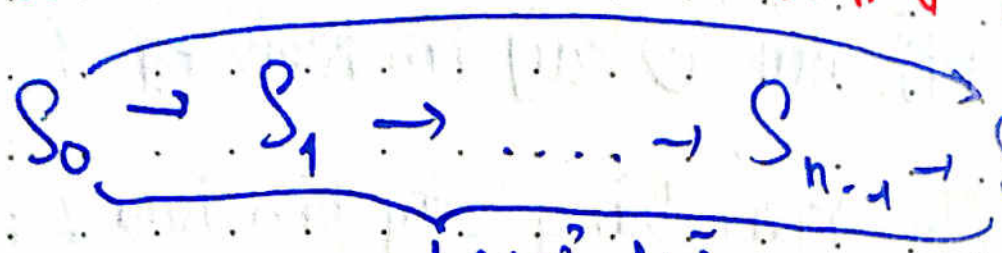
- Logic - Chứng minh

VẤN ĐỀ VỀ BÀI TOÁN TÌM KIẾM



$S \Rightarrow G$
$S_0 \Rightarrow S_n$

↓ Biểu diễn

Đồ thị
- Đỉnh: TT (Trạng thái)
- Cung: →

1. Định nghĩa vấn đề tìm kiếm

- s_{start} : Trạng thái bắt đầu
- Actions (s) : Hành động khả thi
- Cost (s, a) : Chi phí hành động
- Succ (s, a) : Trạng thái kế tiếp
- IsEnd (s) : Trạng thái kết thúc ??

Date: ..../..../....

- Bài toán tìm kiếm có năm thành phần:
  Q, S, G, succs, cost
  + Q: Tập hữu hạn các trạng thái
  + $S \subseteq Q$: Tập khác rỗng gồm các trạng thái ban đầu
  + $G \subseteq Q$: Tập khác rỗng gồm các trạng thái đích
  + succs(s): Là hàm nhận 1 trạng thái s làm đầu vào và trả về 1 tập trạng thái có thể đến từ s 0 1 bước
  + cost(s, s'): Là hàm nhận hai trạng thái s và s' làm đầu vào và trả về chi phí di chuyển 1 bước từ s đến s'. Hàm chỉ có ý nghĩa khi s' là trạng thái con của s.

2. Đồ thị không gian trạng thái vs. Cây tìm kiếm
- Mỗi nút ở cây tìm kiếm là 1 đường đi ở đồ thị không gian trạng thái.
- Tìm kiếm bằng cây tìm kiếm: Thực hiện:
  + Mở rộng ra các nút kề nó
  + Duy trì 1 bộ nhớ fringe.
  + Cố gắng mở rộng càng ít nút càng tốt.

3. Tìm kiếm theo chiều rộng (Breadth-First Search - BFS)
- Với trạng thái s bất kì đã gán nhãn:
  + previous(s) là trạng thái trước đó ở đường đi ngắn nhất tới

trạng thái START đến s.

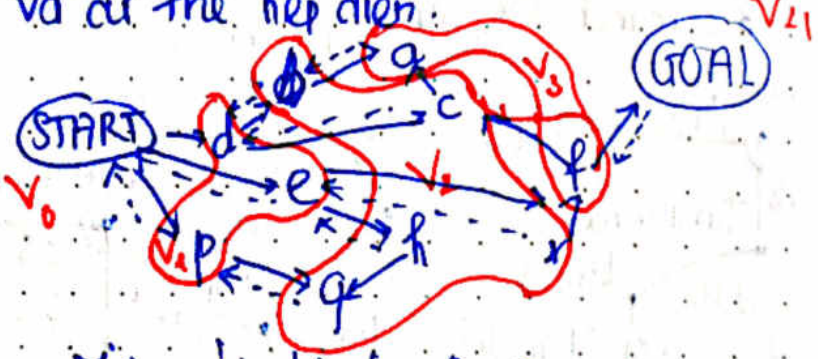
+ Tại vòng lặp thứ k của thuật toán, bắt đầu với V_k là tập các trạng thái mà đường đi ngắn nhất từ START đi đến chúng có đúng k bước.

+ Tiếp đó, ở vòng lặp, hình V_{k+1} là tập các trạng thái mà đường đi ngắn nhất từ START có đúng k+1 bước.

+ Ta bắt đầu với $k=0$, $V_0 = \{START\}$ và định nghĩa previous (START) = NULL.

+ Tiếp đó, thêm vào trạng thái đi 1 bước từ START vào V_1.

Và cứ thế tiếp diễn.



- Một cách khác: Đi lùi

+ Gán nhãn. Ứ trạng thái có thể đi đến GO k bước ⇒ có thể đi đến O ít hơn k bước.

+ ... Cho đến khi đi đến trạng thái START

+ Các nhãn "số bước đến đích" xác định đường đi ngắn nhất. Không cần thêm thông tin lưu trữ.

- Một số nhận xét:

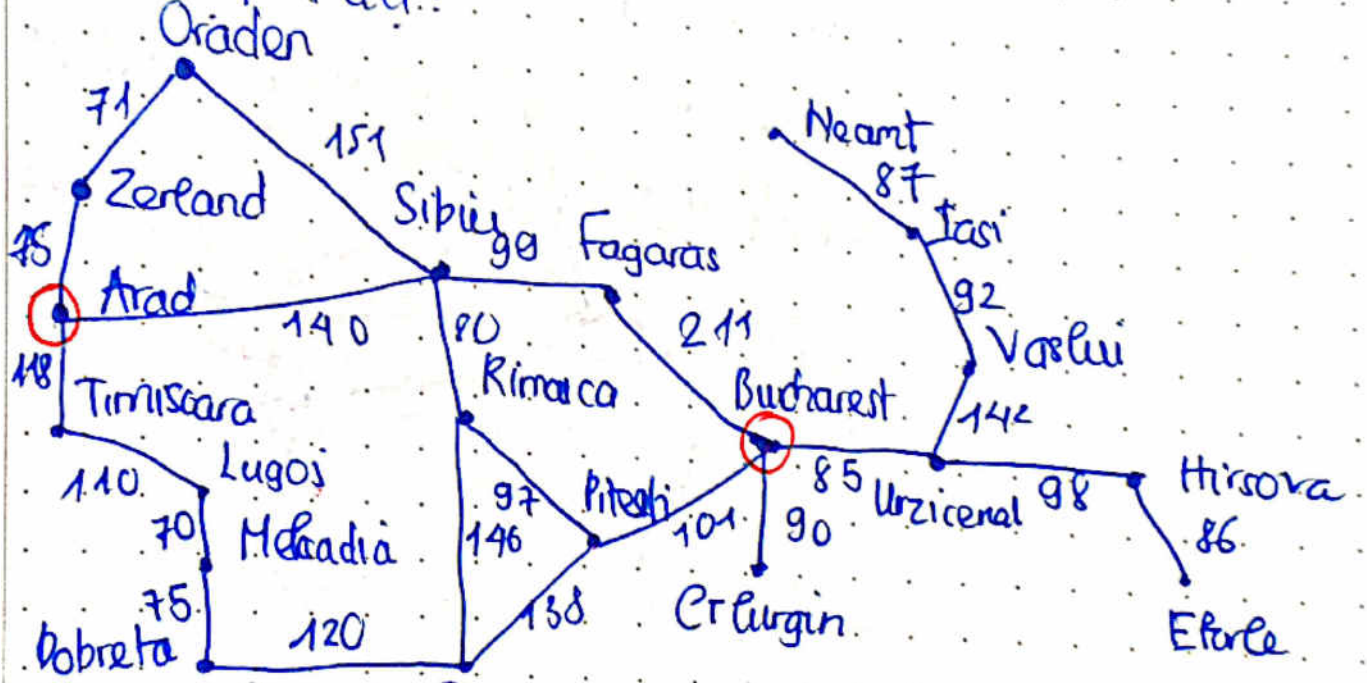
+ Vẫn chạy bình thường khi có ít nhất 1 trạng thái đích hoặc ít hơn 1 trạng thái ban đầu.

+ Thuật toán thực hiện lần lượt từ đỉnh đích lần lượt đến khi gặp trạng thái ban đầu.

Thuật toán thực hiện lần lượt từ trạng thái ban đầu gọi là suy diễn tiến/lùi.

+ Thuật toán này rất giống thuật toán Dijkstra's.

- Bài tập ví dụ:



$V_0 = \{Arad\}$ Craiova

$V_1 = \{Zerland, Sibiu, Timisoara\}$

$V_{12} = \{Oradea, Sibiu, Rimnicu, Lugoj, Fagaras\}$

$V_{23} = \{Craiova, Pitesti, Mehadia, Bucharest\}$

$V_3 = \{$(Sibiu đã có $\in V_1$ thì ko đưa vào V_3 nữa)

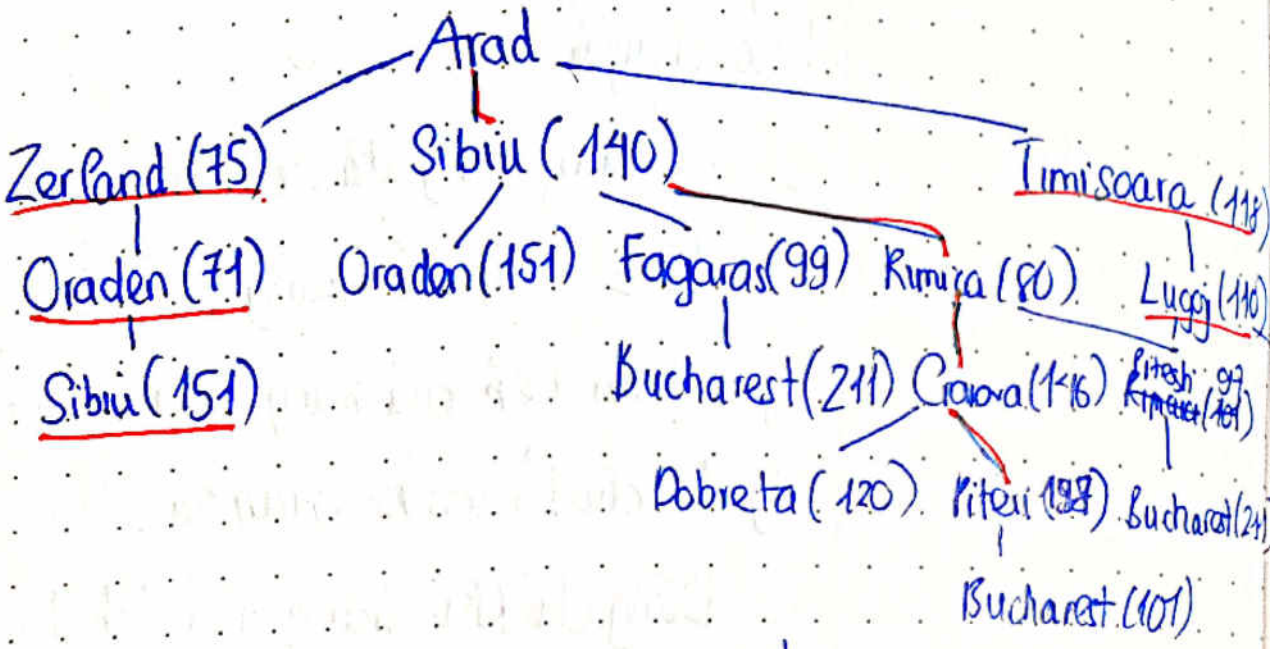
Date: ...../...../.....

Đã tìm thấy Goal, dừng thuật toán.

Đường đi là: Arad → Sibiu → Fagaras → Bucharest

Chi phí: 140 + 99 + 211 = 450

- Minh hoạ UCS



2. Tìm kiếm chi phí đồng nhất (Uniform - Cost Search)

2.1 Hàng đợi ưu tiên

- Cấu trúc dữ liệu cho phép thêm vào và lấy ra các cặp (thing, value) cần thao tác:

+ Init - PrioQueue (PQ): Khởi tạo hàng đợi ưu tiên rỗng

+ Insert - PrioQueue (PQ, thing, value): Thêm (thing, value) vào hàng đợi

+ Pop - Leaset (PQ): Trả về cặp (thing, value) có giá trị nhỏ nhất và loại nó ra khỏi hàng đợi

Date: ...../...../.....

- Hàng đợi ưu tiên có thể cài đặt sao cho chi phí thêm vào và lấy ra là O(log(số phần tử ở hàng đợi ưu tiên))

2.2 Tìm kiếm CP đúng nhất

- 1 hướng tiếp cận kiểu BFS đơn giản khi có chi phí trên các bước chuyển

- Sử dụng hàng đợi ưu tiên:

+ KQ = Tập các trạng thái đã được mở hay đang ~~được~~ đợi mở

+ Độ ưu tiên của trạng thái s = g(s) = chi phí đến s đúng đắn đã cho bởi con trỏ quay lui.

- Dừng chỉ khi trạng thái đích được lấy ra khỏi hàng đợi ưu tiên. Nếu KQ, ta sẽ bỏ lỡ 1 đường đi ≠ ngắn hơn.

- Đánh giá thuật toán:

+ Tính đầy đủ: Bao đảm tìm ra lời giải, nếu có

+ Tính tối ưu: Tìm đường đi có chi phí ít nhất

+ Đ PT về tg tính: Số node phát sinh

+ Đ PT về KG tính: Lg bộ nhớ sd

* Biến: N: Số trạng thái, B: số con trung bình (B>1), L: độ dài đường đi từ S → G vs số bước (chi phí) ít nhất, Q: kích thước hàng đợi ưu tiên trung bình.

Date: ...../...../.....

+ BFS : Tgian: $O(\min(N, B^L))$, Kgian: $O(\min(N, B^L))$
+ LCBFS: Tgian: $O(\min(N, B^L))$, Kgian: $O(\min(N, B^L))$
+ UCS: Tgian: $O(\log(Q) * \min(N, B^L))$, Kgian: $O(\min(N, B^L))$
+ DFS: Tgian $(O(B^{LMAX}))$, Kgian: $O(LMAX)$

- Ví dụ minh họa:

PQ = {(Arad, 0)}

PQ = {Zerind (75), Sibiu (140), Timisoara (118)}

PQ = {Oradea (146), Sibiu (140), Timisoara (118)}

PQ = {Oradea (146), Sibiu (140), Lugoj (229)}

PQ = {Oradea (146), Fagaras (239), RV (220), Lugoj (229)}

PQ = {Fagaras (239), RV (220), Lugoj (229)}

PQ = {Fagaras (239), Pitesti (317), Craiova (366), Lugoj (229)}

PQ = {Fagaras (239), Pitesti (317), Craiova (366), Mehadia (299)}

PQ = {Bucharest (450), Pitesti (317), Craiova (366), Mehadia (299)}

PQ = {Bucharest (450), Pitesti (317), Craiova (366), Dobrete (374)}

PQ = {~~Bucharest (450)~~, Bucharest (418), Craiova (366), Dobrete (374)}

PQ = {~~Bucharest (450)~~, Bucharest (418), ~~Pitesti (504)~~, Dobrete (374)}

PQ = {Bucharest (418), ~~Pitesti (504)~~, ~~Craiova (494)~~}

PQ = {}

Arad → Sibiu → RV → Pitesti → Bucharest - Chi phí: 418

Key points

Date: ...../...../.....

(S) Cost function g(n) → (n) h(n) heuristic function → (G) h(G)=0

g(S)=0

State space

- Ràng buộc để định nghĩa 1 hàm heuristic hợp lệ:

+ Không âm (Ko quan tâm dấu đi âm), tính đặc trưng cho mỗi bài toán.

+ n là goal, h(n) = 0

g(n) (n) h(n)

g(S)=0 (S) - UCS f(n) = g(n) (G) h(G)=0

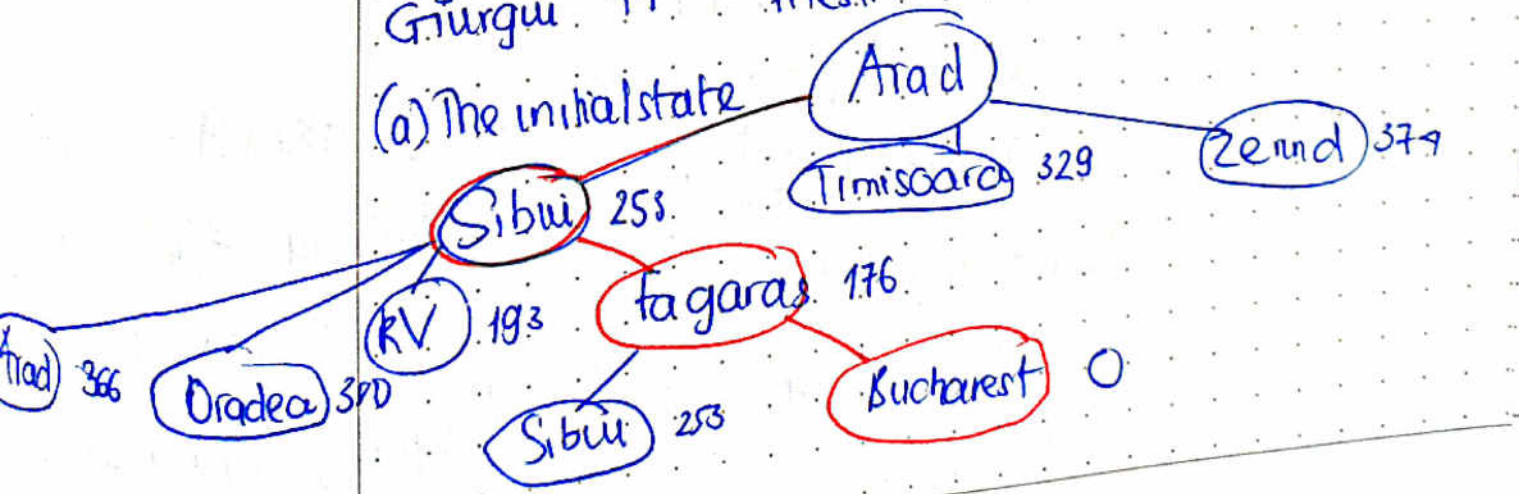
2. Greedy Best-first search chỉ xét đoạn của h(n)

f(n) = h(n)

VD: Có heuristic hước: Straight-line distance to Bucharest

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

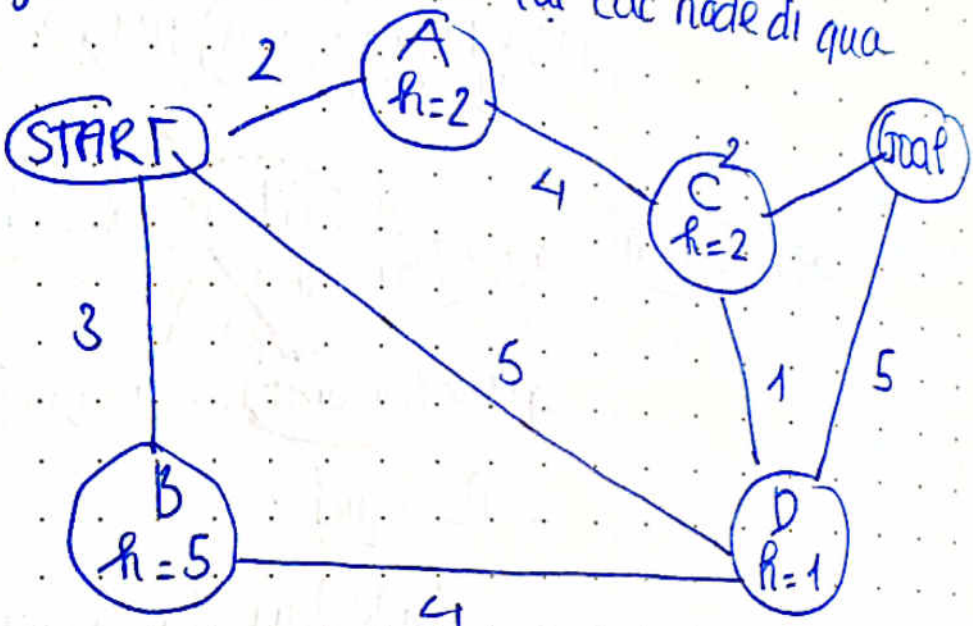
(a) The initial state



Summary

- Đánh giá: Frontier có 1 cấu trúc DL lưu trữ các nút chờ đc xét, tùy theo ngữ cảnh dùng gì priority queue, stack, ... f(n) = g(n) + h(n)
+ K° đảm bảo hình tối ưu
+ Tính đầy đủ ∈ vào Ràng heuristic có thể bị kẹt ở đây
+ Đpt tgian: $O(b^m)$ → m là độ sâu
+ Đpt kº gian: $O(b^m)$ lưu lại các node đi qua

Quiz 1:



Chi phí = 10
START → D → Goal

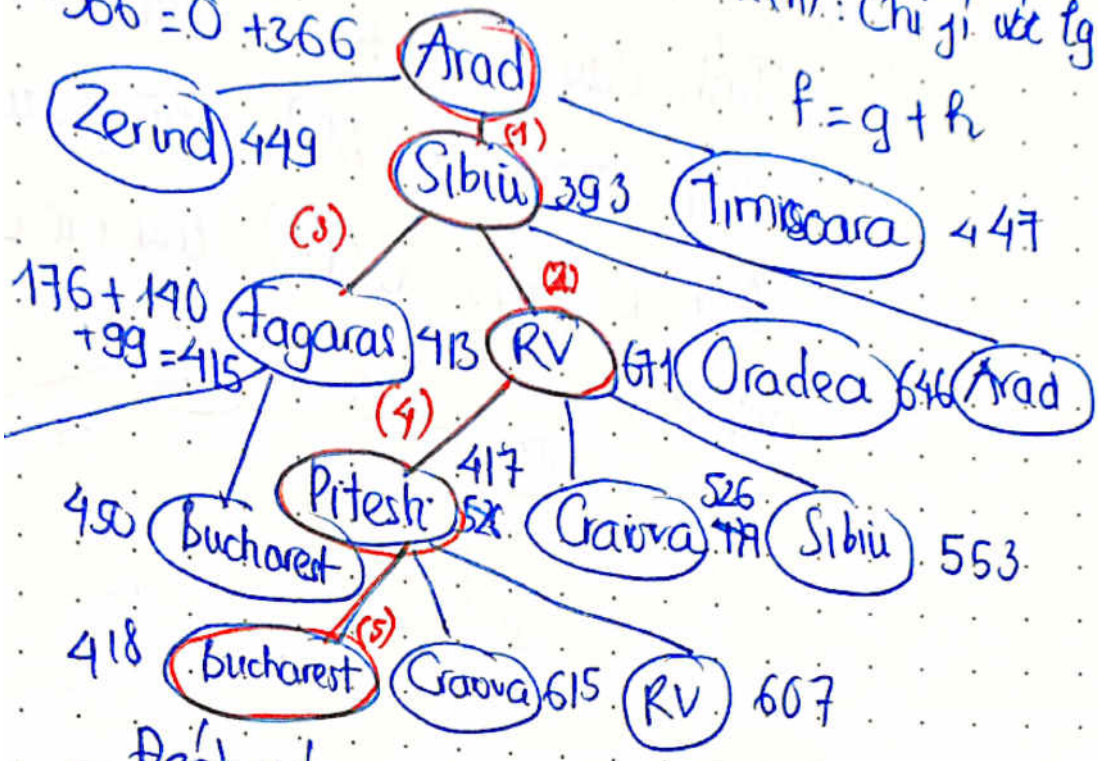
3. Tìm kiếm A*

- Giải thuật tốt nhất ở phần này.
- Sd heuristic ⇒ ko gải nguồn thông tin duy nhất, hành động đi có chi phí lớn → yên tâm sẽ nhận đc đường đi ngắn nhất (cần kiểm tra tính hợp lệ)

$f(n) = g(n) + h(n)$ — $g(n)$: chi phí từ n

$h(n)$: Chi phí ước lượng từ n đến G

$f = g + h$



- Đánh giá:

+ Tính đầy đủ, tối ưu hay nhiên với các điều kiện nhất định trên heuristic sử dụng (hệ thống tìm nhánh hữu hạn

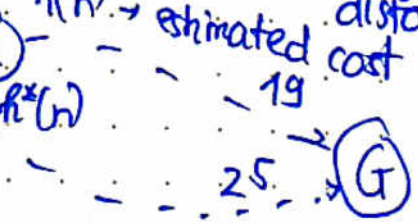
+ Độ phức tạp không gian, thời gian: Độ phức tạp hàm mũ phụ thuộc vào heuristic

(A^* không gian luôn nào cũng tối ưu)

- Một số điều kiện cho heuristic:

+ Tính chấp nhận được: không bao giờ ước lượng quá chi phí đến đỉnh goal (nhỏ hơn chi phí bỏ ra thực sự) h_{LSD} (the straight-line distance)

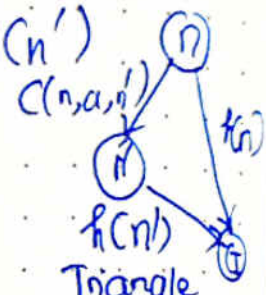
$$h(n) \leq h^*(n)$$



$h(n)$ sẽ không ước lượng quá True cost tới goal: $h^*(n)$ không xác định chính xác thế nào. $h^*(n)$ độ chính xác ước lượng = hoàn hảo

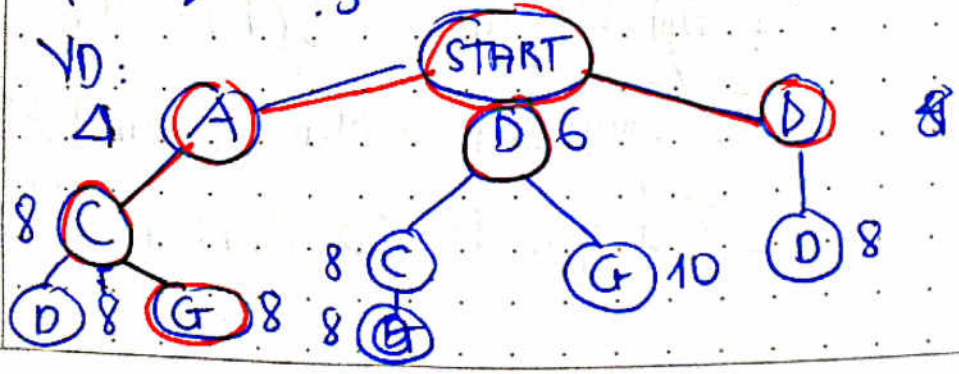
Date: ...../...../.....

- Đối với giải thuật A* dùng TREE-SEARCH, h(n) chấp nhận được thì đảm bảo tối ưu (Đc tiếp mở rộng chính đã đi qua)
- Đối với giải thuật A* dùng GRAPH-SEARCH, sd tập đóng close nếu đỉnh nào đã đi vào close k⁰ đc tiếp đi ra nữa, h(n) có 1 heuristic nhất quán.
- Consistency heuristic (Heuristic nhất quán)
+ Nếu tất cả các node n, ∀ con n' of n được sinh bởi hành động A bất kì: $h(n) \leq c(n,a,n') + h(n')$



+ Cũng chính là h admissible heuristic
- Đặc điểm A*: Ưu điểm
+ Mở rộng contour theo hình elip (UCS tròn)
+ Tối ưu, hiệu quả, k⁰ có các thuật toán nào tối ưu hơn.
- Nhược điểm
+ Các đỉnh mở vẫn lớn s tăng theo hàm mũ. (có thể hết tốn bộ k⁰ gian lưu trữ)
+ K⁰ gian trạng thái lớn có thể k⁰ phù hợp.

VD:



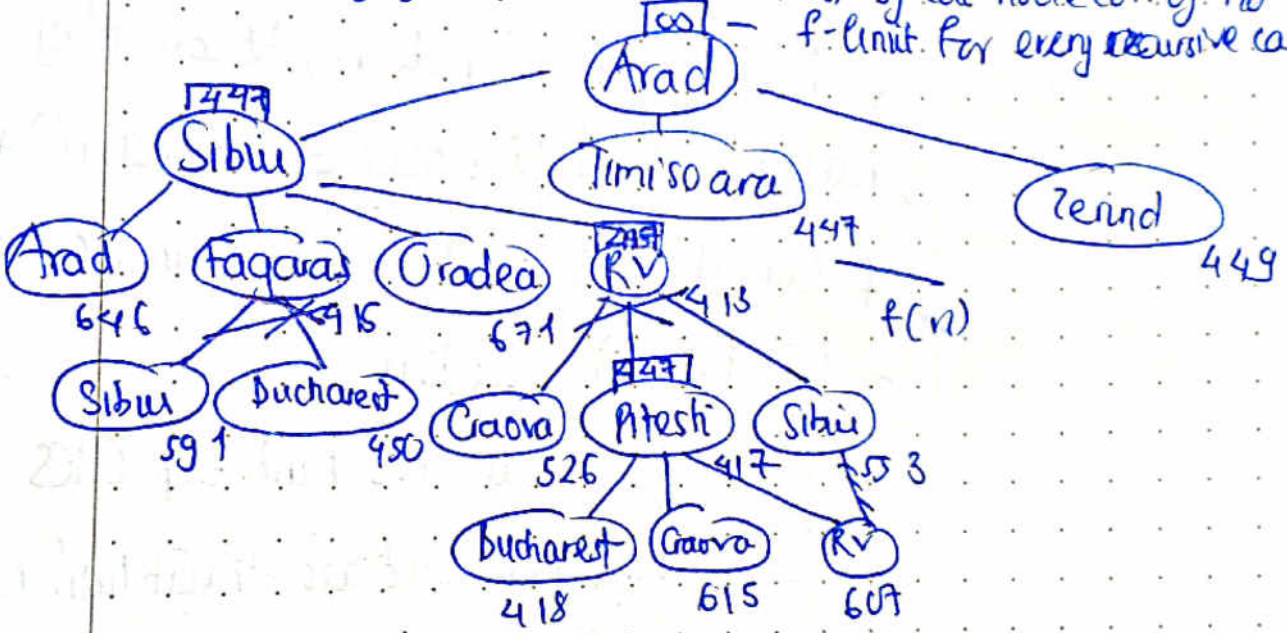
Key points

Date: ...../...../.....

4. Cải tiến Recursive Best-First Search (RBFS)

- Lưu f_value g đg đi thay thế tốt nhất g bất kì nút tổ tiên nào g nút đang đứng. Quay lui nếu ghi nút đang đứng > f_limit

- Trong quá trình quay lui, thay thế ghi f_value g mỗi node bằng giá trị đg đi tốt nhất g các node con g nó



- Đánh giá:

+ Đạt hình tới đích nếu h(n) chấp nhận đc

+ Đpt tgian: Phụ ∈ h(n), nhảy 1-1 các nhánh (chỉ giống nhất nên chạy BFS)

+ Đpt kgian: O(bd) Tốn ít bộ nhớ để tìm lại hay thăm

5. (Simplified) Memory-bound A* - (S)MA*

- Xoá ghi f-value lớn nhất nếu bộ nhớ full