HỆ THỐNG TÌM KIẾM (PHẦN 4)

Trần Trung Kiên

ttkien@fit.hcmus.edu.vn

Tổng thể

- Ôn lại các buổi học trước
- Cải tiến các thuật toán tìm kiếm: không mở lại các trạng thái đã mở

Check

- Có bao nhiêu bạn cảm thấy là mình hiểu DFS, BFS, UCS?
- Có bao nhiêu bạn cảm thấy là mình hiểu Greedy, A*?
- Nếu không hiểu bài thì nên tận dùng moodle để hỏi
  - Có thể có nhiều bạn có cùng câu hỏi với bạn
  - Có thể có nhiều bạn trả lời được; nếu không thì Thầy sẽ trả lời cho bạn

Ôn lại

- Fringe là gì?
- DFS chọn kế hoạch nào từ fringe để mở rộng?
- BFS chọn kế hoạch nào từ fringe để mở rộng?
- UCS chọn kế hoạch nào từ fringe để mở rộng?

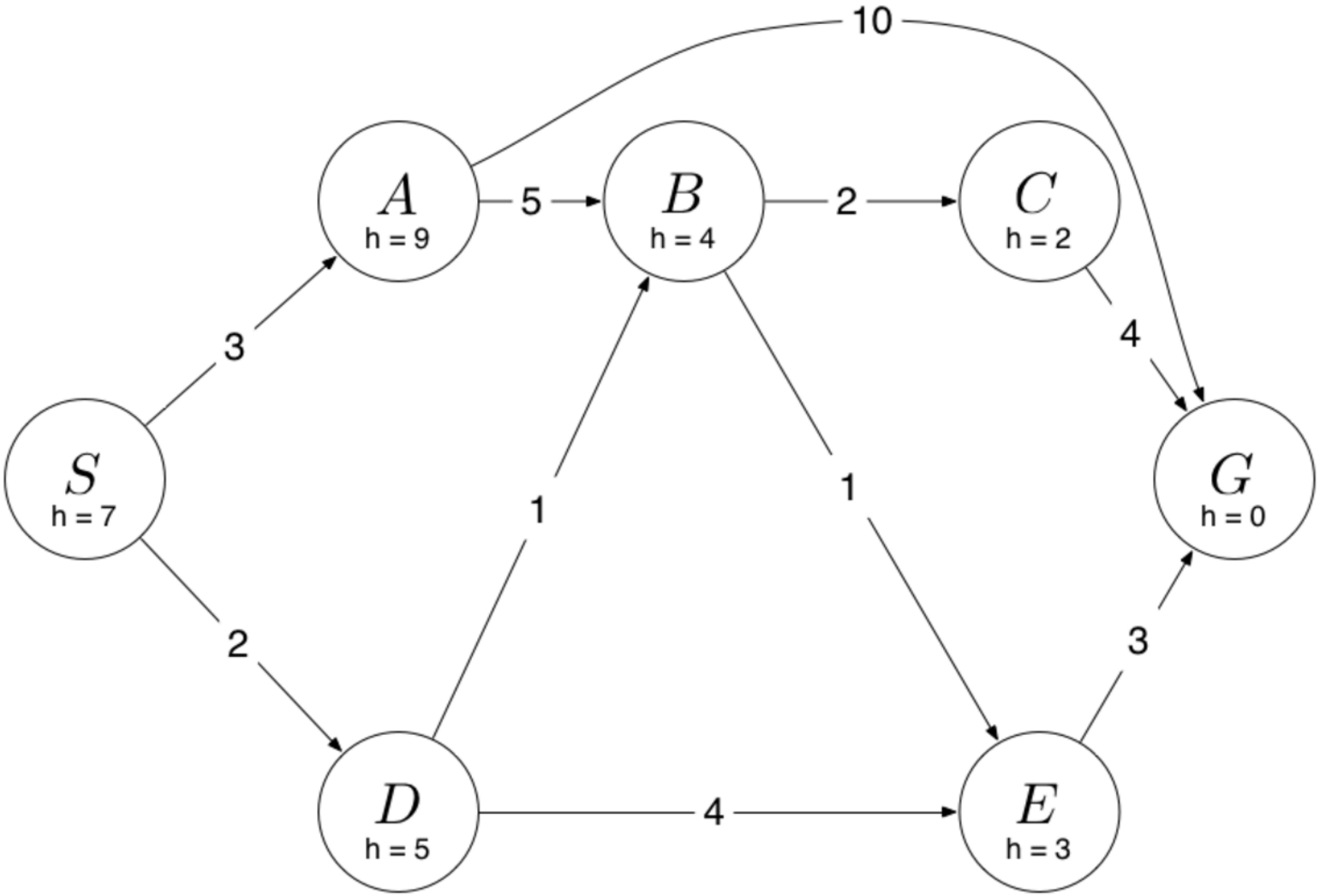
Ôn lại

- Trong 3 thuật toán DFS, BFS, UCS, thuật toán nào đảm bảo tìm được lời giải tối ưu?
- BFS sẽ tìm được lời giải tối ưu khi chi phí của các cạnh như thế nào?
- Ưu và nhược điểm của UCS?

Ôn lại

- Heuristic là gì?
- Greedy chọn kế hoạch nào từ fringe để mở rộng?
- Ưu và nhược điểm của Greedy?

Chạy thử Greedy với đồ thị ở dưới

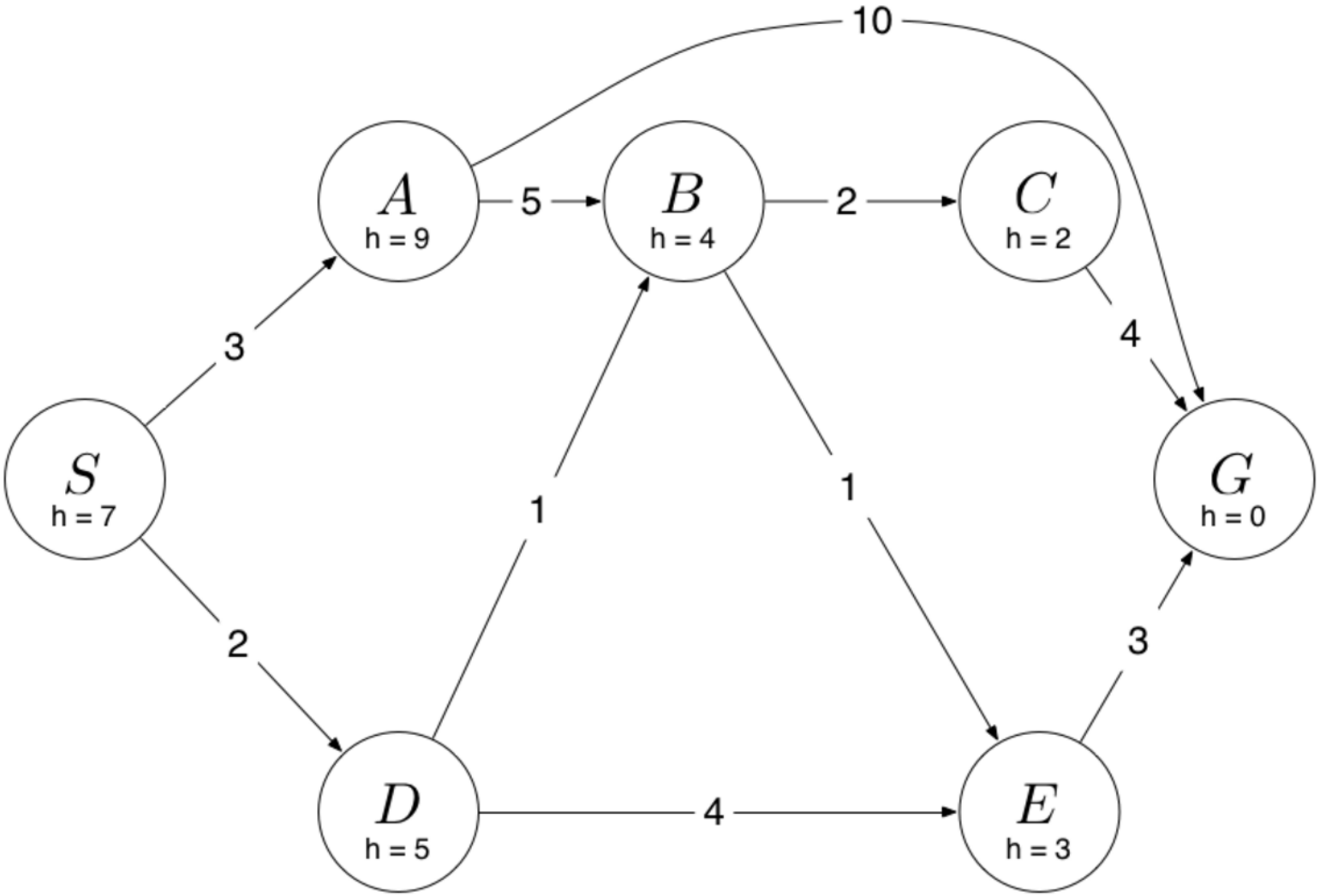


Nguồn ảnh: UC Berkely. CS188 Intro to AI

Ôn lại

- A* chọn kế hoạch nào từ fringe để mở rộng?
- So sánh A* với UCS và Greedy?

Chạy thử A* với đồ thị ở dưới



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Ôn lại

- Để A^* tìm được lời giải tối ưu thì heuristic phải thỏa điều kiện gì?
- Với một bài toán, làm sao để tạo ra heuristic?
- Heuristic càng chính xác thì sẽ có điểm lợi gì và có điểm hại gì?

Tổng thể

- Ôn lại các buổi học trước
- Cải tiến các thuật toán tìm kiếm: không mở lại các trạng thái đã mở

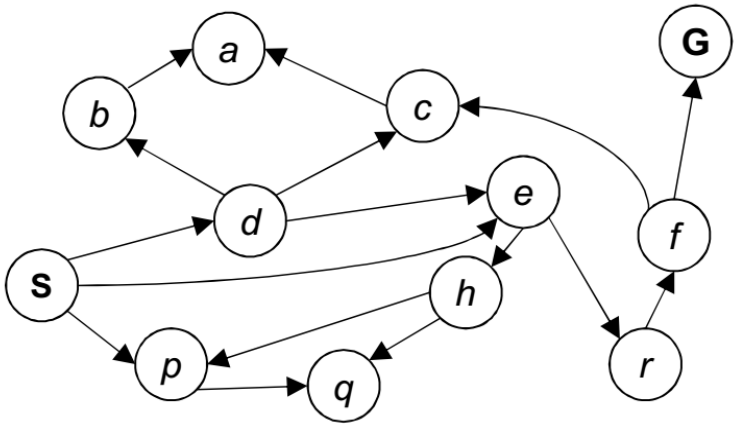
Cải tiến các thuật toán tìm kiếm

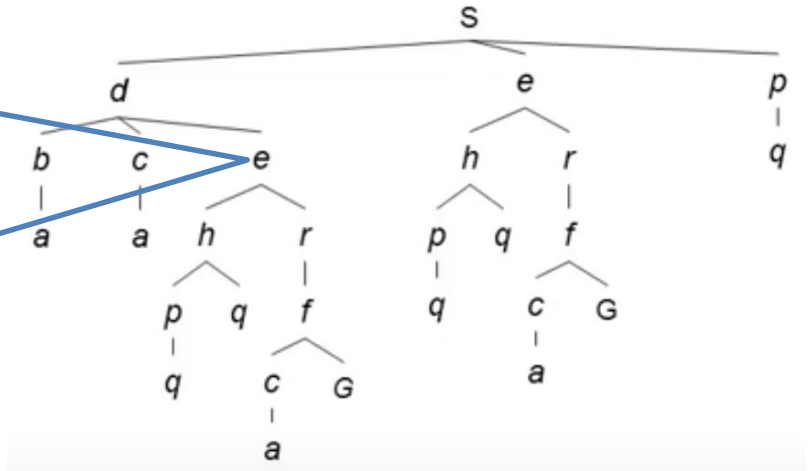
- *Ý tưởng: không mở lại các trạng thái đã mở*
- Tại sao lại muốn làm vậy?

BFS sẽ mở trạng thái e này. Việc này là không cần thiết vì: e trước đó đã được mở rồi (ở nhánh S → e)

Phần bên dưới của hai e này là giống hệt nhau ⇒ nếu e này không đi được tới G thì e kia cũng vậy ⇒ chỉ cần giữ lại một e là đủ

Nhưng giữ e được mở đầu tiên (ở S → e) có đảm bảo là BFS vẫn tìm được lời giải nông nhất?

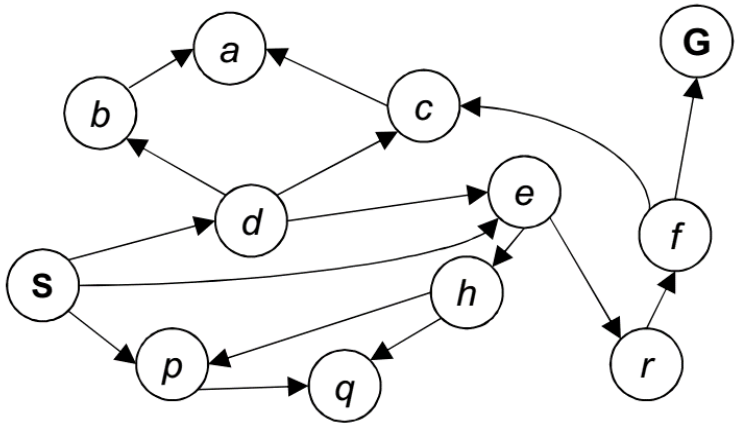




Nguồn ảnh: UC Berkely. CS188 Intro to AI

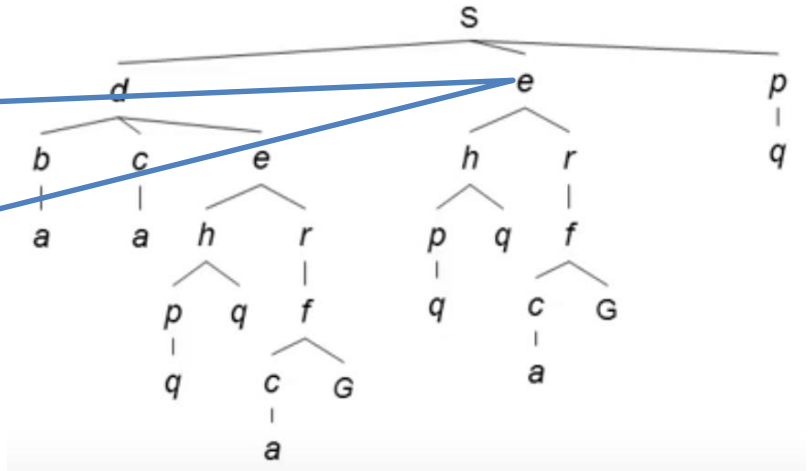
Cải tiến các thuật toán tìm kiếm

- *Ý tưởng: không mở lại các trạng thái đã mở*
- Tại sao lại muốn làm vậy?



DFS sẽ mở trạng thái e này. Việc này là không cần thiết vì: e trước đó đã được mở rồi (ở nhánh S → d → e)

Phần bên dưới của hai e này là giống hệt nhau ⇒ nếu e ở S → d → e đã không đi được tới G thì e ở S → e cũng sẽ không đi được tới G



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Cải tiến các thuật toán tìm kiếm

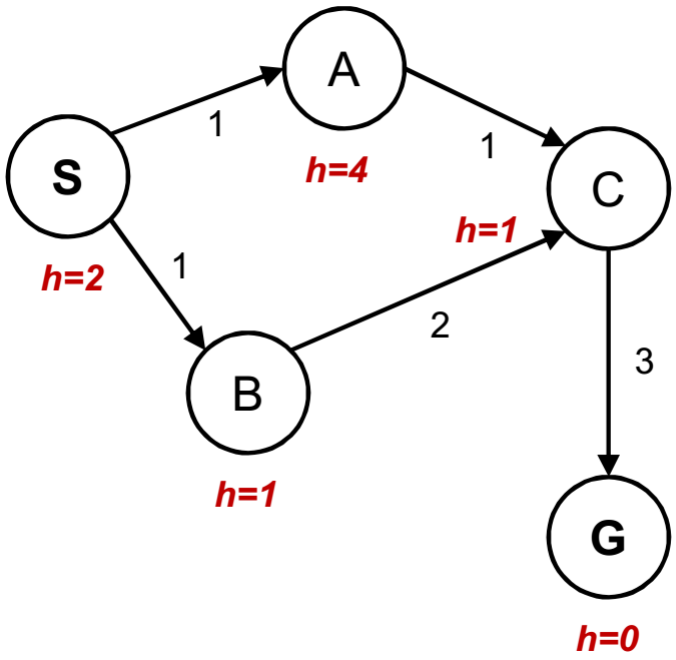
- *Ý tưởng: không mở lại các trạng thái đã mở*
- Cụ thể hơn:
  - Dùng biến **closed set** để lưu lại các trạng thái đã mở
  - Trước khi mở một trạng thái, kiểm xem đã có trong closed set hay chưa? Nếu chưa có thì mới mở, nếu có rồi thì bỏ qua
- Nếu trên cây tìm kiếm có nhiều trạng thái n thì theo trên ta sẽ chỉ giữ lại n được mở đầu tiên; nhưng nếu ta bỏ qua việc mở các trạng thái n còn lại thì có thể xảy ra việc không tìm được G vì G nằm ở dưới các trạng thái n đã bị bỏ qua không?
  - Không, vì ta đã giữ lại n được mở đầu tiên

Cải tiến các thuật toán tìm kiếm

- *Ý tưởng: không mở lại các trạng thái đã mở*
- Cụ thể hơn:
  - Dùng biến **closed set** để lưu lại các trạng thái đã mở
  - Trước khi mở một trạng thái, kiểm xem đã có trong closed set hay chưa? Nếu chưa có thì mới mở, nếu có rồi thì bỏ qua
- Nếu trên cây tìm kiếm có nhiều trạng thái n thì theo trên ta sẽ chỉ giữ lại n được mở đầu tiên; nhưng nếu ta bỏ qua việc mở các trạng thái n còn lại thì có thể xảy ra việc không tìm được G tối ưu vì G tối ưu nằm ở dưới các trạng thái n đã bị bỏ qua không?
  - DFS: không sao cả, vì DFS không quan tâm đến G tối ưu
  - BFS: BFS vẫn sẽ tìm được G nông nhất, vì n được mở đầu tiên là n nông nhất
  - UCS: UCS vẫn sẽ tìm được G tối ưu, vì n được mở đầu tiên là n tối ưu
  - A*: ?

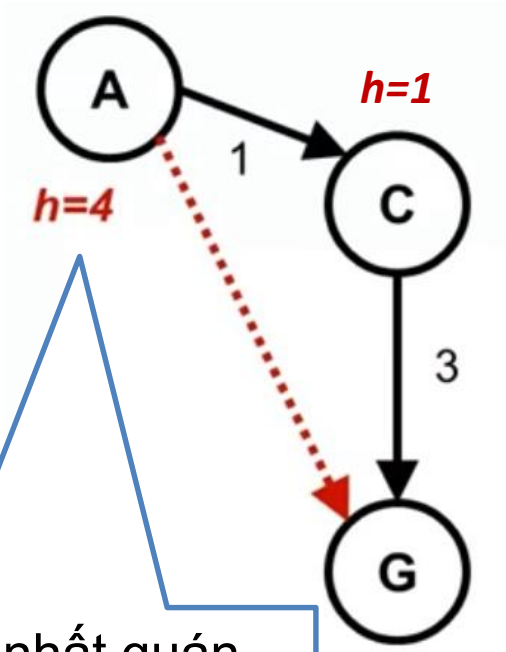
A* dùng closed set có tìm được lời giải tối ưu?

- Chạy thử A* dùng closed set với đồ thị ở bên …
  - Lời giải tìm được có tối ưu?
- Nếu chạy A* <u>không</u> dùng closed set thì sao?



Điều kiện để A* dùng closed set tìm được lời giải tối ưu

- Để A* không dùng closed set tìm được lời giải tối ưu thì heuristic phải hợp lệ: với mỗi trạng thái, chi phí ước lượng từ trạng thái đó đến đích ≤ chi phí thật sự từ trạng thái đó đến đích
  - Vd với A: h(A) ≤ chí phí thật sự từ A đến G
- Để A* dùng closed set tìm được lời giải tối ưu thì heuristic phải nhất quán (consistent): với mỗi cạnh, chi phí ước lượng ≤ chi phí thật sự
  - Vd với cạnh AC: h(A) - h(C) ≤ chi phí từ A đến C



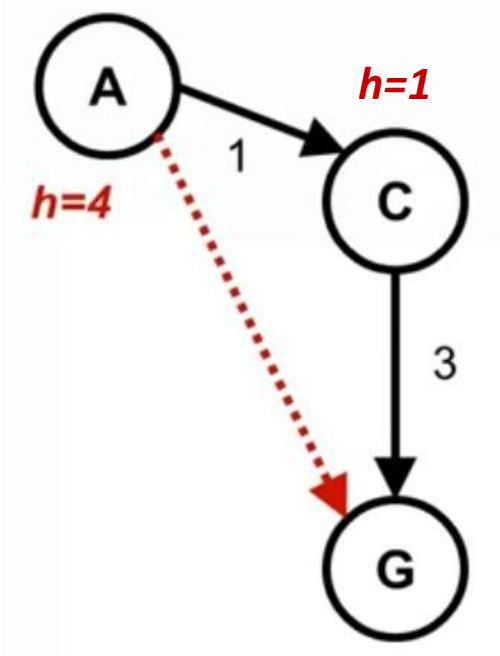
Hợp lệ nhưng không nhất quán

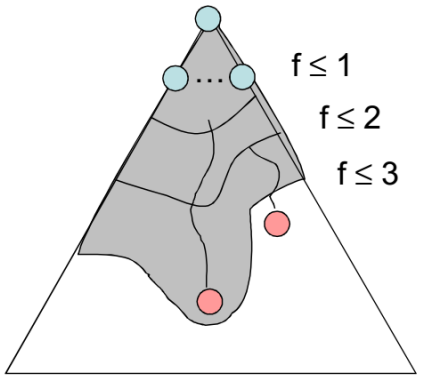
Thử thay h = 2 để nhất quán và chạy lại A* trong ví dụ trước …

Nguồn ảnh: UC Berkely. CS188 Intro to AI

Điều kiện để A* dùng closed set tìm được lời giải tối ưu

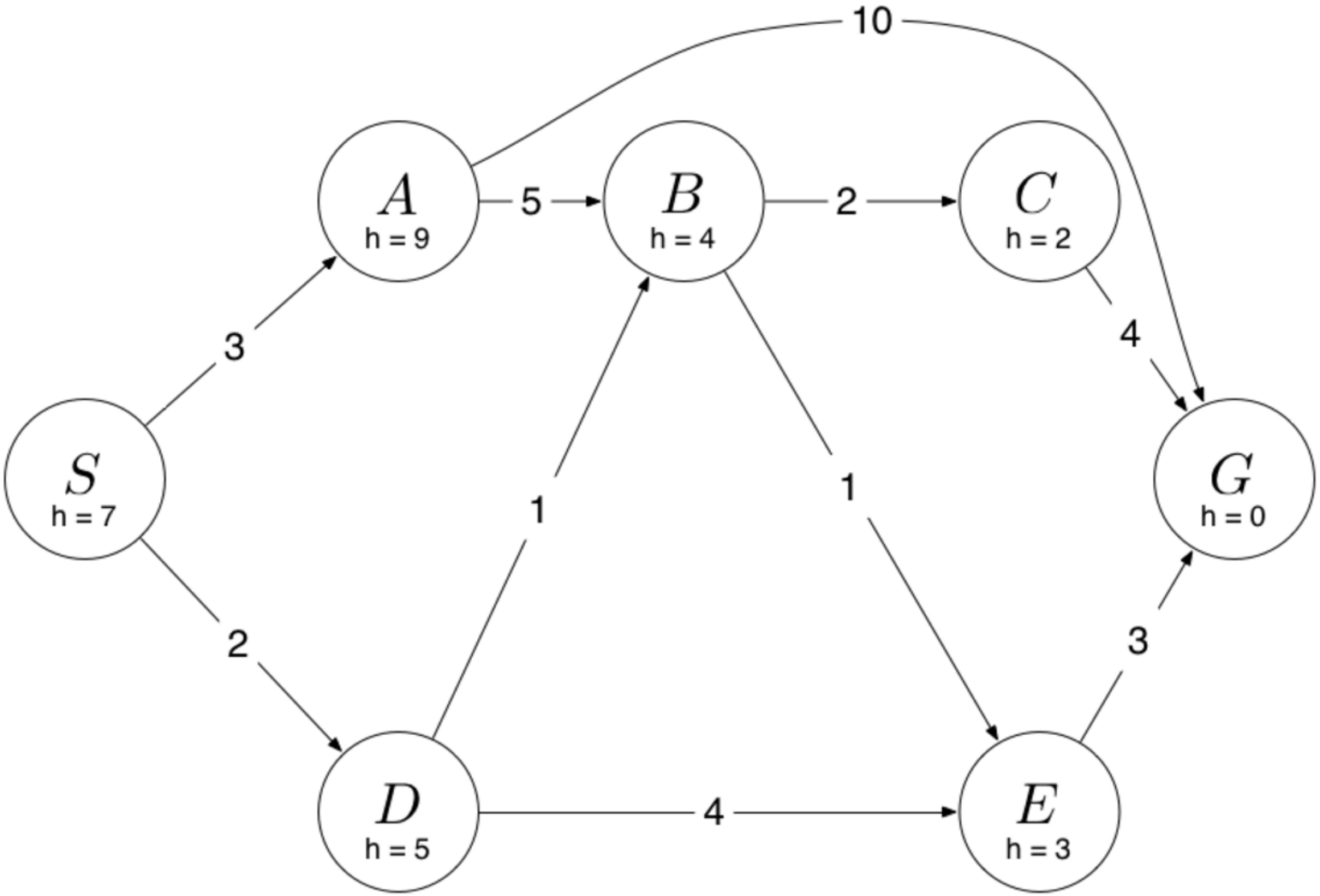
- Để A* dùng closed set tìm được lời giải tối ưu thì heuristic phải nhất quán (consistent): với mỗi cạnh, chi phí ước lượng ≤ chi phí thật sự
  - Vd với cạnh AC: h(A) - h(C) ≤ chi phí từ A đến C
- Tại sao? ★
  - Nếu h nhất quán thì giá trị f dọc theo một đường không bao giờ giảm
    - h(A) ≤ chi phí từ A đến C + h(C)
  - A* sẽ mở theo f tăng dần
  - Trong các trạng thái n trên cây, n đầu tiên được mở (và được giữ lại) là n có f nhỏ nhất; đây cũng là n có g nhỏ nhất





Nguồn ảnh: UC Berkely. CS188 Intro to AI

Quiz: h có nhất quán không?



Nguồn ảnh: UC Berkely. CS188 Intro to AI

Tổng kết về tính tối ưu của A*

- Để A* không dùng closed set tìm được lời giải tối ưu thì h phải hợp lệ
- Để A* dùng closed set tìm được lời giải tối ưu thì h phải nhất quán
- h nhất quán thì h sẽ hợp lệ, nhưng h hợp lệ thì không chắc là nhất quán
- Các h hợp lệ có được bằng cách nới lỏng bài toán thường sẽ nhất quán

Thuật toán tìm kiếm dùng closed set (còn được gọi là Graph Search, thuật toán tìm kiếm không dùng closed set được gọi là Tree Search)

Đầu vào: trạng thái bắt đầu, hàm successor, hàm kiểm tra trạng thái đích

Đầu ra: kế hoạch tìm được (chuỗi các hành động để đi từ trạng thái bắt đầu đến trạng thái đích)

Quá trình thực hiện:

- Khởi tạo:
  - `fringe`: gồm một kế hoạch ứng với trạng thái bắt đầu
  - `closed set`: rỗng
- Trong khi `fringe` chưa rỗng:
  - Lấy một kế hoạch ra khỏi `fringe` theo một chiến lược nào đó
  - Nếu kế hoạch này đi tới đích (dùng hàm kiểm tra trạng thái đích): RETURN kế hoạch!
  - Nếu trạng thái cuối của kế hoạch này chưa có trong `closed set`:
    - Đưa trạng thái vào `closed set`
    - Mở rộng ra (dựa vào hàm successor) và đưa các kế hoạch mới vào `fringe`
- Nếu ra được khỏi vòng lặp nghĩa là: không tìm thấy lời giải