



Học viện Công nghệ Bưu chính Viễn thông
Khoa Công nghệ thông tin 1

Nhập môn trí tuệ nhân tạo

Tìm kiếm có thông tin
(informed search)

Ngô Xuân Bách

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

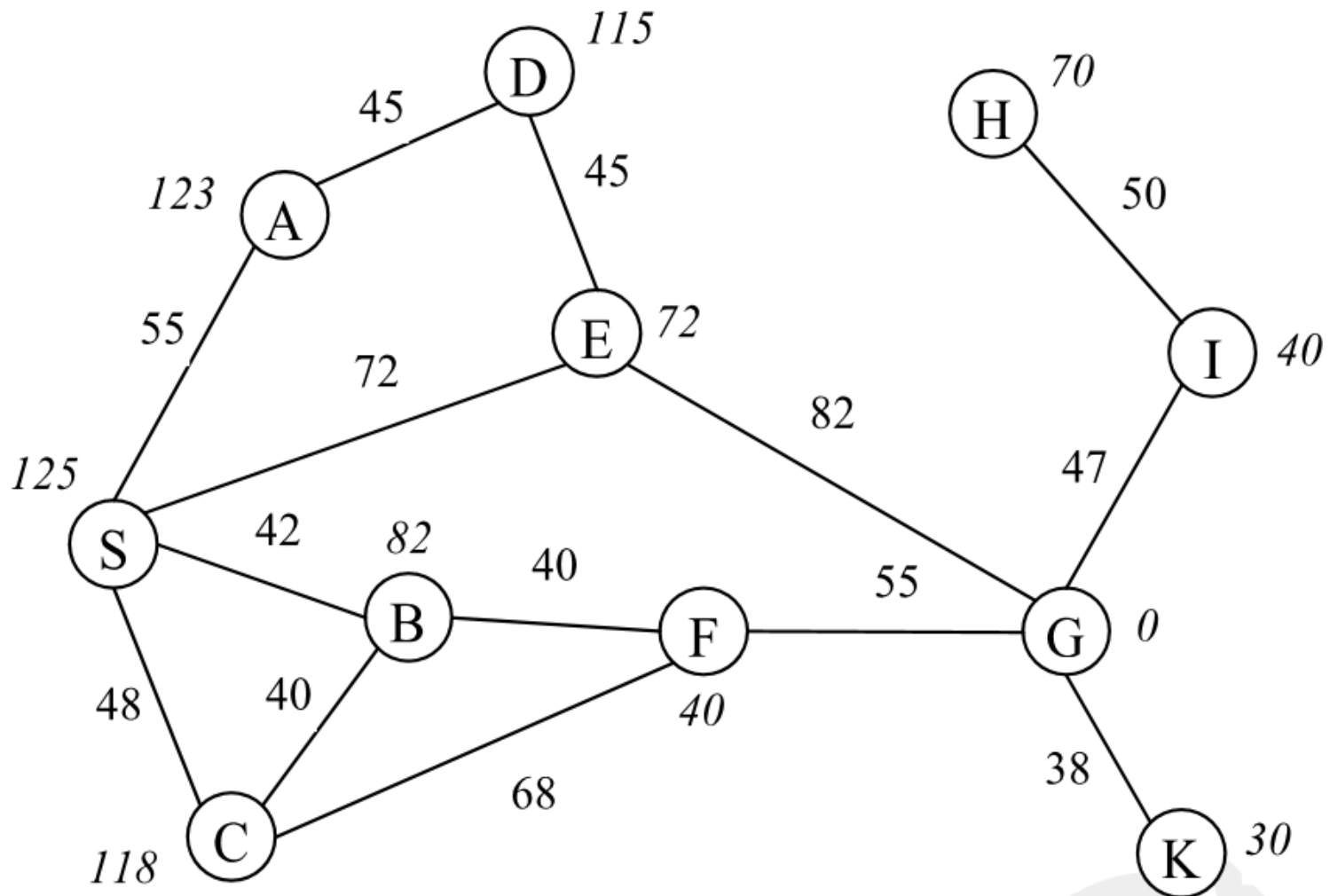
► Tìm kiếm có thông tin

- Sử dụng thông tin phụ từ bài toán để **định hướng tìm kiếm**
- Sử dụng một hàm **$f(n)$** đánh giá độ “tốt” tiềm năng của nút n , từ đó chọn nút tốt nhất để mở rộng trước
 - Tìm kiếm tốt nhất đầu tiên (best-first search)
 - **Xây dựng hàm $f(n)$ thế nào?**

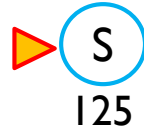
Nội dung

- ▶ Tìm kiếm tham lam (greedy search)
- ▶ Thuật toán A^*
- ▶ Các hàm heuristic
- ▶ Thuật toán A^* sâu dẫn (IDA^*)

Ví dụ đồ thị cho bài toán tìm kiếm

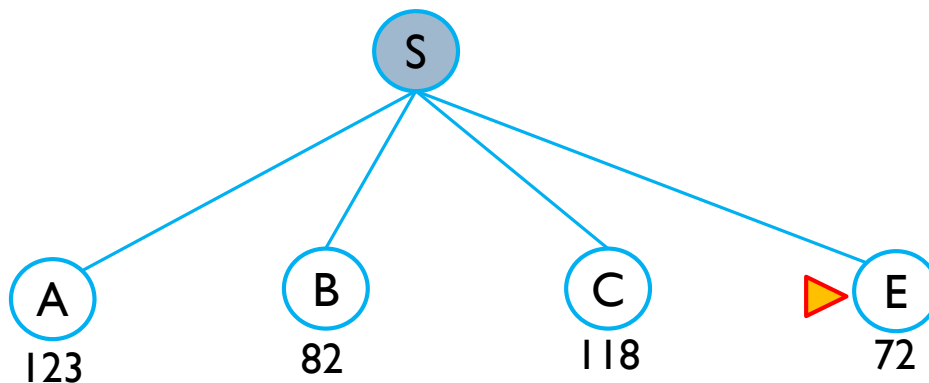


Ví dụ tìm kiếm tham lam (1 / 4)



Ví dụ tìm kiếm tham lam (2/4)

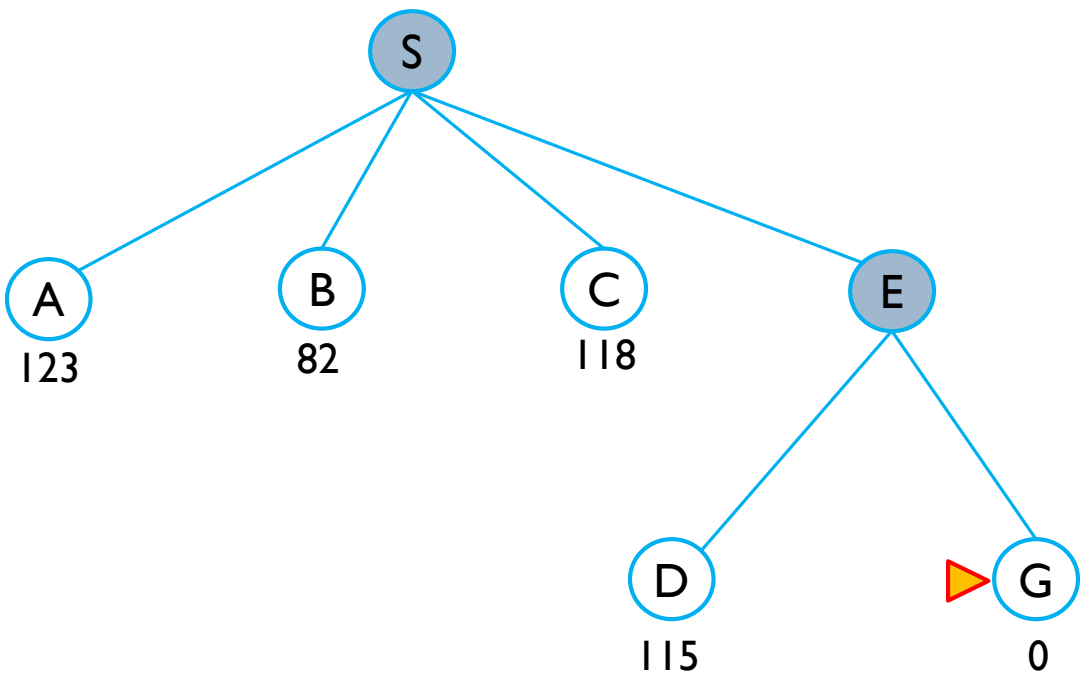
Mở rộng S



Ví dụ tìm kiếm tham lam (3/4)

Mở rộng S

Mở rộng E

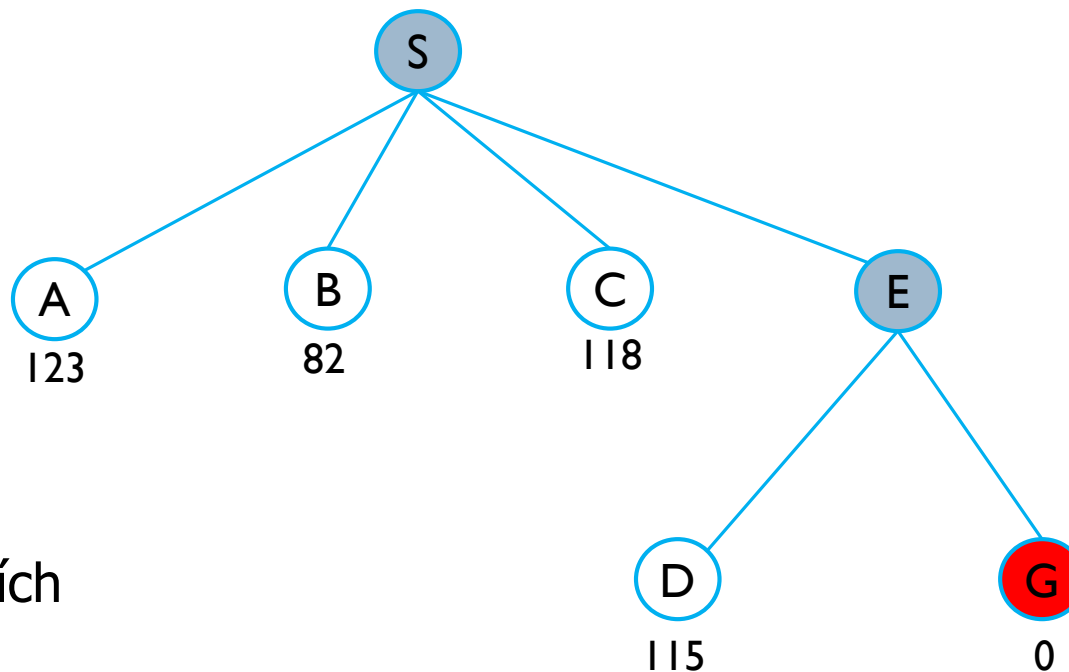


Ví dụ tìm kiếm tham lam (4/4)

Mở rộng S

Mở rộng E

Mở rộng G: Đích



Tính chất của tìm kiếm tham lam

▶ Đầy đủ?

- Không (có thể tạo thành vòng lặp, hoặc có nhánh gồm vô hạn nút có giá trị hàm h nhỏ nhưng không dẫn tới đích)

▶ Tối ưu?

- Không

▶ Thời gian?

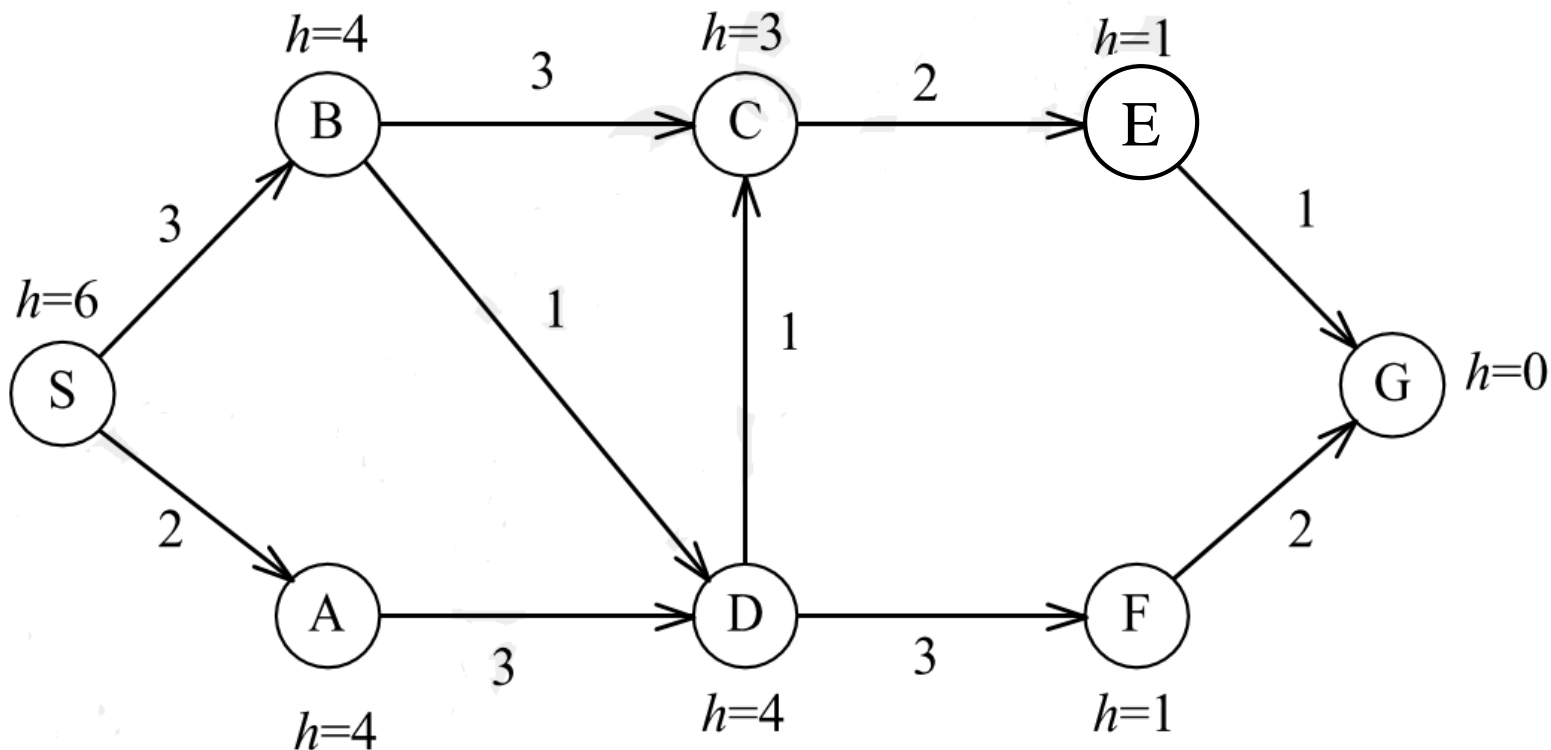
- $O(b^m)$
- Nếu hàm *heuristic* tốt, thuật toán có thể sẽ nhanh hơn rất nhiều

▶ Bộ nhớ?

- $O(b^m)$: lưu tất cả các nút trong bộ nhớ
- Nếu hàm *heuristic* tốt, số nút cần lưu có thể giảm đi rất nhiều

Bài tập 1

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



(Phuong TM, 2016)

Nội dung



- ▶ Tìm kiếm tham lam (greedy search)
- ▶ Thuật toán A^*
- ▶ Các hàm heuristic
- ▶ Thuật toán A^* sâu dẫn (IDA^*)

Thuật toán A^* : ý tưởng

- ▶ Khắc phục nhược điểm của tìm kiếm tham lam
 - Tham lam: chỉ quan tâm tới đường đi tới đích
 - A^* : quan tâm cả đường đi từ điểm xuất phát tới nút đang xét
 - Không mở rộng đường đi có giá thành lớn

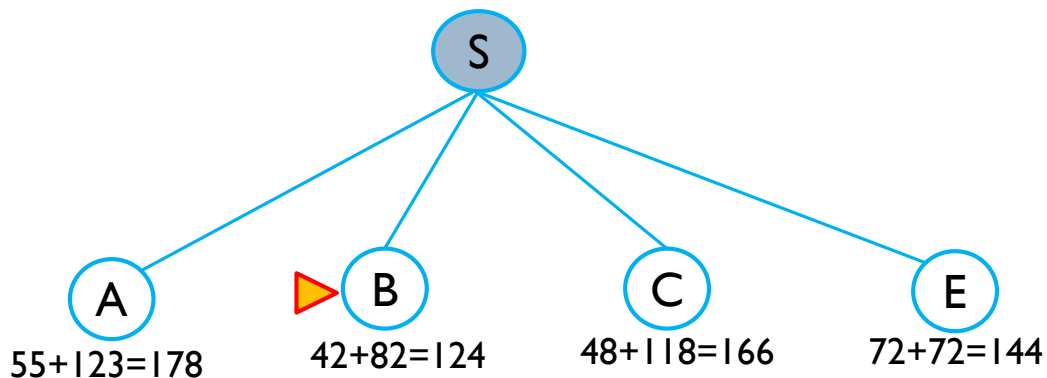
- ▶ **Phương pháp:** $f(n) = g(n) + h(n)$
 - $g(n)$: giá thành đường đi từ điểm xuất phát tới nút n
 - $h(n)$: hàm heuristic ước lượng giá thành đường đi từ n tới đích
 - $f(n)$: ước lượng giá thành đường đi từ điểm xuất phát, qua n , tới đích

Ví dụ thuật toán A* (1/5)

  S
 $0 + 125 = 125$

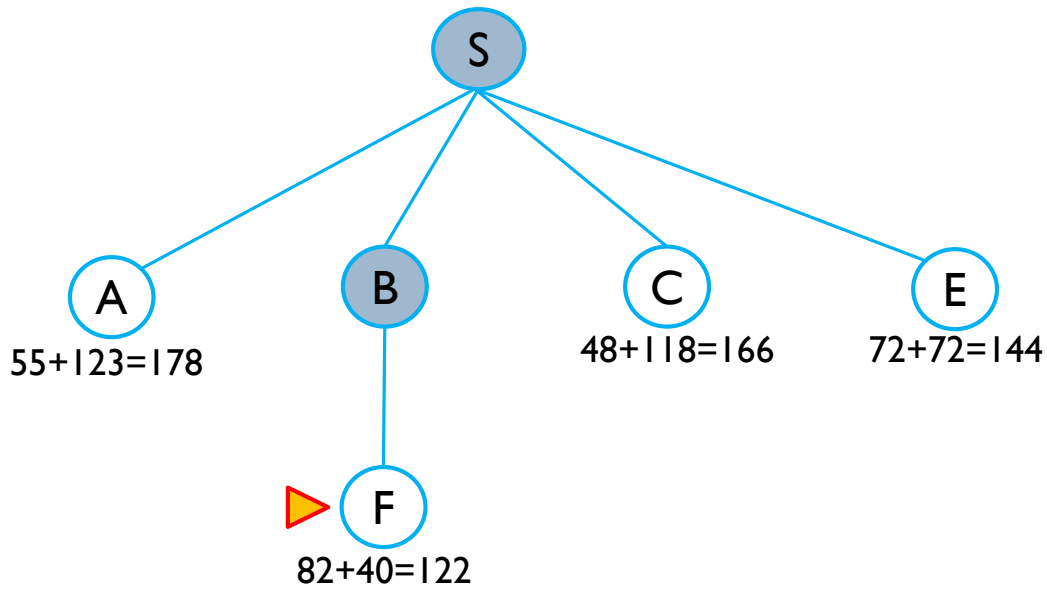
Ví dụ thuật toán A* (2/5)

Mở rộng S



Ví dụ thuật toán A* (3/5)

Mở rộng S



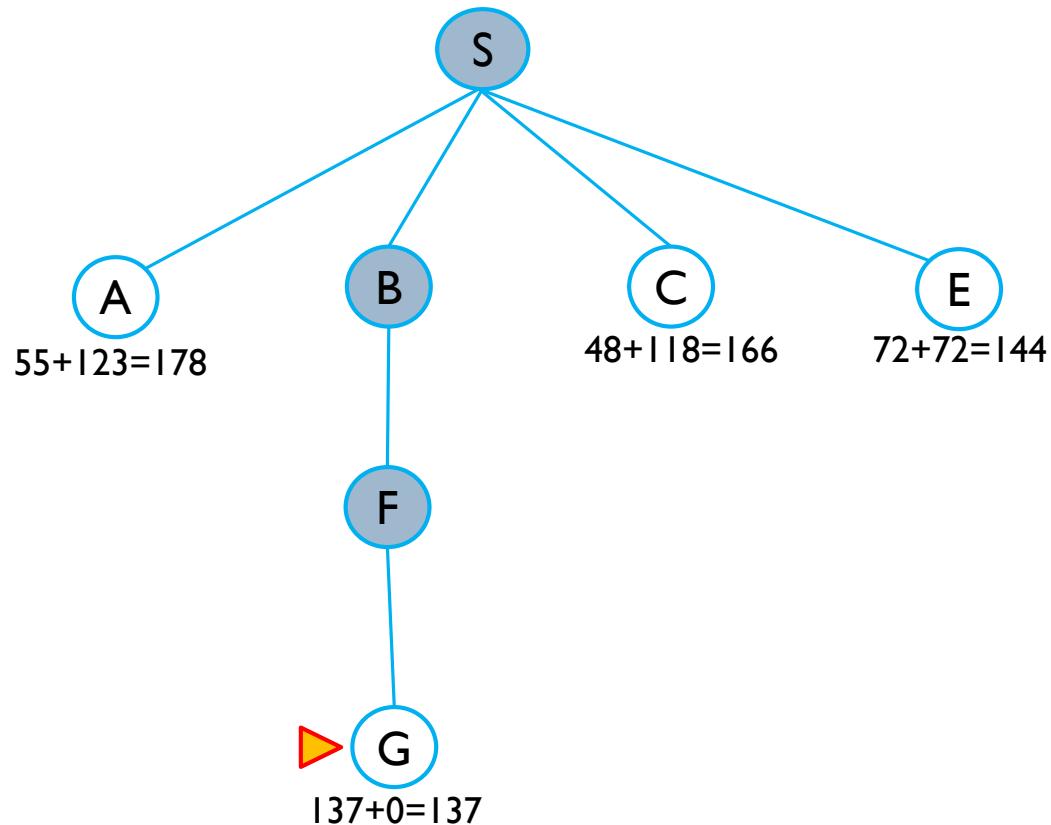
Mở rộng B

Ví dụ thuật toán A* (4/5)

Mở rộng S

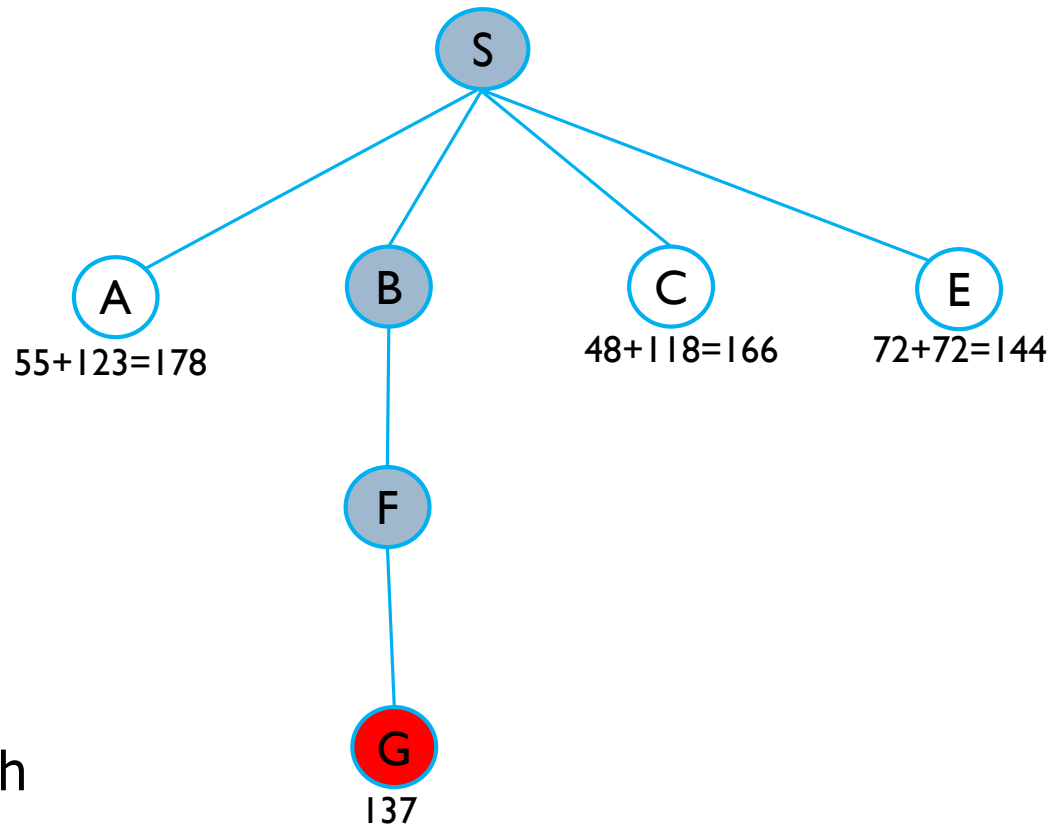
Mở rộng B

Mở rộng F



Ví dụ thuật toán A* (5/5)

Mở rộng S



Mở rộng B

Mở rộng F

Mở rộng G: Đích

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

Tính chất của thuật toán A^*

▶ Đầy đủ?

- Có (trừ khi có vô số nút n với $f(n) \leq f(G)$)

▶ Tối ưu?

- Có (nếu hàm *heuristic* là chấp nhận được)

▶ Thời gian?

- $O(b^m)$
- Nếu hàm *heuristic* tốt, thuật toán có thể sẽ nhanh hơn rất nhiều

▶ Bộ nhớ?

- $O(b^m)$: lưu tất cả các nút trong bộ nhớ
- Nếu hàm *heuristic* tốt, số nút cần mở rộng có thể giảm đi rất nhiều

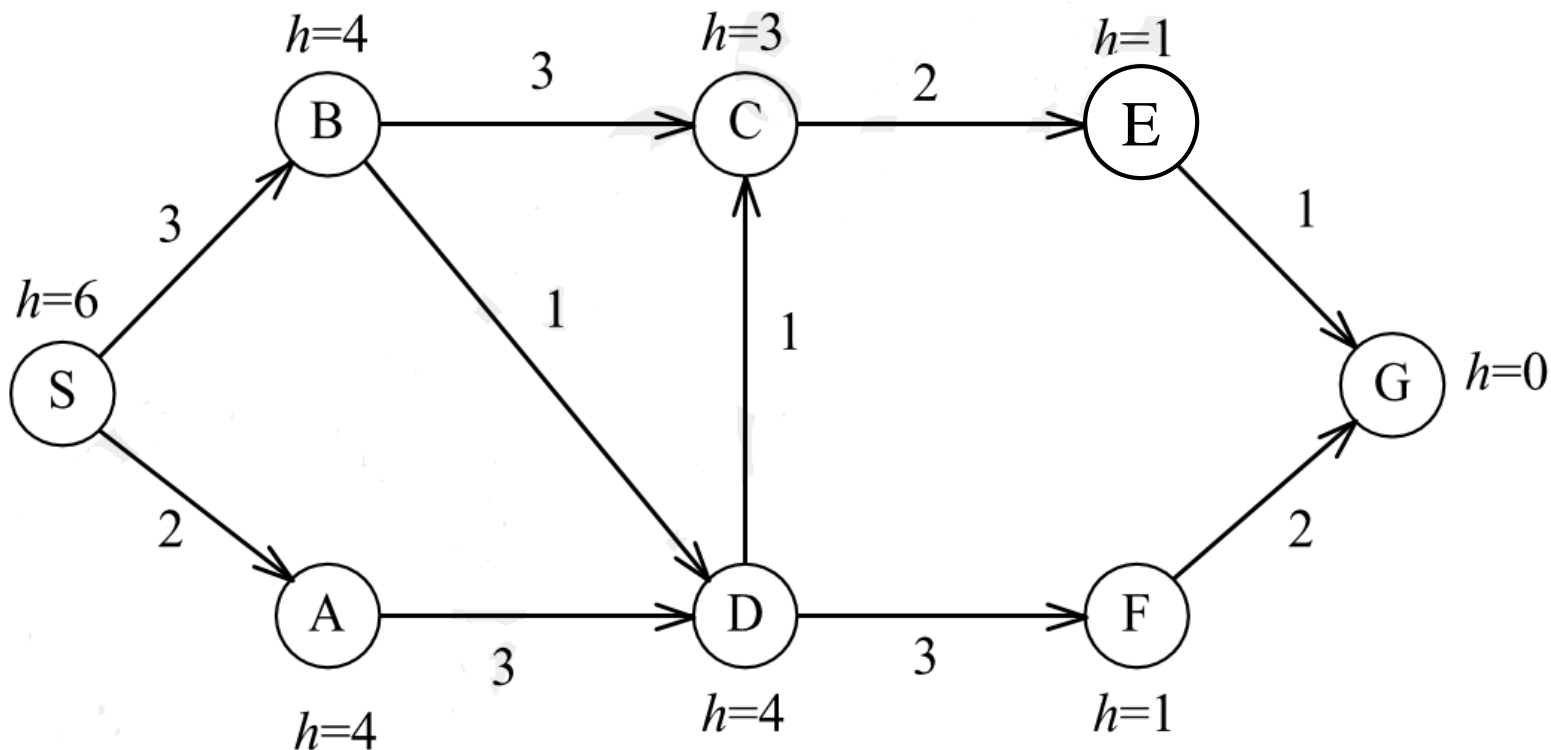
Tính tối ưu của A^*

- ▶ Hàm heuristic chấp nhận được
 - Mọi nút n thì $h(n) \leq h^*(n)$, với $h^*(n)$ là giá thành thực để đi từ n tới đích
 - Ví dụ: hàm heuristic đo khoảng cách đường chim bay là chấp nhận được

- ▶ **Định lý:** Thuật toán A^* sẽ cho kết quả **tối ưu** nếu $h(n)$ là hàm **chấp nhận được**

Bài tập 2

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



(Phuong TM, 2016)

Nội dung

- ▶ Tìm kiếm tham lam (greedy search)
- ▶ Thuật toán A^*
- ▶ Các hàm heuristic
- ▶ Thuật toán A^* sâu dẫn (IDA*)

Các hàm heuristic

- ▶ Các hàm heuristic được xây dựng tùy vào từng bài toán cụ thể
 - Một bài toán có thể có nhiều hàm heuristic
 - Chất lượng hàm heuristic ảnh hưởng rất nhiều tới quá trình tìm kiếm

- ▶ Hàm heuristic trội
 - Nếu $h_1(n)$ và $h_2(n)$ là 2 hàm heuristic chấp nhận được thỏa mãn $h_1(n) \leq h_2(n)$ với mọi nút n , thì h_2 **trội hơn** (tốt hơn) h_1

Ví dụ hàm heuristic

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- ▶ $h_1(n)$: số ô đặt sai chỗ
 - $h_1(S) = 8$
- ▶ $h_2(n)$: tổng khoảng cách Manhattan
 - $h_2(S) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$

Nội dung

- ▶ Tìm kiếm tham lam (greedy search)
- ▶ Thuật toán A^*
- ▶ Các hàm heuristic
- ▶ Thuật toán A^* sâu dẫn (IDA*)

Tìm kiếm A* sâu dần – IDA*

- ▶ **Mục tiêu:** giải quyết vấn đề bộ nhớ trong tìm kiếm 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

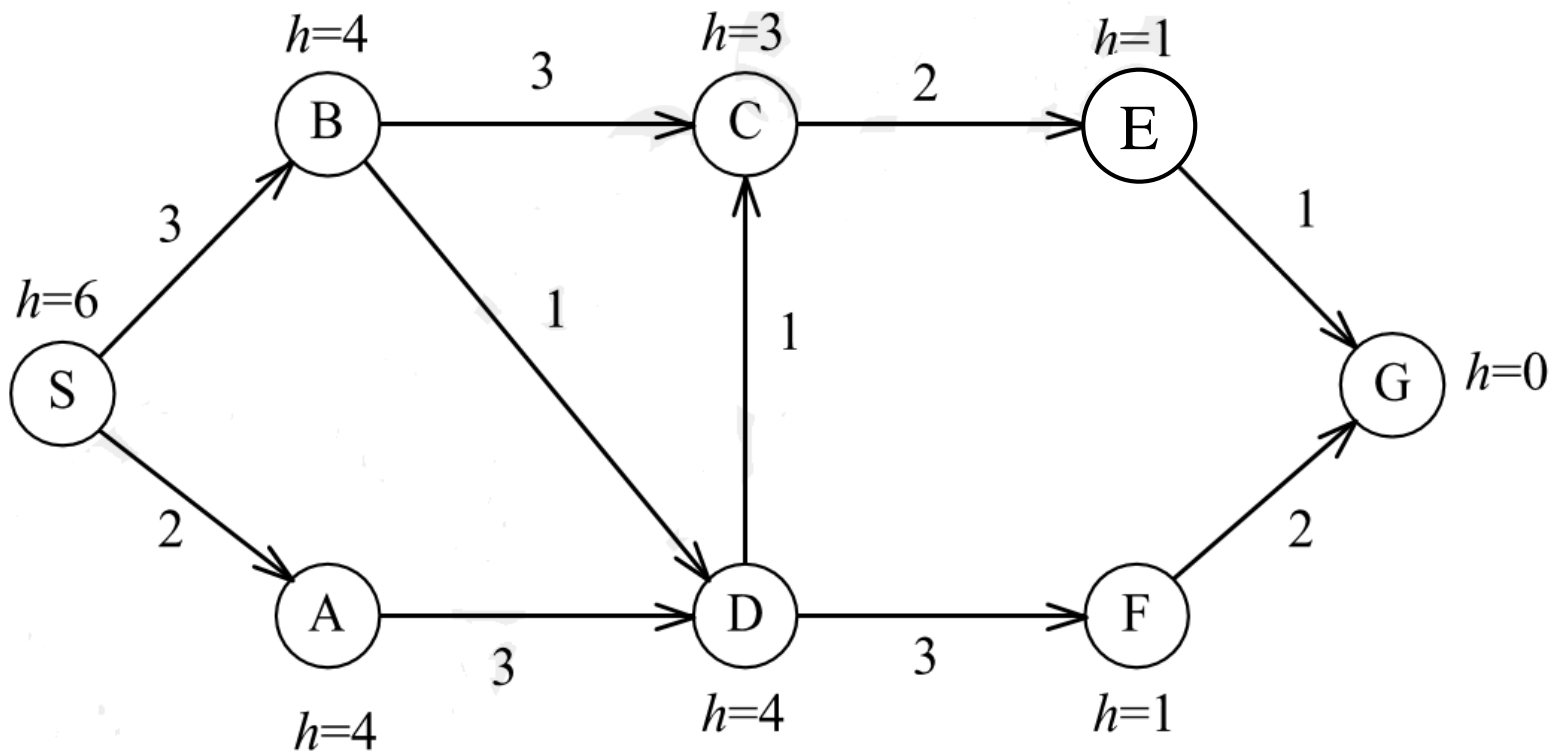
2. $i \leftarrow i + \beta, O \leftarrow S$

Tính chất của 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 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)

Khi nào đưa nút lặp vào danh sách?

▶ Tham lam

- **Không**: Việc đưa vào không làm thay đổi thuật toán (có thể dẫn đến vòng lặp)

▶ A^*

- Trong trường hợp nút lặp có giá thành (chi phí) tốt hơn, nó sẽ được **đưa lại danh sách** (nếu đã phát triển rồi) hoặc **cập nhật thay nút cũ** có giá thành kém hơn (nếu đang trong danh sách)

▶ IDA*:

- **Có**