Trí tuệ nhân tạo (Artificial Intelligence)

Tìm kiếm với thông tin bổ sung

By Hoàng Hữu Việt
Email: viethh@vinhuni.edu.vn
Viện Kỹ thuật và Công nghệ, Đại học Vinh

Vinh, 3/2019

Tài liệu

- Tài liệu chính

  [1] Stuart Russell, Peter Norvig. Artificial Intelligence. A modern approach. 3rd ed. Prentice Hall, 2009.

- Tài liệu khác

  [2] Milos Hauskrecht. Artificial Intelligence, 2013.

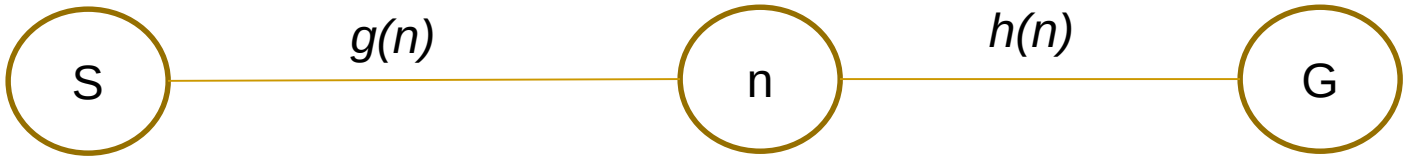
  people.cs.pitt.edu/~milos/courses/cs1571-Fall2013/

Nội dung

- Giới thiệu
- Greedy best-first search
- A* search
- Ảnh hưởng hàm heuristic đến hiệu quả thuật toán
- Bài tập

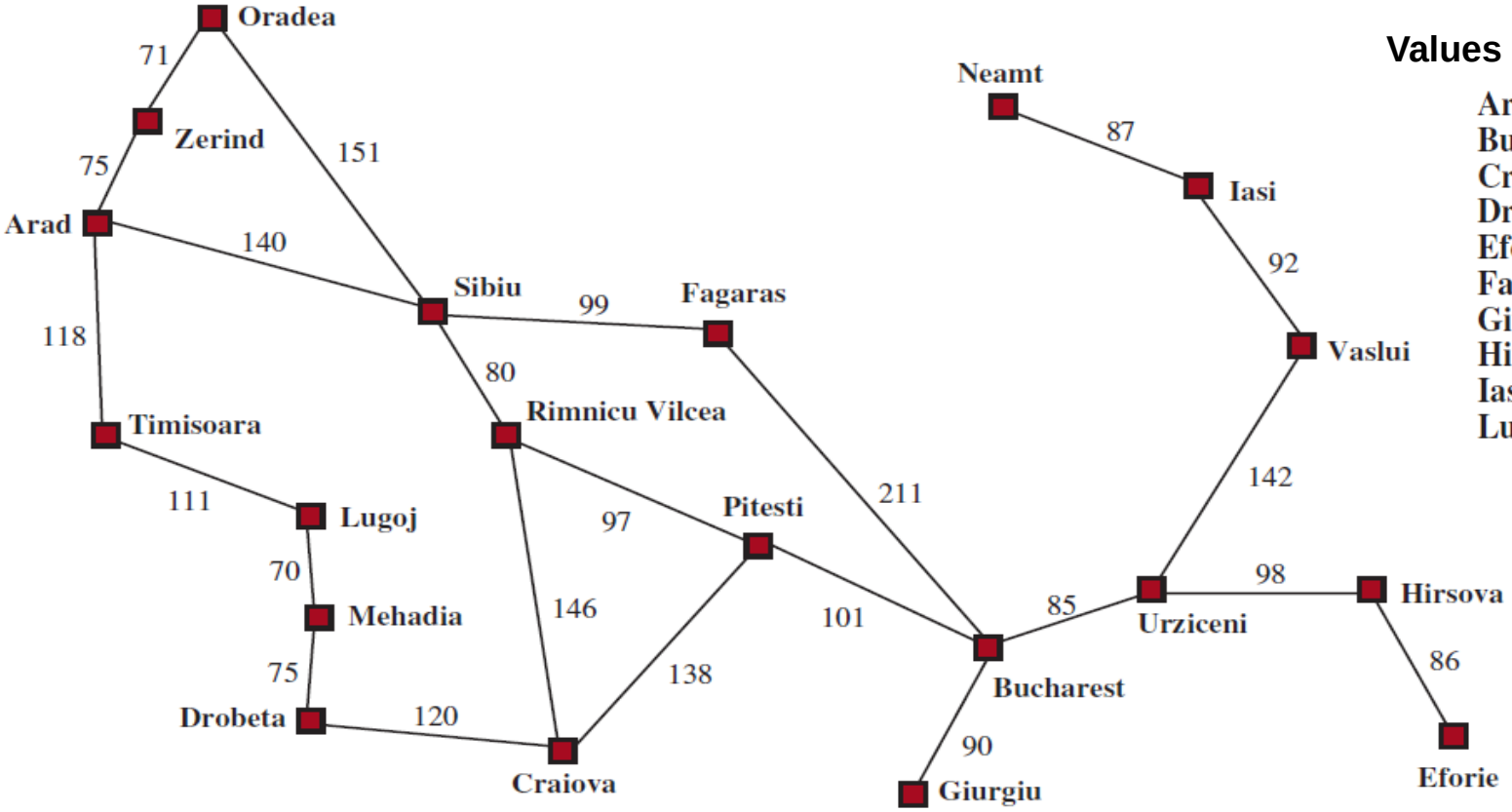
Thông tin bổ sung cho tìm kiếm

- Uninformed (or blind) search methods
  - Chỉ sử dụng thông tin trong định nghĩa bài toán.
  - Chi phí của đường chỉ tính đến nút hiện tại, tức hàm $g(n)$.
- Informed (or heuristic) search methods
  - Hướng tiếp cận tổng quát: tìm kiếm tốt nhất đầu tiên (best-first search).
  - Có thể là một thể hiện của TREE-SEARCH hoặc GRAPH-SEARCH, trong đó một nút được chọn để mở rộng dựa trên một hàm đánh giá, f (chiến lược tìm kiếm !).
  - Hàm f bao gồm một hàm đánh giá theo kinh nghiệm (heuristic function), $h(n)$.
    - $f(n) = g(n) + h(n)$, $h(n)$ = ước lượng chi phí của đường đi tốt nhất từ trạng thái n đến đích.



Thuật toán tìm kiếm tham lam

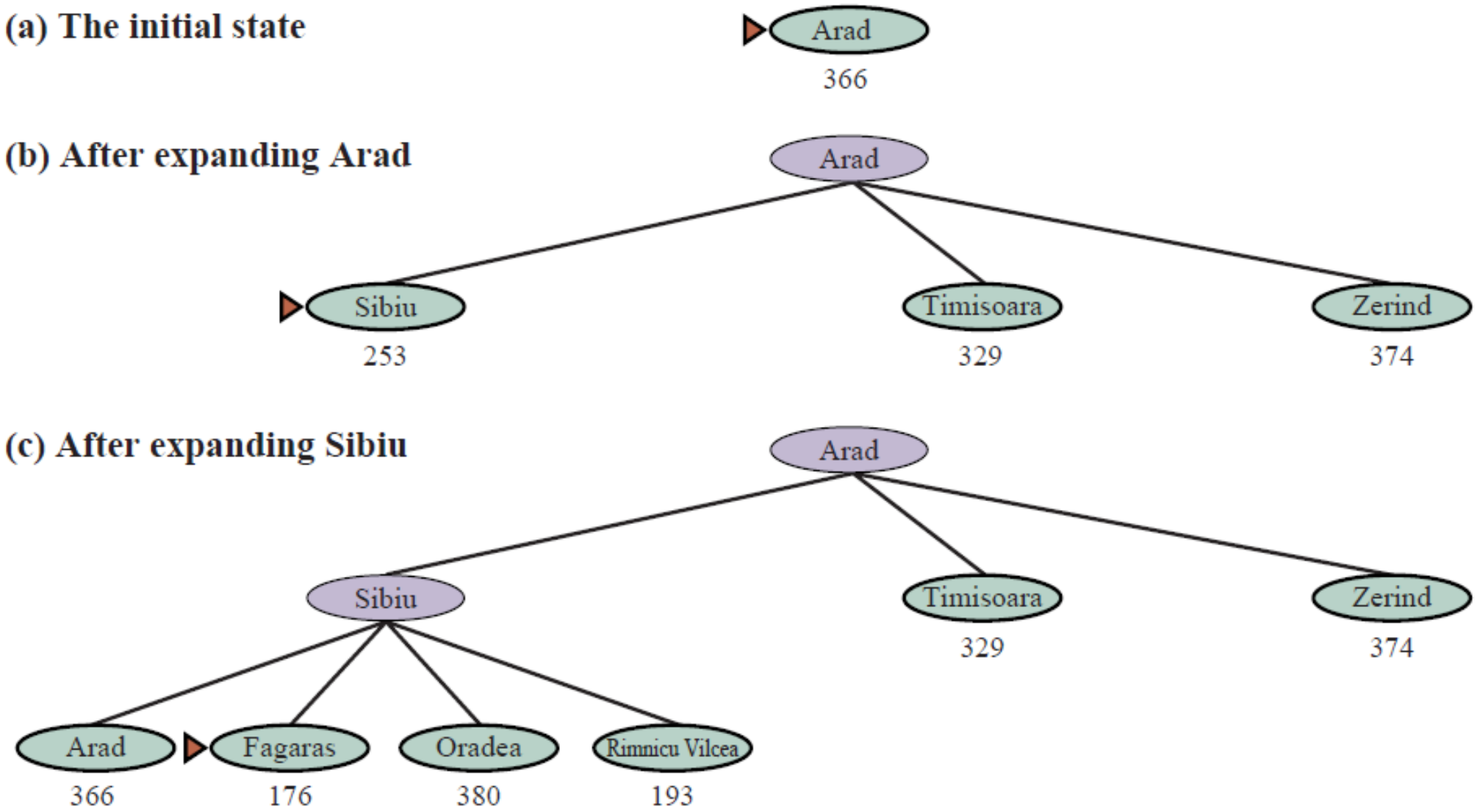
- Thuật toán tìm kiếm tham lam (greedy best – first search) là thuật toán UCS với $f(n) = h(n)$.
- Ví dụ 1. tìm đường đi từ Arad → Bucharest



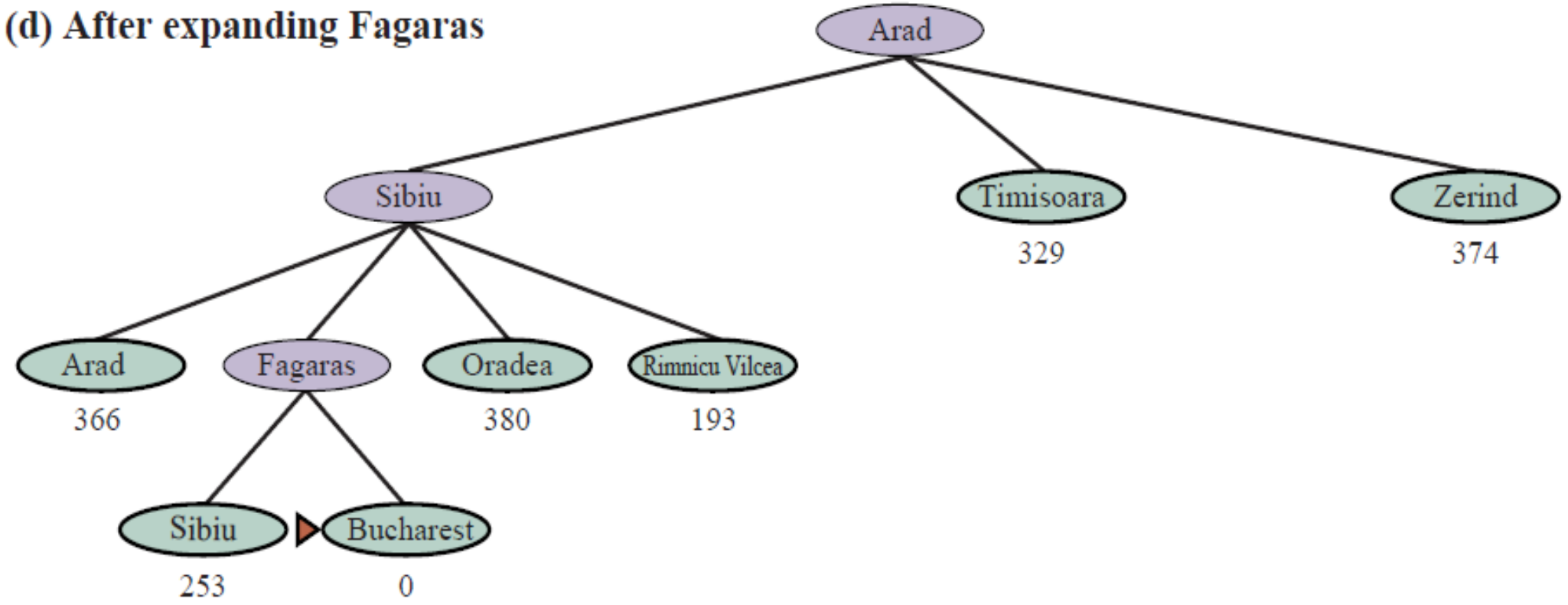
Values of hSLD—straight-line distances to Bucharest

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Thuật toán tìm kiếm tham lam



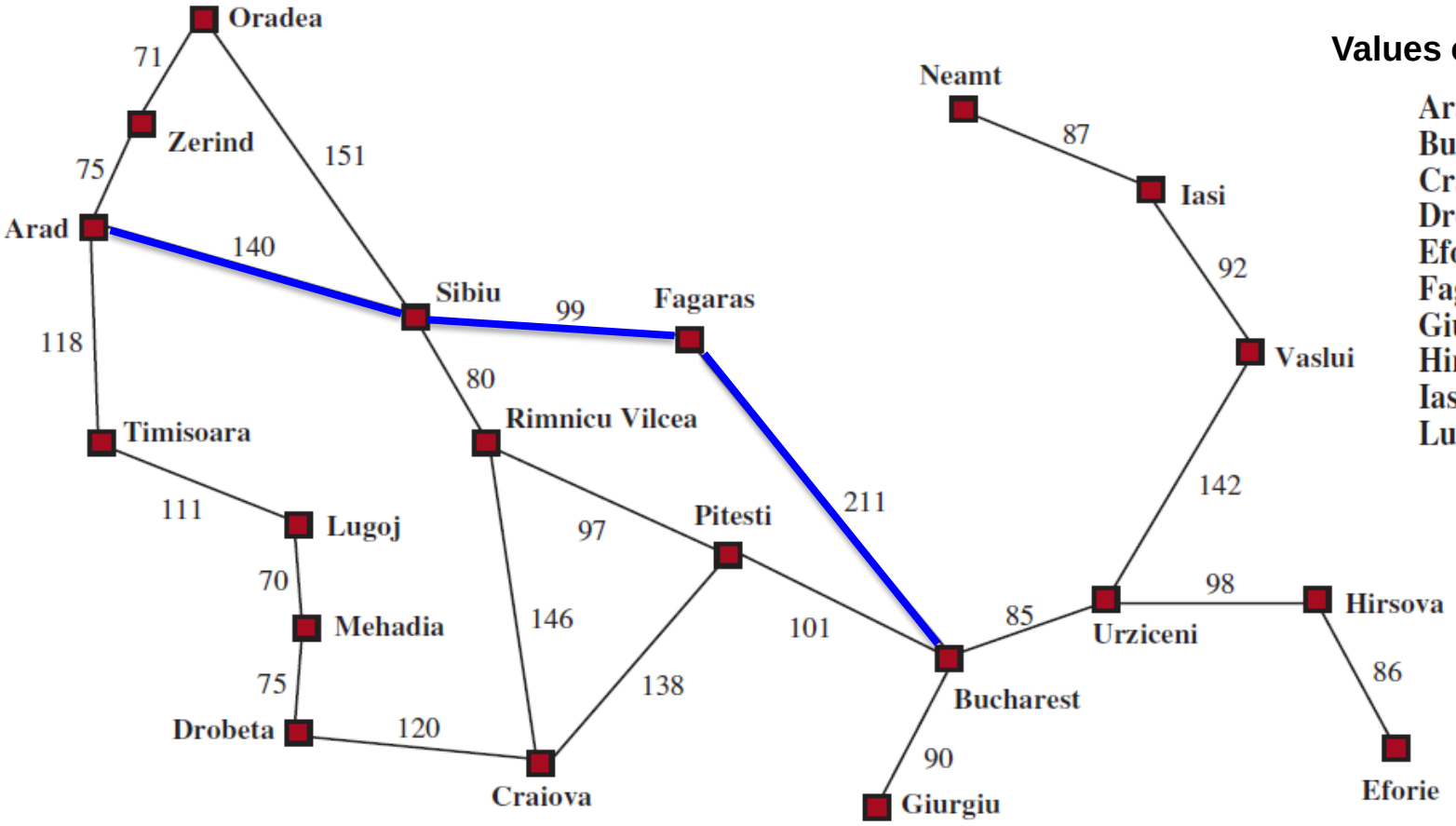
Thuật toán tìm kiếm tham lam



- Tiếp theo mở rộng Fagaras → Bucharest -> dừng.

Thuật toán tìm kiếm tham lam

- Nghiệm: Arad – Sibiu – Fagaras – Bucharest: 450
- Tối ưu (optimality)? No



Values of hSLD—straight-line distances to Bucharest

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Thuật toán tìm kiếm tham lam

- Ví dụ 2. tìm đường đi ngắn nhất từ đỉnh S đến đỉnh G cho bản đồ sau, biết:
  - Tại mỗi ô chỉ di chuyển 4 hướng trái, phải, lên, xuống nhưng không được di vào ô màu xám.
  - Hàm heuristic $h(n)$ là (i) khoảng cách Euclide và (ii) khoảng cách Manhattan.

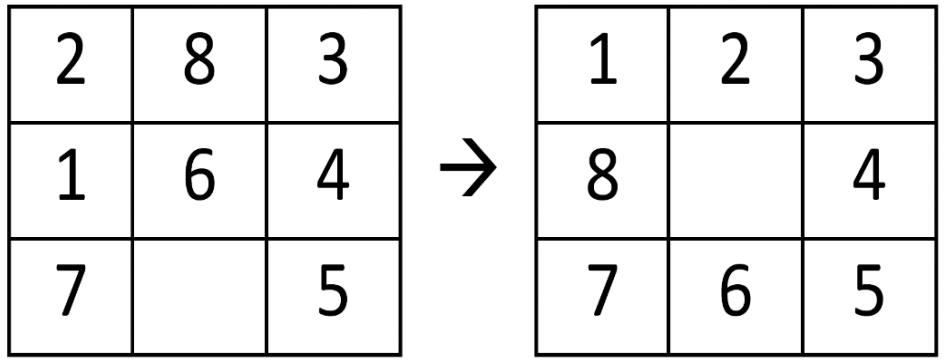
				G
		S		

			$\sqrt{1}$	G
		$\sqrt{5}$	$\sqrt{2}$	$\sqrt{1}$
	$\sqrt{13}$	S	$\sqrt{5}$	$\sqrt{4}$
		$\sqrt{13}$	$\sqrt{10}$	

Thuật toán tìm kiếm tham lam

- Bài tập
  - Tìm dãy dịch chuyển từ trạng thái đầu đến trạng thái đích cho bài toán 8 số, trong đó hàm:
    - Heuristic $h(n)$ là tổng số ô nằm sai vị trí so với vị trí ở trạng thái đích.

      $h(n) = 1 + 1 + 0 + 1 + 1 + 0 + 0 + 0 = 4$

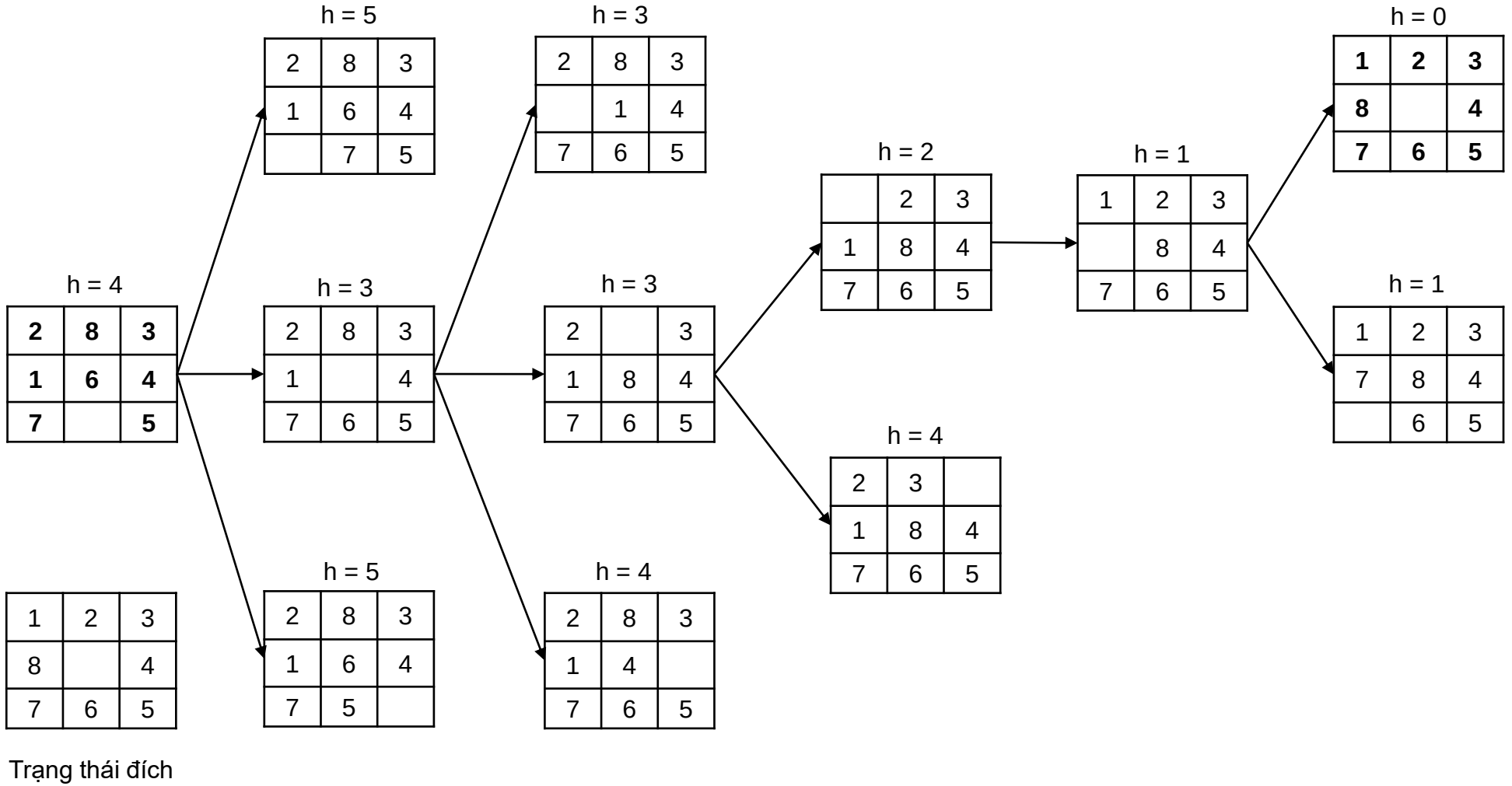


    - Heuristic $h(n)$ là tổng khoảng cách từ tất cả cả ô đến các ô trong trạng thái đích (Manhattan distance).

      $h(n) = 1 + 2 + 0 + 1 + 1 + 0 + 0 + 0 = 5$

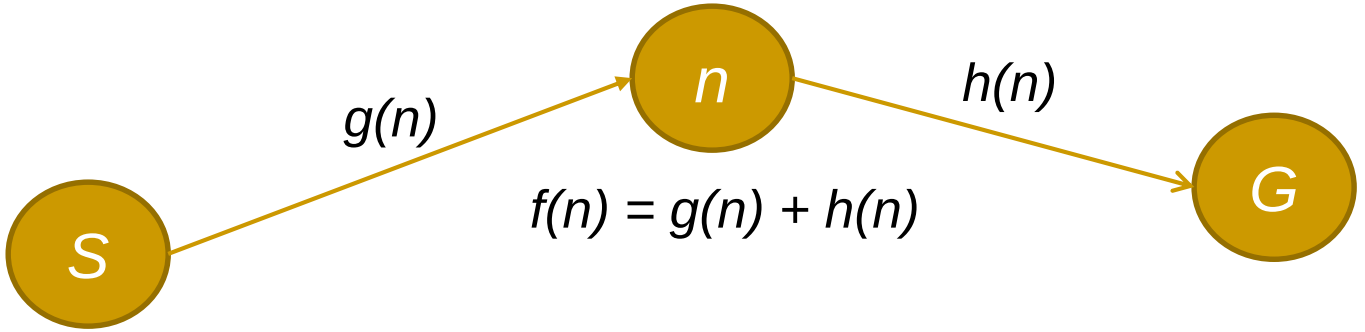
Thuật toán tìm kiếm tham lam

- Heuristic $h(n)$ là tổng số ô nằm sai vị trí so với vị trí ở trạng thái đích.



Thuật toán tìm kiếm A*

- Hàm đánh giá: $f(n) = g(n) + h(n)$
  - $g(n)$: chi phí từ trạng thái đầu đến n.
  - $h(n)$: đánh giá chi phí từ trạng thái n đến đích.
  - $f(n)$: đánh giá chi phí từ trạng thái đầu đến trạng thái đích đi qua n.



- Trường hợp đặc biệt
  - Uniform-cost search: $f(n) = g(n)$, tức là $h(n) = 0$.
  - Greedy best-first search: $f(n) = h(n)$, tức là $g(n) = 0$.

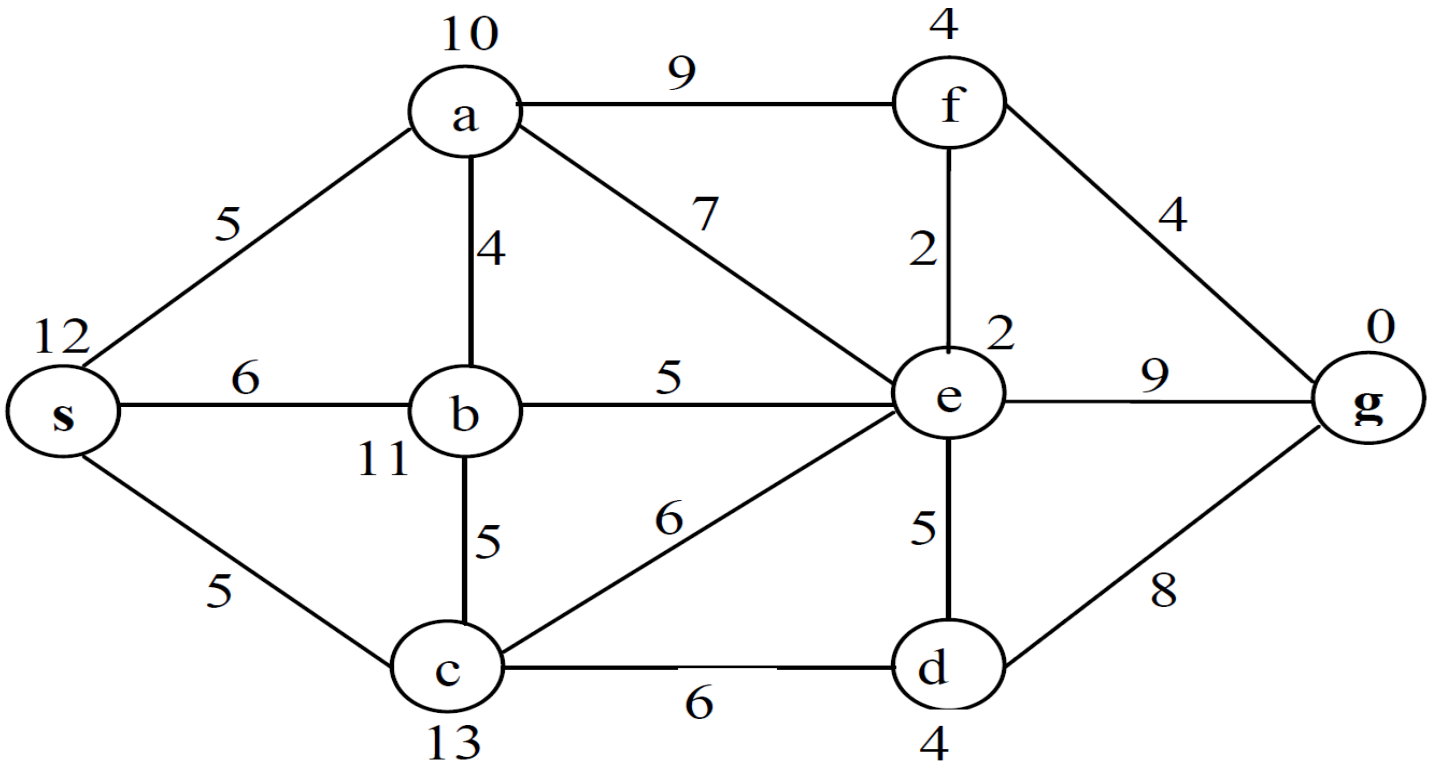
Thuật toán tìm kiếm A*

- Sử dụng thuật toán UCS với $f(n) = g(n) + h(n)$.

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
    node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
    frontier ← a priority queue ordered by PATH-COST, with node as the only element
    explored ← an empty set
    loop do
        if EMPTY?(frontier) then return failure
        node ← POP(frontier)  /* chooses the lowest-cost node in frontier */
        if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                frontier ← INSERT(child, frontier)
            else if child.STATE is in frontier with higher PATH-COST then
                replace that frontier node with child
```

Thuật toán tìm kiếm A*

- Ví dụ 1. tìm đường đi từ s->g và vẽ cây tìm kiếm bằng các thuật toán:
  - UCS (Uniform – Cost – Search)
  - Greedy search
  - A* search

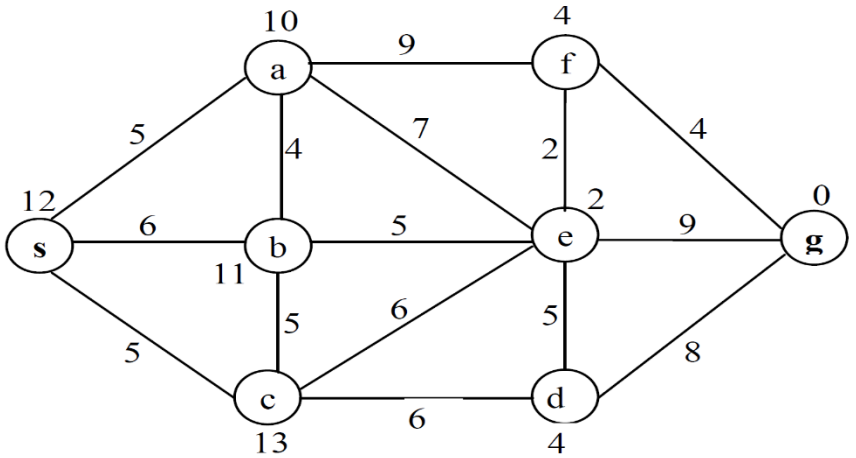


Thuật toán tìm kiếm A*

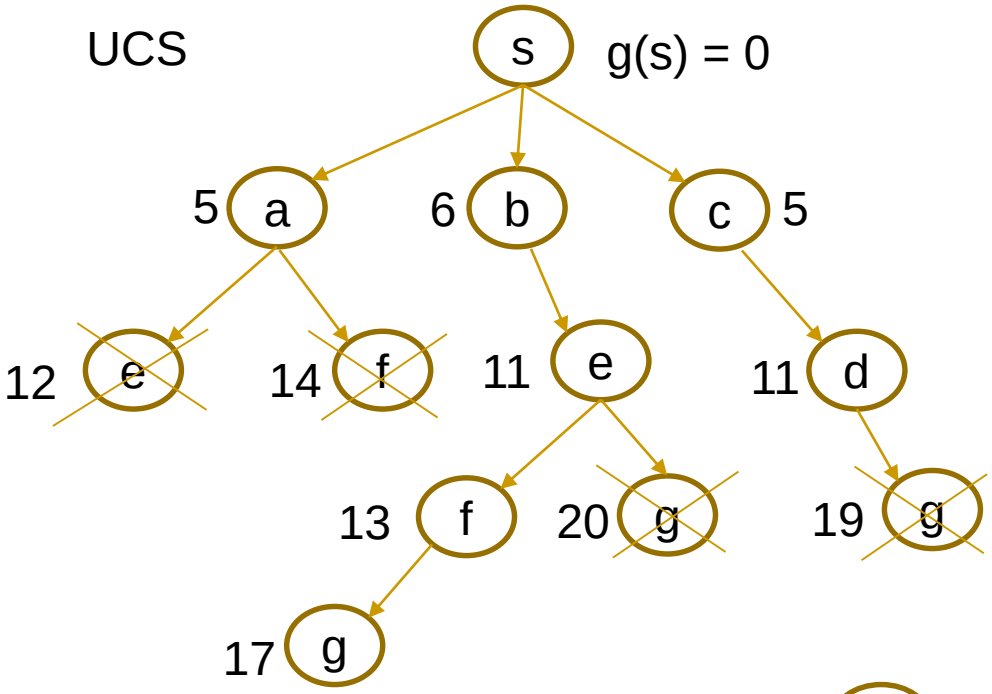
Iter.	Node	Explored	Frontier (**UCS: f(n) = g(n)**)
0	s	{}	(s,0)
1	s	s	(a,5), (b,6), (c,5)
2	a	s,a	(b,6), (c,5), (e,12), (f,14)
3	c	s,a,c	(b,6), ~~(e,12)~~, (f,14), (e,11), (d,11)
4	b	s,a,c,b	(f,14), (e,11), (d,11)
5	e	s,a,c,b,e	~~(f,14)~~,(d,11), (f,13), (g,20)
6	d	s,a,c,b,e,d	(f,13), ~~(g,20)~~, (g,19)
7	f	s,a,c,b,e,d,f	~~(g,19)~~, (g,17)
8	g	Dừng → đường đi:s – b - e – f – g: 17	

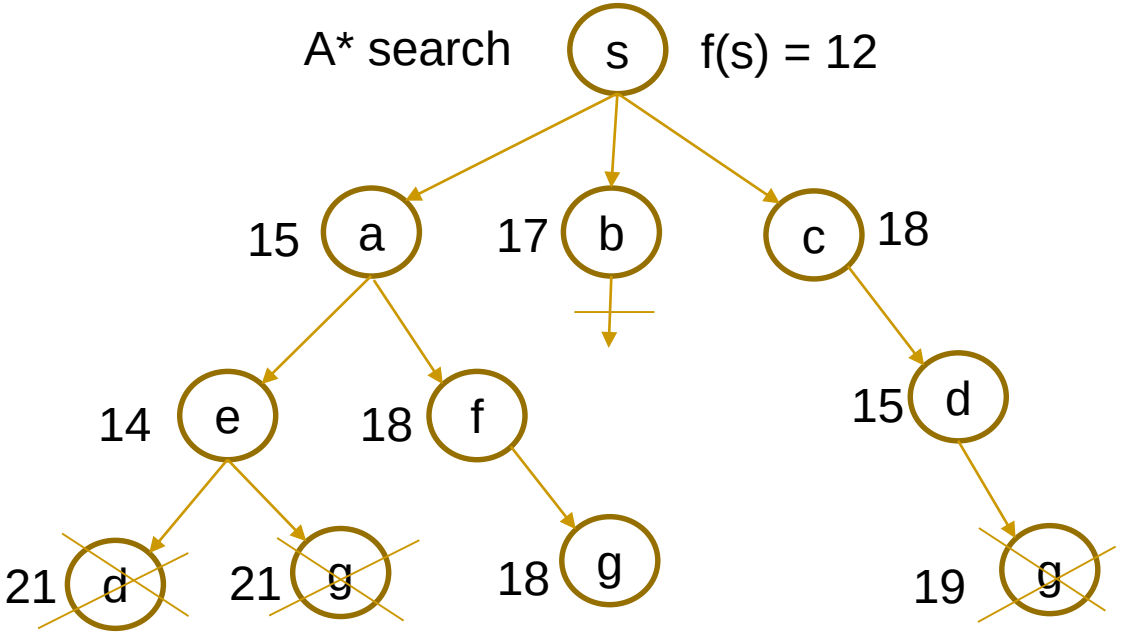
Iter.	Node	Explored	Frontier (**A* Search: f(n) = g(n) + h(n)**)
0	s	{}	(s,12)
1	s	s	(a,15), (b,17), (c,18)
2	a	s,a	(b,17), (c,18), (e,14), (f,18)
3	e	s,a,e	(b,17), (c,18), (f,18), (d,21),(g,21)
4	b	s,a,e,b	(c,18), (f,18), (d,21),(g,21)
5	c	s,a,e,b,c	(f,18), ~~(d,21)~~,(g,21),(d,15)
6	d	s,a,e,b,c,d	(f,18),~~(g,21)~~, (g,19)
7	f	s,a,e,b,c,d,f	~~(g,19),~~ (g,18)
8	g	Dừng: đường đi: s – a –f – g:18	

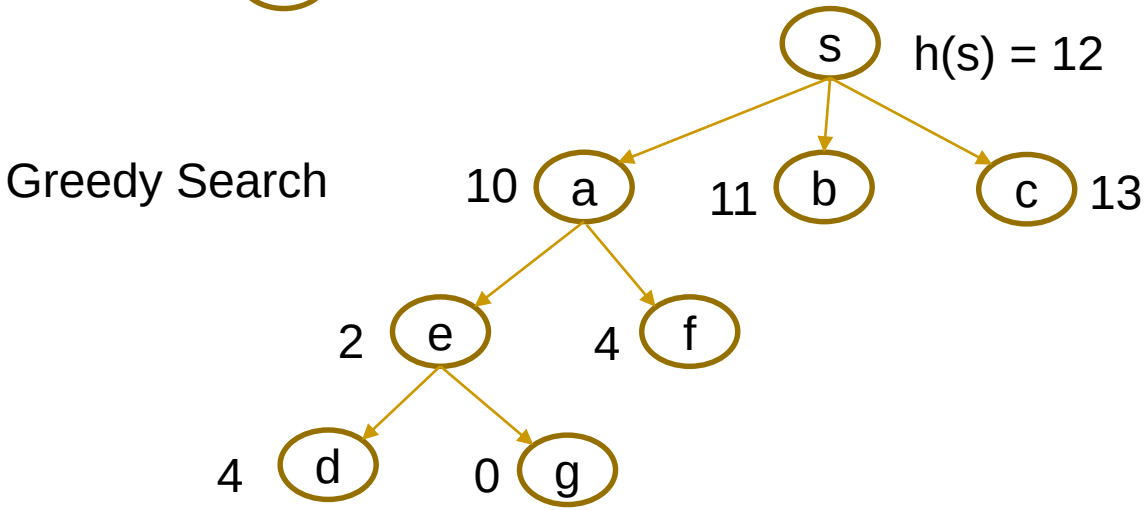
Iter.	Node	Explored	Frontier (**GREEDY: f(n) = h(n)**)
0	s	{}	(s,12)
1	s	s	(a,10), (b,11), (c,13)
2	a	s,a	(b,11), (c,13), (e,2), (f,4)
3	e	s,a,e	(b,11), (c,13), (e,2), (g,0)
4	g	Dừng: đường đi: s – a – e – g: 21	

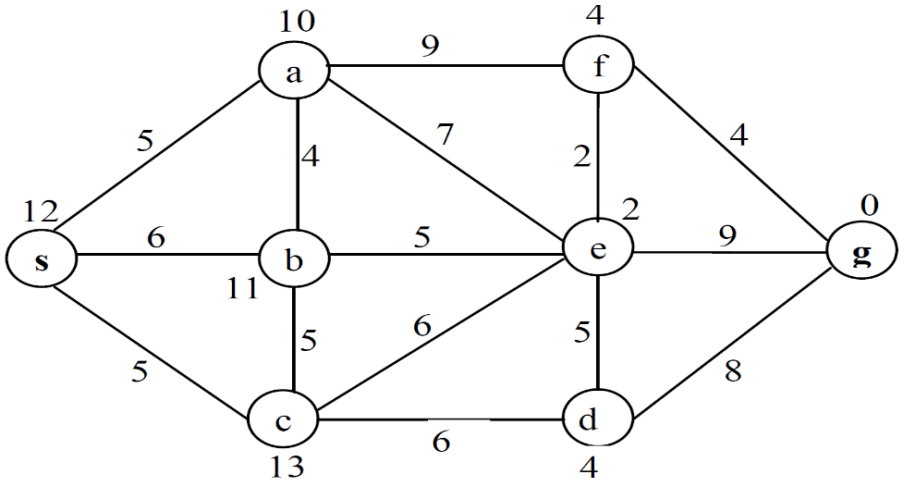


Thuật toán tìm kiếm A*



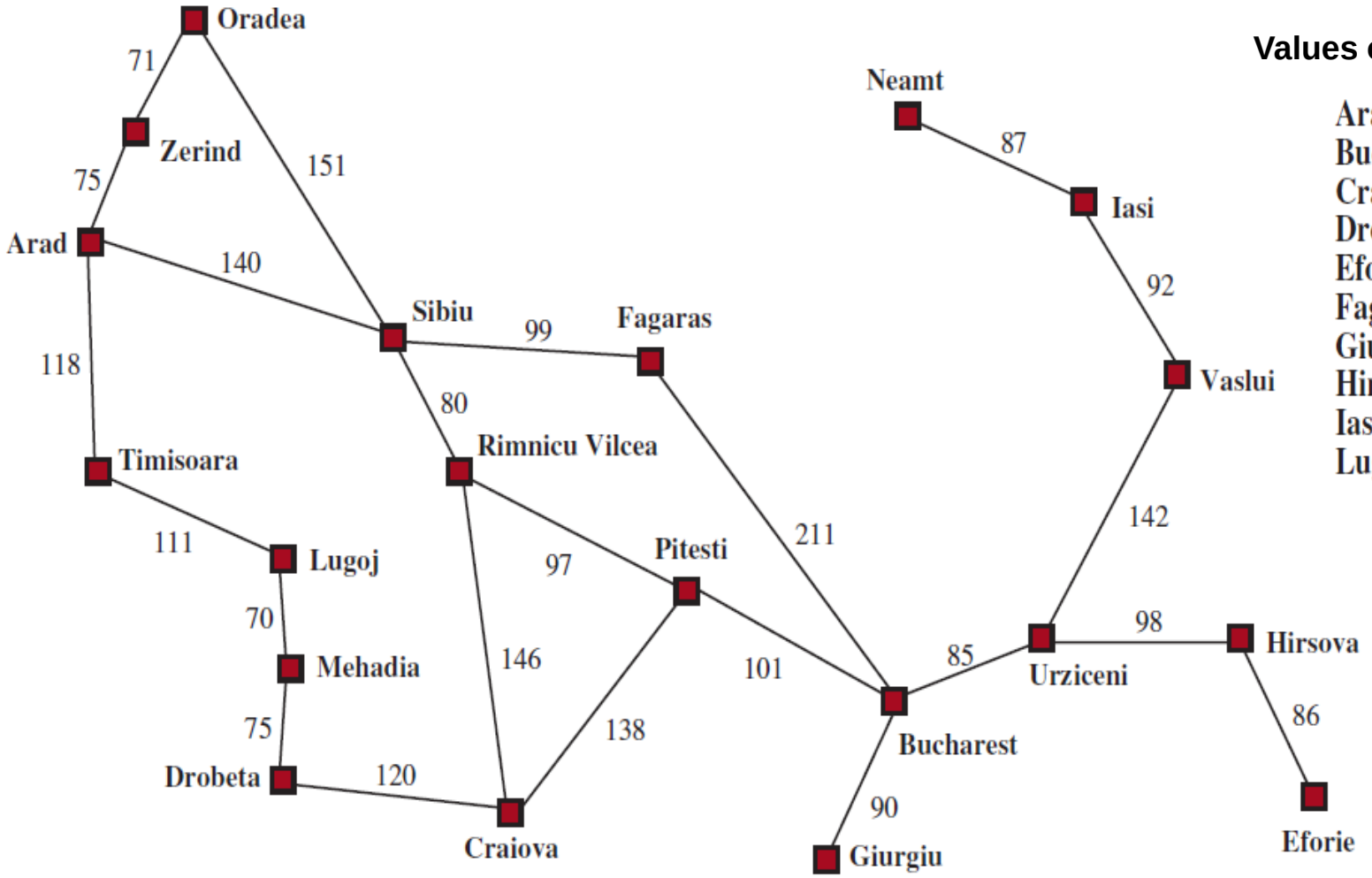






Thuật toán tìm kiếm A*

- Ví dụ 2. tìm đường đi từ Arad → Bucharest

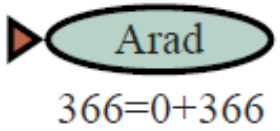


Values of hSLD—straight-line distances to Bucharest

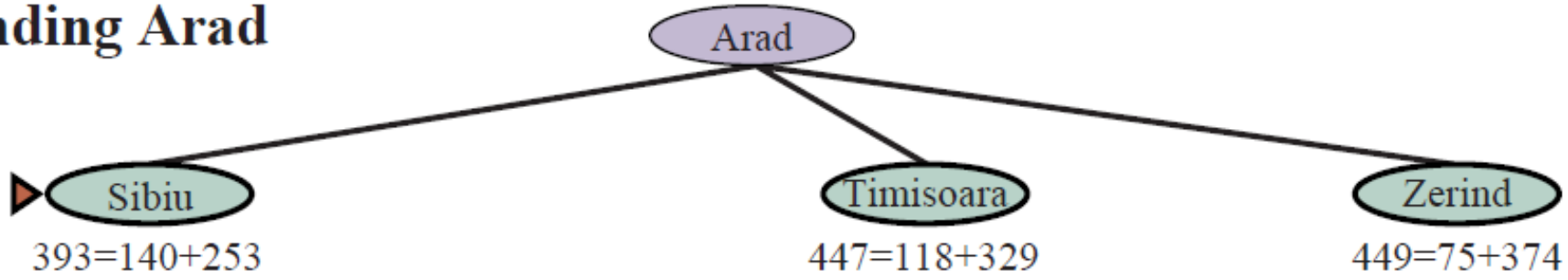
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Thuật toán tìm kiếm A*

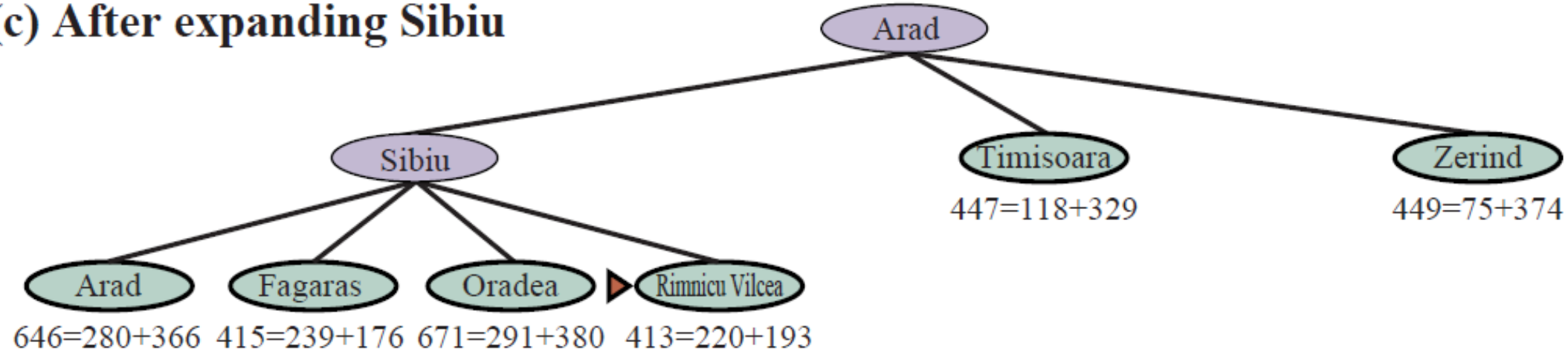
(a) The initial state



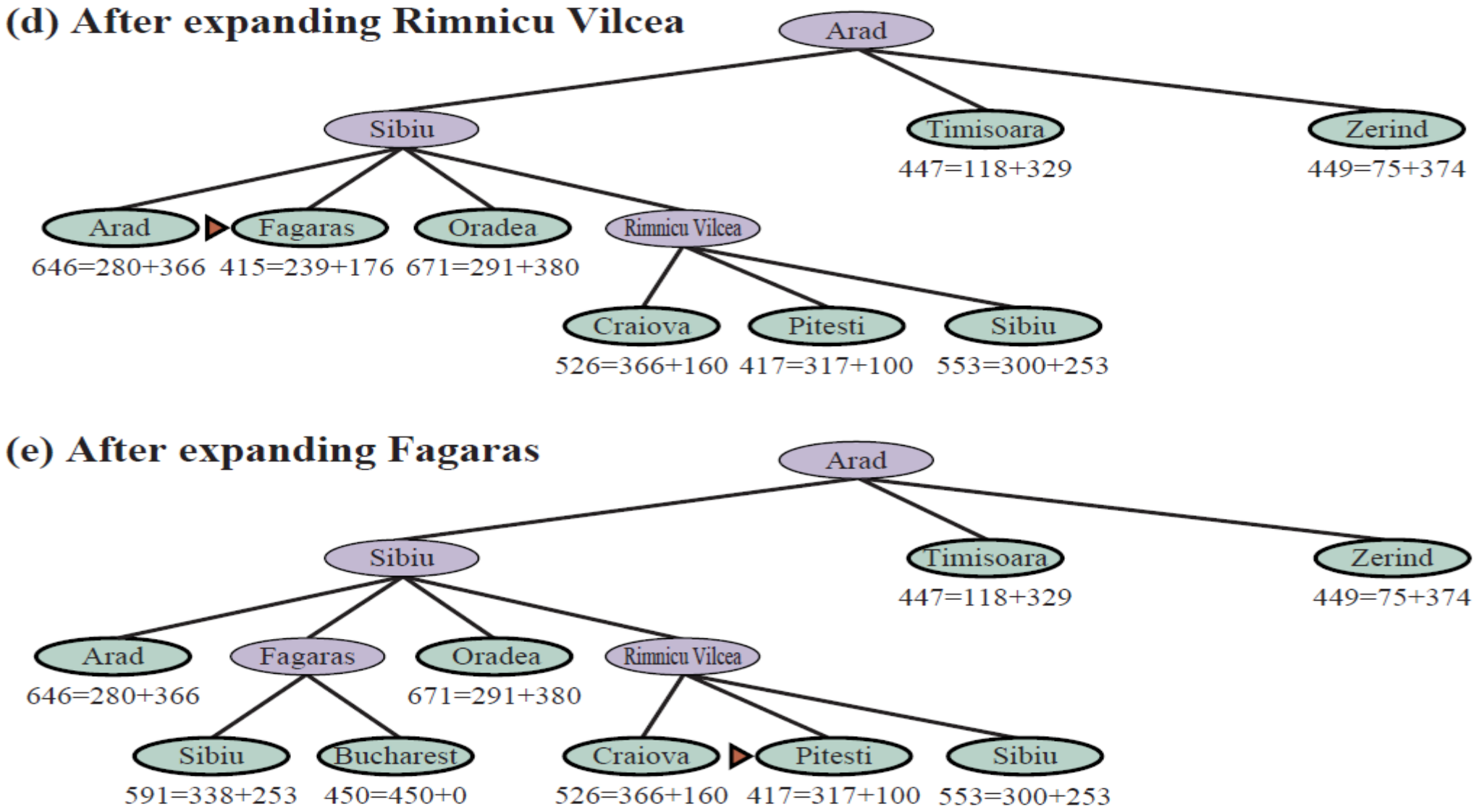
(b) After expanding Arad



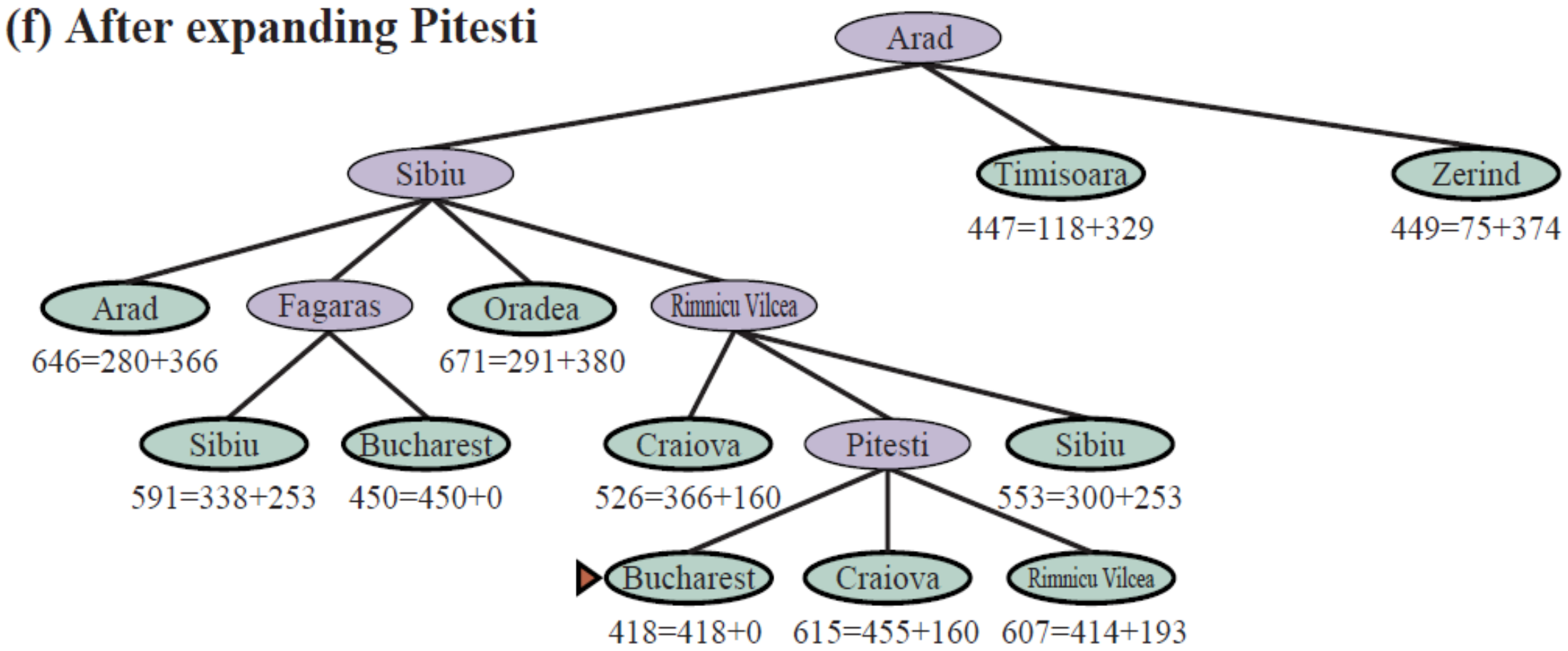
(c) After expanding Sibiu



Thuật toán tìm kiếm A*



Thuật toán tìm kiếm A*

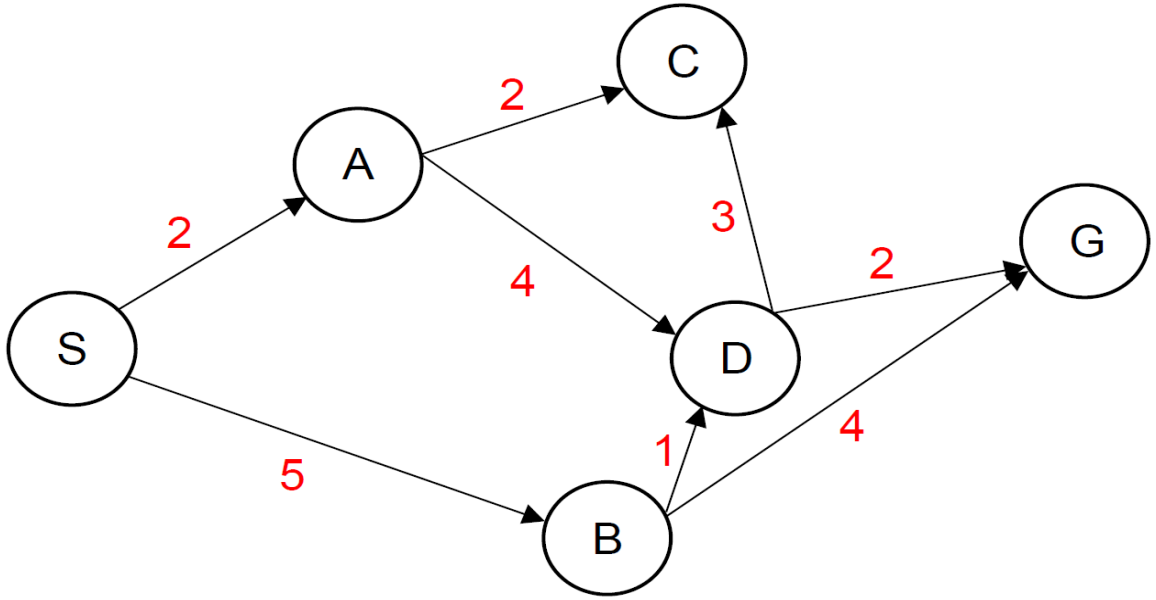


Thuật toán tìm kiếm A*

- Các tính chất của tìm kiếm A*
  - Hoàn chỉnh (completeness)? Yes
  - Tối ưu (optimality)?
  - Độ phức tạp thời gian (time complexity)?
  - Độ phức tạp không gian (space complexity)?

Thuật toán tìm kiếm A*

- Điều kiện tối ưu
  - Một hàm heuristic $h(n)$ là **chấp nhận được (admissible heuristic)** nếu $h(n) \leq h^*(n)$ với mọi nút n, trong đó $h^*(n)$ là giá thực tế từ nút n đến đích.
    - Nếu $h(n)$ là khoảng cách Euclidean *thì* $h(n)$ là một heuristic chấp nhận được.
  - Hàm heuristic *h(n)* của đồ thị sau có phải là một hàm heuristic chấp nhận được không, vì sao?
  - Hàm *h(n)* không phải hàm chấp nhận được vì: h(D) = 4 > h*(D) = 2

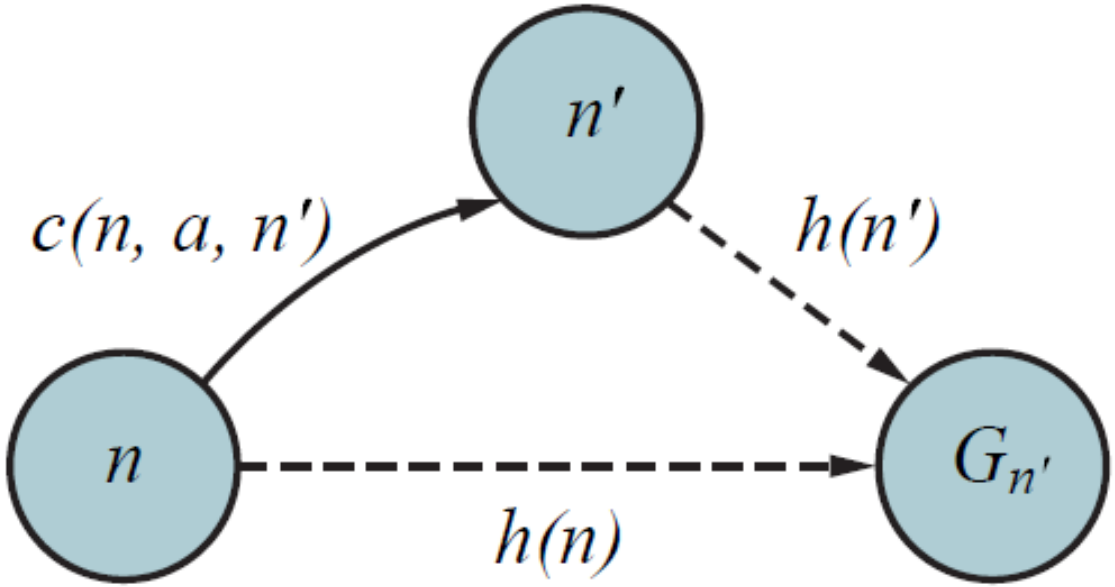


Heuristic Values

A = 2	C = 1	S = 10
B = 3	D = 4	G = 0

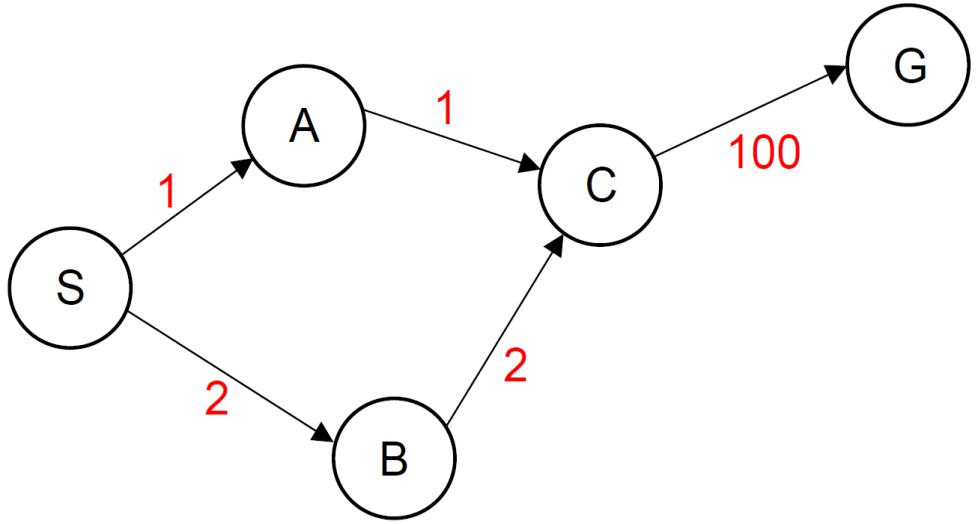
Thuật toán tìm kiếm A*

- Một hàm $h(n)$ là **đồng nhất (consistent)** hay đơn điệu (monotonicity) nếu $h(n) \leq c(n,a,n') + h(n')$ với mọi nút n, trong đó $h(n')$ là hàm đánh giá tại nút n' được sinh ra bởi toán tử a từ nút n và $c(n,a,n')$ là giá thực tế từ nút n đến n'.
  - Tính đồng nhất là bất đẳng tam giác (triangle inequality).
  - Một hàm $h(n)$ là đồng nhất thì $h(n)$ là chấp nhận được, nhưng ngược lại thi chưa chắc đúng.



Thuật toán tìm kiếm A*

- Ví dụ hàm heuristic $h(n)$ của đồ thị sau có phải là một hàm đồng nhất không, vì sao?
- Xét tại đỉnh C, ta có:
  - h(C) = 90, c(C,a,G) = 100, h(G) = 0 → thõa mãn
- Xét tại đỉnh B, ta có:
  - h(B) = 1, c(B,a,C) = 2, h(C) = 90 → thõa mãn
- Xét tại đỉnh A, ta có:
  - h(A) = 100, c(A,a,C) = 1, h(C) = 90
  - h(A) > c(A,a,C) + h(C) → vi pham điều kiện.
- Hàm h(n) không đồng nhất.



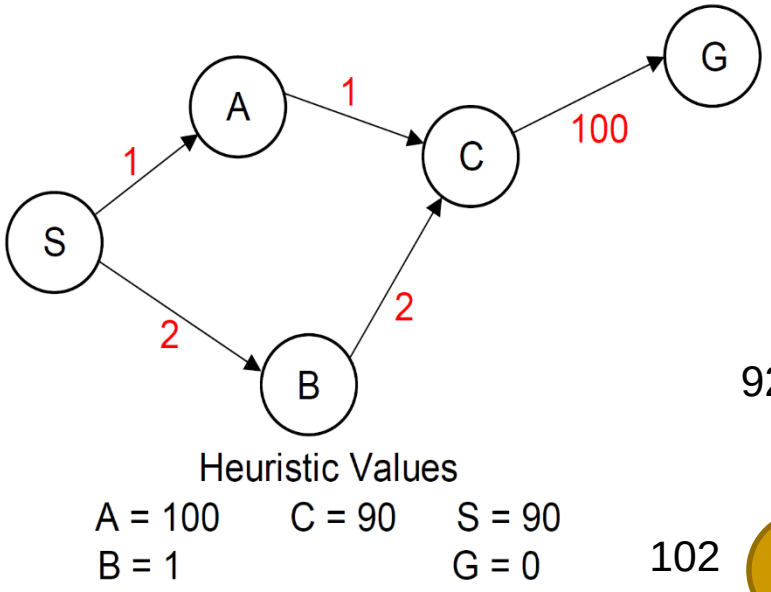
Heuristic Values

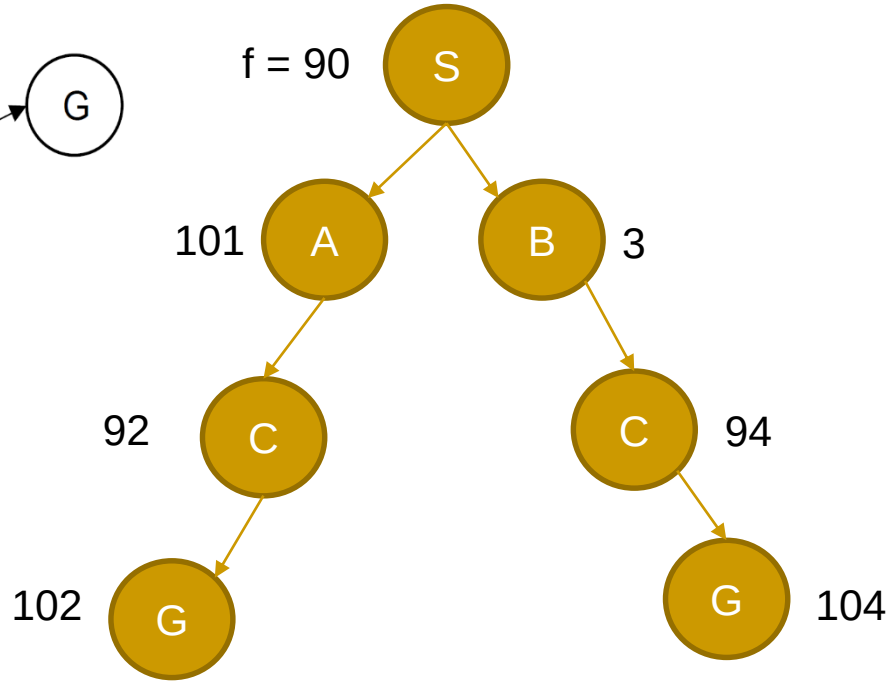
A = 100 C = 90 S = 90

B = 1 G = 0

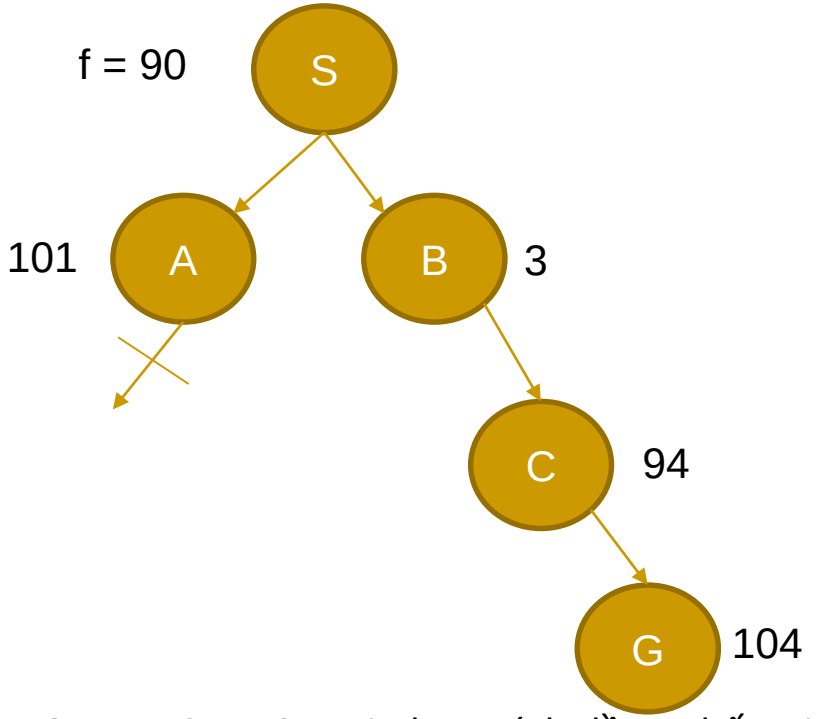
Thuật toán tìm kiếm A*

- "The tree-search version of A* is optimal if $h(n)$ is admissible, while the graph-search version is optimal if $h(n)$ is consistent."
- Tìm đường đi ngắn nhất từ S đến G của đồ thị sau theo A* dựa trên TREE-SEARCH và GRAPH-SEARCH. Thuật toán nào tìm được đường đi ngắn nhất, vì sao?





TREE-SEARCH: thõa mãn tính chấp nhận được.



GRAPH-SEARCH: Vi phạm tính đồng nhất tại đỉnh A.

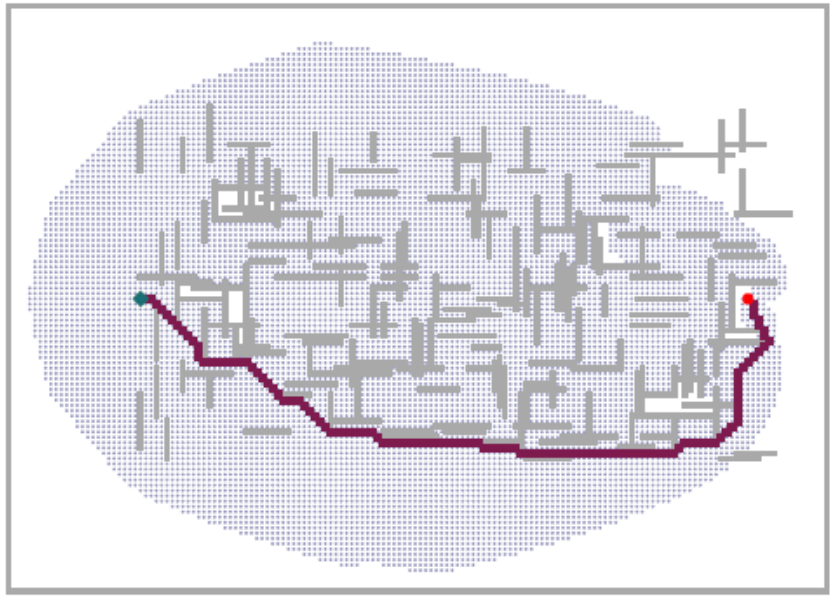
Thuật toán tìm kiếm A*

- Hoàn chỉnh (completeness)? Yes
- Tối ưu (optimality)?
  - Thuật toán A* dựa trên TREE-SEARCH là tối ưu nếu $h(n)$ là chấp nhận được (admissible): $h(n) \leq h^*(n)$.
  - Thuật toán A* dựa trên GRAPH-SEARCH là tối ưu nếu $h(n)$ là đồng nhất (consistent): $h(n) \leq c(n,a,n') + h(n')$.
  - Xem các chứng minh trong tài liệu [1].
- Độ phức tạp thời gian (time complexity)?
  - Xem tài liệu [1]
- Độ phức tạp không gian (space complexity)?
  - Xem tài liệu [1]

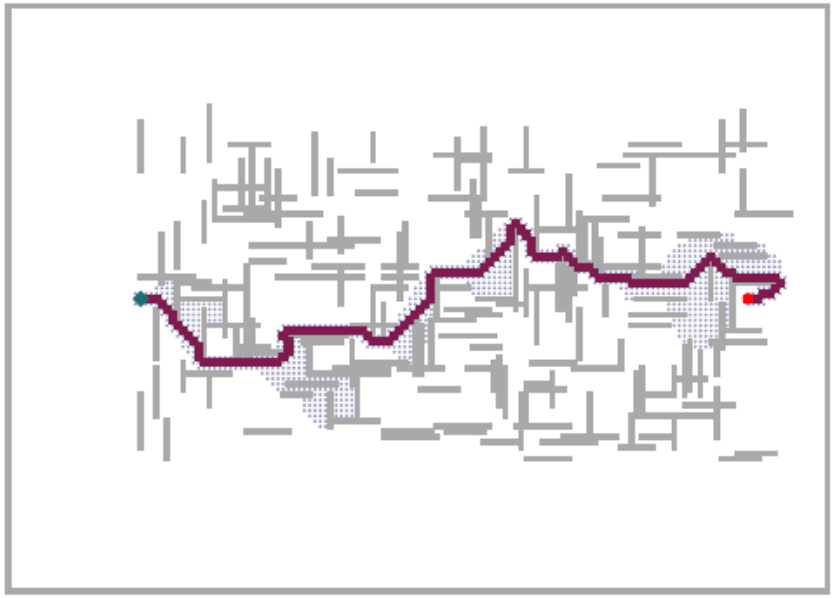
Thuật toán tìm kiếm A* với trọng số

- Tên gọi: **weighted A* search**
- Hàm đánh giá: *f(n) = g(n) + W*h(n), W > 1.*
  - *A* search: W = 1*
  - *UCS: W = 0*
  - *Greedy best first search: W = ∞*

- Ví dụ
  - *(a) W* = 0 (UCS)
  - *(b) W = 2*



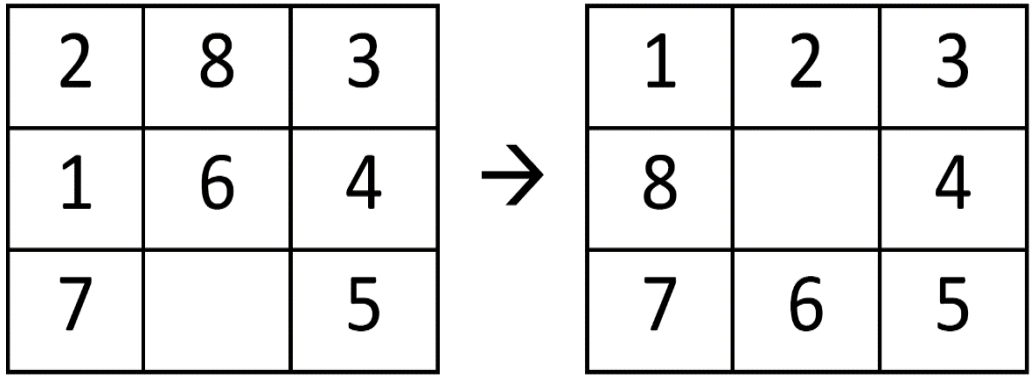
(a)



(b)

Ảnh hưởng hàm heuristic

- Hai hàm heuristics chấp nhận được $h_1(n)$ và $h_2(n)$, điều kiện để một trong 2 hàm tìm được nghiệm nhanh hơn?
- Ví dụ hai hàm heuristic $h_1(n)$ và $h_2(n)$ như sau:
  - $h_1(n)$: số ô nằm sai vị trí tro với trạng thái đích.
  - $h_2(n)$: tổng khoảng cách từ tất cả cả ô đến các ô trong trạng thái đích (Manhattan distance).
  - Nên chọn $h_1(n)$ hay $h_2(n)$?

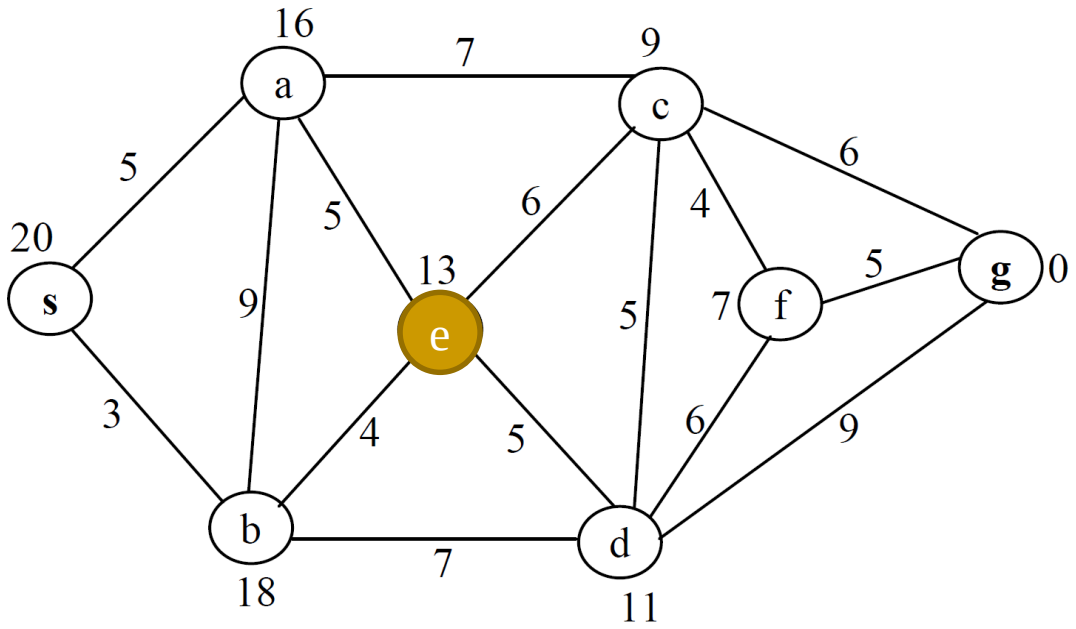


Ảnh hưởng hàm heuristic

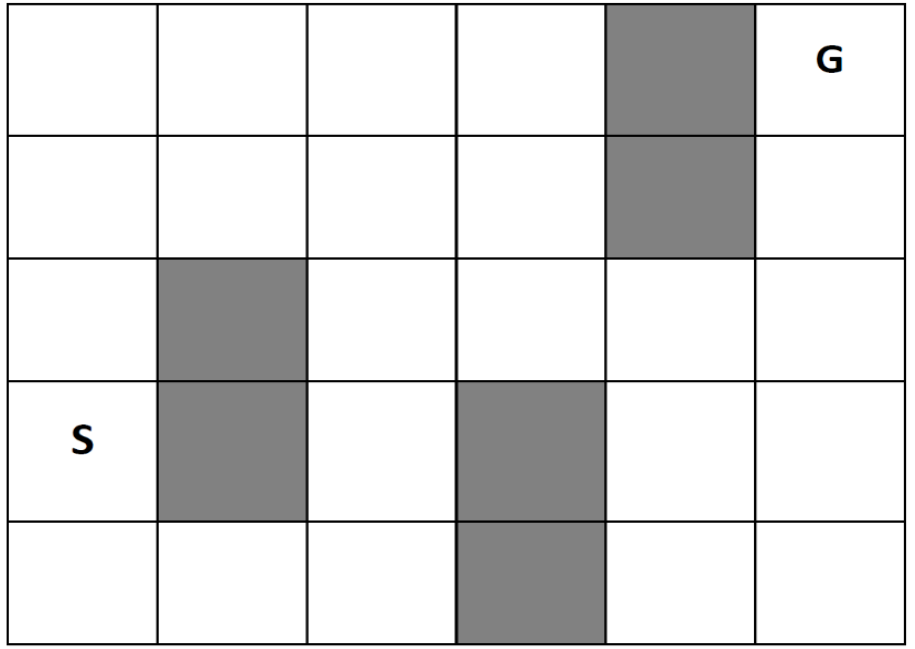
- Với mọi n, nếu $h_2(n) > h_1(n)$ *thì* $h_2(n)$ tốt hơn $h_1(n)$.
- Hai hay nhiều hàm heuristics chấp nhận được có thể được kết hợp để tạo ra một hàm heuristics chấp nhận khác.
  - Ví dụ có 2 hàm heuristic chấp nhận được $h_1(n)$ và $h_2(n)$ thì $h_3(n) = max(h_1(n),h_2(n))$ cũng là một hàm heuristics chấp nhận được.

Bài tập

- Sử dụng các giải thuật BFS, UCS, Best-First Search và A* mô tả các bước lặp và vẽ cây tìm kiếm với các đồ thị sau.



a) Đồ thị 1: các số trên đỉnh n là hàm $h(n)$.



b) Đồ thị 2, tại mỗi ô di chuyển đến 4 ô lân cận theo 4 hướng trái, phải, lên, xuống nhưng không di chuyển vào ô màu xám.