Tìm kiếm có thông tin (Informed search)

Từ Minh Phương
Bộ môn: khmt

Nội dung

- Best-first search
- Greedy best-first search
- A* search
- Heuristics
- Local search algorithms
- Hill-climbing search
- Simulated annealing search
- Local beam search
- Genetic algorithms

Mở rộng nút tốt nhất trước tiên

- (Best-first search)
- Ý tưởng:
  - Sử dụng hàm ước lượng $f(n)$ cho mỗi nút: là ước lượng độ tốt của nút n

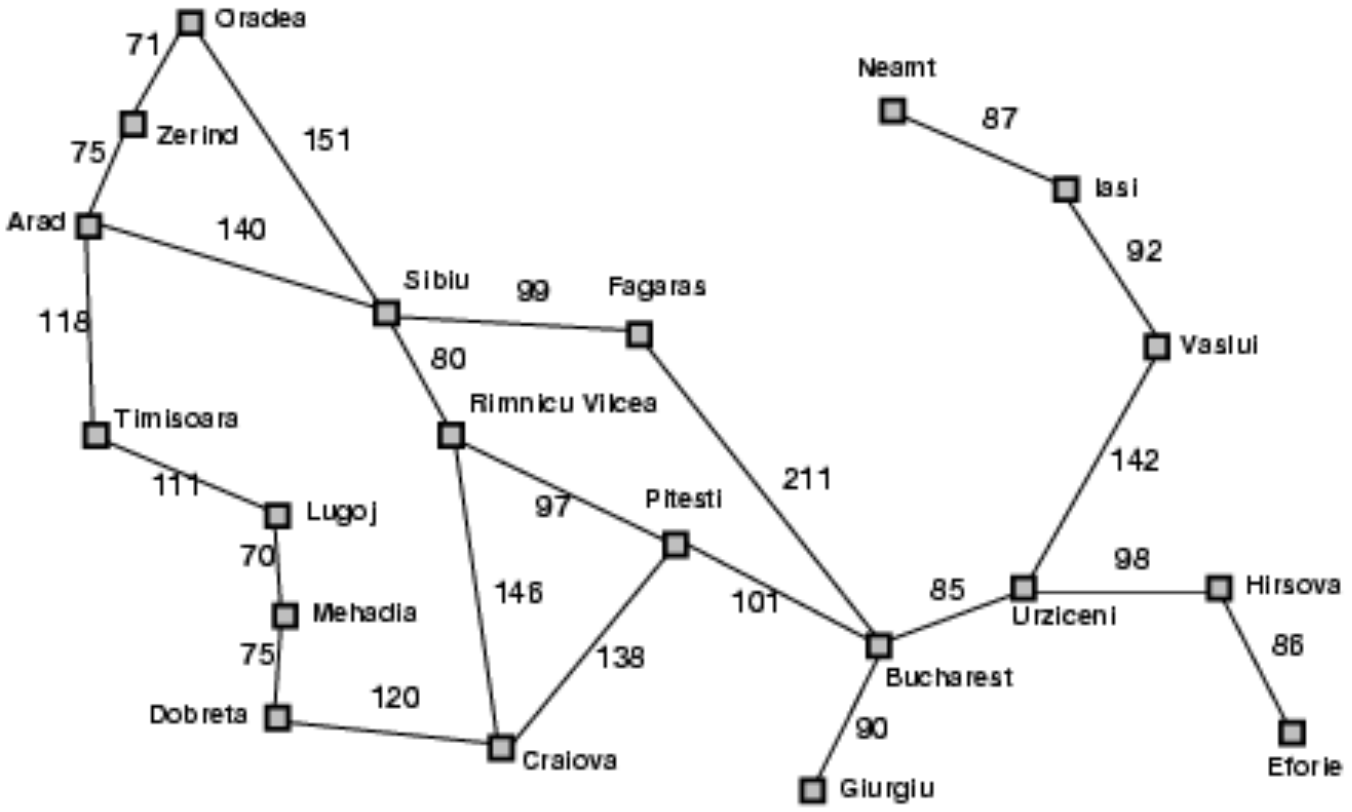
  → Mở rộng nút n có giá trị $f(n)$ nhỏ nhất

- Triển khai thuật toán:

  Sắp xếp các nút cần mở rộng theo thứ tự tăng dần của hàm $f(n)$

- Các trường hợp riêng:
  - Tìm kiếm tham lam mở rộng nút tốt nhất trước (greedy best-first search)
  - A^*

Romania with step costs in km



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

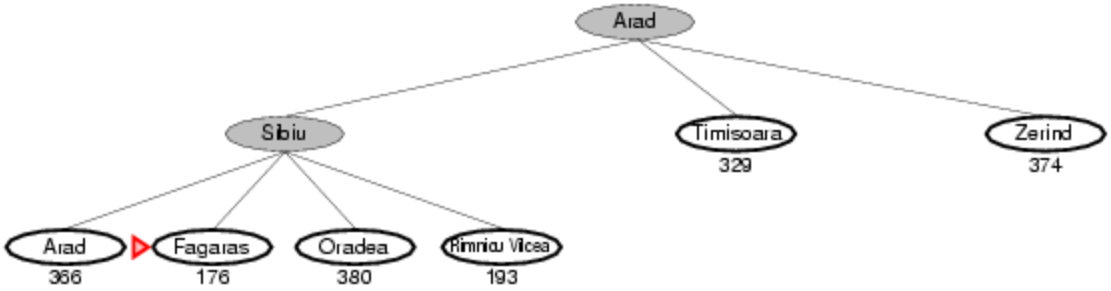
Tìm kiếm tham lam mở rộng nút tốt nhất trước

- Sử dụng hàm $f(n) = h(n)$ (heuristic)
- = ước lượng giá thành đường đi từ *n* tới đích
- Ví dụ., $h_{SLD}(n)$ = đường chim bay từ *n* tới đích (Bucharest)
- Phương pháp này mở rộng nút trông có vẻ gần đích nhất

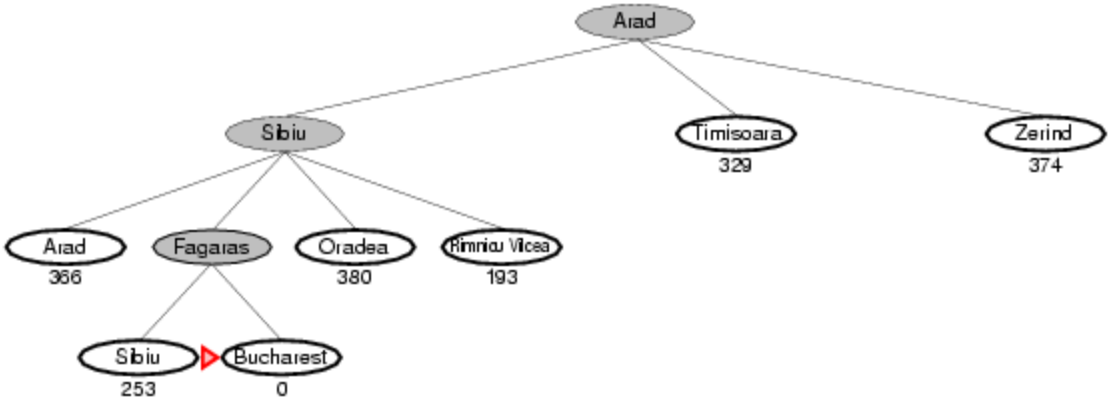
Ví dụ

Ví dụ

Ví dụ


Arad
Sibiu
Timisoara
329
Zerind
374
Arad
366
Fagaras
176
Oradea
380
Rimnicu Vilcea
193

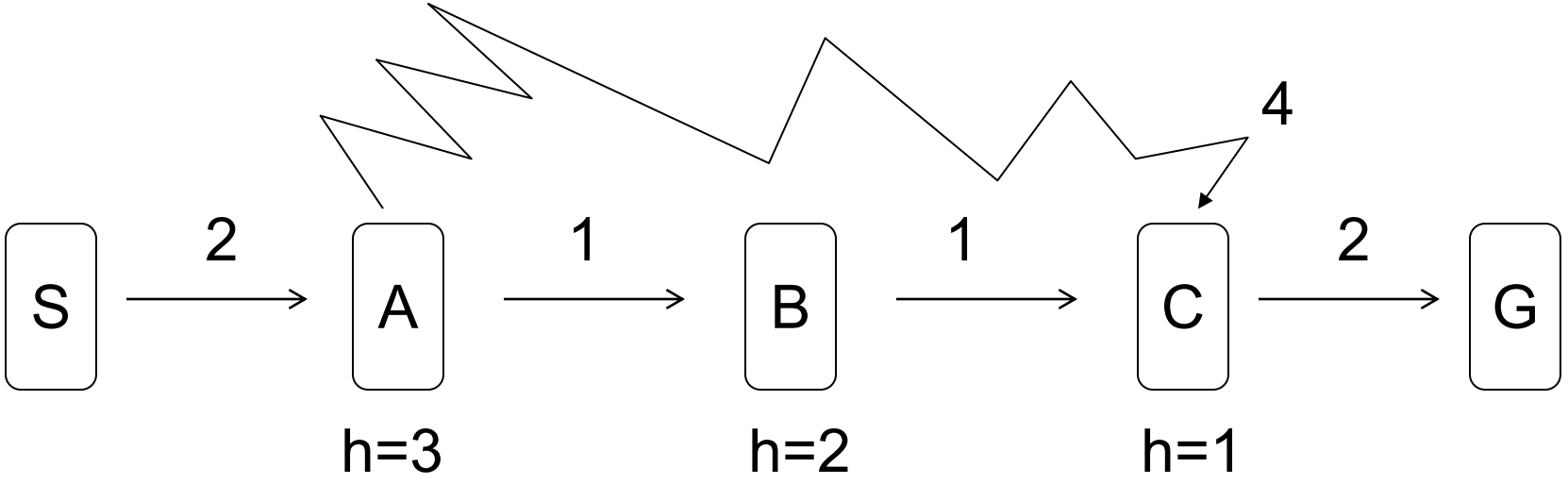
Ví dụ


Arad
Sibiu
Timisoara
329
Zerind
374
Arad
366
Fagaras
Oradea
380
Rimnicu Vilcea
193
Sibiu
253
Bucharest
0

Đặc điểm

- Đầy đủ? Không- có thể bị lặp, ví dụ, Iasi → Neamt → Iasi → Neamt →
-
- Thời gian? $O(b^m)$, có thể nhanh hơn nhiều nếu có heuristic tốt
- Không gian? $O(b^m)$ – Lưu tất cả nút trong bộ nhớ
- Tối ưu? Không

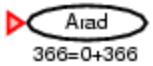
Tìm kiếm tham lam không cho kết quả tối ưu



Thuật toán A^*

- Khắc phục các nhược điểm của tìm kiếm tham lam
- Ý tưởng: không tiếp tục mở rộng các đường đi đang có giá thành lớn.
- Hàm đánh giá $f(n) = g(n) + h(n)$
- $g(n)$ = giá thành đường đi từ nút xuất phát đến n
- $h(n)$ = giá thành ước lượng từ n tới đích
- $f(n)$ = giá thành ước lượng từ nút xuất phát, qua n tới đích

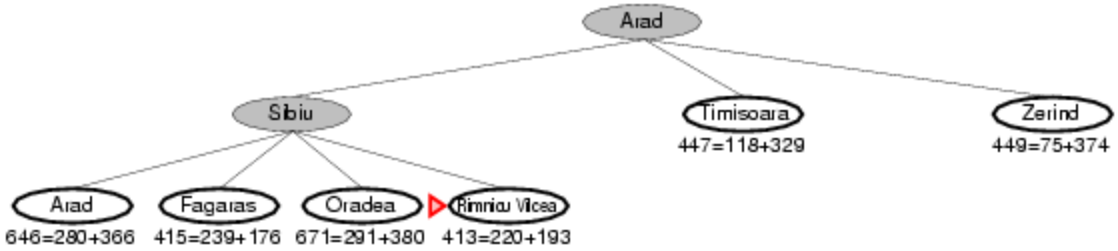
A* :ví dụ



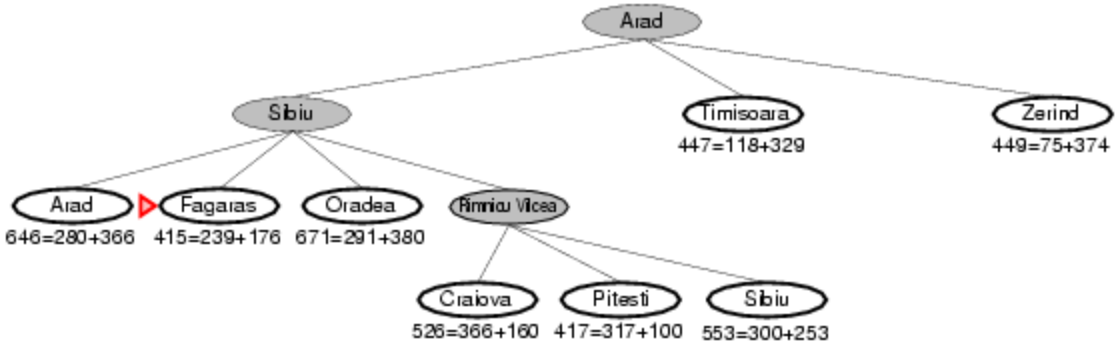
A* : ví dụ


Arad
Sibiu
393=140+253
Timisoara
447=118+329
Zerind
449=75+374

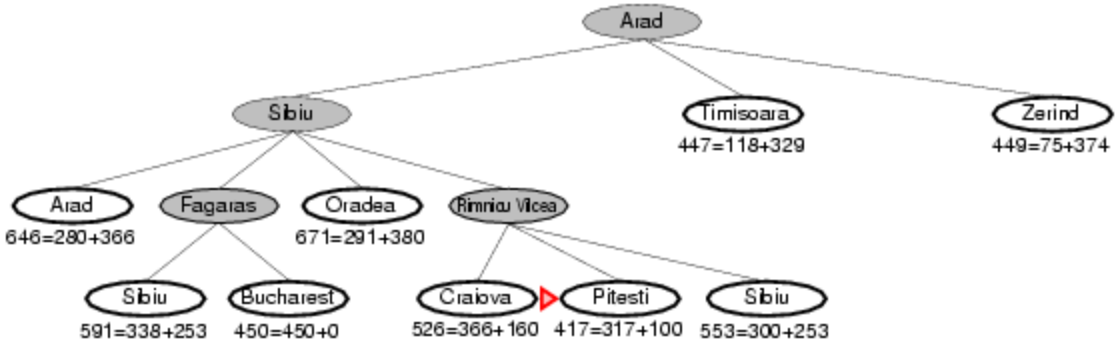
A* :ví dụ


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
413=220+193

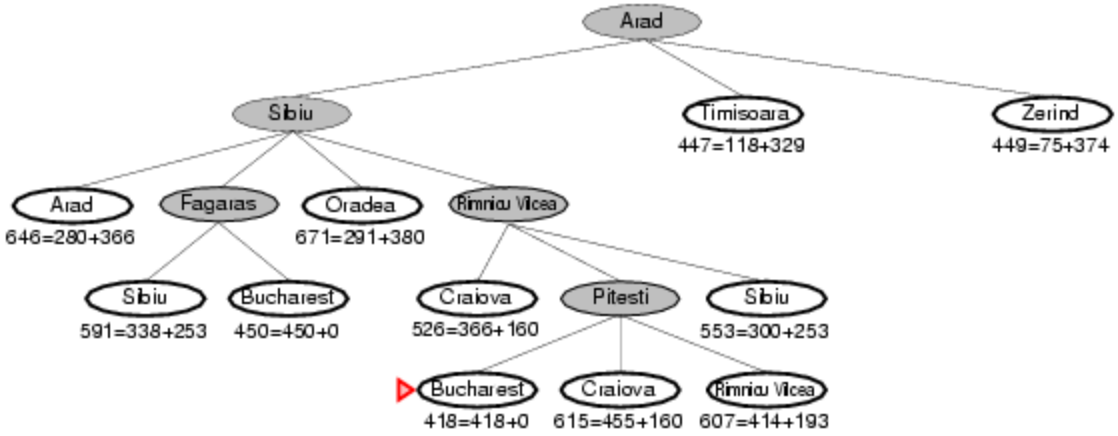
A* :ví dụ


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
415=239+176
Oradea
671=291+380
Rimnicu Vilcea
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

A* :ví dụ


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
417=317+100
Sibiu
553=300+253

A* :ví dụ


Arad
Sibiu
Timisoara
447=118+329
Zerind
449=75+374
Arad
646=280+366
Fagaras
Oradea
671=291+380
Rimnicu Vilcea
Sibiu
591=338+253
Bucharest
450=450+0
Craiova
526=366+160
Pitesti
Sibiu
553=300+253
Bucharest
418=418+0
Craiova
615=455+160
Rimnicu Vilcea
607=414+193

Thuật toán A*

A*(Q,S,G,P,c,h)

đầu vào: bài toán tìm kiếm
hàm heuristics h

đầu ra: đường tới nút đích

khởi tạo: tập các nút biên (nút mở) O = S

while (O không rỗng) **do**

1. Lấy nút n khỏi O sao cho f(n) là nhỏ nhất
2. **nếu** n ∈ G, **return** đường đi tới n
3. với mọi m ∈ P(n)
   a) g(m) = g(n) + c(m,n)
   b) f(m) = g(m) + h(m)
   c) thêm m vào O cùng với giá trị f(m)

return không tìm được đường đi

A* có tối ưu không ?

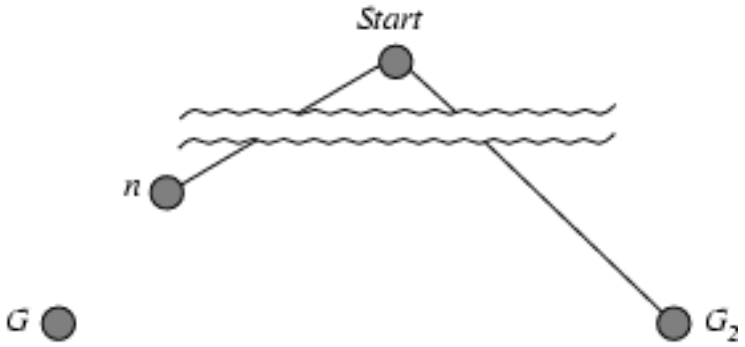
Chỉ tối ưu nếu thoả mãn điều kiện sau

Admissible heuristics

- *Admissible = chấp nhận được*
- Hàm heuristic *h(n)* được gọi là chấp nhận được nếu với mọi nút *n*, ta có $h(n) \leq h^*(n)$, *trong đó* $h^*(n)$ là giá thành thực để đi từ n tới đích.
- Ví dụ : khoảng cách đường chim bay là hàm heuristics chấp nhận được
- Định lý: nếu *h(n) chấp nhận được thì thuật toán* A^* tìm được kết quả tối ưu

Chứng minh tính tối ưu của A*

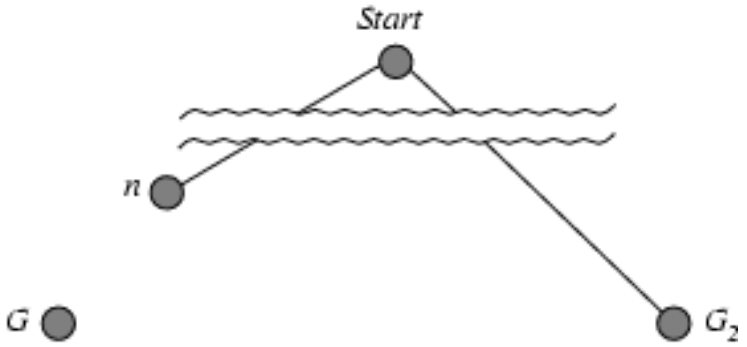
- giả sử nút đích không tối ưu G_2 đang nằm trong hàng đợi. Giả sử n là nút cũng đang nằm trong hàng đợi và *n* nằm trên đường đi ngắn nhất tới nút đích tối ưu *G*



- $f(G_2) = g(G_2)$ vì $h(G_2) = 0$
- $g(G_2) > g(G)$ vì G_2 không tối ưu
- $f(G) = g(G)$ vì $h(G) = 0$
- $f(G_2) > f(G)$

Chứng minh (tiếp theo)

- giả sử nút đích không tối ưu G_2 đang nằm trong hàng đợi. Giả sử n là nút cũng đang nằm trong hàng đợi và *n* nằm trên đường đi ngắn nhất tới nút đích tối ưu *G*



- $f(G_2)$ > $f(G)$ từ trang trước
- $h(n)$ ≤ $h^*(n)$ vì h chấp nhận được
- $g(n) + h(n)$ ≤ $g(n) + h^*(n)$
- $f(n)$ ≤ $f(G)$
-

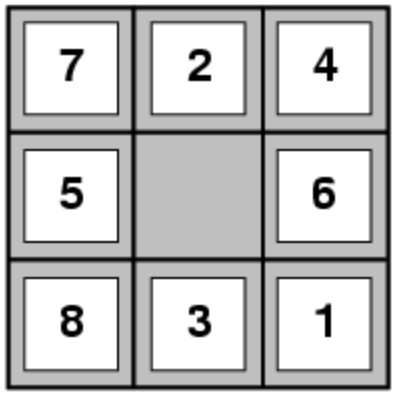
Suy ra $f(G_2) > f(n)$, do vậy A^* sẽ không chọn G_2 để mở rộng

Đặc điểm của A*

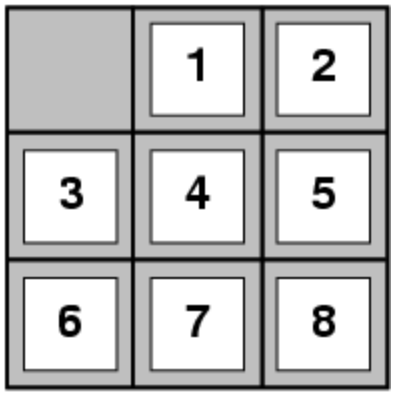
- Đầy đủ? Có (trừ khi có vô số nút với hàm f ≤ *f(G)*)
- Thời gian? $O(b^m)$. Có thể nhanh hơn nhiều nếu có heuristics tốt
- Bộ nhớ? cần lưu tất cả các nút -> $O(b^m)$
- Tối ưu? Có

Ví dụ heuristics chấp nhận được

- $h_1(n)$ = số ô đặt sai chỗ
- $h_2(n)$ = khoảng cách Manhattan



Start State

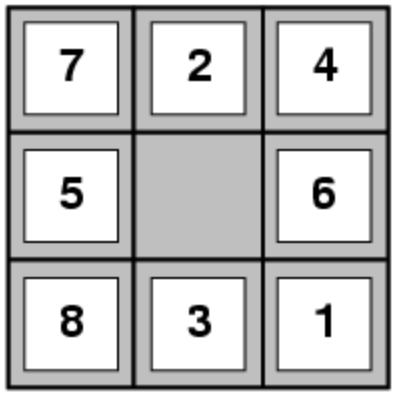


Goal State

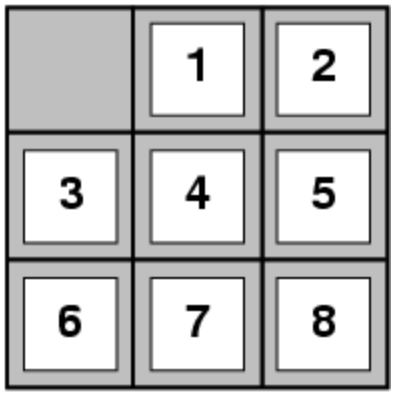
- $h_1(S) = ?$
- $h_2(S) = ?$
-

Ví dụ heuristics chấp nhận được

- $h_1(n)$ = số ô đặt sai chỗ
- $h_2(n)$ = khoảng cách Manhattan



Start State



Goal State

- $h_1(S) = ?$ 8
- $h_2(S) = ?$ 3+1+2+2+2+3+3+2 = 18

Tính trội

- Nếu $h_2(n) \geq h_1(n)$ với mọi n (cả hai hàm đều chấp nhận được)
- thì h_2 trội hơn (tốt hơn) h_1
- h_2 cho phép tìm kiếm nhanh hơn

Tìm kiếm A* sâu dần (IDA*)

1. Tìm kiếm sâu (DFS), không mở rộng nút có f(n) > 0. Nếu tìm được đích thì dừng lại
2. Tìm kiếm sâu (DFS), không mở rộng nút có f(n) > α. Nếu tìm được đích thì dừng lại
3. Tìm kiếm sâu (DFS), không mở rộng nút có f(n) > 2 α. Nếu tìm được đích thì dừng lại
4. ….

- Tính chất của IDA*:
  - đầy đủ
  - tối ưu
  - yêu cầu bộ nhớ tuyến tính
  - độ phức tạp tính toán lớn hơn A*

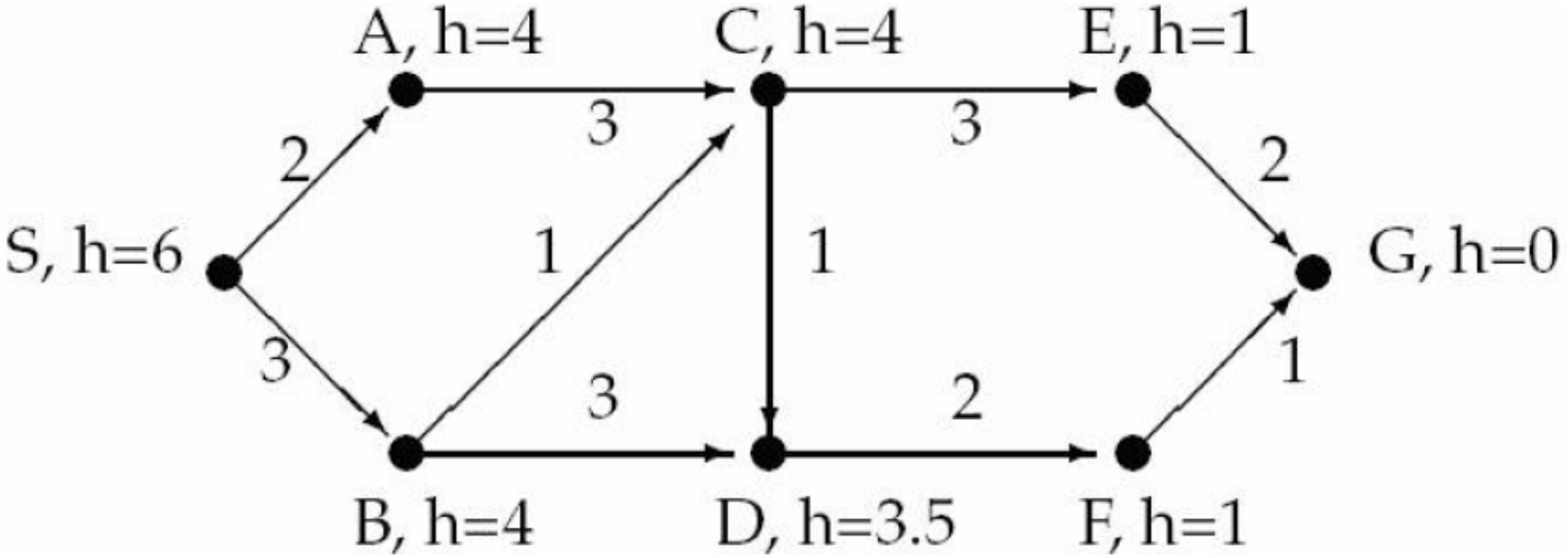
Thuật toán: A*(Q, S, G, P, c, h)

- Đầu vào: bài toán tìm kiếm, hàm heuristic h
- Đầu ra: đường đi ngắn nhất từ nút xuất phát đến nút đích
- Khởi tạo: danh sách các nút biên (nút mở) O ← S

  giá trị $i = 0$ là ngưỡng cho hàm f

While(1) do

1. while (O không rỗng) do
   a) Lấy nút n từ đầu O
   b) Nếu n thuộc G, return(đường đi tới n)
   c) Với mọi m ∈ P(n)
      i) g(m) = g(n) + c(m, n)
      ii) f(m) = g(m) + h(m)
      iii) If f(m) ≤ i then Thêm m vào đầu O
2. i ← i + β, $O \leftarrow S$


A, h=4
C, h=4
E, h=1
S, h=6
G, h=0
B, h=4
D, h=3.5
F, h=1
2
3
3
2
1
1
3
3
2
1