NHẬP MÔN TRÍ TUỆ NHÂN TẠO

ThS Nguyễn Thị Trang
CNTT1
Học viện Công nghệ Bưu chính Viễn thông
Email: trangnguyen.hust117@gmail.com

Nhập môn trí tuệ nhân tạo

TÌM KIẾM CÓ THÔNG TIN

Tìm kiếm mù & tìm kiếm có thông tin

- Tìm kiếm mù
  - Mở rộng các nút tìm kiếm theo một quy luật có trước, không dựa vào thông tin hỗ trợ của bài toán
  - Di chuyển trong không gian trạng thái không có định hướng, phải xem xét nhiều trạng thái
  - Không phù hợp trong các bài toán có không gian trạng thái lớn.

Tìm kiếm mù & tìm kiếm có thông tin

- Tìm kiếm có thông tin : sử dụng *các tri thức cụ thể của bài toán* → Quá trình tìm kiếm hiệu quả hơn
  - Các giải thuật tìm kiếm best-first (Greedy best-first, A*)
  - Các giải thuật tìm kiếm cục bộ (Hill-climbing, Simulated annealing, Local beam, Genetic algorithms)
  - Các giải thuật tìm kiếm đối kháng (MiniMax, Alpha-beta pruning)

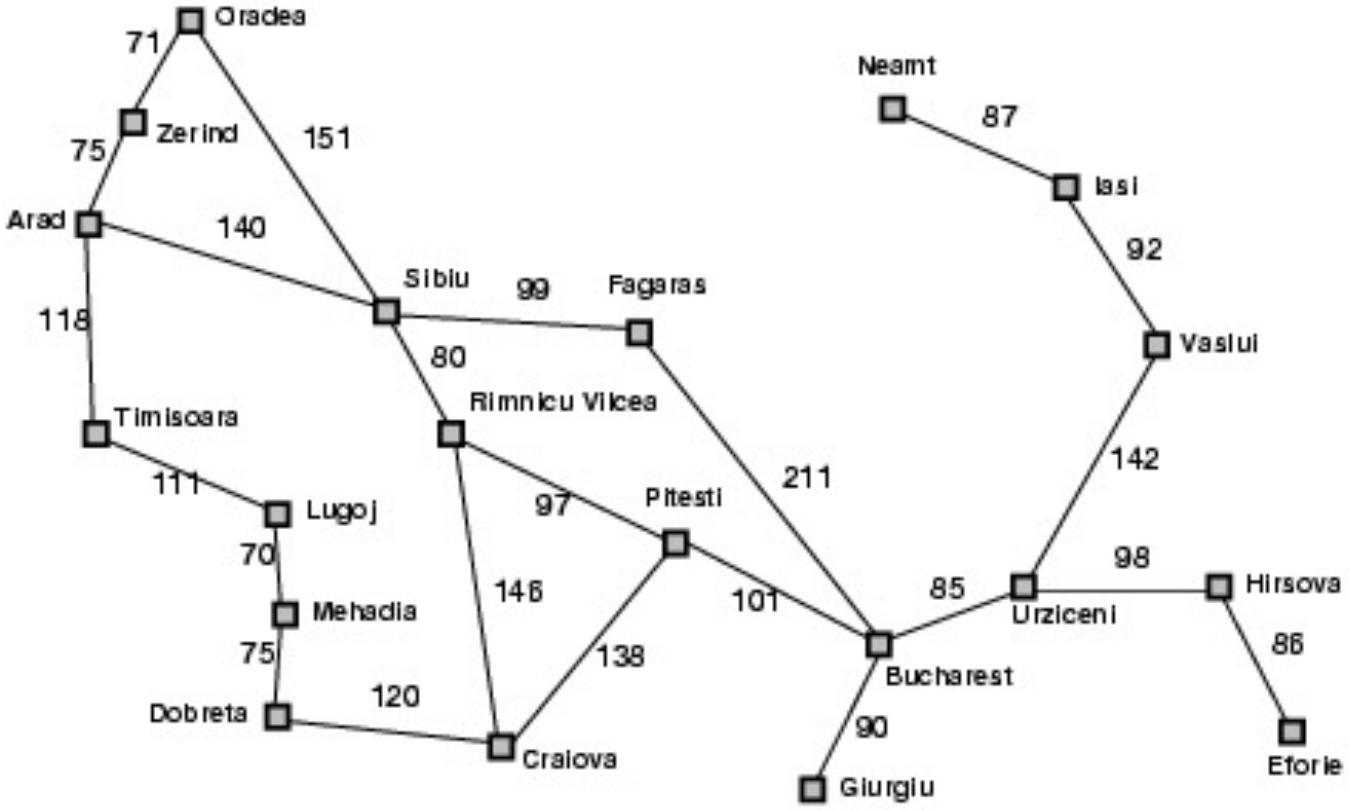
Best-first search

- Ý tưởng: Sử dụng một *hàm đánh giá f(n)* cho mỗi nút của cây tìm kiếm
  - Để đánh giá mức độ "phù hợp" của nút đó
  - → Trong quá trình tìm kiếm, ưu tiên xét các nút có mức độ phù hợp cao nhất
- Cài đặt giải thuật
  - Sắp thứ tự các nút trong cấu trúc fringe theo trật tự giảm dần về mức độ phù hợp
- Các trường hợp đặc biệt của giải thuật Best-first search
  - Greedy best-first search
  - A* search

Nội dung

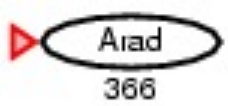
- Tìm kiếm tham lam (greedy search)
- Thuật toán A*
- Các hàm Heuristic
- Thuật toán A* sâu dần (IDA*)

Greedy best-first search – Ví dụ (1)

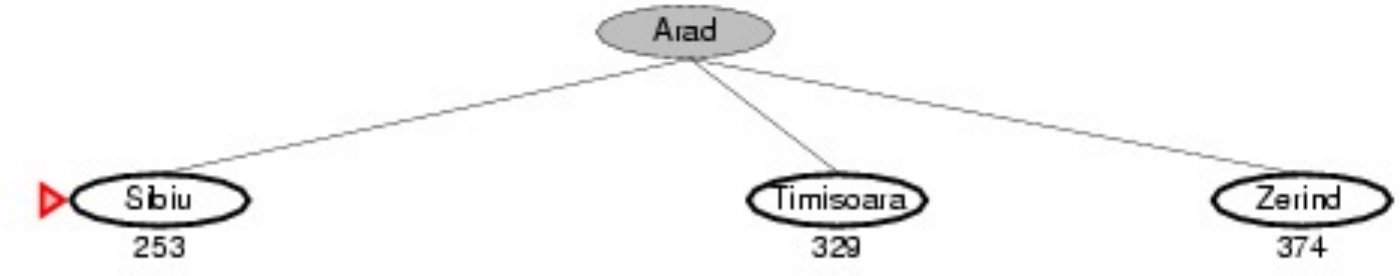


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

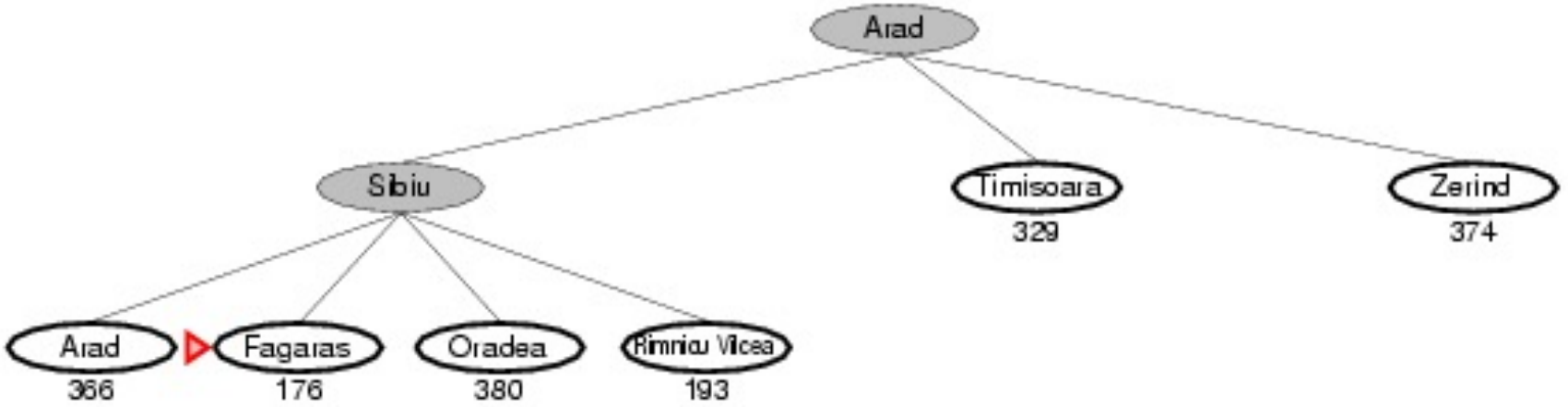
Greedy best-first search – Ví dụ (2)



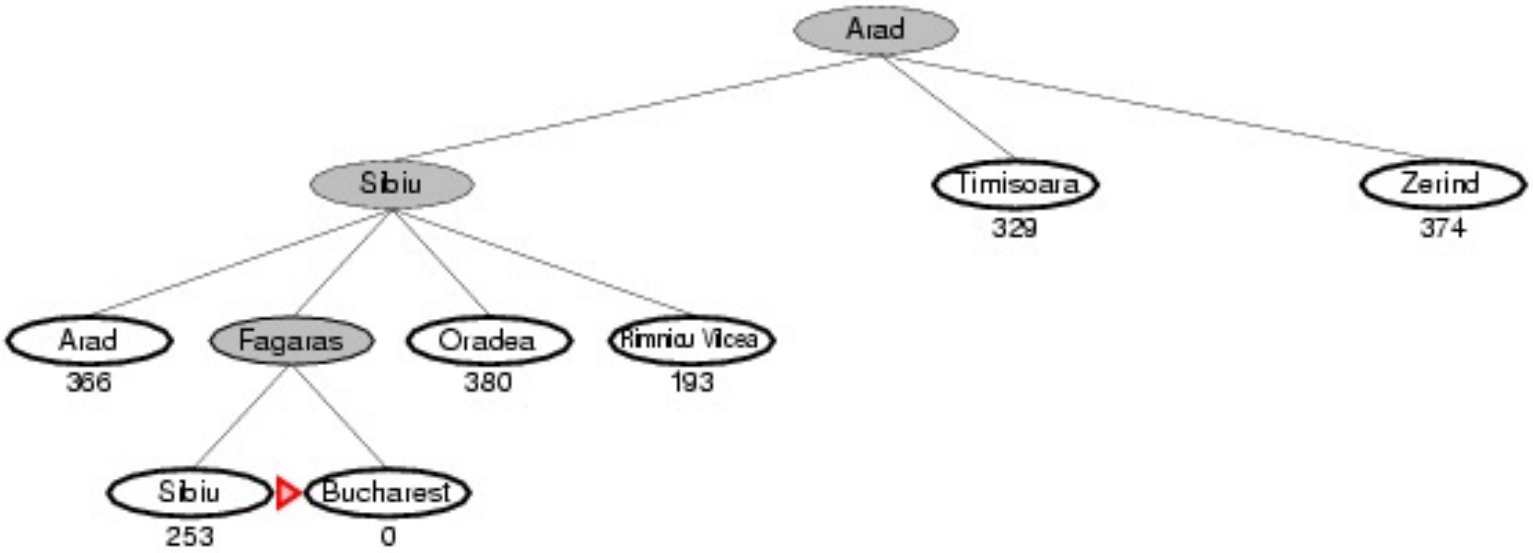
Greedy best-first search – Ví dụ (3)



Greedy best-first search – Ví dụ (4)



Greedy best-first search – Ví dụ (5)

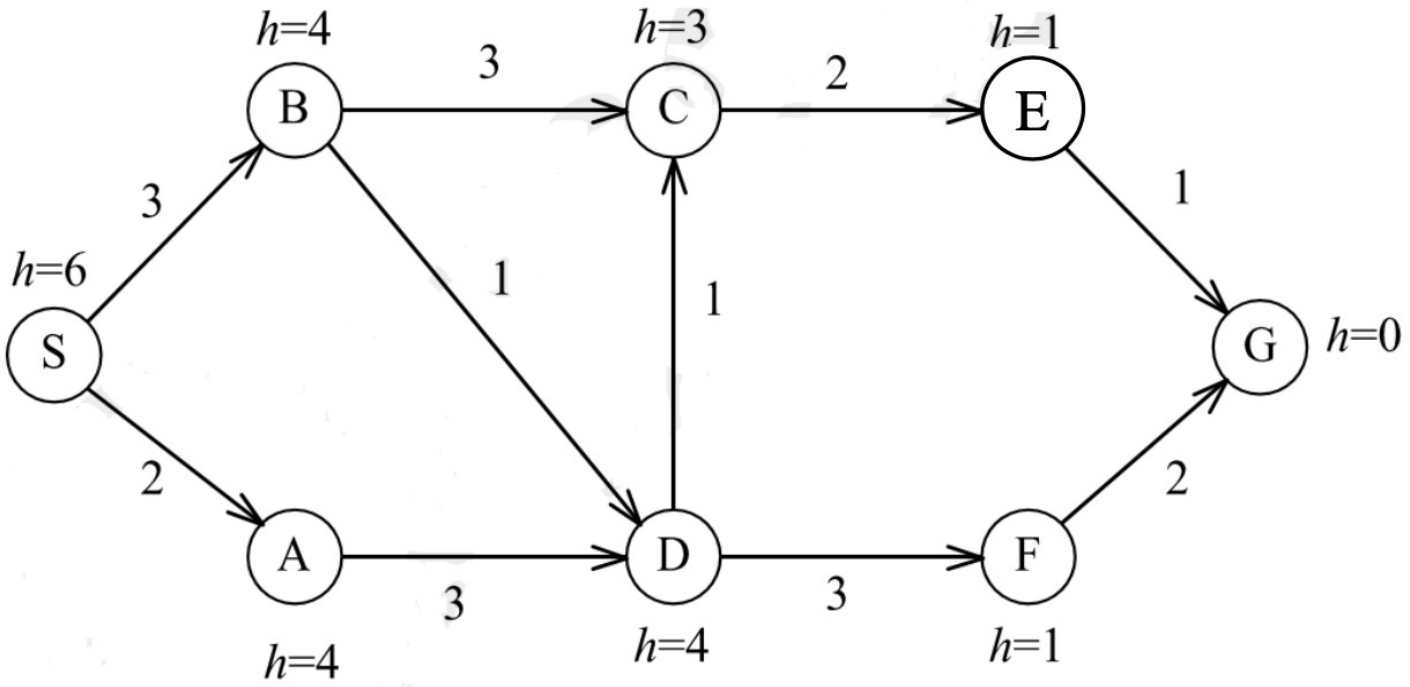


Greedy best-first search – Các đặc điểm

- Tính hoàn chỉnh?
  - Không – Vì có thể vướng (chết tắc) trong các vòng lặp kiểu như: Iasi → Neamt → Iasi → Neamt →…
- Độ phức tạp về thời gian?
  - $O(b^m)$
  - Một hàm heuristic tốt có thể mang lại cải thiện lớn
- Độ phức tạp về bộ nhớ?
  - $O(b^m)$ – Lưu giữ tất cả các nút trong bộ nhớ
- Tính tối ưu?
  - Không

Bài tập 1

- Sử dụng thuật toán tìm kiếm tham lam tìm đường đi từ S tới G?

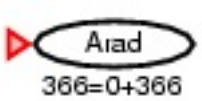


(Phuong TM, 2016)

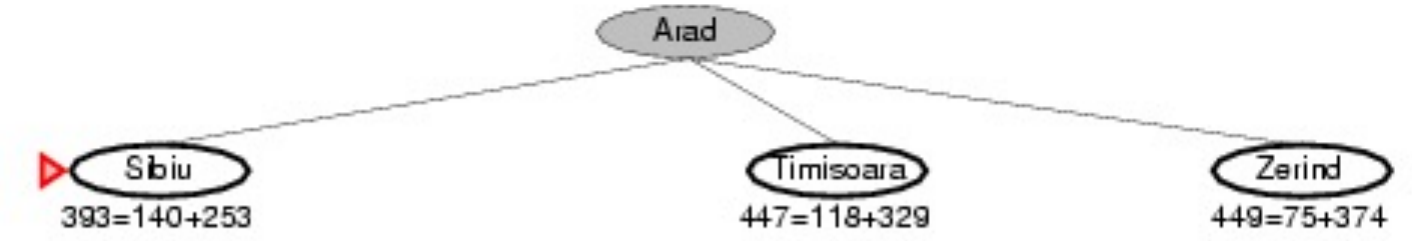
A* search

- Ý tưởng: Tránh việc xét (phát triển) các nhánh tìm kiếm đã xác định (cho đến thời điểm hiện tại) là có chi phí cao
- Sử dụng hàm đánh giá $f(n) = g(n) + h(n)$
  - $g(n)$ = chi phí từ nút gốc cho đến nút hiện tại n
  - $h(n)$ = chi phí ước lượng từ nút hiện tại n tới đích
  - $f(n)$ = chi phí tổng thể ước lượng của đường đi qua nút hiện tại n đến đích

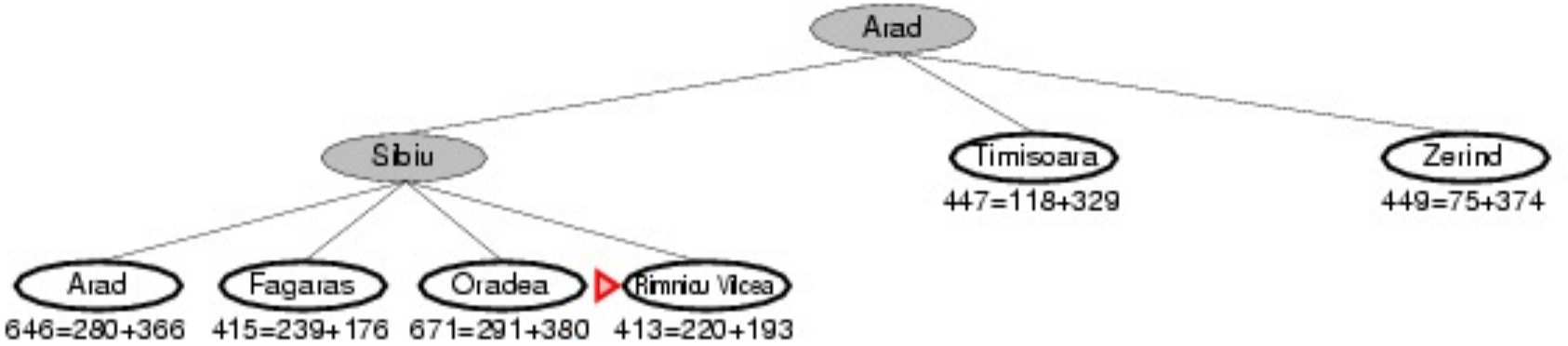
A* search – Ví dụ (1)



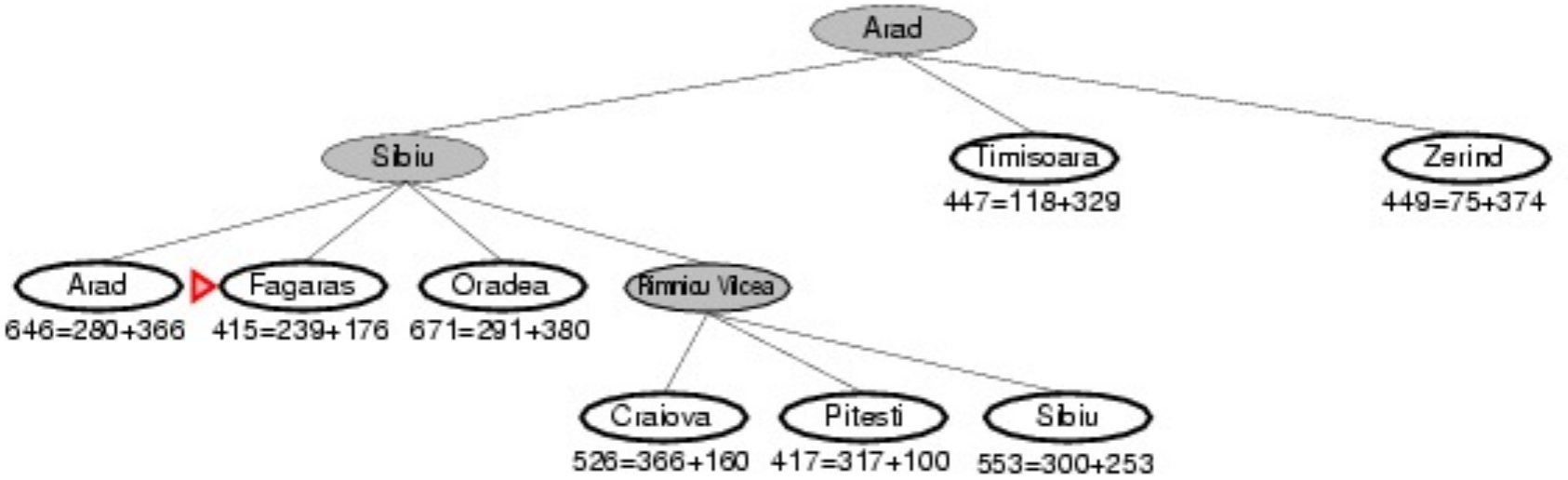
A* search – Ví dụ (2)



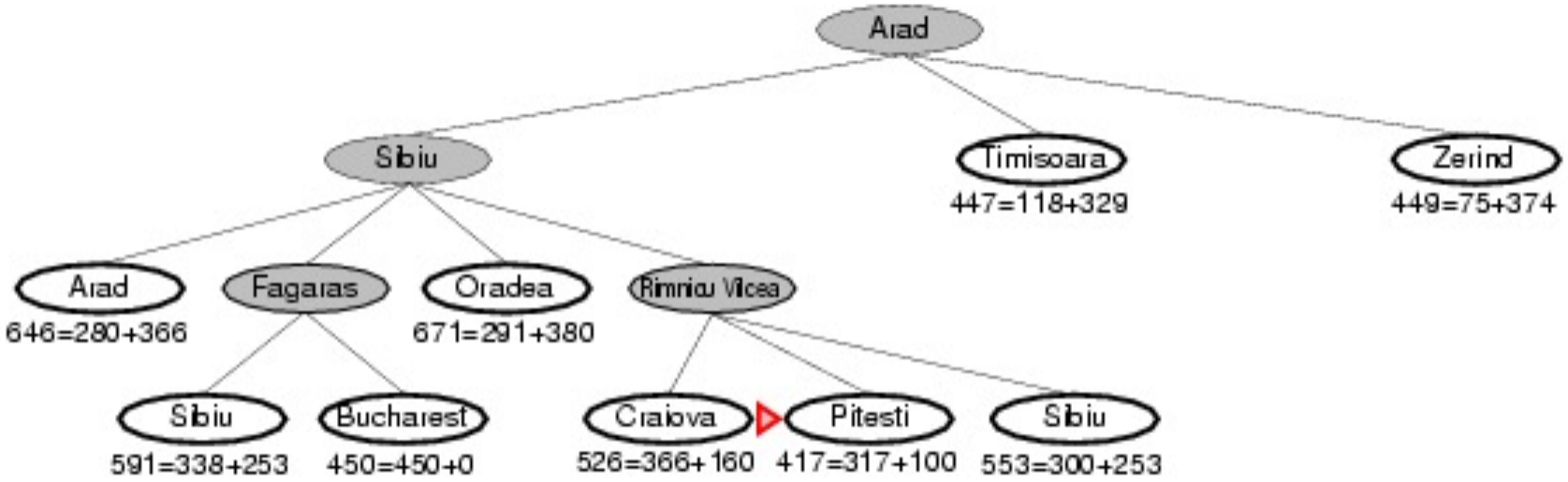
A* search – Ví dụ (3)



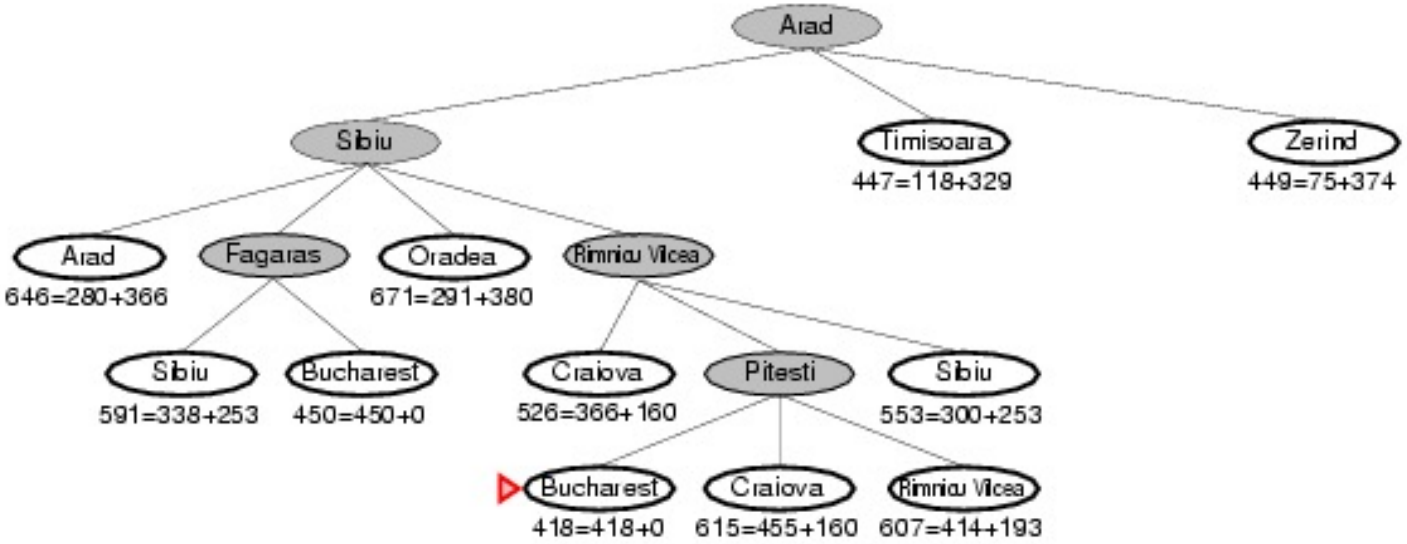
A* search – Ví dụ (4)



A* search – Ví dụ (5)



A* search – Ví dụ (6)



Thuật toán A*

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

(Q: không gian trạng thái, S: trạng thái bắt đầu, G: đích, P: hành động, c: giá, h: heuristic)

Đầu vào: bài toán tìm kiếm, hàm heuristic h

Đầu ra: đường tới nút đích

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

while $(O \neq \emptyset)$ **do**

1. lấy nút n có $f(n)$ là nhỏ nhất khỏi O
2. **if** $n \in G$, **return** (đường đi tới n)
3. với mọi $m \in P(n)$
   a) $g(m) = g(n) + c(n, m)$
   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* search: các đặc điểm

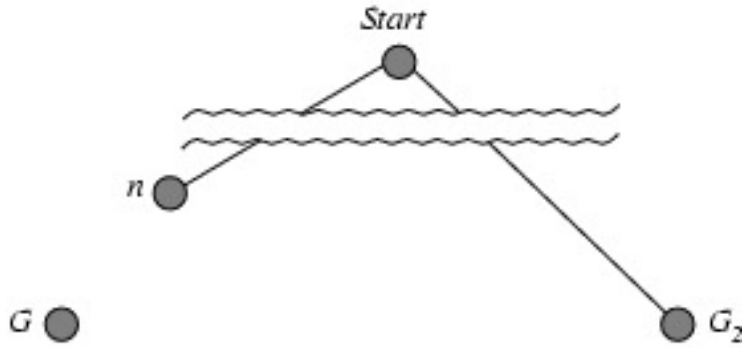
- Nếu *không gian các trạng thái là hữu hạn* và *có giải pháp để tránh việc xét (lặp) lại các trạng thái*, thì giải thuật A* là hoàn chỉnh (tìm được lời giải) – nhưng không đảm bảo là tối ưu
- Nếu *không gian các trạng thái là hữu hạn* và *không có giải pháp để tránh việc xét (lặp) lại các trạng thái*, thì giải thuật A* là không hoàn chỉnh
- Nếu *không gian các trạng thái là vô hạn*, thì giải thuật A* là không hoàn chỉnh

Các ước lượng chấp nhận được

- Một ước lượng $h(n)$ được xem là chấp nhận được nếu đối với mọi nút n: $0 \le h(n) \le h^*(n)$, trong đó $h^*(n)$ là chi phí thật (thực tế) để đi từ nút n đến đích
- Một ước lượng chấp nhận được không bao giờ đánh giá quá cao (overestimate) đối với chi phí để đi tới đích
  - Thực chất, ước lượng chấp nhận được có xu hướng đánh giá "lạc quan"
- Ví dụ: Ước lượng $h_{SLD}(n)$ đánh giá thấp hơn khoảng cách đường đi thực tế
- **Định lý:** Nếu $h(n)$ là đánh giá chấp nhận được, thì phương pháp tìm kiếm A* sử dụng giải thuật `TREE-SEARCH` là tối ưu

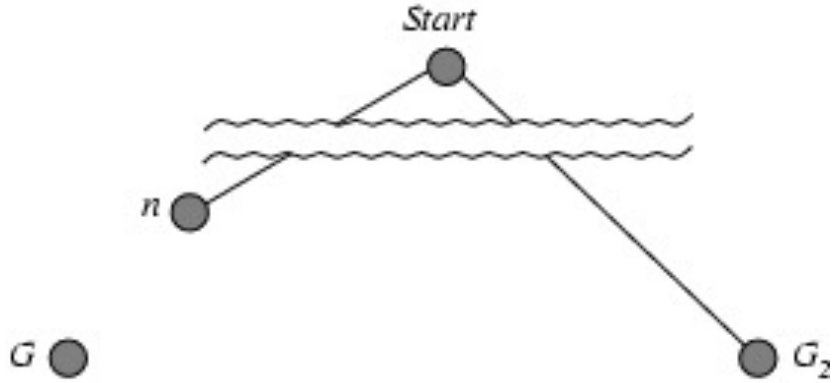
Tính tối ưu của A* - Chứng minh (1)

- Giả sử có một đích không tối ưu (suboptimal goal) G_2 được sinh ra và lưu trong cấu trúc *fringe*. Gọi *n* là một nút chưa xét trong cấu trúc *fringe* sao cho *n* nằm trên một đường đi ngắn nhất đến một đích tối ưu (optimal goal) *G*



- Ta có: 1) $f(G_2) = g(G_2)$ vì $h(G_2) = 0$
- Ta có: 2) $g(G_2) > g(G)$ vì G_2 là đích không tối ưu
- Ta có: 3) $f(G) = g(G)$ vì $h(G) = 0$
- Từ 1)+2)+3) suy ra: 4) $f(G_2) > f(G)$

Tính tối ưu của A* - Chứng minh (2)



Ta có: 5) $h(n) \le h^*(n)$ vì *h* là ước lượng chấp nhận được

Từ 5) suy ra: 6) $g(n) + h(n) \le g(n) + h^*(n)$

Ta có: 7) $g(n) + h^*(n) = f(G)$ vì *n* nằm trên đường đi tới *G*

Từ 6)+7) suy ra: 8) $f(n) \le f(G)$

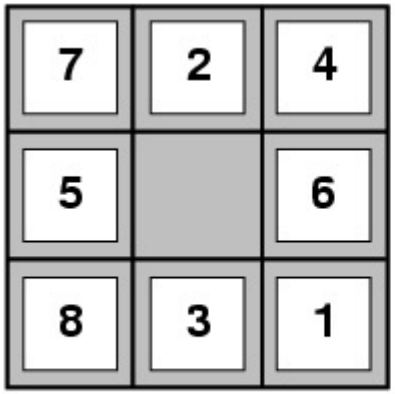
Từ 4)+8) suy ra: $f(G_2) > f(n)$. Tức là, giải thuật A* không bao giờ xét G_2

Các ước lượng chấp nhận được (1)

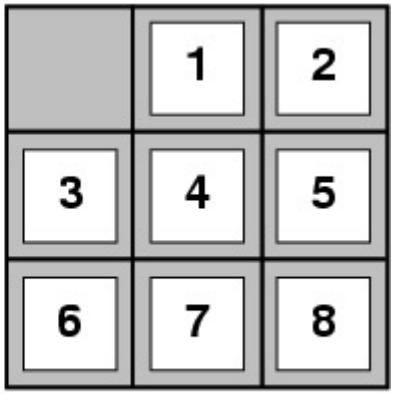
Ví dụ đối với trò chơi ô chữ 8 số:

- $h_1(n)$ = số các ô chữ nằm ở sai vị trí (so với vị trí của ô chữ đấy ở trạng thái đích)
- $h_2(n)$ = khoảng cách dịch chuyển ($\leftarrow,\rightarrow,\uparrow,\downarrow$) ngắn nhất để dịch chuyển các ô chữ nằm sai vị trí về vị trí đúng

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



Start State

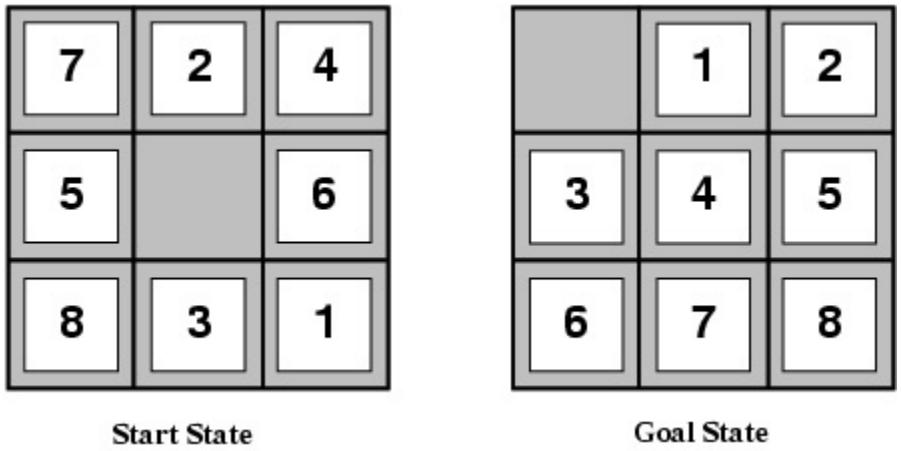


Goal State

Các ước lượng chấp nhận được (2)

Ví dụ đối với trò chơi ô chữ 8 số:

- $h_1(n)$ = số các ô chữ nằm ở sai vị trí (so với vị trí của ô chữ đấy ở trạng thái đích)
- $h_2(n)$ = khoảng cách dịch chuyển ($\leftarrow,\rightarrow,\uparrow,\downarrow$) ngắn nhất để dịch chuyển các ô chữ nằm sai vị trí về vị trí đúng
- $h_1(S) = 8$
- $h_2(S) = 3+1+ 2+2+ 2+3+ 3+2 = 18$

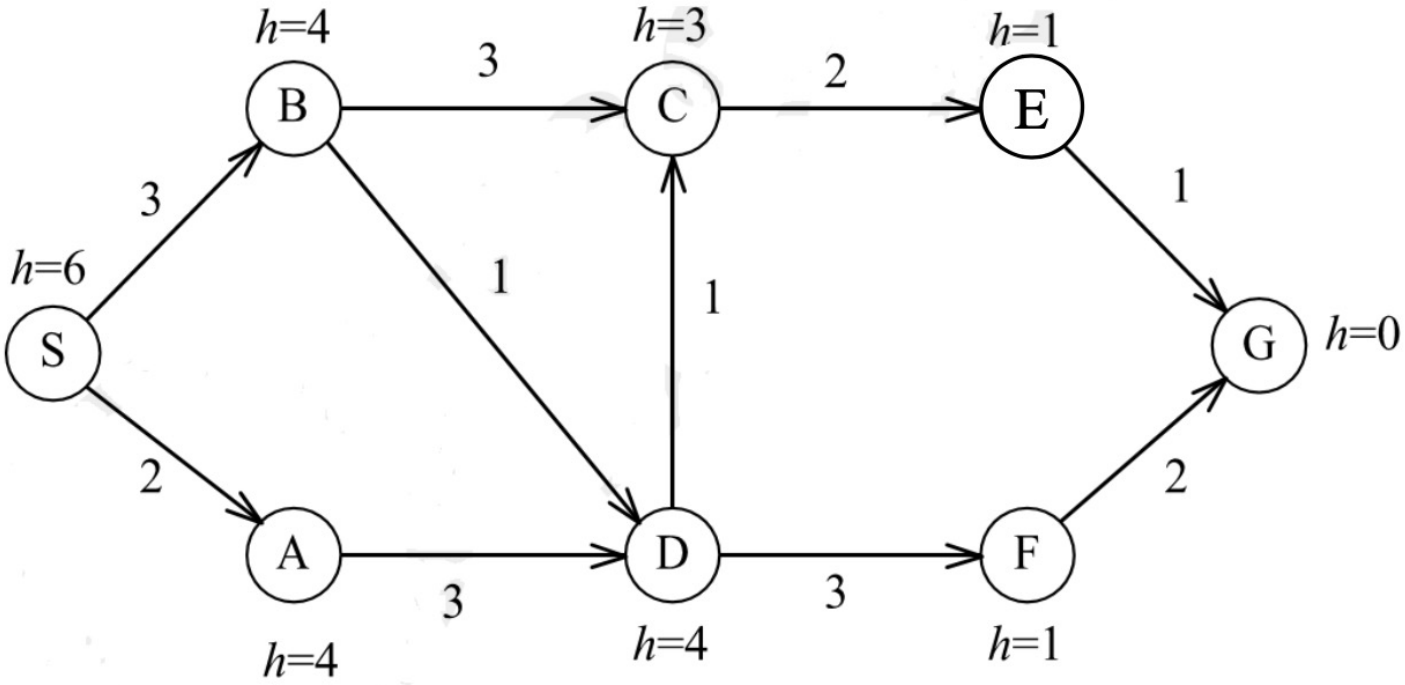


Ước lượng ưu thế

- Ước lượng h_2 được gọi là ưu thế hơn / trội hơn (dominate) ước lượng h_1 nếu:
  - $h^*(n) \geq h_2(n) \geq h_1(n)$ đối với tất cả các nút n
- Nếu ước lượng h_2 ưu thế hơn ước lượng h_1, thì h_2 tốt hơn (nên được sử dụng hơn) cho quá trình tìm kiếm
- Trong ví dụ (ô chữ 8 số) ở trên: Chi phí tìm kiếm = Số lượng trung bình của các nút phải xét:
  - Với độ sâu d =12
    - IDS (Tìm kiếm sâu dần): 3.644.035 nút phải xét
    - A*(sử dụng ước lượng h_1): 227 nút phải xét
    - A*(sử dụng ước lượng h_2): 73 nút phải xét
  - Với độ sâu d =24
    - IDS (Tìm kiếm sâu dần): Quá nhiều nút phải xét
    - A*(sử dụng ước lượng h_1): 39.135 nút phải xét
    - A*(sử dụng ước lượng h_2): 1.641 nút phải xét

Bài tập 2

- Sử dụng thuật toán tìm kiếm A* tìm đường đi từ S tới G?



(Phuong TM, 2016)

Tìm kiếm A* sâu dần – IDA*

- **Mục tiêu**: Giải quyết vấn đề bộ nhớ của thuật toán A*
- **Phương pháp**: Lặp lại việc tìm kiếm theo chiều sâu trên các cây tìm kiếm con có giá trị hàm f(n) không lớn hơn một ngưỡng.
  - Giá trị ngưỡng được tăng dần sau mỗi vòng lặp, để mỗi vòng lặp có thể xét thêm các nút mới

Thuật toán IDA*

$IDA^*(Q, S, G, P, c, h)$

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) $O \leftarrow S$

giá trị $i \leftarrow 0$ là ngưỡng cho hàm f

while (1) **do**

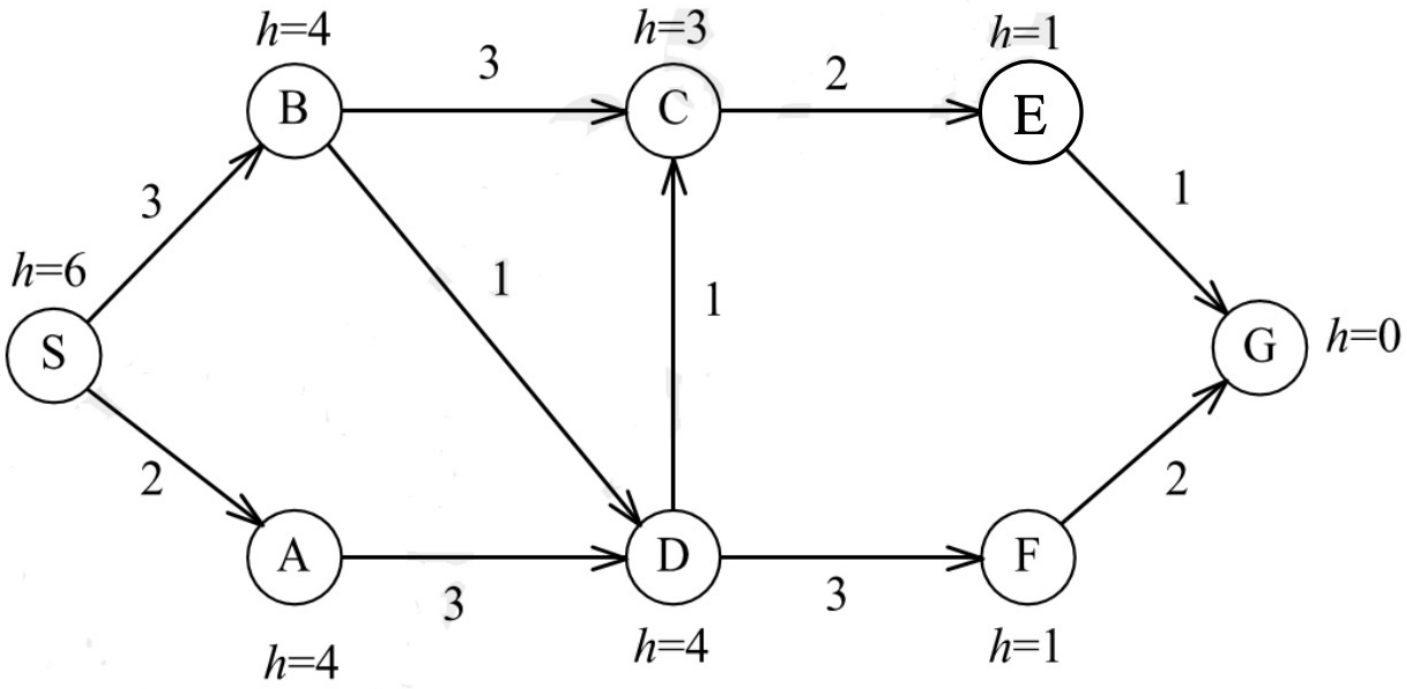
1. **while** $(O \neq \emptyset)$ **do**
   a) lấy nút n từ đầu O
   b) **if** n thuộc G, **return** (đường đi tới n)
   c) với mọi $m \in P(n)$
      i) $g(m) = g(n) + c(m, n)$
      ii) $f(m) = g(m) + h(m)$
      iii) **if** $f(m) \leq i$ **then** thêm m vào đầu O
2. $i \leftarrow i + \beta, O \leftarrow S$

Tính chất IDA*

- Đầy đủ ? Có
- Tối ưu?
  - β – tối ưu (giá thành của lời giải tìm được không vượt quá β so với giá thành của lời giải tối ưu)
- Thời gian?
  - Độ phức tạp tính toán lớn hơn của thuật toán A*
- Bộ nhớ
  - Yêu cầu bộ nhớ tuyến tính

Bài tập 3

- Sử dụng thuật toán tìm kiếm A* sâu dần tìm đường đi từ S tới G, cho biết bước nhảy $\beta = 2$?



(Phuong TM, 2016)