$ = khoảng cách thực sự từ n đến trạng thái bắt đầu
- $h(n)$ = ước lượng heuristic cho khoảng cách từ trạng thái n đến trạng thái đích

84

84

Cài đặt hàm đánh giá Heuristic

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

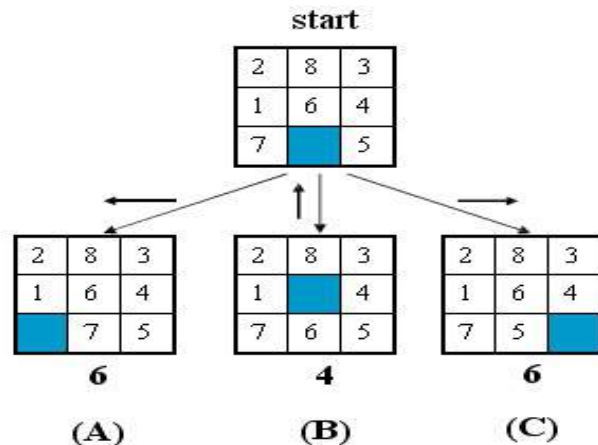
1	2	3
8		4
7	6	5

goal

$$g(n) = 0$$

$$g(n) = 1$$

$$f(n) =$$



85

Cài đặt hàm đánh giá Heuristic

- Việc ước lượng heuristic cho khoảng cách từ trạng thái n đến trạng thái đích như thế nào?
- Giả sử có các định nghĩa sau:
 - $h_1(n)$ = số vị trí sai khác của trạng thái n so với goal
 - $h_2(n)$ = khoảng cách dịch chuyển ($\leftarrow, \rightarrow, \uparrow, \downarrow$) ngắn nhất để dịch chuyển các ô chữ nằm sai vị trí về vị trí đúng (khoảng cách Manhattan)

$$\Rightarrow h_1(n) = ???$$

$$\Rightarrow h_2(n) = ???$$

START

6	4	2
3		5
0	1	7

GOAL

0	1	2
3	4	5
6	7	

86

86

Cài đặt hàm đánh giá Heuristic

- Việc ước lượng heuristic cho khoảng cách từ trạng thái n đến trạng thái đích như thế nào?
- Giả sử có các định nghĩa sau:
 - $h1(n)$ = số vị trí sai khác của trạng thái n so với goal
 - $h2(n)$ = khoảng cách Manhattan
- $\Rightarrow h1(n) = ???$
- $\Rightarrow h2(n) = ???$

■ $h1(n) = 6.$

■ $h2(n) = 8.$

6	4	2
3		5
0	1	7

START

0	1	2
3	4	5
6	7	

GOAL

87

Tìm kiếm tốt nhất đầu tiên (Best-first-search)

- **Ý tưởng:** sử dụng hàm đánh giá trong quá trình phát triển cây tìm kiếm, hàm đánh giá ước lượng độ gần của mỗi đỉnh trạng thái với trạng thái đích, chỉ triển khai cây tìm kiếm theo nhánh có triển vọng đi đến đích nhanh nhất
- *Tìm kiếm tốt nhất đầu tiên* (best-first search) là tiếp cận tổng quát của tìm kiếm với thông tin bổ sung.
- Việc chọn nút để triển khai dựa trên một *hàm lượng giá* (evaluation function): $f(n)$

88

88

Tìm kiếm tốt nhất đầu tiên (Best-first-search)

- *hàm lượng giá* (evaluation function): $f(n)$ được xây dựng dựa trên việc ước lượng chi phí và nút có giá ước lượng nhỏ nhất được ưu tiên triển khai trước. Sự lựa chọn hàm sẽ quyết định chiến lược tìm kiếm.
- Hàm $h(n)$ là *chi phí ước lượng* của đường đi tốt nhất từ trạng thái của nút đến trạng thái mục tiêu.
 - $f(n) = g(n) + h(n)$
 - $g(n)$: *chi phí thực tế* đi từ gốc đến n
 - $h(n)$: *chi phí ước lượng* đi từ n đến nút mục tiêu
- Hàm heuristic là cách thường dùng nhất để sử dụng kiến thức bổ sung trong quá trình tìm kiếm: $h(n)$ là một hàm không âm bất kỳ và *phụ thuộc vào vấn đề đang giải quyết*. Nếu là nút mục tiêu thì $h(n) = 0$.

89

89

Tìm kiếm tốt nhất đầu tiên (Best-first-search)

- **Dùng mảng có sắp xếp theo hàm đánh giá**
- Tương tự DFS và BFS, best-first search dùng các danh sách để lưu trữ các trạng thái:
 - *OPEN*: lưu các trạng thái sắp được kiểm tra
 - *CLOSED*: lưu các trạng thái đã duyệt qua
- **Các trạng thái trong OPEN list sẽ được sắp xếp theo thứ tự dựa trên 1 hàm Heuristic nào đó (đặt thứ tự ưu tiên cho các giá trị gần trạng thái đích)**
- Do đó, mỗi lần lặp sẽ xem xét trạng thái tiềm năng nhất trong OPEN list

90

90

Tìm kiếm tốt nhất đầu tiên (Best-first-search)

```

PNode bestFirstSearch(PState init_state) {
    PNode root = new Node();
    root->state = init_state;
    root->parent = NULL;
    root->f = 0;
    frontier.insert(root);
    explored.clear();
    while (!empty(frontier)) {
        //Lấy 1 nút từ đường biên có f(n) nhỏ nhất
        //và loại bỏ nó ra khỏi đường biên
        Node* node = frontier.pop();
        if (node là nút mục tiêu)
            return node;
        insert(node, explored);
    }
}

```

```

for (child là nút con của node) {
    Tính child->f;
    if (child->state không thuộc frontier và
        child->state không thuộc explored) {
        child->parent = node;
        frontier.insert(child);
    } else if (child->state nằm trong đường biên
        và có f lớn hơn child->f) {
        Thay thế nút nằm trên đường biên bằng child
    } else if (child->state nằm trong explored
        và có f lớn hơn child->f) {
        Loại bỏ nút chứa child->state
        ra khỏi explored
        frontier.insert(child);
    }
}
return NULL; //Thất bại, không tìm thấy lời giải
}

```

91

Sử dụng
tìm kiếm tốt
nhất đầu
tiên để tìm
trạng thái
goal

- Xét trò chơi 8-ô, mỗi trạng thái n, một giá trị $f(n)$:

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

- $g(n)$ = khoảng cách thực sự từ n đến trạng thái bắt đầu
- $h(n)$ = hàm heuristic đánh giá khoảng cách từ trạng thái n đến mục tiêu.

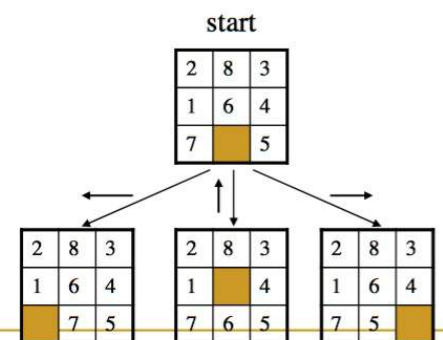
1	2	3
8		4
7	6	5

goal

$$g(n) = 0$$

$$h(n): \text{số lượng các vị trí còn sai}; g(n) = 1$$

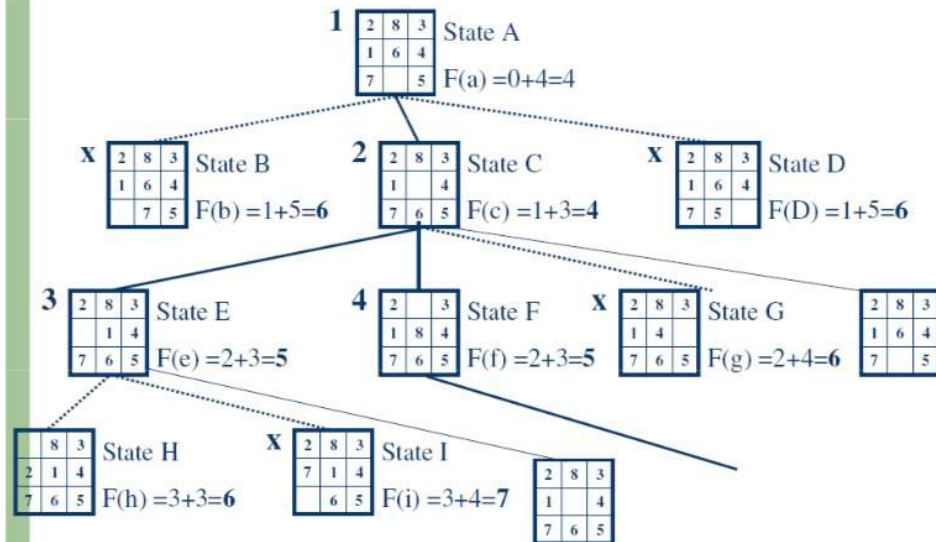
$$f(n) = \begin{matrix} 6 & 4 & 6 \end{matrix} \quad 21$$



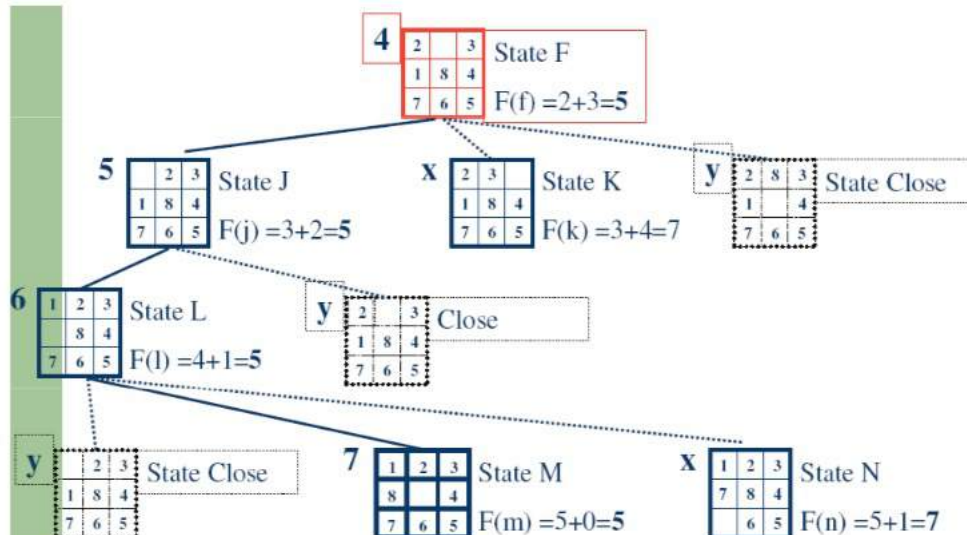
92

92

Ví dụ



93



94

Tìm kiếm háu ăn (greedy best-first search)

- *Tìm kiếm háu ăn* (greedy best-first search) hay còn gọi là *tìm kiếm chỉ sử dụng heuristic* (pure heuristic search) **cố gắng triển khai nút “gần” với mục tiêu nhất.**
- Vì thế, nó chỉ dùng hàm heuristic $h(n)$ để lượng giá các nút, tức là

$$f(n) = h(n)$$

95

95

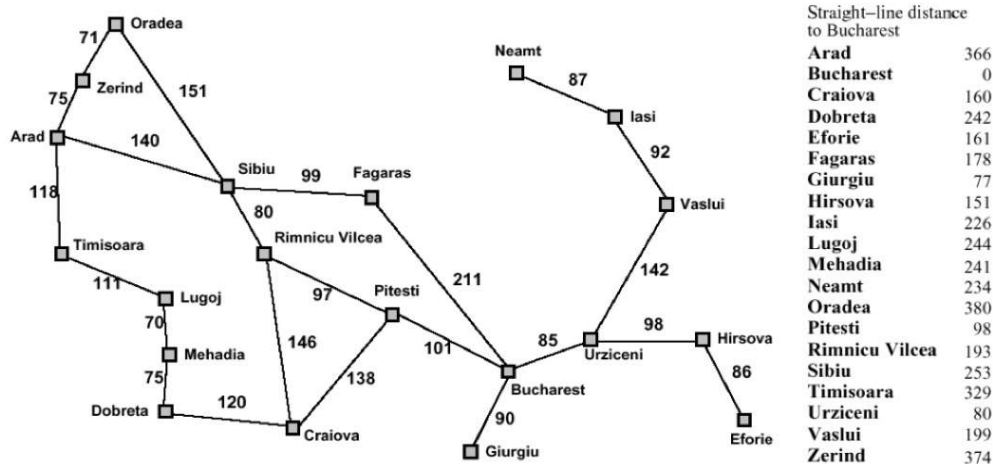
Tìm kiếm heuristic

- Bài toán tìm đường:
 - Thành phố xuất phát: Arad
 - Thành phố đích: Bucharest
 - Các cạnh biểu diễn đường nối trực tiếp giữa hai thành phố, các con số ghi trên các cạnh là chi phí đi giữa hai thành phố.
 - Cột bên phải là khoảng cách Euclid từ các thành phố đến thành phố đích Bucharest. ($h(n)$)

96

Tìm kiếm heuristic

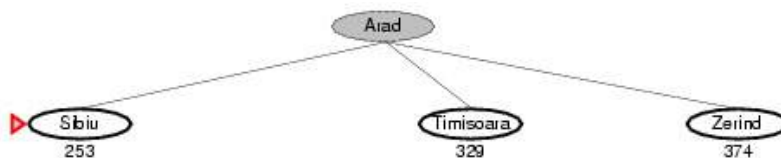
- Initial State = **Arad**
- Goal State = **Bucharest**



97

Tìm kiếm heuristic

- Initial State = **Arad**
- Goal State = **Bucharest**



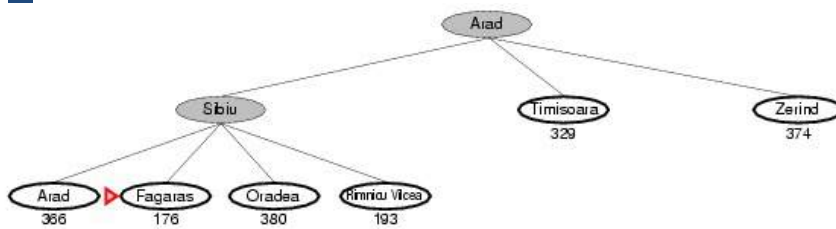
Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

98

Tìm kiếm heuristic

- Initial State = **Arad**
- Goal State = **Bucharest**



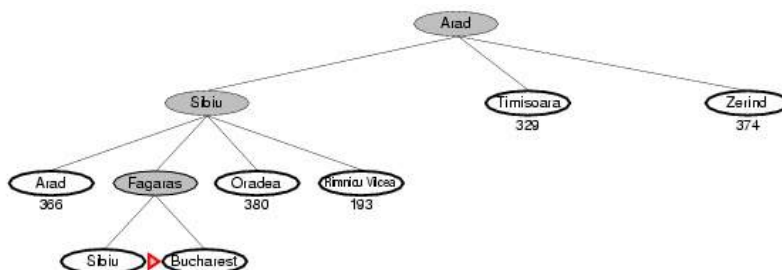
Straight-line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

99

Tìm kiếm heuristic

- Initial State = **Arad**
- Goal State = **Bucharest**

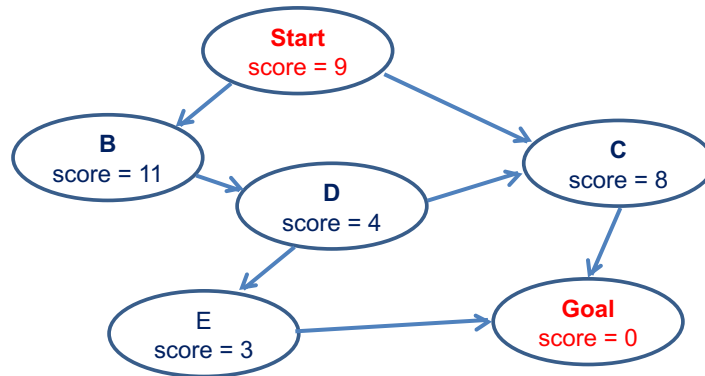


Straight-line distance
to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

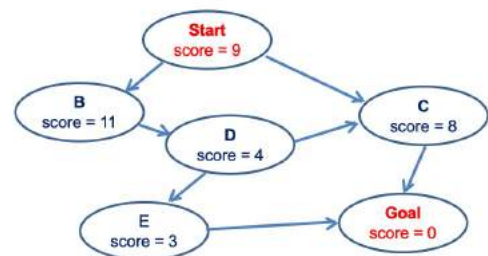
100

Tìm kiếm tốt nhất đầu tiên, tìm kiếm rộng, tìm kiếm sâu để tìm kiếm trạng thái “Goal” từ trạng thái “Start”



101

101



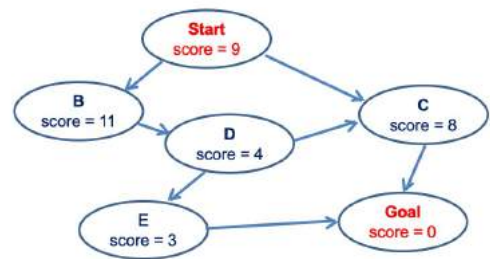
BFS - Tìm kiếm theo chiều rộng

Step# OPEN CLOSED X CHILDREN RemainingCHILDREN

1	{ S }	{ }	S	{ S, B, C }	{ B, C }
2	{ B, C }	{ S }	B	{ D }	{ D }
3	{ C, D }	{ S, B }	C	{ G }	{ G }
4	{ D, G }	{ S, B, C }	D	{ C, E }	{ E }
5	{ G, E }	{ S, B, C, D }	G	DONE	

- note we check for **GOALS** upon removal from OPEN in order to get **SHORTEST PATHS** in later algo's

102



DFS Tìm kiếm theo chiều sâu

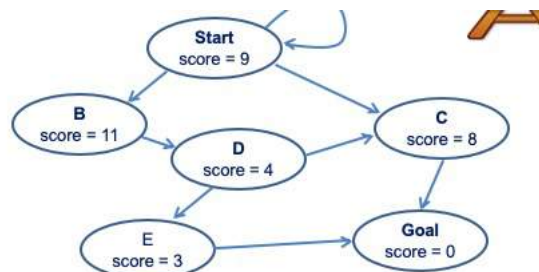
Step#	OPEN	CLOSED	X	CHILDREN	RemainingCHILDREN
1	{ S }	{ }	S	{ S ^S , B ^S , C ^S }	{ B ^S , C ^S }
2	{ B ^S , C ^S }	{ S }	B ^S	{ D ^B }	{ D ^B }
3	{ D ^B , C ^S }	{ S, B ^S }	D ^B	{ C ^D , E ^D }	{ E ^D }
4	{ E ^D , C ^S }	{ S, B ^S , D ^B }	E ^D	{ G ^E }	{ G ^E }
5	{ G ^E , C ^S }	{ S, B ^S , D ^B , E ^D }	G ^E	DONE	

103

103

BEST

Step#	OPEN	CLOSED	X	CHILDREN	RemainingCHILDREN
1	{ S ₉ }	{ }	S ₉	{ S ₉ , B ₁₁ , C ₈ }	{ B ₁₁ , C ₈ }
2	{ C ₈ , B ₁₁ }	{ S ₉ }	C ₈	{ G ₀ }	{ G ₀ }
3	{ G ₀ , B ₁₁ }	{ S ₉ , C ₈ }	G ₀	DONE	



10/4/16

CS 540 - Fall 2016 (Shavlik@), Lecture 8, Week 5

Lecture 1, Slide 104

104

Giải thuật A*

- Giải thuật A* là một trường hợp đặc biệt Best first search (việc cập nhật lại đường đi dựa trên giá trị $g(n)$ thay vì dựa trên giá trị $f(n)$ tổng quát)
 - $f(n) = g(n) + h(n)$
 - *$h(n)$ phụ thuộc vào trạng thái n nên $f(n)$ chỉ thay đổi khi $g(n)$ thay đổi hay nói cách khác khi ta tìm được một đường đi mới đến n tốt hơn đường đi cũ => cập nhật lại g khi đường đi mới tốt hơn)*
- Mỗi trạng thái n tùy ý sẽ gồm 4 yếu tố ($g(n)$, $h(n)$, $f(n)$, $cha(n)$)
 Cha(n) là nút cha của nút đang xét n

105

105

```

PNode AStarSearch(PState init_state) {
    PNode root = new Node();
    root->state = init_state;
    root->parent = NULL;
    root->f = 0;
    frontier.insert(root);
    explored.clear();
    while (!empty(frontier)) {
        //Lấy 1 nút từ đường biên có f(n) nhỏ nhất
        //và loại bỏ nó ra khỏi đường biên
        Node* node = frontier.pop();
        if (node là nút mục tiêu)
            return node;
        insert(node, explored);

        for (child là nút con của node) {
            child->g = node->g + stepCost(node, child);
            child->h = estimate(child->state);
            child->f = child->g + child->h;
            if (child->state không thuộc frontier và
                child->state không thuộc explored) {
                child->parent = node;
                frontier.insert(child);
            } else if (child->state nằm trong đường biên
                        và có g lớn hơn child->g)
                (*) Thay thế nút nằm trên đường biên
                    bằng child
            }
        }
        return NULL; //Thất bại, không tìm thấy lời giải
    }
}

```

106

Mã giả giải thuật A*

```

g(n0)=0; f(n0)=h(n0);
open:=[n0]; closed:=[];
while open <> [] do
    loại n bên trái của open và đưa n vào closed;
    if (n là một đích) then thành công, thoát
    else
        Sinh các con m của n;
        For m thuộc con(n) do
            g(m)=g(n)+c[n,m];
            If m không thuộc open hay closed then
                f(m)=g(m)+h(m); cha(m)=n; Bỏ m vào open;
            If m thuộc open (tồn tại m' thuộc open, sao cho m=m') then
                If g(m)<g(m') then g(m')=g(m); f(m')=g(m')+h(m'); Cha(m')=n;
            If m thuộc closed (tồn tại m' thuộc closed, sao cho m=m') then
                If g(m)<g(m') then f(m)=g(m)+h(m); cha(m)=n;
                Đưa m vào open; loại m' khỏi closed;
        Sắp xếp open để t.thái tốt nhất nằm bên trái;

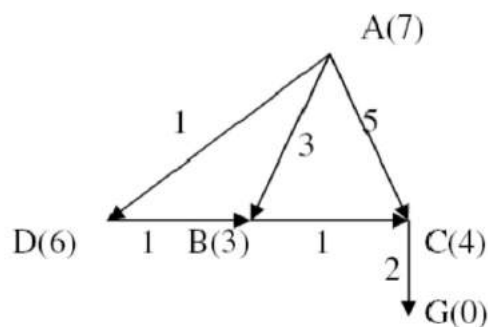
```

107

107

Giải thuật A*

- Sử dụng giải thuật A* để tìm đường đi từ A đến G với giá trị g(n) được ghi trên cạnh của đồ thị và h(n) ghi ngay đỉnh của đồ thị



108

108

■ Mỗi trạng thái n tùy ý sẽ gồm bốn yếu tố $(g(n), h(n), f(n), \text{cha}(n))$

■ **Bước 1:**

- $\text{Open} = \{A(0,7,7,-)\}$; $\text{close} = \{\}$

■ **Bước 2:**

- Các con của A : D, B, C

- Xét D

■ $g(D) = g(A) + c[A, D] = 0 + 1 = 1$ (do đề bài cung cấp) ($g(m) = g(n) + c[m, n]$)

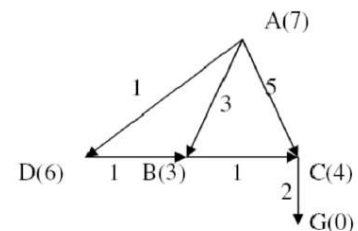
■ D không thuộc **Open; Close**

- Tính giá trị $f(D) = g(D) + h(D) = 1 + 6 = 7$

- Cập nhật cha của D : A

- Đưa D vào **open**: $D(1,6,7,A)$

- Xét B



109

■ **Bước 2:**

- Các con của A : D, B, C

-

- Xét B

■ $g(B) = g(A) + c[A, B] = 0 + 3 = 3$ (do đề bài cung cấp) ($g(m) = g(n) + c[m, n]$)

■ B không thuộc **Open; Close**

- Tính giá trị $f(B) = g(B) + h(B) = 3 + 3 = 6$

- Cập nhật cha của B : A

- Đưa B vào **open**: $B(3,3,6,A)$

- Xét C

■ $g(C) = g(A) + c[A, C] = 0 + 5 = 5$ (do đề bài cung cấp) ($g(m) = g(n) + c[m, n]$)

■ C không thuộc **Open; Close**

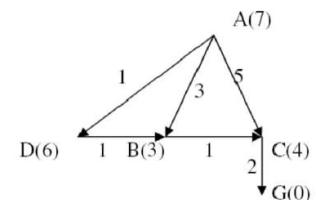
- Tính giá trị $f(C) = g(C) + h(C) = 5 + 4 = 9$

- Cập nhật cha của C : A

- Đưa C vào **open**: $C(5,4,9,A)$

- Sắp xếp các phần tử trong **open** để trạng thái tốt nhất bên trái

$\text{Open} = \{B(3,3,6,A), D(1,6,7,A), C(5,4,9,A)\}$; $\text{close} = \{A(0,7,7,-)\}$

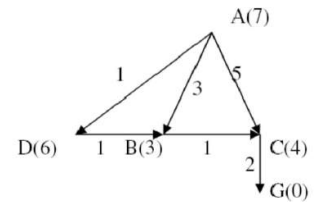


110

110

■ Bước 3:

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **B** ra khỏi Open đưa vào Close
 $Open\{D(1,6,7,A), C(5,4,9,A)\} \quad ; \quad close=\{A(0,7,7,-), \mathbf{B(3,3,6,A)}\}$
- Các con của B: C
- Xét C
 - $g(C) = g(B) + c[B,C] = 3 + 1 = 4$ ($g(m) = g(n) + c[m,n]$)
 - **C thuộc Open;**
 - So sánh giá trị $g(C_n)$ hiện tại và $g(C_o)$ đã tồn tại trong Open
 - $G(C_o) = 5 > g(C_n) = 4 \Rightarrow$ cập nhật lại C
 - Tính giá trị $f(C) = g(C) + h(C) = 4 + 4 = 8$
 - Cập nhật cha của C: B
 - Đưa C vào open: $C(4,4,8,B)$
- Sắp xếp các phần tử trong open để trạng thái tốt nhất bên trái
 $Open\{D(1,6,7,A), C(4,4,8,B)\} \quad ; \quad close=\{A(0,7,7,-), \mathbf{B(3,3,6,A)}\}$

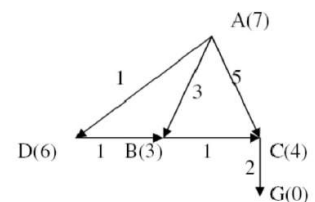


111

111

■ Bước 4:

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **D** ra khỏi Open đưa vào Close
 $Open\{C(4,4,8,B)\} \quad ; \quad close=\{A(0,7,7,-), B(3,3,6,A), \mathbf{D(1,6,7,A)}\}$
- Các con của D: B
- Xét B
 - $g(B) = g(D) + c[D,B] = 1 + 1 = 2$ ($g(m) = g(n) + c[m,n]$)
 - **B thuộc Closed;**
 - So sánh giá trị $g(B_n)$ hiện tại và $g(B_o)$ đã tồn tại trong Open
 - $G(B_o) = 3 > g(B_n) = 2 \Rightarrow$ cập nhật lại B
 - Tính giá trị $f(B) = g(B) + h(B) = 2 + 3 = 5$
 - Cập nhật cha của B: D
 - Đưa B(2,3,5,D) vào open
 - Loại B(3,3,6,A) ra khỏi Closed
- Sắp xếp các phần tử trong open để trạng thái tốt nhất bên trái
 $Open\{B(2,3,5,D), C(4,4,8,B)\} \quad ; \quad close=\{A(0,7,7,-), \mathbf{D(1,6,7,A)}\}$

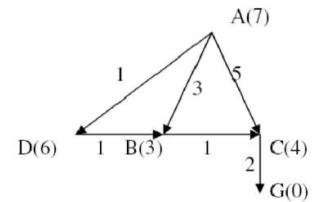


112

112

■ Bước 5:

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **B** ra khỏi Open đưa vào Close
 $Open\{C(4,4,8,B)\}; \quad close=\{A(0,7,7,-), D(1,6,7,A), B(2,3,5,D)\}$
- Các con của B: C
- Xét C
 - $g(C) = g(B) + c[B,C] = 1 + 2 = 3$ ($g(m) = g(n) + c[m,n]$)
 - **C thuộc Open;**
 - So sánh giá trị $g(C_n)$ hiện tại và $g(C_o)$ đã tồn tại trong Open
 - $G(C_o) = 4 > g(C_n) = 3 \Rightarrow$ cập nhật lại C
 - Tính giá trị $f(C) = g(C) + h(C) = 3 + 4 = 7$
 - Cập nhật cha của C: B
 - Đưa C vào open: C(3,4,7,B)
- Sắp xếp các phần tử trong open để trạng thái tốt nhất bên trái
 $Open\{C(3,4,7,B)\} \quad ; \quad close=\{A(0,7,7,-), D(1,6,7,A), B(2,3,5,D)\}$

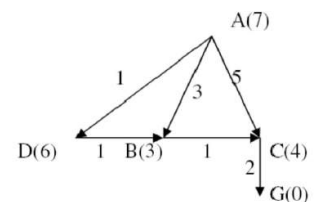


113

113

■ Bước 6:

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **C** ra khỏi Open đưa vào Close
 $Open\{\}; \quad close=\{A(0,7,7,-), D(1,6,7,A), B(2,3,5,D), C(3,4,7,B)\}$
- Các con của C: G
- Xét G
 - $g(G) = g(C) + c[C,G] = 3 + 2 = 5$ ($g(m) = g(n) + c[m,n]$)
 - G không thuộc Open; Closed
 - Tính giá trị $f(G) = g(G) + h(G) = 5 + 0 = 5$
 - Cập nhật cha của G: C
 - Đưa G vào open: G(5,0,5,C)
- Sắp xếp các phần tử trong open để trạng thái tốt nhất bên trái
 $Open\{G(5,0,5,C)\} \quad ; \quad close=\{A(0,7,7,-), D(1,6,7,A), B(2,3,5,D)\}$



114

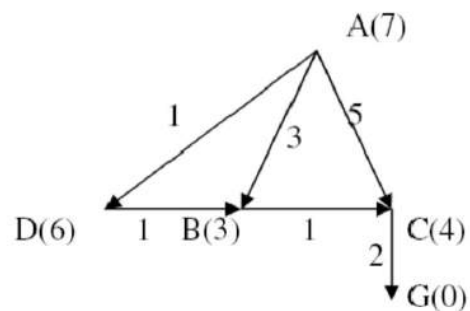
114

■ Bước 6:

- Quay lại đầu vòng lặp *while*
- Lấy phần tử *C* ra khỏi *Open* đưa vào *Close*
 $Open\{\}; \quad closed=\{A(0,7,7,-), D(1,6,7,A), B(2,3,5,D), C(3,4,7,B),$
 $G(5,0,5,C)\}$
- Các $G(5,0,5,C)$ là trạng thái đích \Rightarrow giải thuật dừng lại

Ta có đường đi

$closed=\{A(0,7,7,-), D(1,6,7,A), B(2,3,5,D),$
 $A \Rightarrow D \Rightarrow B \Rightarrow C \Rightarrow G$



115

Tìm kiếm heuristic

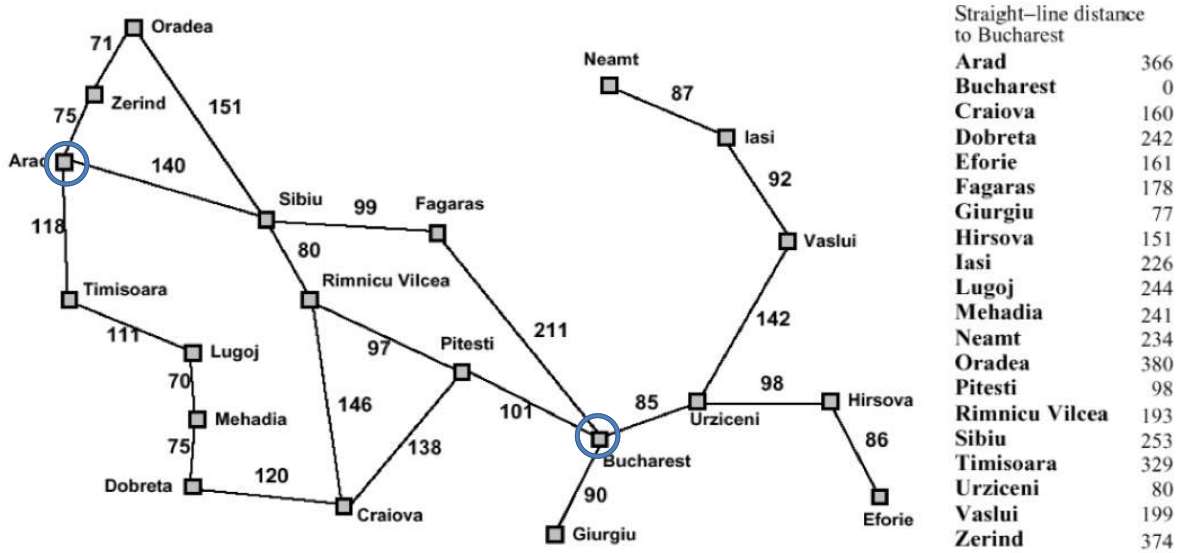
■ Bài toán tìm đường:

- Thành phố xuất phát: Arad
- Thành phố đích: Bucharest
- Các cạnh biểu diễn đường nối trực tiếp giữa hai thành phố, các con số ghi trên các cạnh là chi phí đi giữa hai thành phố.
- Cột bên phải là khoảng cách Euclid từ các thành phố đến thành phố đích Bucharest.

- Sử dụng phương pháp tìm kiếm A^* (hàm ước lượng $f(n) = g(n) + h(n)$, với $g(n)$ là chi phí từ thành phố xuất phát đến n và $h(n)$ là khoảng cách Euclid từ n đến đích)

116

Tìm kiếm heuristic



117

Tìm kiếm heuristic

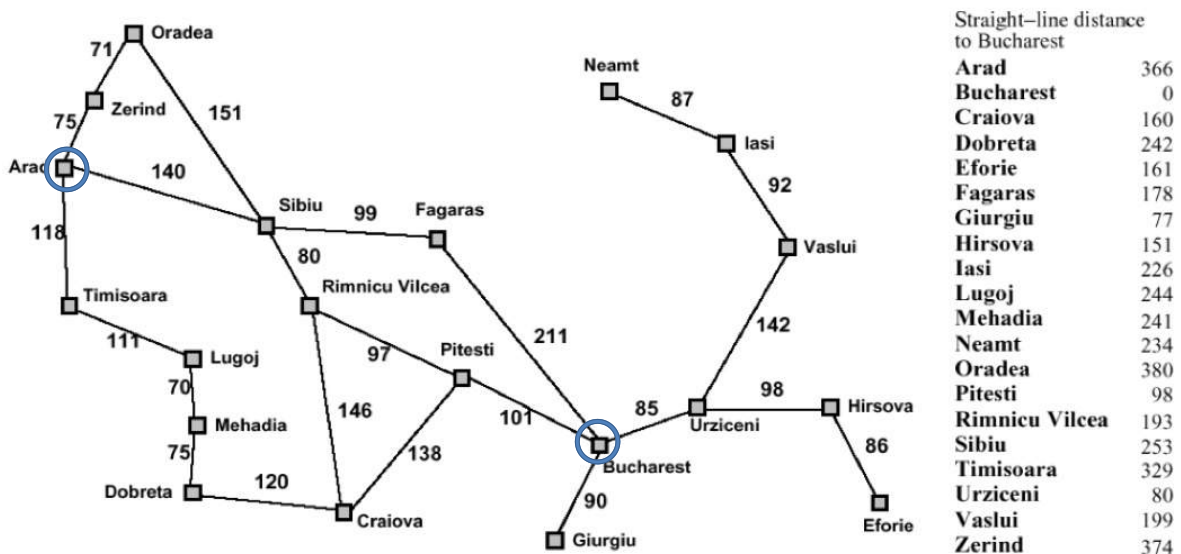
Bài toán tìm đường:

- Thành phố xuất phát: Arad
- Thành phố đích: Bucharest
- Các cạnh biểu diễn đường nối trực tiếp giữa hai thành phố, các con số ghi trên các cạnh là chi phí đi giữa hai thành phố.
- Cột bên phải là khoảng cách Euclid từ các thành phố đến thành phố đích Bucharest.

Sử dụng phương pháp tìm kiếm A* (hàm ước lượng $f(n) = g(n) + h(n)$, với $g(n)$ là chi phí từ thành phố xuất phát đến n và $h(n)$ là khoảng cách Euclid từ n đến đích)

116

Tìm kiếm heuristic



117

Mã giả giải thuật A*

```

g(n0)=0; f(n0)=h(n0);
open:=[n0]; closed:=[];
while open <> [] do
    loại n bên trái của open và đưa n vào closed;
    if (n là một đích) then thành công, thoát
    else
        Sinh các con m của n;
        For m thuộc con(n) do
            g(m)=g(n)+c[n,m];
            If m không thuộc open hay closed then
                f(m)=g(m)+h(m); cha(m)=n; Bỏ m vào open;
            If m thuộc open (tồn tại m' thuộc open, sao cho m=m') then
                If g(m)<g(m') then g(m')=g(m); f(m')=g(m')+h(m'); Cha(m')=n;
            If m thuộc closed (tồn tại m' thuộc closed, sao cho m=m') then
                If g(m)<g(m') then f(m)=g(m)+h(m); cha(m)=n;
                Đưa m vào open; loại m' khỏi closed;
        Sắp xếp open để t.thái tốt nhất nằm bên trái;

```

118

118

- Mỗi trạng thái n tùy ý sẽ gồm: (g(n), h(n), f(n), cha(n))

- **Bước 1:**

- Open={ Arad (0,366,366,-)}; close={}

- **Bước 2:**

- Các con của Arad: Timisoara, Sibiu, Zerind
- Xét Timisoara

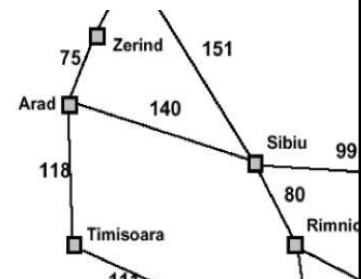
- $g(\text{Timisoara}) = g(\text{Arad}) + c[\text{Arad}, \text{Timisoara}] = 0 + 118 = 118$ (do đề bài cung cấp) ($g(m) = g(n) + c[m,n]$)

- Timisoara không thuộc Open; Close

- Tính giá trị $f(\text{Timisoara}) = g(\text{Timisoara}) + h(\text{Timisoara})$
 $= 118 + 329 = 447$

- Cập nhật cha của Timisoara : Arad

- Đưa Timisoara vào open: Timisoara(118,329,447,Arad)



119

119

■ Bước 2:

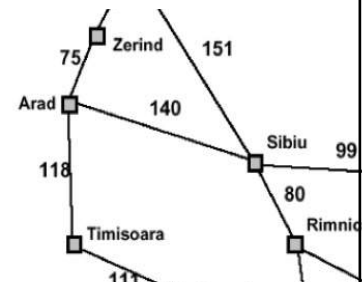
-

- Xét Sibiu

$$\begin{aligned} \blacksquare g(\text{Sibiu}) &= g(\text{Arad}) + c[\text{Arad}, \text{Sibiu}] \\ &= 0 + 140 = 140 \text{ (do đề bài cung cấp)} \\ &\quad (g(m) = g(n) + c[m, n]) \end{aligned}$$

■ Sibiu không thuộc Open; Close

- Tính giá trị $f(\text{Sibiu}) = g(\text{Sibiu}) + h(\text{Sibiu}) = 140 + 253 = 393$
- Cập nhật cha của Sibiu : Arad
- Đưa Sibiu vào open: Sibiu (140, 253, 393, Arad)



120

120

■ Bước 2:

-

- Xét Zerind

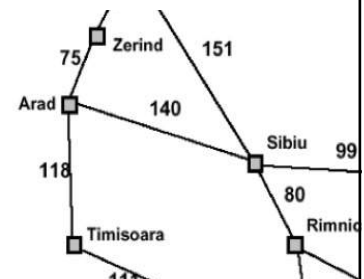
$$\begin{aligned} \blacksquare g(\text{Zerind}) &= g(\text{Arad}) + c[\text{Arad}, \text{Zerind}] \\ &= 0 + 75 = 75 \text{ (do đề bài cung cấp)} \\ &\quad (g(m) = g(n) + c[m, n]) \end{aligned}$$

■ Zerind không thuộc Open; Close

- Tính giá trị $f(\text{Zerind}) = g(\text{Zerind}) + h(\text{Zerind}) = 75 + 374 = 449$
- Cập nhật cha của Zerind : Arad
- Đưa Zerind vào open: Zerind(75, 374, 449, Arad)

- Sắp xếp các phần tử trong open để trạng thái tốt nhất bên trái

Open{Sibiu (140, 253, 393, Arad), Timisoara(118, 329, 447, Arad), Zerind(75, 374, 449, Arad)};
Closed = { Arad (0, 366, 366, -) }



121

121

■ Bước 3

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **Sibiu** ra khỏi *Open* đưa vào *Closed*

Open{Timisoara(118,329,447,Arad), Zerind(75,374,449,Arad)};

Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad) }

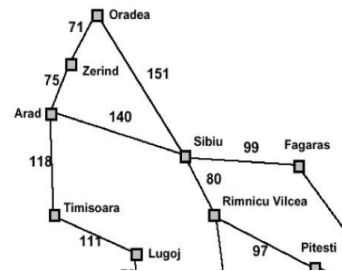
- Các con của **Sibiu** : Rimnicu Vilces, Fagaras, Arad, Oradea
- Xét Rimnicu

$$\begin{aligned} \blacksquare \quad g(\text{Rimnicu}) &= g(\text{Sibiu}) + c[\text{Sibiu}, \text{Rimnicu}] = 140 + 80 = 220 \\ & \quad (g(m) = g(n) + c[m,n]) \end{aligned}$$

■ Rimnicu không thuộc *Open*; *Close*

$$\begin{aligned} - \text{Tính giá trị } f(\text{Rimnicu}) &= g(\text{Rimnicu}) + h(\text{Rimnicu}) \\ &= 220 + 193 = 413 \end{aligned}$$

- Cập nhật cha của Rimnicu : Sibiu
- Đưa Rimnicu vào *open*: Rimnicu(220,193,413,Sibiu)



122

122

■ Bước 3

-
- Các con của **Sibiu** : Rimnicu Vilces, Fagaras, Arad, O
- Xét Fagaras

$$\blacksquare \quad g(\text{Fagaras}) = g(\text{Sibiu}) + c[\text{Sibiu}, \text{Fagaras}] = 140 + 99 = 239 \quad (g(m) = g(n) + c[m,n])$$

■ Fagaras không thuộc *Open*; *Close*

$$- \text{Tính giá trị } f(\text{Fagaras}) = g(\text{Fagaras}) + h(\text{Fagaras}) = 239 + 178 = 417$$

$$- \text{Cập nhật cha của Fagaras : Sibiu}$$

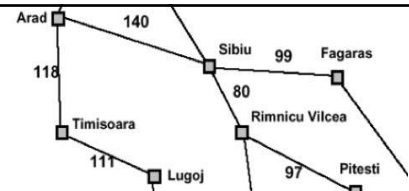
$$- \text{Đưa Fagaras vào open: Fagaras}(239,178,417,\text{Sibiu})$$

- Sắp xếp các phần tử trong *open* để trạng thái tốt nhất bên trái

Open{Rimnicu(220,193,413,Sibiu), Fagaras(239,178,417,Sibiu),

Timisoara(118,329,447,Arad), Zerind(75,374,449,Arad)};

Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad) }



123

123

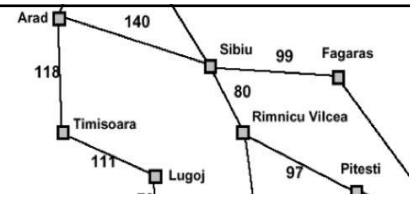
■ Bước 3

-
- Các con của **Sibiu** : Rimnicu, Fagaras, Arad, Oradea
- Xét Oradea

■ $g(\text{Oradea}) = g(\text{Sibiu}) + c[\text{Sibiu}, \text{Oradea}] = 140 + 151 = 291$ ($g(m) = g(n) + c[m,n]$)

■ Oradea không thuộc Open; Close

- Tính giá trị $f(\text{Oradea}) = g(\text{Oradea}) + h(\text{Oradea}) = 291 + 380 = 671$
- Cập nhật cha của Oradea : Sibiu
- Đưa Oradea vào open: Oradea(291,380,671,Sibiu)
- Sắp xếp các phần tử trong open để trạng thái tốt nhất bên trái
 $\text{Open}\{\text{Rimnicu}(220,193,413,\text{Sibiu}), \text{Fagaras}(239,178,417,\text{Sibiu}),$
 $\text{Timisoara}(118,329,447,\text{Arad}), \text{Zerind}(75,374,449,\text{Arad}), \text{Oradea}(291,380,671,\text{Sibiu})\};$
 $\text{Closed} = \{ \text{Arad} (0,366,366,-), \text{Sibiu} (140,253,393,\text{Arad}) \}$



124

124

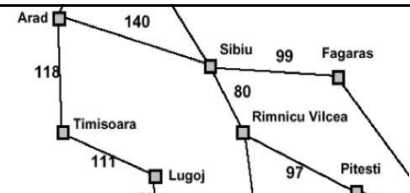
■ Bước 3

-
- Các con của **Sibiu** : Rimnicu, Fagaras, Arad, Oradea
- Xét Arad

■ $g(\text{Arad}) = g(\text{Sibiu}) + c[\text{Sibiu}, \text{Arad}] = 140 + 140 = 280$ ($g(m) = g(n) + c[m,n]$)

■ Arad thuộc Closed

- $G(\text{Arad})=280 > G(\text{Arad}') \Rightarrow$ ko làm gì cả (ko đưa vào open hay closed)
- Sắp xếp các phần tử trong open để trạng thái tốt nhất bên trái
 $\text{Open}\{\text{Rimnicu}(220,193,413,\text{Sibiu}), \text{Fagaras}(239,178,417,\text{Sibiu}),$
 $\text{Timisoara}(118,329,447,\text{Arad}), \text{Zerind}(75,374,449,\text{Arad}), \text{Oradea}(291,380,671,\text{Sibiu})\};$
 $\text{Closed} = \{ \text{Arad} (0,366,366,-), \text{Sibiu} (140,253,393,\text{Arad}) \}$



125

125

■ Bước 4

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **Rimnicu** ra khỏi Open đưa vào Closed

Open{Fagaras(239,178,417,Sibiu),Timisoara(118,329,447,Arad),
Zerind(75,374,449,Arad), Oradea(291,380,671,Sibiu)};

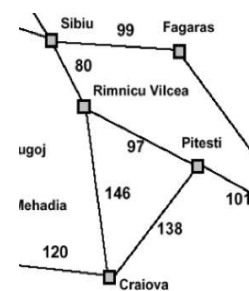
Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad), Rimnicu(220,193,413,Sibiu) }

- Các con của **Rimnicu** : Craiova, Pitesti, Sibiu
- Xét Craiova

■ $g(\text{Craiova}) = g(\text{Rimnicu}) + c[\text{Rimnicu}, \text{Craiova}] = 220 + 146 = 366$

■ Craiova không thuộc Open; Close

- Tính giá trị $f(\text{Craiova}) = g(\text{Craiova}) + h(\text{Craiova}) = 366 + 160 = 526$
- Cập nhật cha của Craiova là Rimnicu
- Đưa Craiova vào open: Craiova(366,160,526, Rimnicu)



126

126

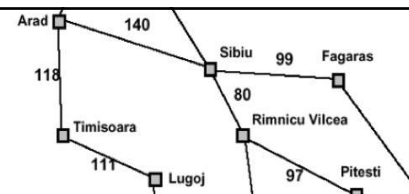
■ Bước 4

-
- Các con của **Rimnicu** : Craiova, Pitesti, Sibiu
- Xét Pitesti

■ $g(\text{Pitesti}) = g(\text{Rimnicu}) + c[\text{Rimnicu}, \text{Pitesti}] = 220 + 97 = 317$

■ Pitesti không thuộc Open; Close

- Tính giá trị $f(\text{Pitesti}) = g(\text{Pitesti}) + h(\text{Pitesti}) = 317 + 98 = 415$
 - Cập nhật cha của Pitesti là Rimnicu
 - Đưa Pitesti vào open: Pitesti (317,98,415, Rimnicu)
 - Sắp xếp các phần tử trong open để trạng thái tốt nhất bên trái
- Open{Pitesti (317,98,415, Rimnicu), Fagaras(239,178,417,Sibiu),Timisoara(118,329,447,Arad),
Zerind(75,374,449,Arad), Craiova(366,160,526, Rimnicu), Oradea(291,380,671,Sibiu)};
- Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad), Rimnicu(220,193,413,Sibiu) }



127

127

■ Bước 4

-

- Các con của **Rimnicu** : Craiova, Pitesti, **Sibiu**

- Xét **Sibiu**

$$\blacksquare g(\text{Sibiu}) = g(\text{Rimnicu}) + c[\text{Rimnicu}, \text{Sibiu}] = 220 + 80 = 400$$

■ **Sibiu thuộc Close**

- Tính giá trị $g(\text{Sibiu}) = 400 > g(\text{Sibiu}') = 140$

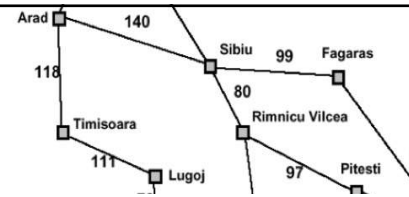
- \Rightarrow không làm gì cả (không đưa vào open hay closed)

-

- Sắp xếp các phần tử trong open để trạng thái tốt nhất bên trái

Open{Pitesti (317,98,415, Rimnicu), Fagaras(239,178,417,Sibiu), Timisoara(118,329,447,Arad),
Zerind(75,374,449,Arad), Craiova(366,160,526, Rimnicu), Oradea(291,380,671,Sibiu)};

Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad), Rimnicu(220,193,413,Sibiu) }



128

128

■ Bước 5

- Quay lại đầu vòng lặp while

- Lấy phần tử **Pitesti** ra khỏi Open đưa vào Closed

Open{Fagaras(239,178,417,Sibiu), Timisoara(118,329,447,Arad),

Zerind(75,374,449,Arad), Craiova(366,160,526, Rimnicu), Oradea(291,380,671,Sibiu)};

Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad), Rimnicu(220,193,413,Sibiu),
Pitesti (317,98,415, Rimnicu) }

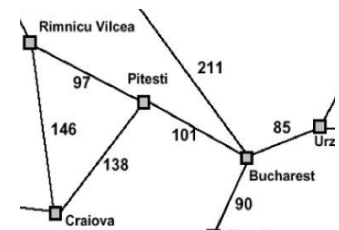
- Các con của **Pitesti** : Craiova, Bucharest, Rimnicu

- Xét **Craiova**

$$\blacksquare g(\text{Craiova}) = g(\text{Pitesti}) + c[\text{Pitesti}, \text{Craiova}] = 317 + 138 = 455$$

■ **Craiova thuộc Open;**

■ $g(\text{Craiova}) = 455 > g(\text{Craiova}') = 368 \Rightarrow$ Không làm gì cả



129

129

■ Bước 5

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **Pitesti** ra khỏi Open đưa vào Closed

Open{Fagaras(239,178,417,Sibiu),Timisoara(118,329,447,Arad),
Zerind(75,374,449,Arad), Craiova(366,160,526, Rimnicu), Oradea(291,380,671,Sibiu)};

Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad), Rimnicu(220,193,413,Sibiu),
Pitesti (317,98,415, Rimnicu) }

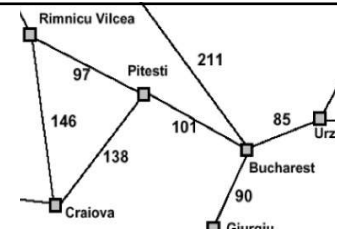
Các con của **Pitesti** : Craiova, Bucharest, Rimnicu

- Xét Bucharest

■ $g(\text{Bucharest}) = g(\text{Pitesti}) + c[\text{Pitesti}, \text{Bucharest}] = 317 + 101 = 418$

■ Bucharest **không thuộc Open; Closed**

- Tính giá trị $f(\text{Bucharest}) = g(\text{Bucharest}) + h(\text{Bucharest}) = 418 + 0 = 418$
- Cập nhật cha của Bucharest : Pitesti
- Đưa Bucharest vào open: Bucharest (418,0,418,Pitesti)



130

130

■ Bước 5

- ...

- Các con của **Pitesti** : Craiova, Bucharest, **Rimnicu**

- Xét Rimnicu

■ $g(\text{Rimnicu}) = g(\text{Pitesti}) + c[\text{Pitesti}, \text{Rimnicu}] = 317 + 138 = 455$

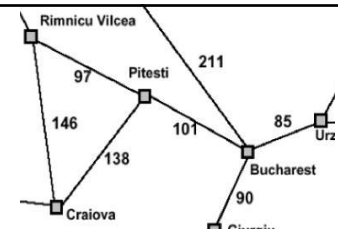
■ Rimnicu **thuộc Closed**

■ Xét $g(\text{Rimnicu})=455 > g(\text{Rimnicu}')=220$

- => Không làm gì cả

Open{Fagaras(239,178,417,Sibiu),Bucharest(418,0,418,Pitesti),Timisoara(118,329,447,Arad),
Zerind(75,374,449,Arad), Craiova(368,160,528, Rimnicu), Oradea(291,380,671,Sibiu)};

Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad), Rimnicu(220,193,413,Sibiu), Pitesti
(317,98,415, Rimnicu) }



131

131

■ Bước 6

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **Fagaras** ra khỏi Open đưa vào Closed

Open{Bucharest(418,0,418,Pitesti),Timisoara(118,329,447,Arad),
Zerind(75,374,449,Arad), Craiova(368,160,528, Rimnicu),Oradea(291,380,671,Sibiu)};

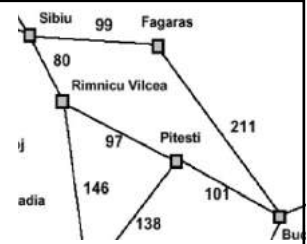
Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad), Rimnicu(220,193,413,Sibiu),
Pitesti (317,98,415, Rimnicu), **Fagaras(239,178,417,Sibiu)** }

- Các con của **Fagaras** : Bucharest, Sibiu
- Xét Sibiu

■ $g(\text{Sibiu}) = g(\text{Fagaras}) + c[\text{Fagaras}, \text{Sibiu}] = 239 + 99 = 338$

■ Sibiu thuộc Closed;

■ $g(\text{Sibiu}) = 338 > g(\text{Sibiu}') = 140 \Rightarrow$ Không làm gì cả



132

132

■ Bước 6

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **Fagaras** ra khỏi Open đưa vào Closed

Open{Bucharest(418,0,418,Pitesti),Timisoara(118,329,447,Arad),
Zerind(75,374,449,Arad), Craiova(368,160,528, Rimnicu),Oradea(291,380,671,Sibiu)};

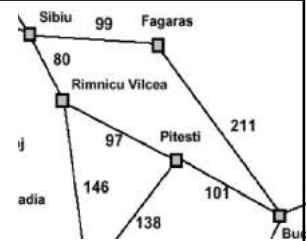
Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad), Rimnicu(220,193,413,Sibiu),
Pitesti (317,98,415, Rimnicu), **Fagaras(239,178,417,Sibiu)** }

- Các con của **Fagaras** : Bucharest, Sibiu
- Xét Bucharest

■ $g(\text{Bucharest}) = g(\text{Fagaras}) + c[\text{Fagaras}, \text{Bucharest}] = 239 + 211 = 450$

■ Bucharest thuộc Open;

■ $g(\text{Bucharest}) = 450 > g(\text{Bucharest}') = 418 \Rightarrow$ Không làm gì cả



133

133

■ Bước 7

- Quay lại đầu vòng lặp *while*
- Lấy phần tử **Bucharest** ra khỏi Open đưa vào Closed

Open{Timisoara(118,329,447,Arad),

Zerind(75,374,449,Arad), Craiova(368,160,528, Rimnicu),Oradea(291,380,671,Sibiu)};

Closed = { Arad (0,366,366,-), Sibiu (140,253,393,Arad), Rimnicu(220,193,413,Sibiu), Pitesti (317,98,415, Rimnicu), Fagaras(239,178,417,Sibiu), **Bucharest(418,0,418,Pitesti)**}

- **Xét Bucharest là trạng thái đích, giải thuật dừng lại.**
- **Đường đi được tìm bằng cách truy ngược cha của nút gốc và các nút kế tiếp:**

Bucharest(418,0,418,Pitesti) => Pitesti (317,98,415, Rimnicu) =>

Rimnicu(220,193,413,Sibiu) => Sibiu (140,253,393,Arad) => Arad (0,366,366,-)

Lưu ý: Các trạng thái trong closed là trạng thái đã xét qua, tuy nhiên **không phải tất cả các trạng thái trong closed đều góp phần trong đường đi lời giải** (Ví dụ đỉnh Fagaras trong bài toán này không thuộc đường đi lời giải)

134

134

Leo đồi (Hill climbing)

- **Ý tưởng:** Tìm kiếm trạng thái đích bằng cách hướng tới trạng thái tốt hơn trạng thái hiện tại (Leo lên đỉnh của một ngọn đồi)
- **Đặc điểm của giải thuật leo đồi:**
 - Trạng thái con tốt nhất sẽ được chọn cho bước tiếp theo
 - Không lưu giữ bất kỳ thông tin về các nút cha và anh em.
 - Quá trình tìm kiếm sẽ dừng lại khi gặp trạng thái đích hoặc trạng thái kế tiếp "xấu" hơn trạng thái đang xét ($f_{\text{đang xét}} < f_{\text{trạng thái kế tiếp}}$)
 - Sử dụng hàm đánh giá để đo tính tốt hơn của một trạng thái so với trạng thái khác

135

135

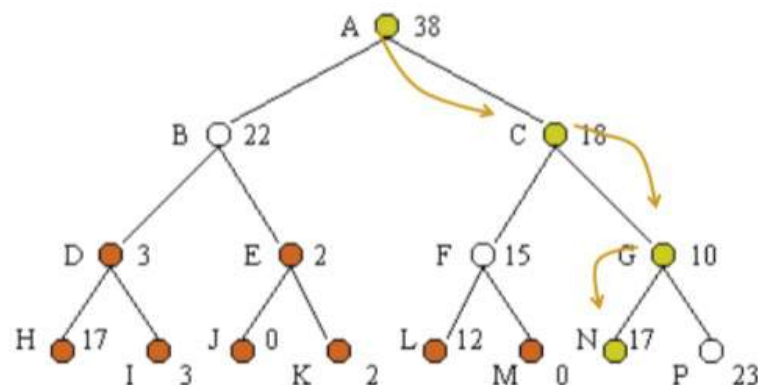
Leo đồi (Hill climbing)

1. **Đánh giá trạng thái khởi đầu.** Nếu nó là trạng thái đích, thoát, nếu không xét nó như trạng thái hiện hành
2. **Lặp lại đến khi tìm thấy một lời giải hoặc đến khi không tìm thấy « toán tử » mới nào có thể áp dụng lên trạng thái hiện hành:**
 - a) *Chọn một toán tử chưa được áp dụng đối với trạng thái hiện hành, áp dụng nó để sinh ra một trạng thái mới NS*
 - b) *Đánh giá trạng thái mới NS*
 - i. Nếu NS là một trạng thái đích, return NS và thoát
 - ii. Nếu NS không là đích nhưng « tốt hơn » trạng thái hiện hành, lấy NS làm trạng thái hiện hành
 - iii. Nếu NS không tốt hơn trạng thái hiện hành, tiếp tục vòng lặp

136

136

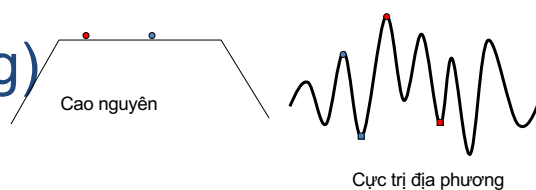
Tìm kiếm leo đồi (tt)



137

137

leo đồi (Hill climbing)



■ Hạn chế của tìm kiếm leo đồi:

- Không thể phục hồi lại từ những thất bại trong chiến lược của nó
- Hiệu quả hoạt động chỉ có thể được cải thiện trong một phạm vi giới hạn nào đó
- Lời giải tìm được không tối ưu hoặc không tìm được lời giải mặc dù có tồn tại lời giải do:
 - Có khuynh hướng sa lầy ở cực đại cục bộ
 - Cao nguyên
 - Chậm chỉ với một phép toán, không cho ra trạng thái « tốt hơn », nhưng với một vài phép toán có thể chuyển đến trạng thái « tốt hơn »

138

THUẬT TOÁN LEO ĐỒI (3) (Hill-Climbing)

■ Một vài giải pháp xử lý các vấn đề này:

- **Quay lui** « một vài bước » trước đó và thử đi theo một hướng khác. Để thực thi chiến lược này, duy trì một danh sách các bước đã trải qua. Giải pháp này đặc biệt phù hợp để xử lý tình huống « Local Optima »
- **Tạo ra một « bước nhảy đột phá » theo một hướng:** để chuyển sang một « vùng » mới trong không gian tìm kiếm. Phù hợp để xử lý tình huống « Plateau »
- **Áp dụng nhiều hơn một toán tử** để nhận được một trạng thái sau đó mới kiểm thử. Phù hợp để xử lý tình huống « Ridge »

139

139