

Informed Search Algorithms

(ARTIFICIAL INTELLIGENCE)

(slides adapted and revised from:

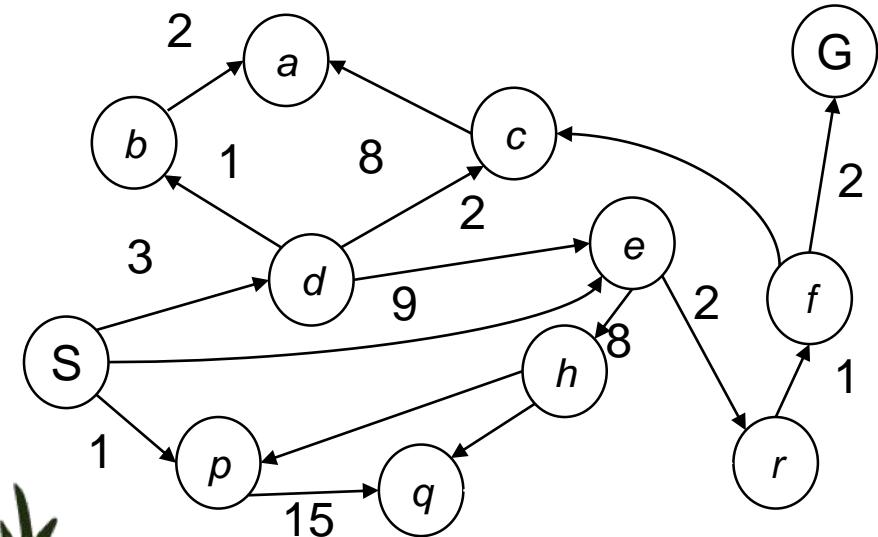
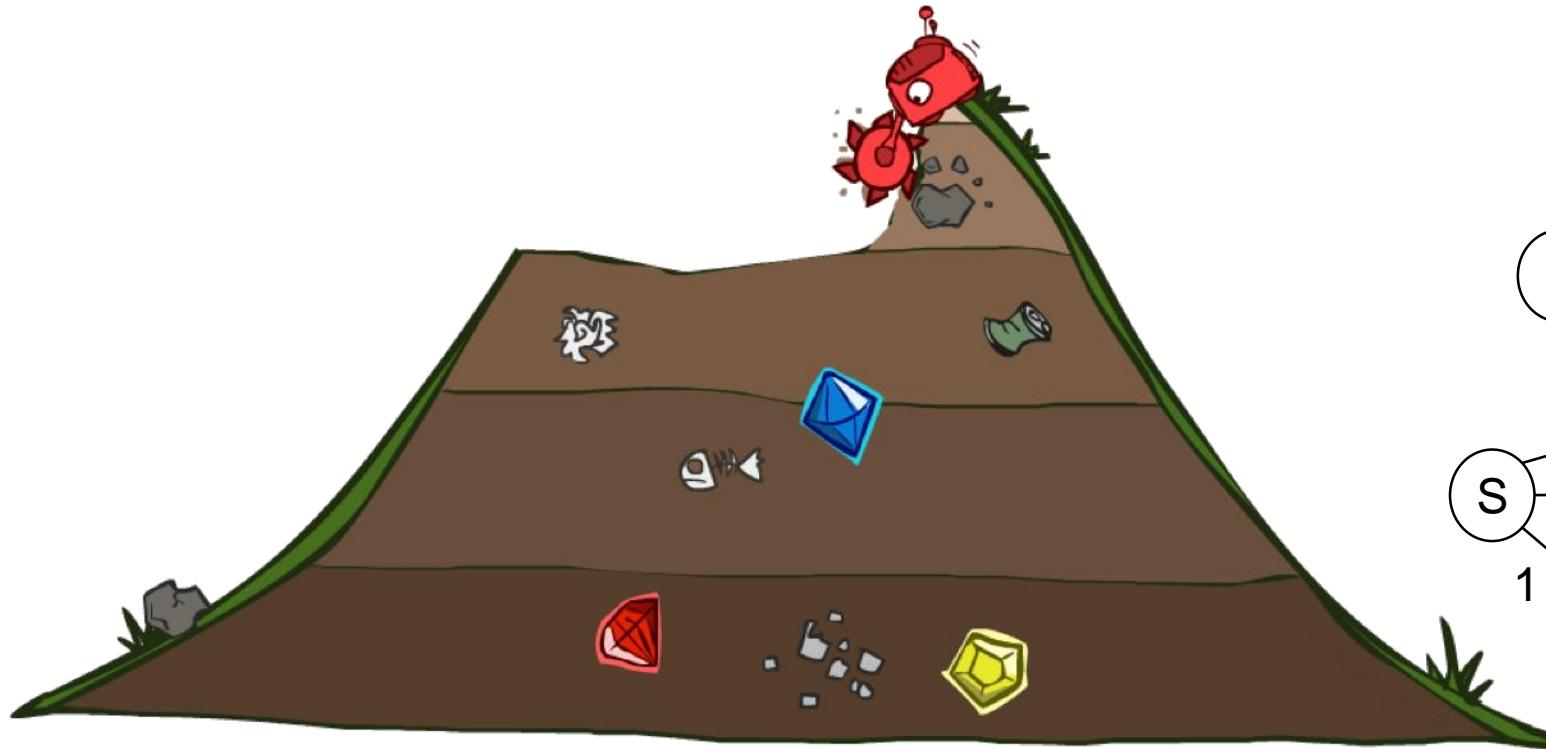
Dan Klein, Pieter Abbeel, Anca Dragan, et al,

Artificial Intelligence. Copyright © 2004 by Massachusetts Institute of Technology,

RUSSELL & NORVIG Artificial Intelligence: A Modern Approach, 3rd ed.

Giáo trình Nhập môn Trí tuệ nhân tạo của Từ Minh Phương)

Uniform Cost Search



Uniform Cost Search

- Phương pháp: Tìm kiếm chi phí đồng nhất sử dụng tổng chiều dài (hoặc chi phí) của một path để quyết định path nào sẽ được mở rộng.

Uniform Cost Search

Đầu vào: Search problem

Đầu ra: Solution

Khởi tạo: $O \leftarrow \text{node } S$ (S là Start state)

//chú ý: node S là một đường đi từ Start state đến state S

//tương tự: node N là một đường đi từ Start state S đến state N

while($O \neq \emptyset$) do

 1. Lấy node N có $g(N)$ thấp nhất **ra khỏi O** ;

 // $g(N)$ là tổng chi phí từ node S đến node hiện tại N

 2. Nếu node $N \in G$ (Goal state) then return Solution;

 //(là đường đi từ state S đến state N)

 3. Với mọi state $M \in P(N)$:

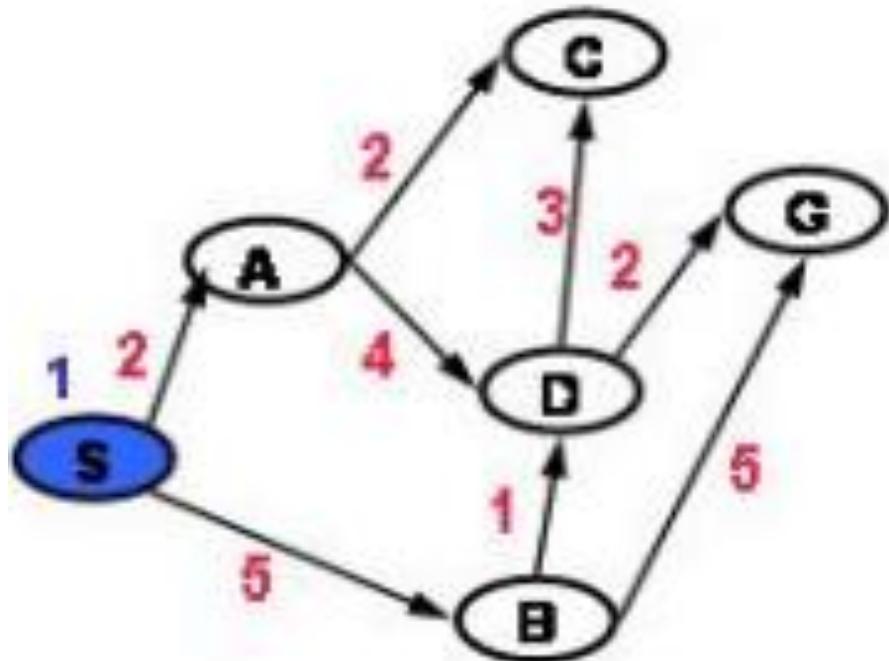
 // $P(N)$ là tập các trạng thái con của state N

 Thêm node M vào O cùng giá trị $g(M)$;

return: không tìm được lời

Uniform Cost Search

Ví dụ:

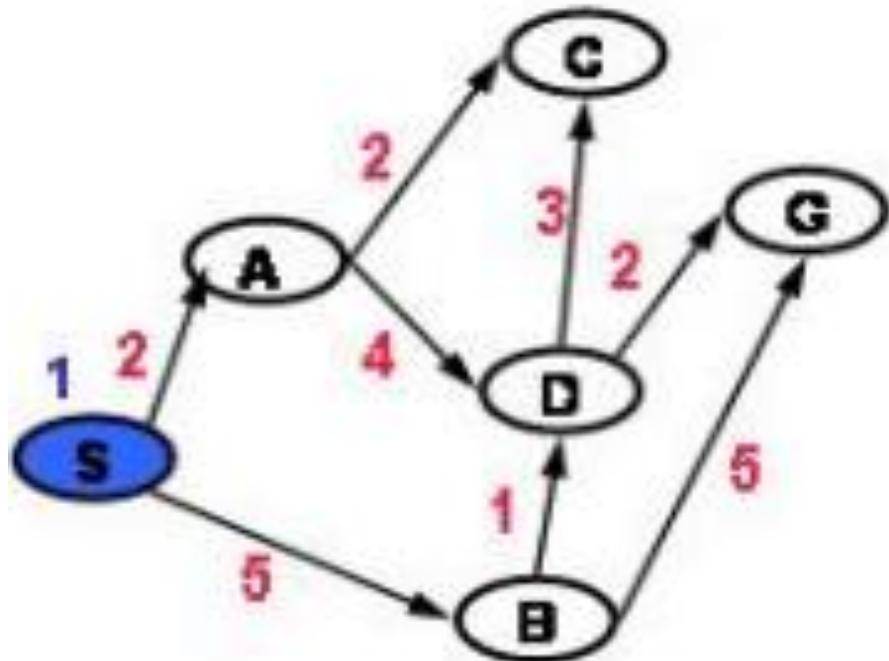


Nút được xét	Tập biên O
	<u>(0, S)</u>

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

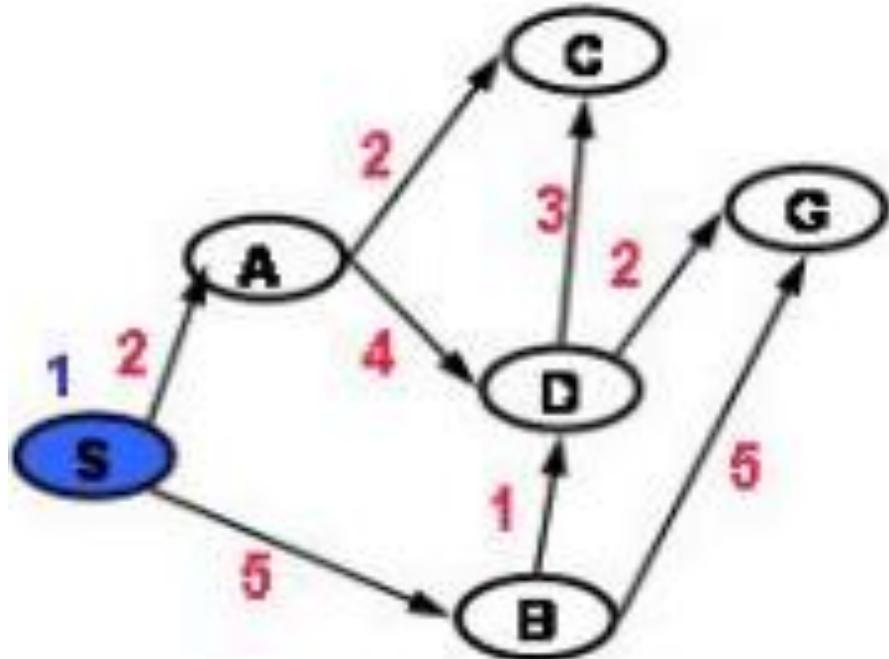


Nút được xét	Tập biên O
(0, S)	
(0, S)	

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

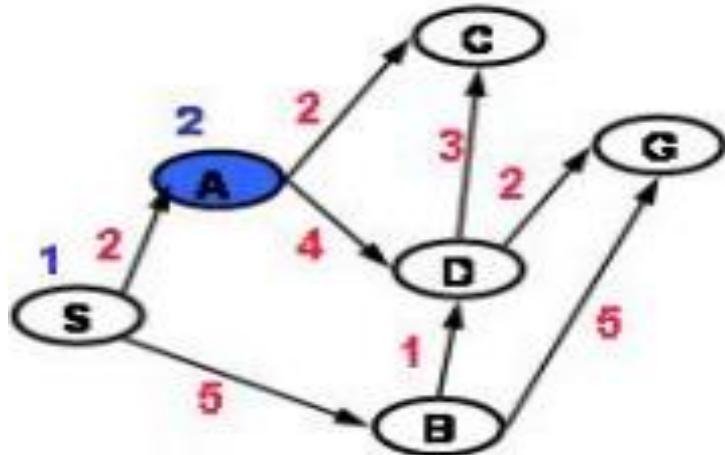


Nút được xét	Tập biên O
	(0, S)
(0, S)	(2,AS) (5,BS)

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

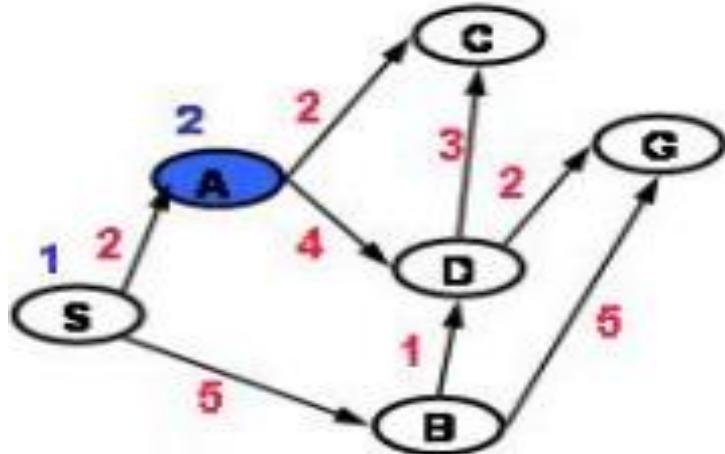


Nút được xét	Tập biên O (priority queue)
	(0, S)
(0, S)	(2,AS) (5,BS)
(2,AS)	

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

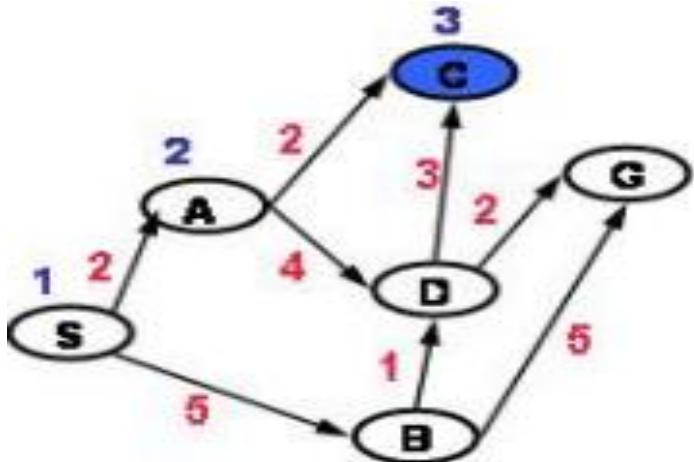


Nút được xét	Tập biên O priority queue
	(0, S)
(0, S)	(2, AS) (5, BS)
(2, AS)	(4, CAS) (6, DAS) (5, B S)

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

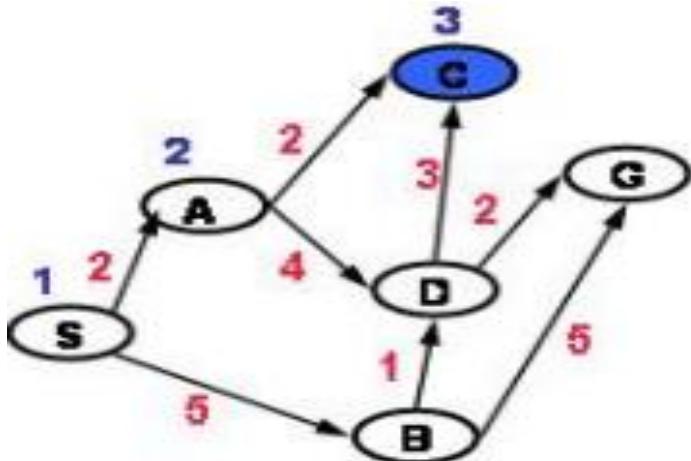


Nút được xét	Tập biên O priority queue
(0, S)	
(0, S)	(2,AS) (5,BS)
(2,AS)	(4, CAS) (6,DAS)(5, BS)
(4, CAS)	

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

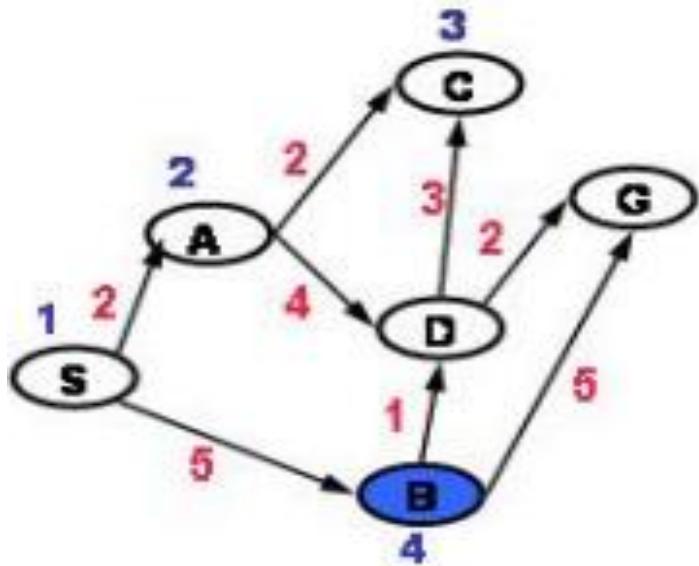


Nút được xét	Tập biên O priority queue
	(0, S)
(0, S)	(2, AS) (5, BS)
(2, AS)	(4, C AS) (6, DAS) (5, BS)
(4, CAS)	(6, DAS)(5, BS)

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

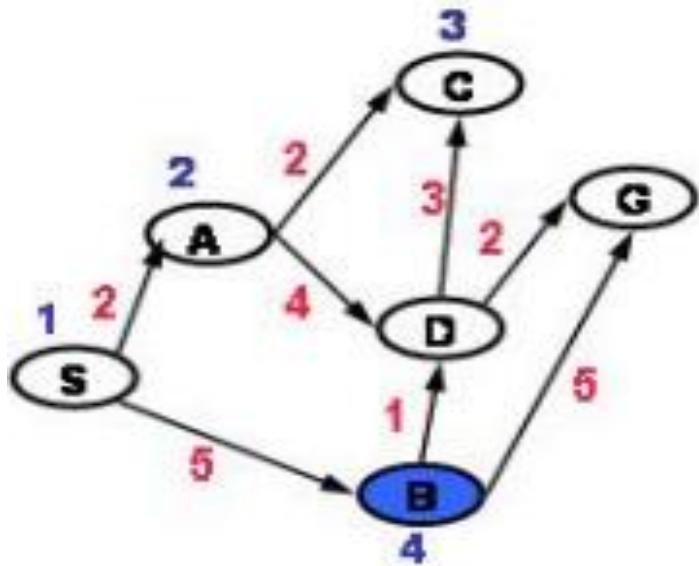


Nút được xét	Tập biên O priority queue
	(0, S)
(0, S)	(2, AS) (5, BS)
(2, AS)	(4, CAS) (6, DAS) (5, BS)
(4, C AS)	(6, DAS)(5, BS)
(5, BS)	

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

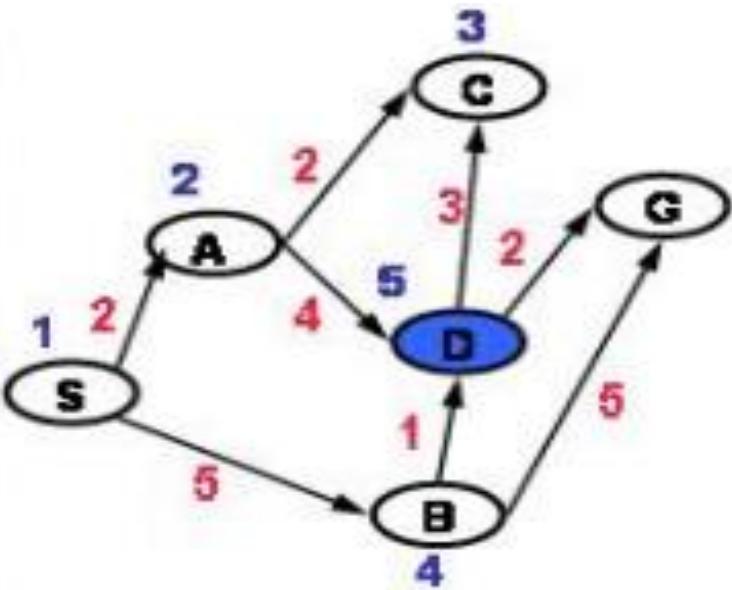


Nút được xét	Tập biên O priority queue
	(0, S)
(0, S)	(2,AS) (5,BS)
(2,AS)	(4, CAS) (6,DAS) (5, BS)
(4, CAS)	(6,DAS)(5, BS)
(5, BS)	<u>(6,DBS) (10,GBS)</u> (6,DAS)

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

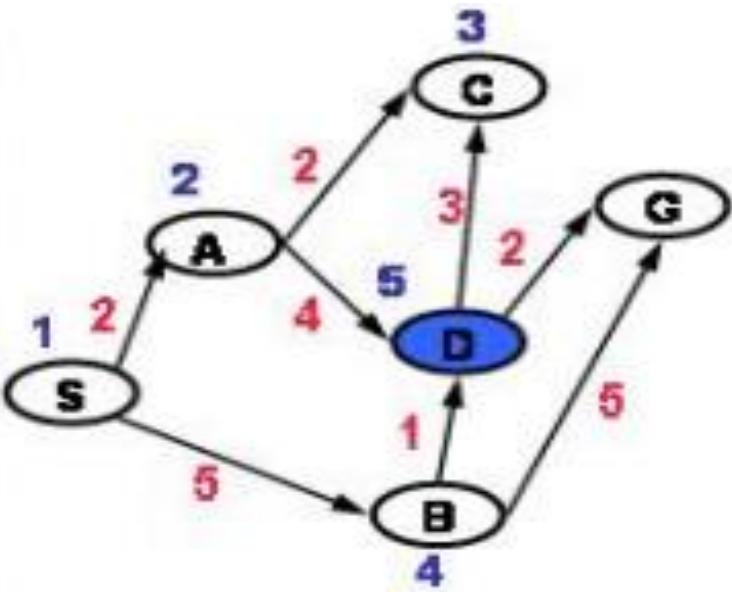


Nút được xét	Tập biên O priority queue
(0, S)	
(0, S)	(2,AS) (5,BS)
(2,AS)	(4, CAS) (6,DAS) (5, BS)
(4, CAS)	(6,DAS) (5, BS)
(5, BS)	(6,DBS) (10,GBS) (6,DAS)
(6,DBS)	

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

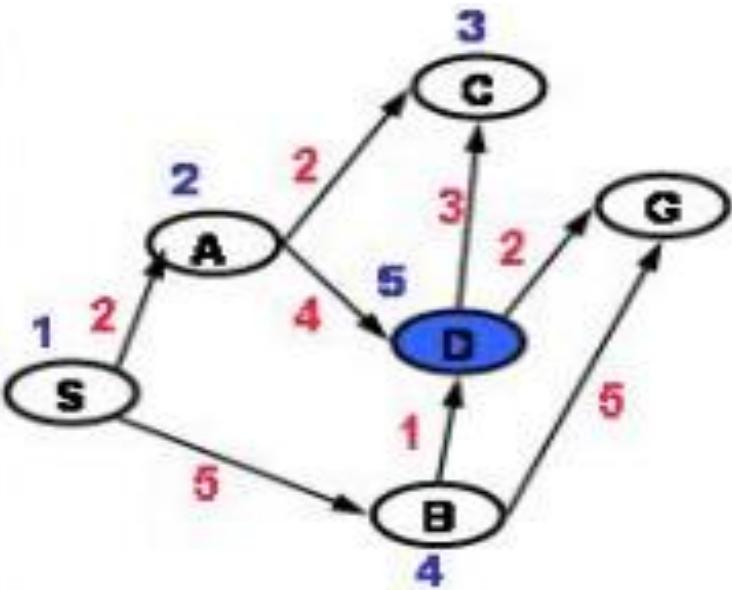


Nút được xét	Tập biên O priority queue
	(0, S)
(0, S)	(2, AS) (5, BS)
(2, AS)	(4, CAS) (6, DAS) (5, BS)
(4, CAS)	(6, DAS) (5, BS)
(5, BS)	(6, DBS) (10, GBS) (6, DAS)
(6, DBS)	<u>(8, GDBS) (9, CDBS) (10, GBS) (6, DAS)</u>

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

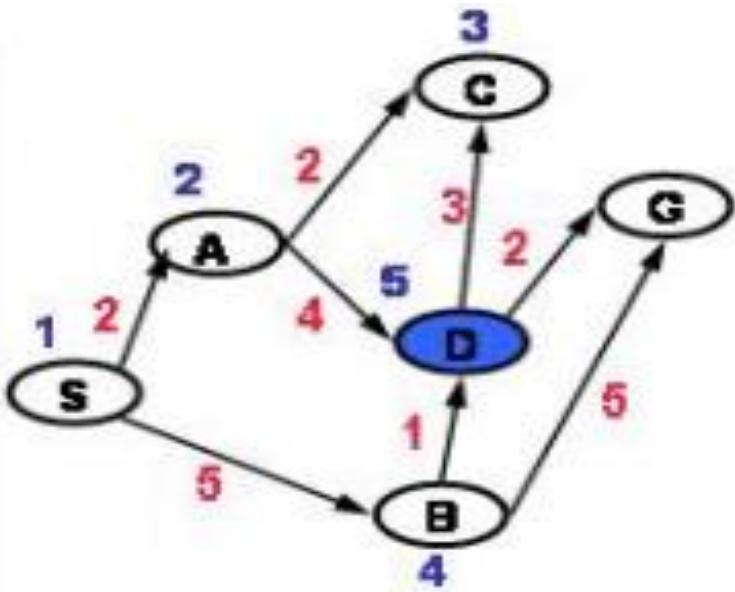


Nút được xét	Tập biên O priority queue
(0, S)	
(0, S)	(2,AS) (5,BS)
(2,AS)	(4, CAS) (6,DAS) (5, BS)
(4, CAS)	(6,DAS) (5, BS)
(5, BS)	(6,DBS) (10,GBS) (6,DAS)
(6,DBS)	<u>(8,GDBS) (9,CDBS)</u> (10,GBS) (6,DAS)
(6,DAS)	

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

Ví dụ:

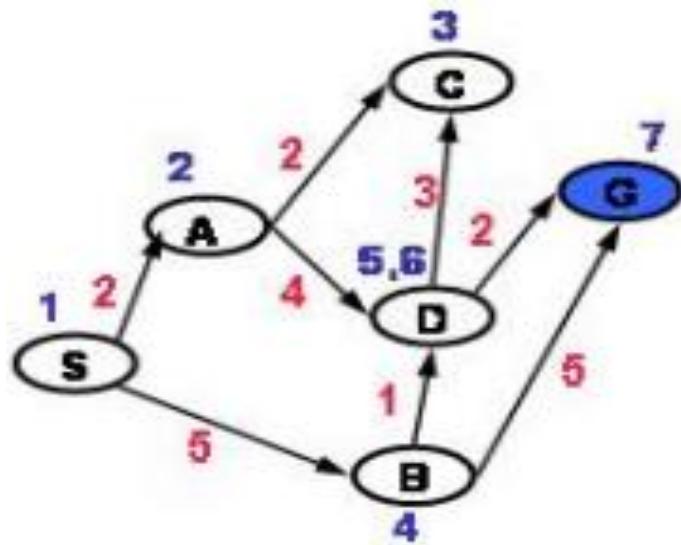


Nút được xét	Tập biên O priority queue
	(0, S)
(0, S)	(2, AS) (5, BS)
(2, AS)	(4, CAS) (6, DAS) (5, BS)
(4, CAS)	(6, DAS) (5, BS)
(5, BS)	(6, DBS) (10, GBS) (6, DAS)
(6, DBS)	(8, GDBS) (9, CDBS) (10, GBS) (6, DAS)
(6, DAS)	<u>(8, GDAS) (9, CDAS)</u> (8, GDBS) (9, CDBS) (10, GBS)

Được gạch chân là các node mới được mở rộng

Uniform Cost Search

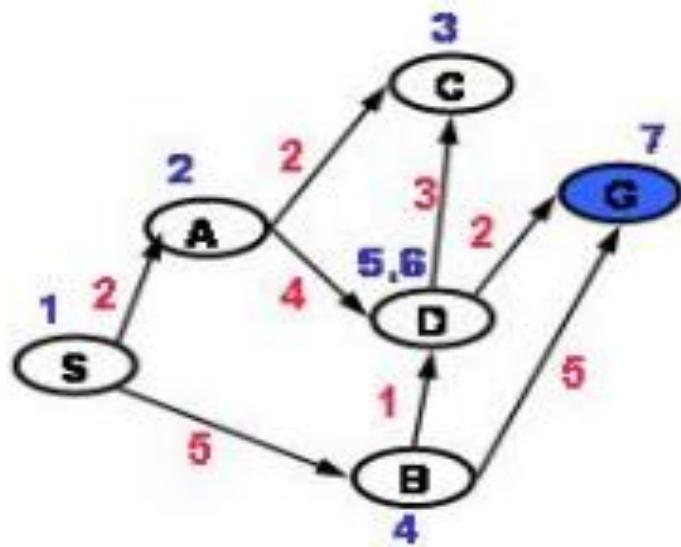
Ví dụ:



Nút được xét	Tập biên O priority queue
	(0, S)
(0, S)	(2,AS) (5,BS)
(2,AS)	(4, CAS) (6,DAS) (5, BS)
(4, CAS)	(6,DAS) (5, BS)
(5, BS)	(6,DBS) (10,GBS) (6,DAS)
(6,DBS)	(8,GDBS) (9,CDBS) (10,GBS) (6,DAS)
(6,DAS)	(8,GDAS) (9,CDAS) (8,GDBS) (9,CDBS) (10,GBS)
(8,GDAS)	Đích

Uniform Cost Search

Ví dụ:

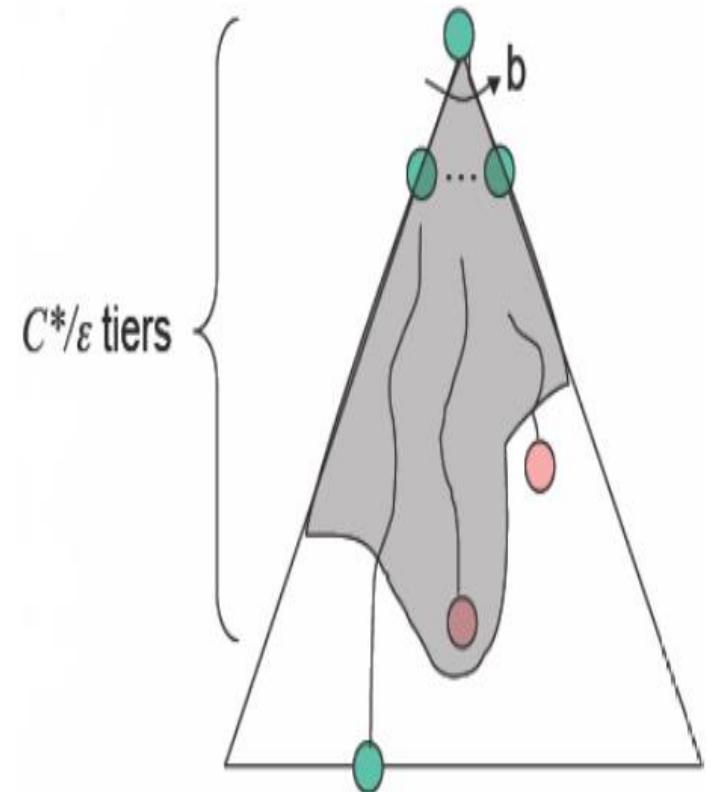


Nút được xét	Tập biên O priority queue
(0, S)	
(0, S)	(2, A, S) (5, B, S)
(2, A, S)	(4, C, A, S) (6, D, A, S) (5, B, S)
(4, C, A, S)	(6, D, A, S) (5, B, S)
(5, B, S)	(6, D, B, S) (10, G, B, S) (6, D, A, S)
(6, D, B, S)	(8, G, D, B, S) (9, C, D, B, S) (10, G, B, S) (6, D, A, S)
(6, D, A, S)	(8, G, D, A, S) (9, C, D, A, S) (8, G, D, B, S) (9, C, D, B, S) (10, G, B, S)
(8, G, D, A, S)	Đích

Chúng ta đã tìm ra con đường ngắn nhất (S->A->D->G) với chi phí bằng 8

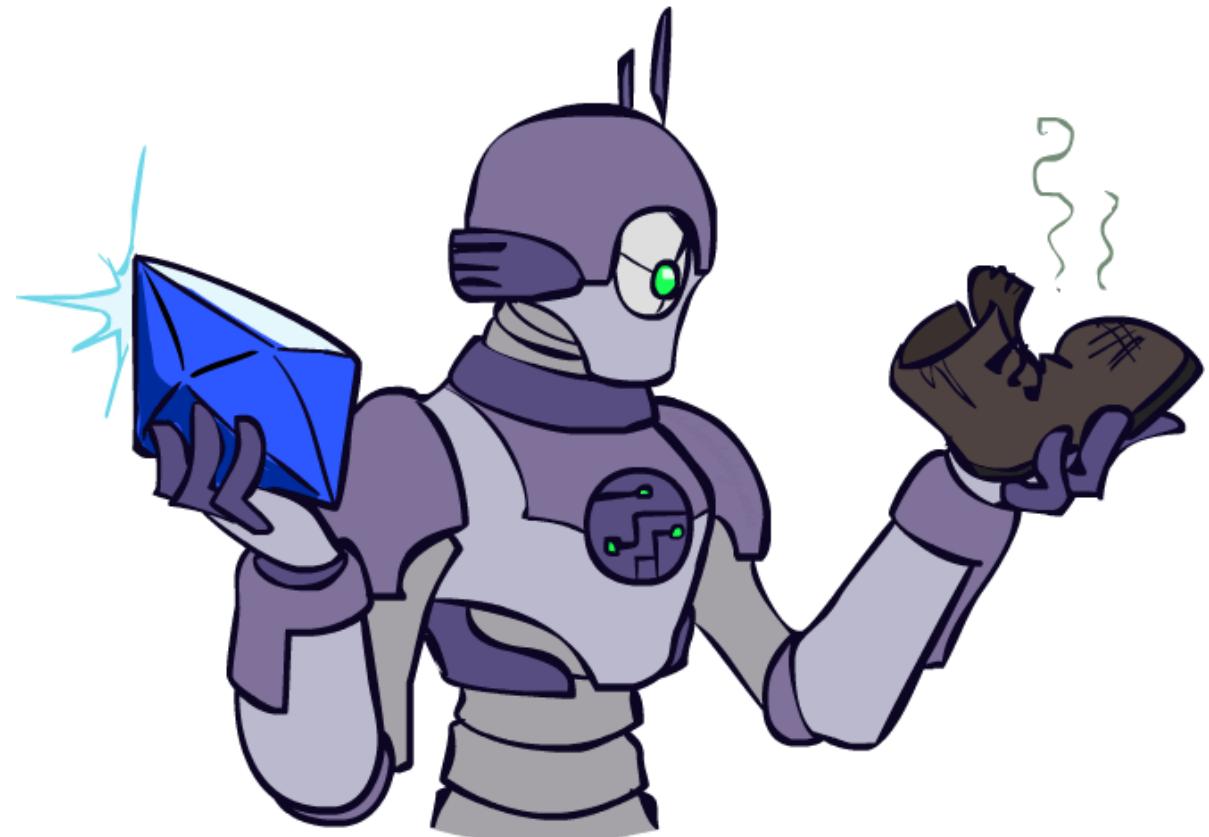
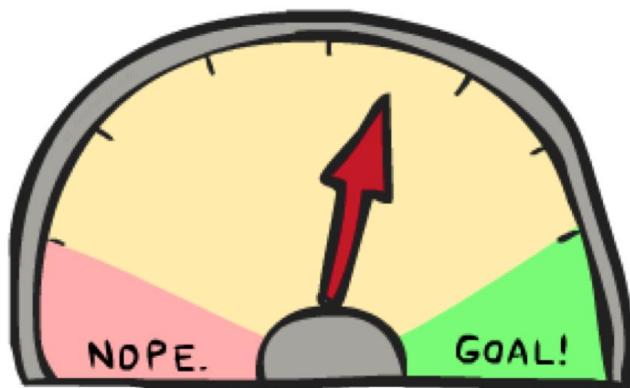
Uniform Cost Search

- Đặc điểm:
 - Thuật toán đầy đủ, tức là đảm bảo tìm ra lời giải nếu có nếu chi phí các bước $\geq \varepsilon$.
 - Độ phức tạp tính toán và độ phức tạp không gian:
 - Xấu nhất: $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$
 - Tốt nhất: $O(b^{d+1})$
 - Tối ưu nếu các nút được mở rộng theo thứ tự tăng dần của $g(n)$



Content

- Informed Search
 - Best first search (Greedy search)
 - Heuristics
 - A* Search



Informed Search

- Các chiến lược tìm kiếm cơ bản (uninformed search): các nút biên lần lượt được mở rộng theo một thứ tự nhất định mà không tính tới việc ưu tiên các nút có khả năng dẫn tới lời giải nhanh hơn
 - Không phù hợp với nhiều bài toán thực tế (do đòi hỏi chi phí quá cao về thời gian và bộ nhớ)
- Các chiến lược tìm kiếm có thông tin (informed search) sử dụng các tri thức bổ sung → Quá trình tìm kiếm hiệu quả hơn

Informed 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 của tập biên theo trật tự giảm dần về mức độ phù hợp
- Các trường hợp đặc biệt:
 - Greedy (best-first) search
 - A* search

Greedy Search



Greedy Search

Đầu vào: Search problem, hàm heuristic h

Đầu ra: Solution

Khởi tạo: $O \leftarrow$ node S (S là Start state)

//chú ý: node S là một đường đi từ Start state đến state S

//tương tự: node N là một đường đi từ Start state S đến state N

while($O \neq \emptyset$) do

 1. Lấy node N có $h(N)$ nhỏ nhất ra khỏi O ;

 // $h(N)$ là ước lượng chi phí từ node N đến node đích G (Goal state)

 2. Nếu node $N \in G$ then return Solution (là đường đi từ state S đến state N);

 3. Với mọi state $M \in P(N)$:

 // $P(N)$ là tập các trạng thái được mở rộng từ state N

 Thêm node M vào O cùng giá trị $h(M)$;

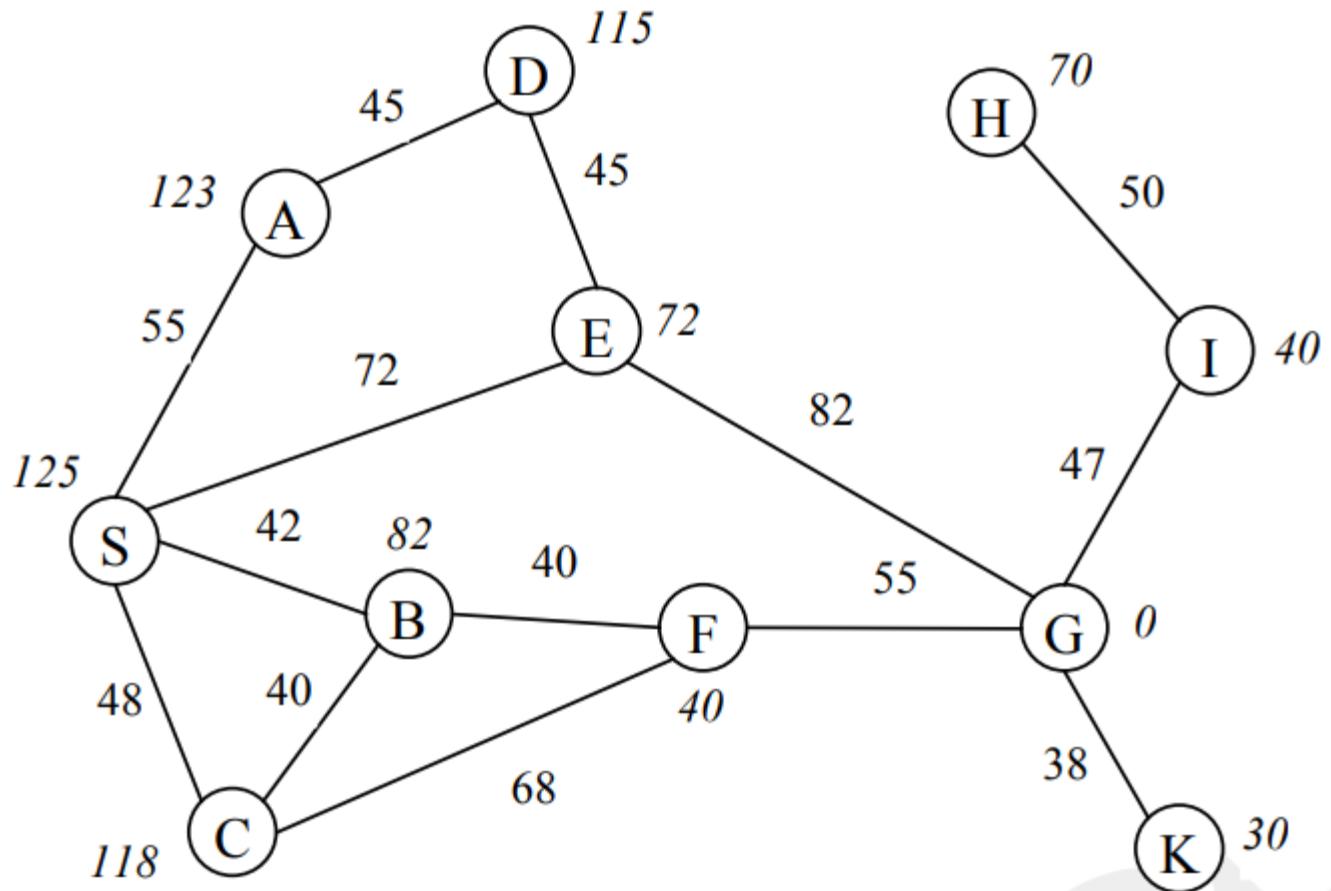
return: không tìm được Solution

Greedy Search

- Hàm ước lượng chi phí (heuristic) $h(N)$:
 - Được xây dựng dựa trên thông tin có được từ Search Problem
 - Phải thỏa mãn hai điều kiện:
 - $h(N)$ không âm
 - $h(N)=0$ nếu N là node đích (Goal state)
- Như vậy: Greedy Search sử dụng hàm heuristic $h(N)$ để ước lượng khoảng cách (chi phí) từ các nodes tới node đích và Solution sẽ ưu tiên node có giá trị hàm $h(N)$ nhỏ nhất.

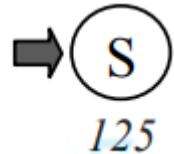
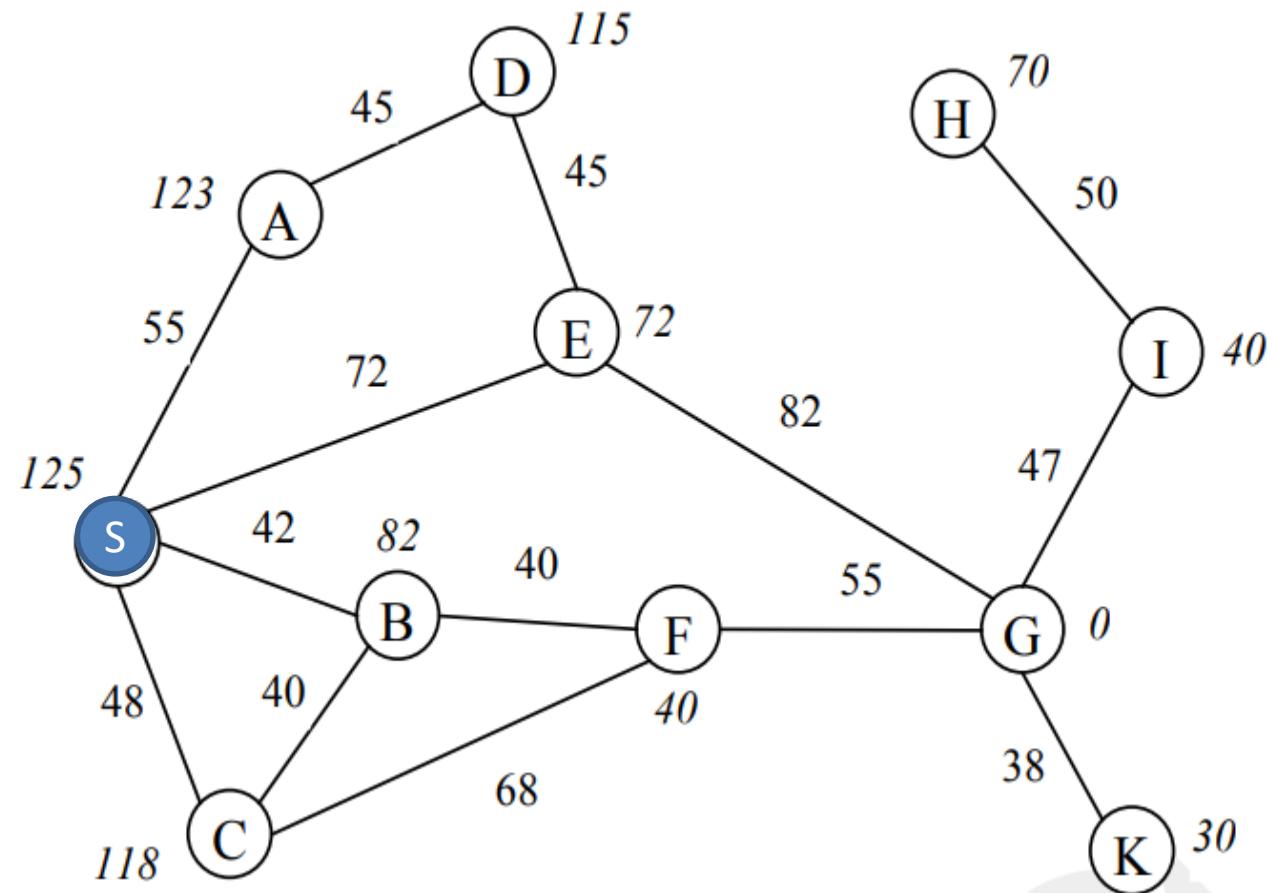
Greedy Search

- Mở rộng nút có vẻ gần nhất.....



Greedy Search

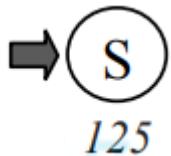
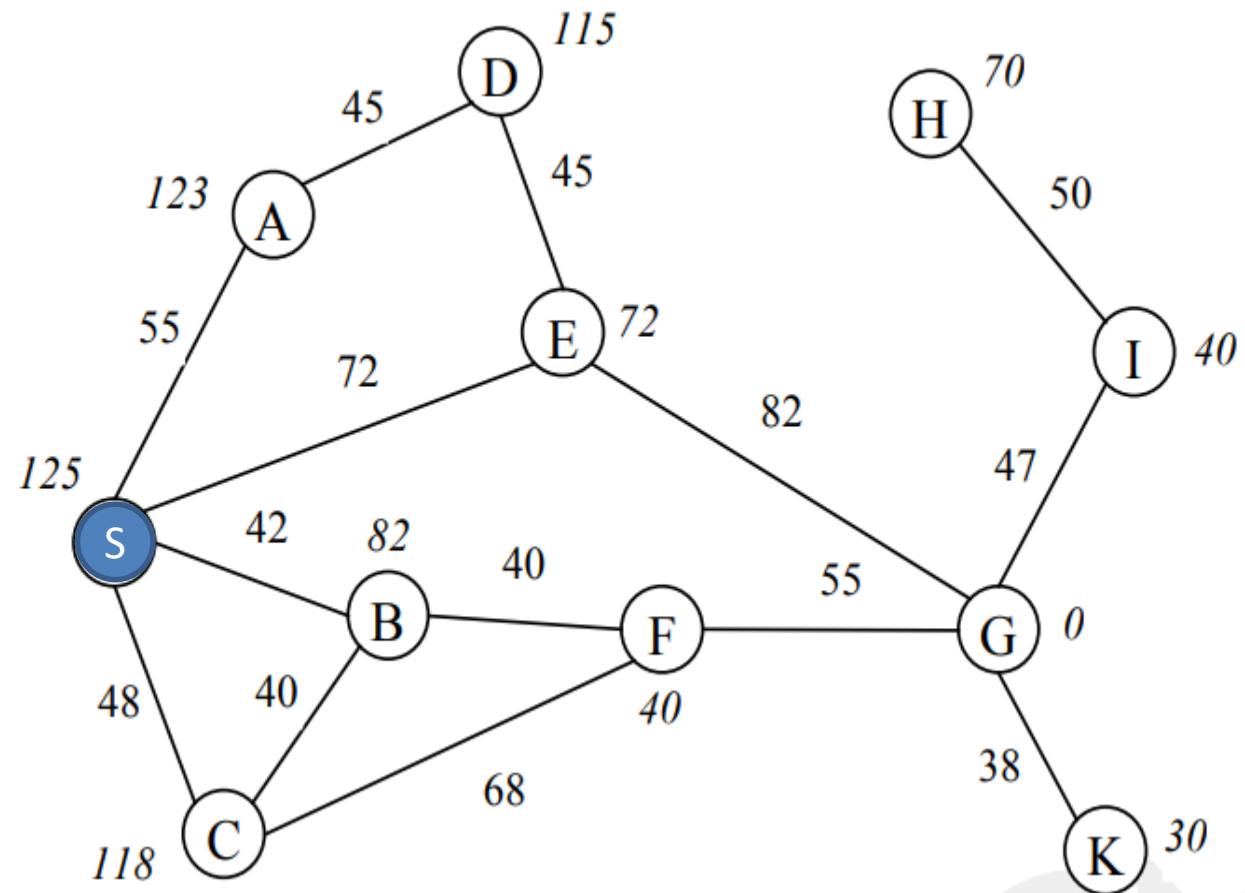
- Mở rộng nút có vẻ gần nhất.....



Nút được xét	Tập biên O priority queue
	(125, S)

Greedy Search

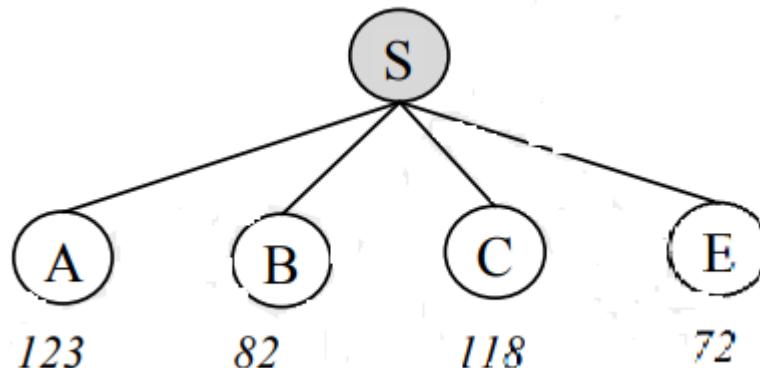
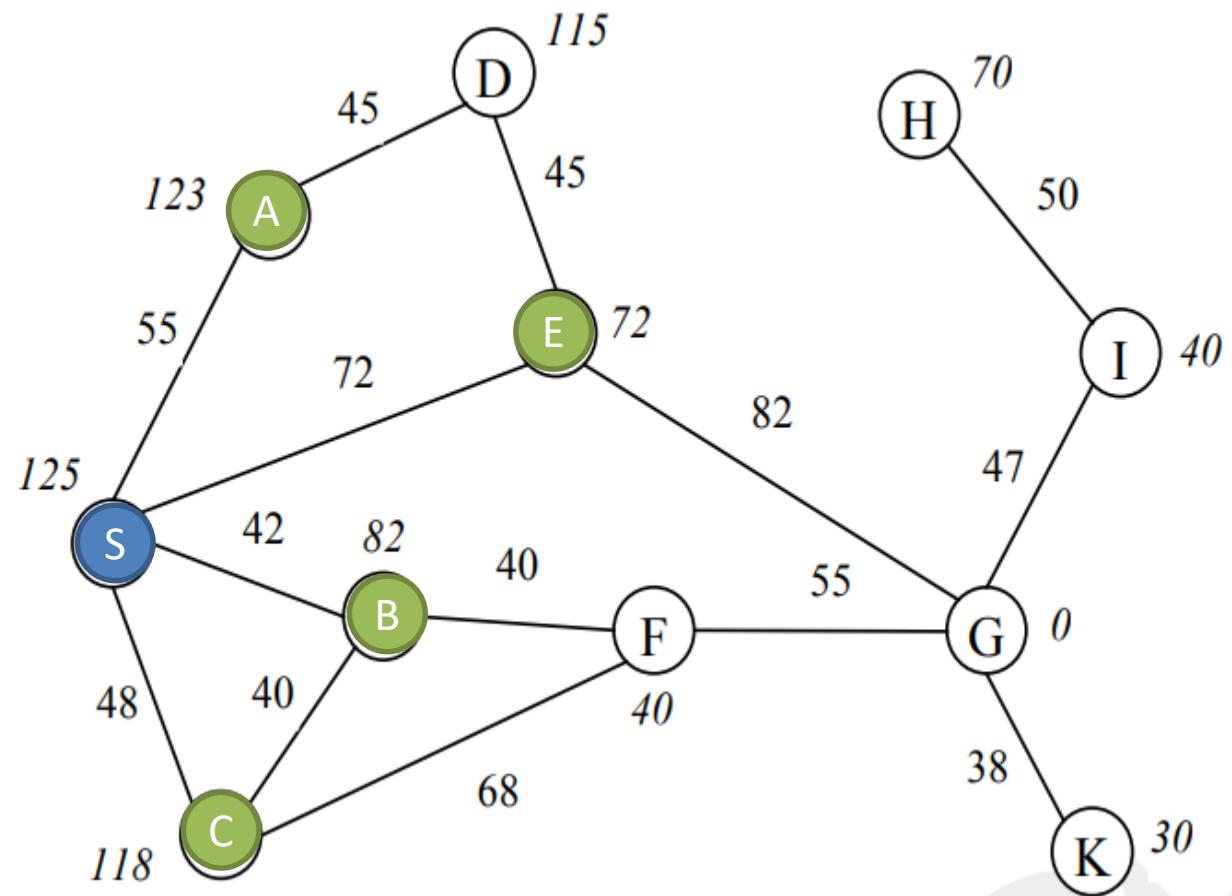
- Mở rộng nút có vẻ gần nhất.....



Nút được xét	Tập biên O priority queue
	(125, S)
(125, S)	

Greedy Search

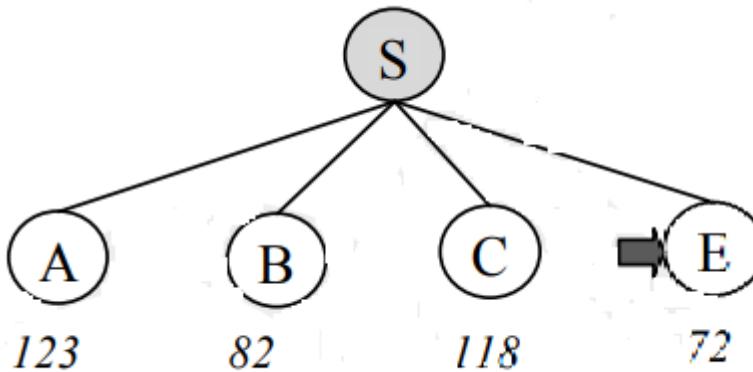
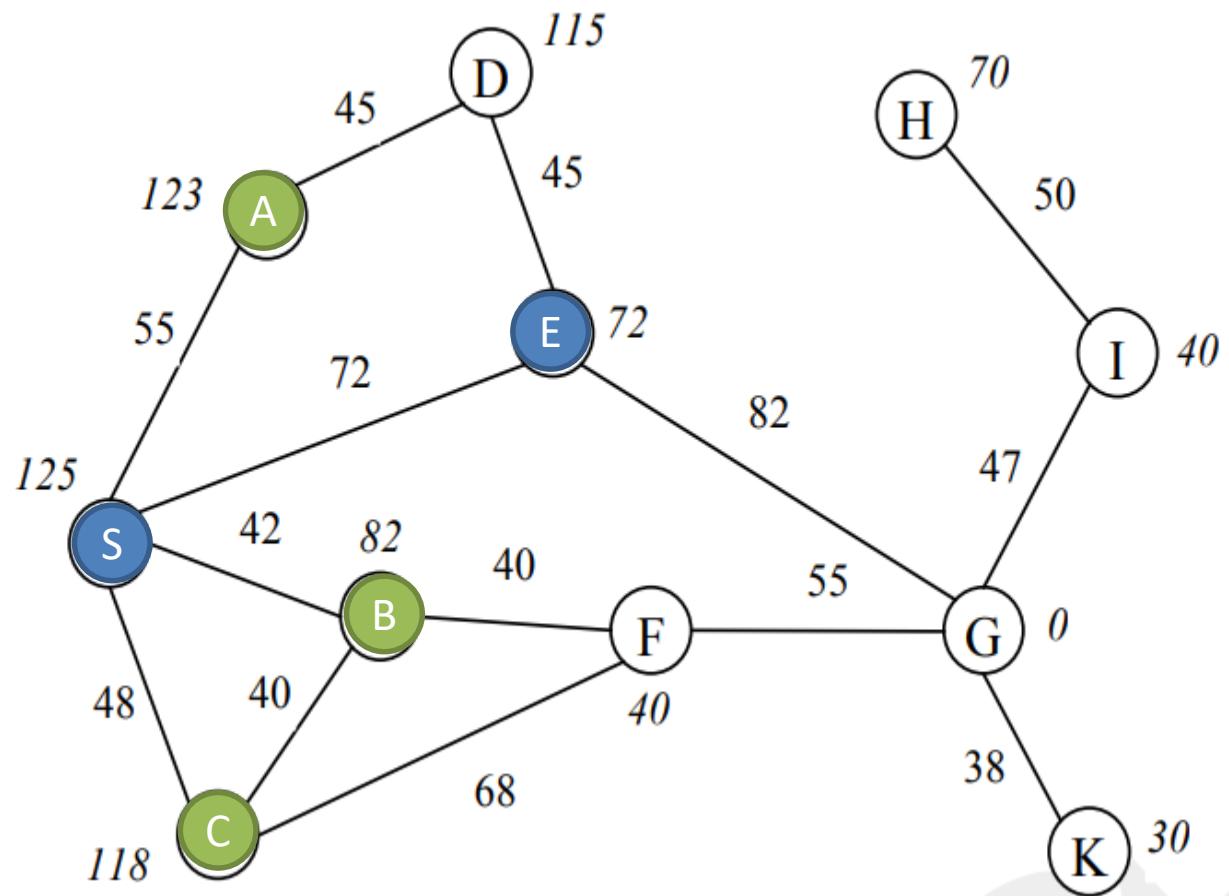
- Mở rộng nút có vế gần nhất.....



Nút được xét	Tập biên O priority queue
	(125, S)
(125, S)	<u>(123, AS)</u> <u>(82, BS)</u> <u>(118 CS)</u> <u>(72, ES)</u>

Greedy Search

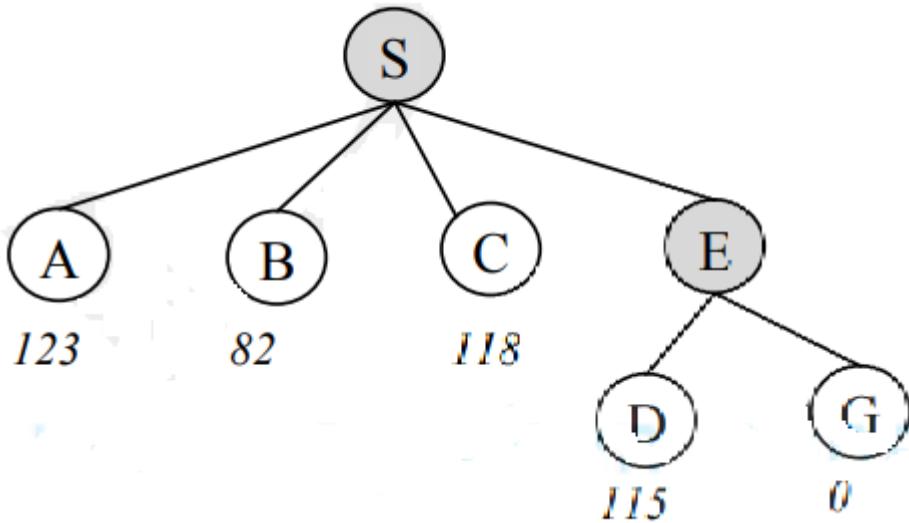
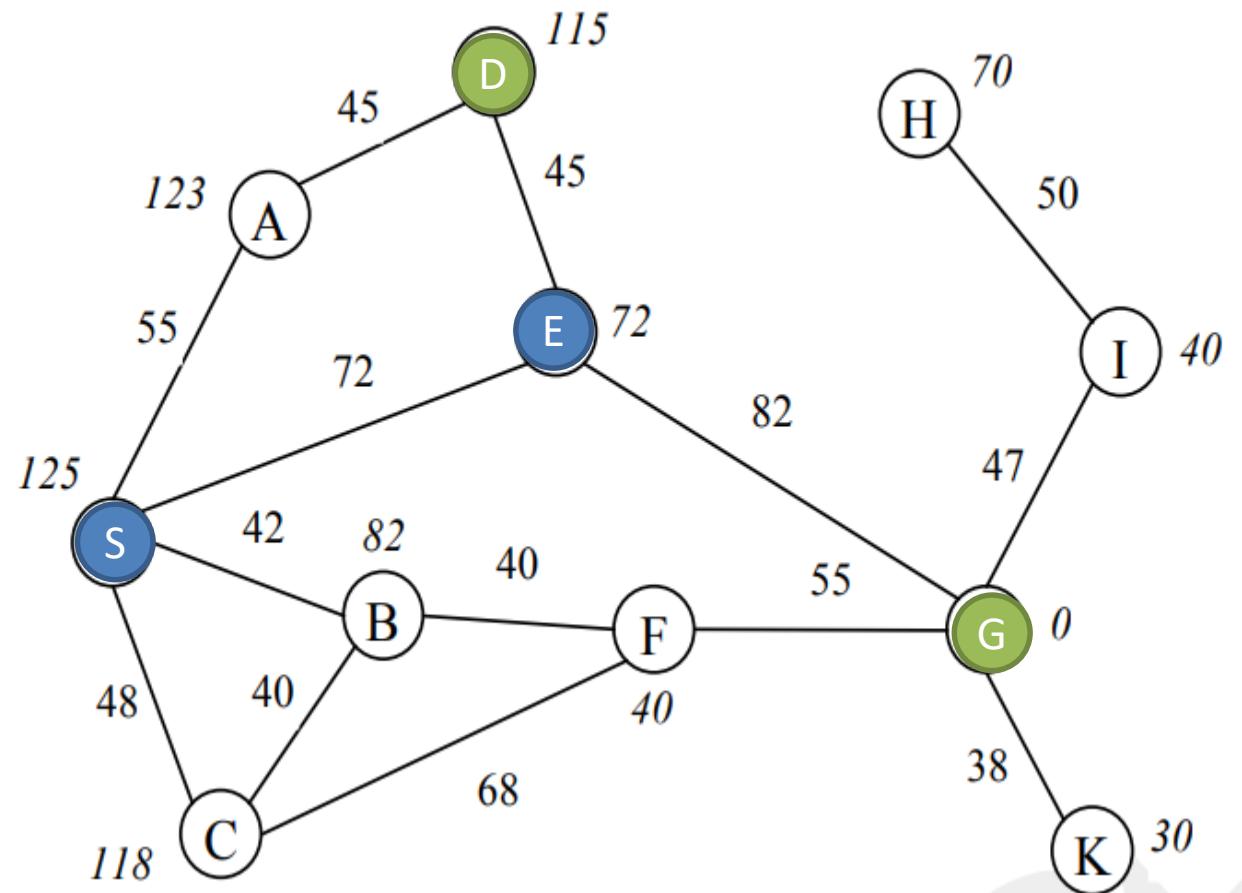
- Mở rộng nút có vẻ gần nhất.....



Nút được xét	Tập biên O priority queue
	(125, S)
(125, S)	(123, AS) (82, BS) (118 CS) (72, ES)
(72, ES)	

Greedy Search

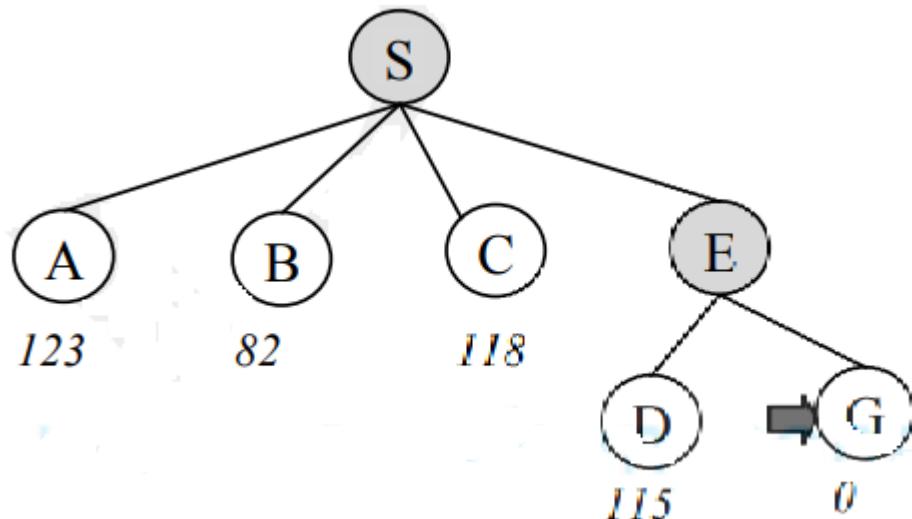
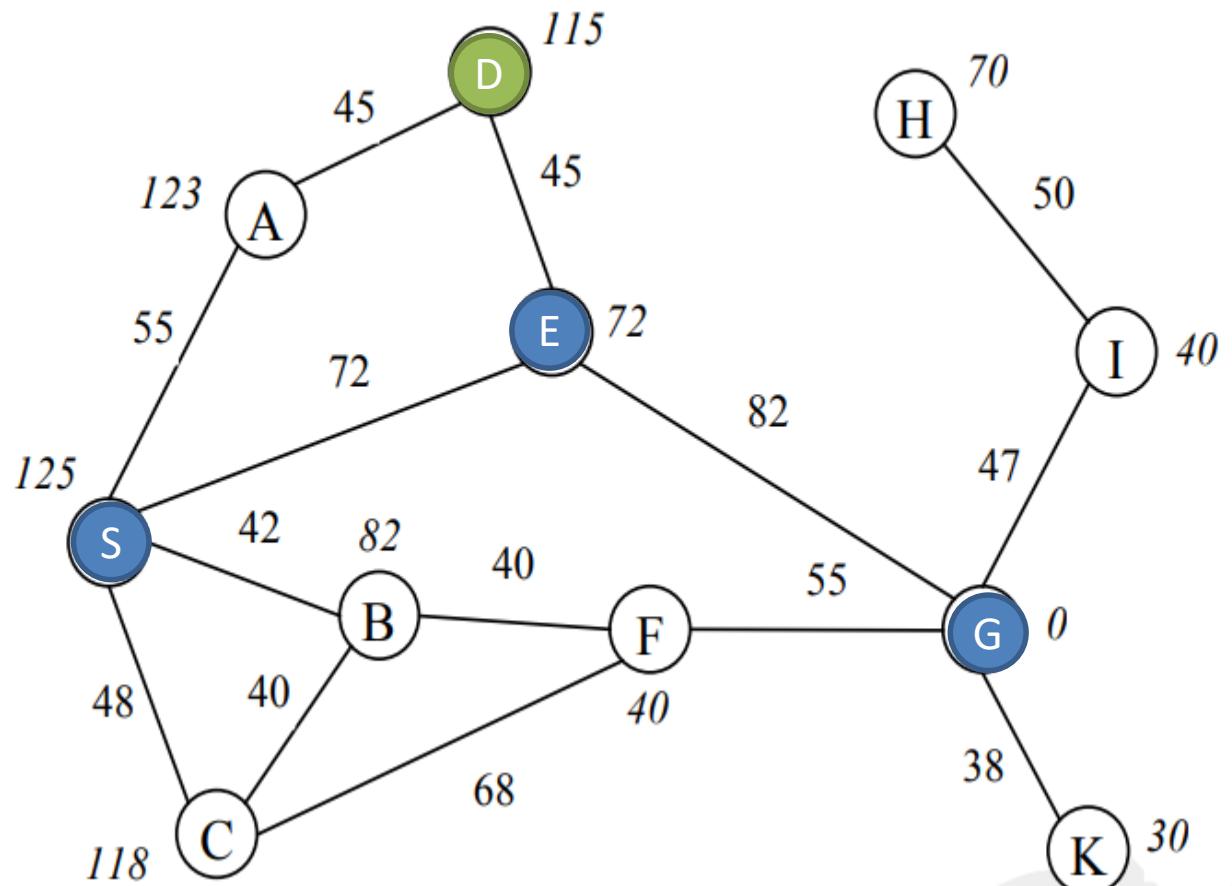
- Mở rộng nút có vẻ gần nhất.....



Nút được xét	Tập biên O priority queue
	(125, S)
(125, S)	(123, AS) (82, BS) (118 CS) (72, ES)
(72, ES)	(123, AS) (82, BS) (118 CS) <u>(115, DES)</u> <u>(0, GES)</u>

Greedy Search

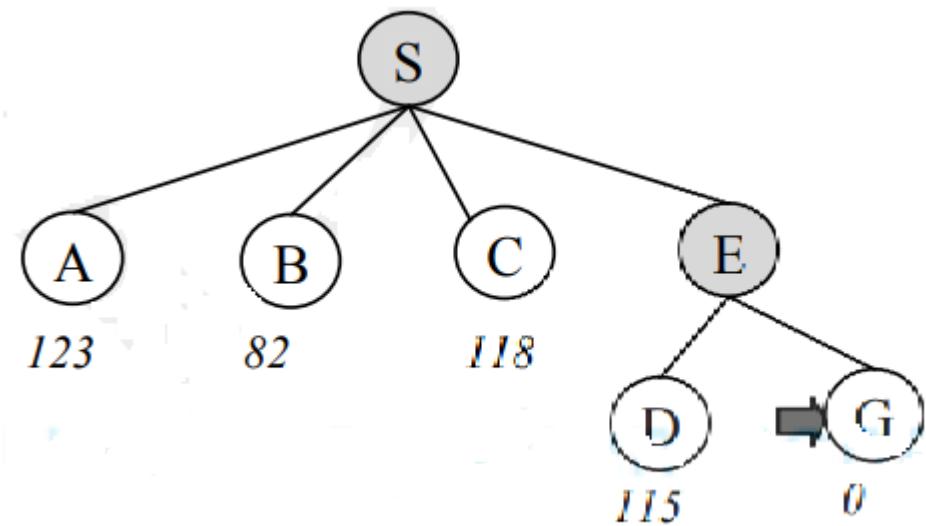
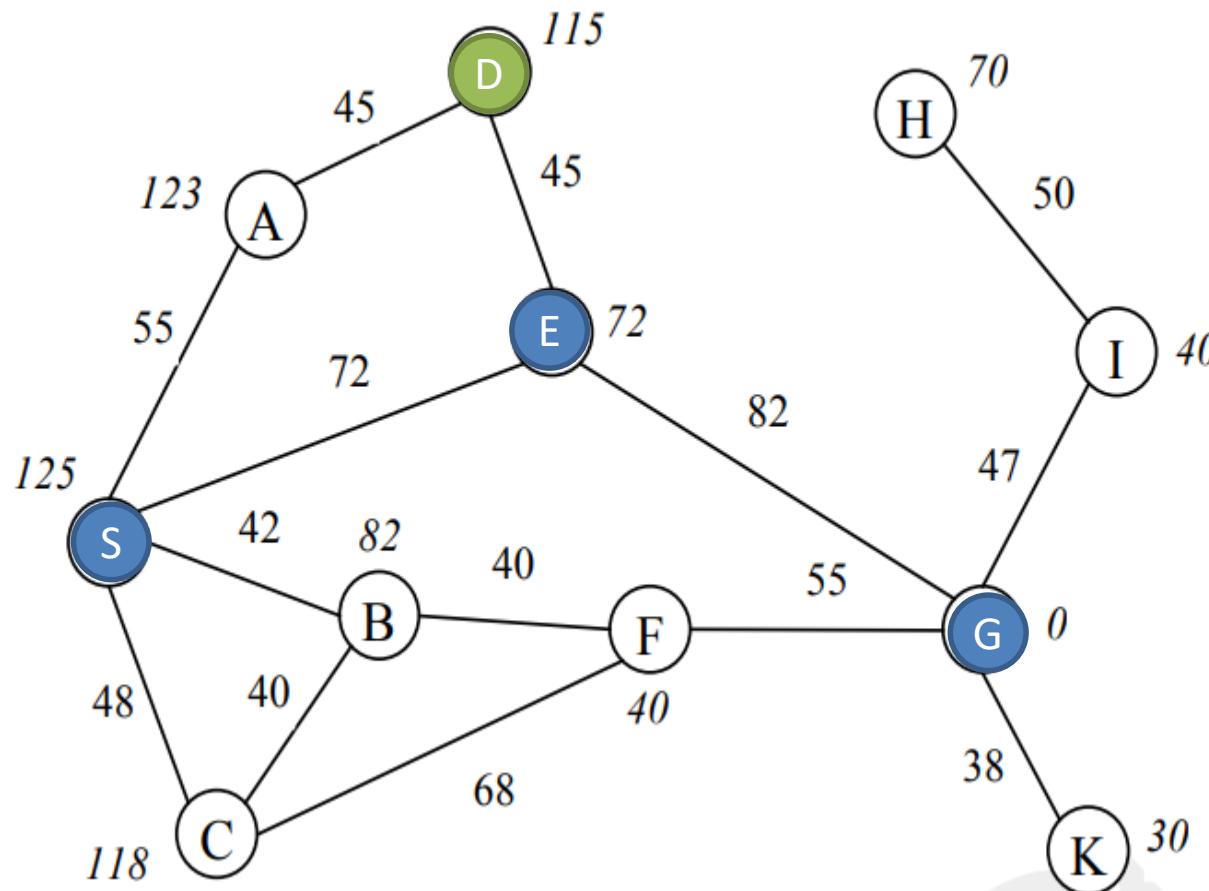
- Mở rộng nút có vẻ gần nhất.....



Nút được xét	Tập biên O priority queue
(125, S)	
(125, S)	(123, AS) (82, BS) (118 CS) (72, ES)
(72, ES)	(123, AS) (82, BS) (118 CS) (115, DES) (0, GES)
(0, GES)	Đích

Greedy Search

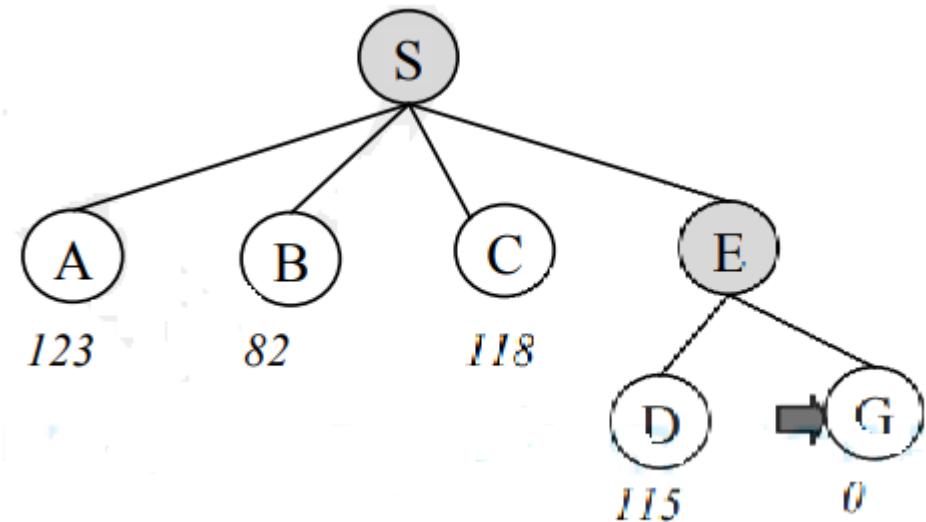
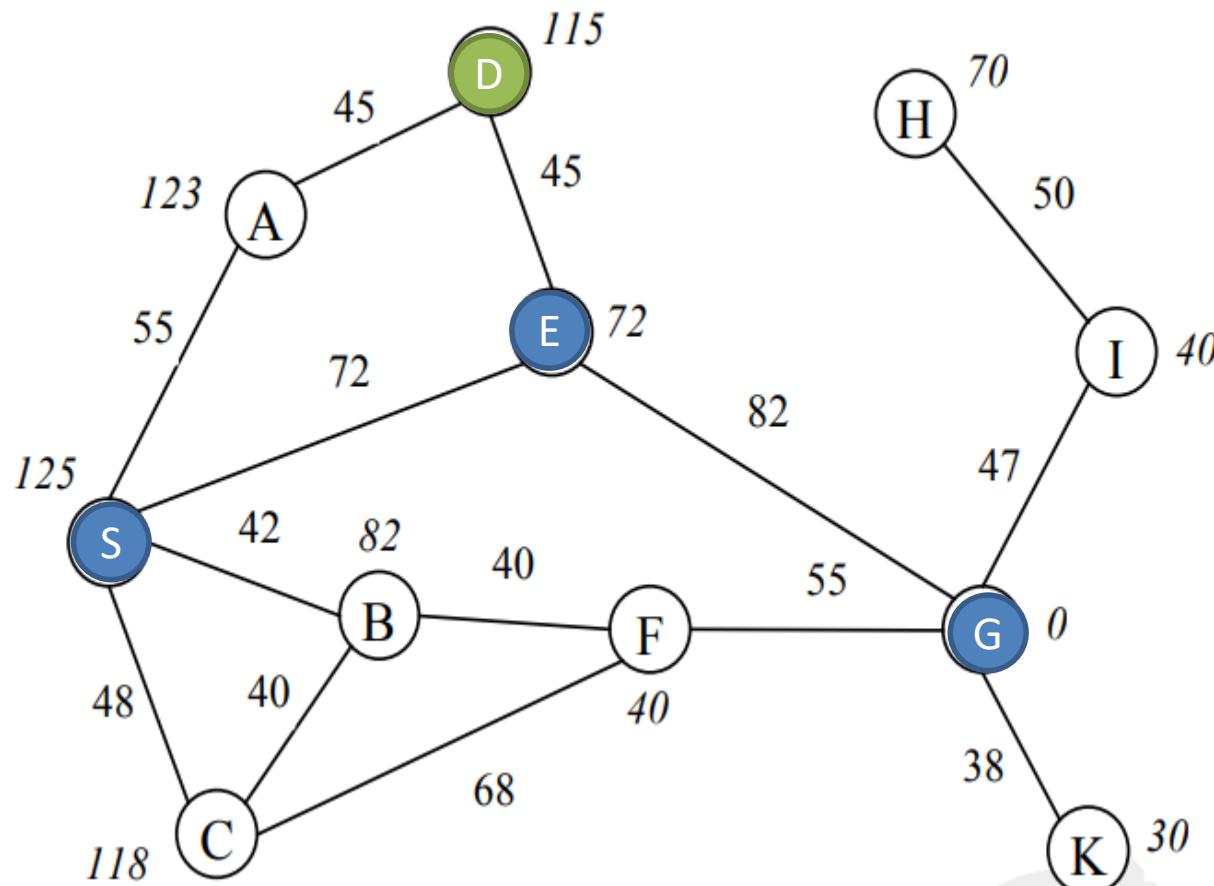
- Mở rộng nút có vế gần nhất.....



Đường đi tìm được là $S \rightarrow E \rightarrow G$.

Greedy Search

- Mở rộng nút có vẻ gần nhất.....

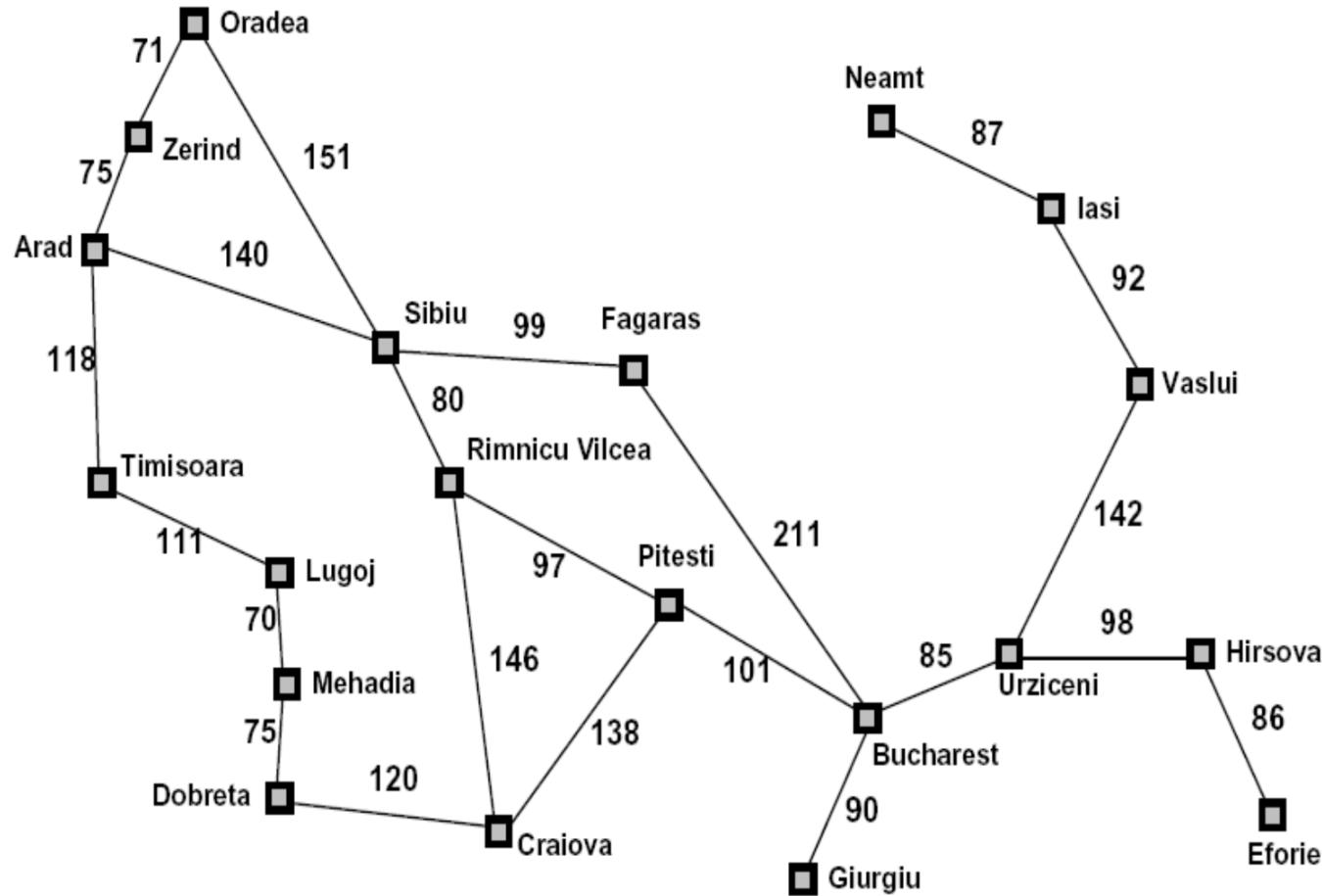


Đường đi tìm được là S→E→G.

Chúng ta đã tìm được con đường tốt nhất chưa???

Greedy Search

- Mở rộng nút có vé gần nhất.....

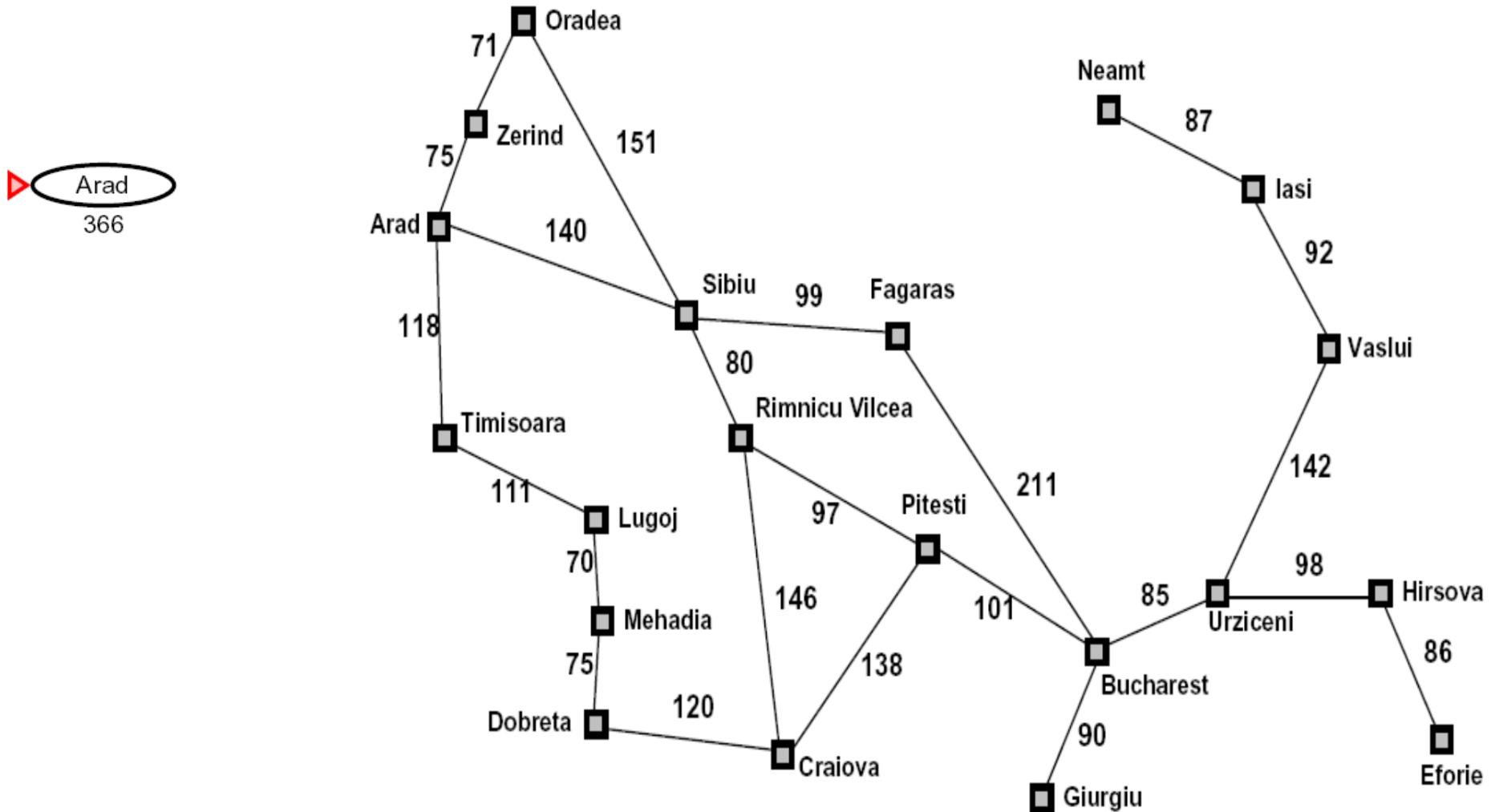


Straight-line distance
to Bucharest

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

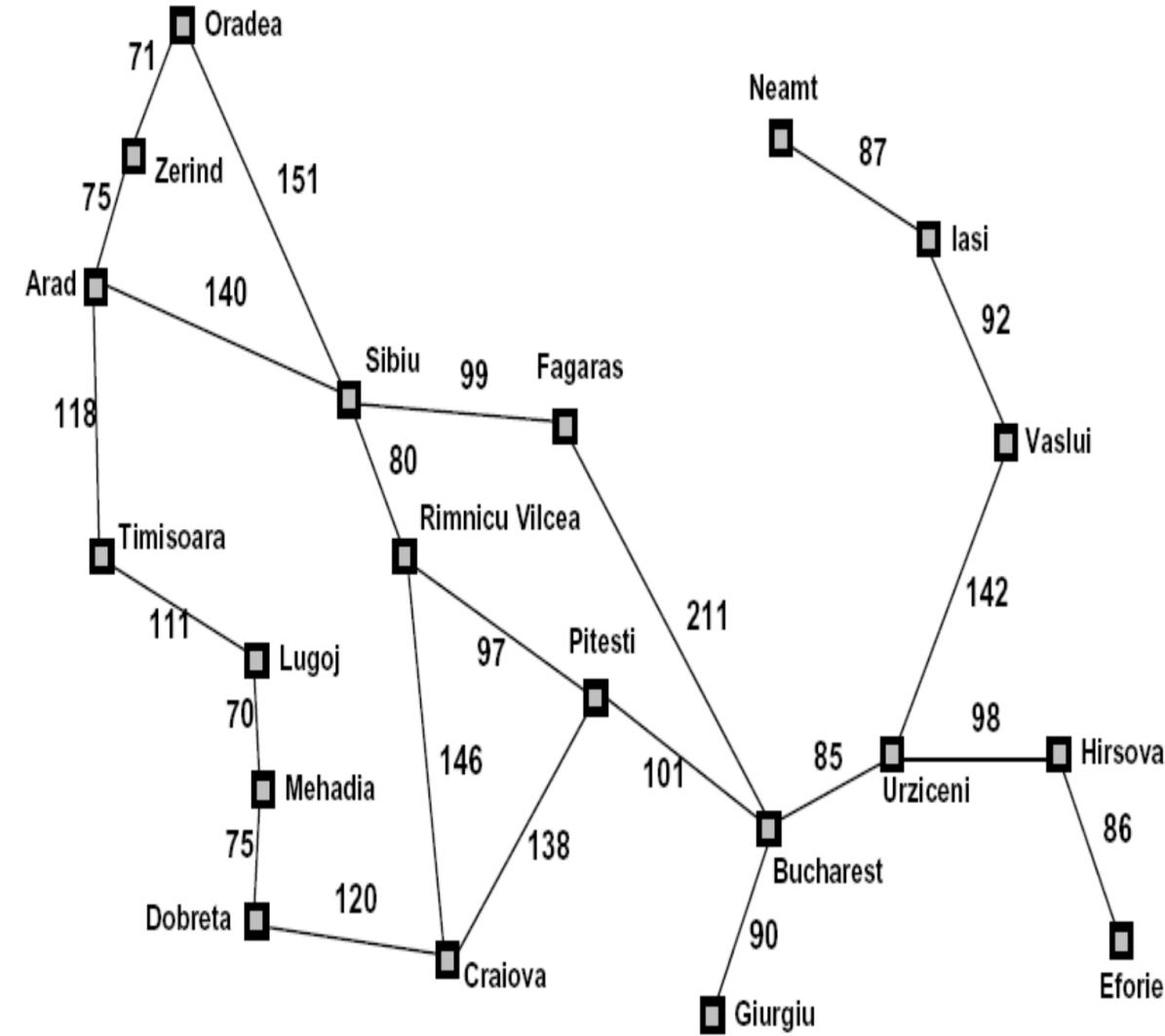
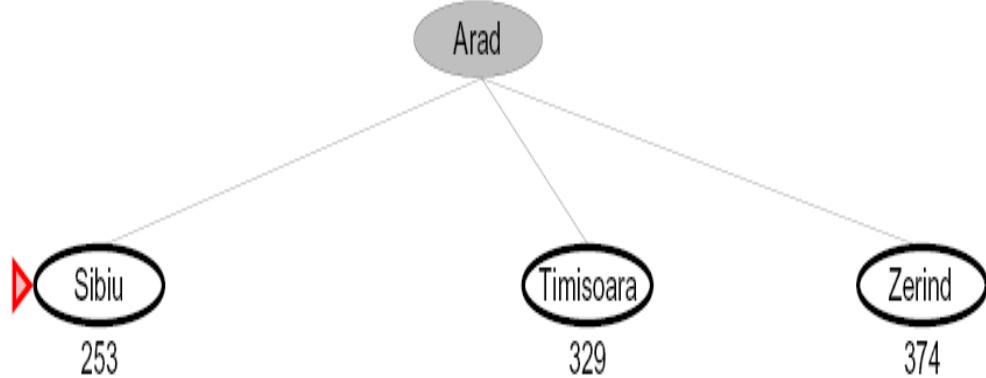
Greedy Search

- Mở rộng nút có vẻ gần nhất.....



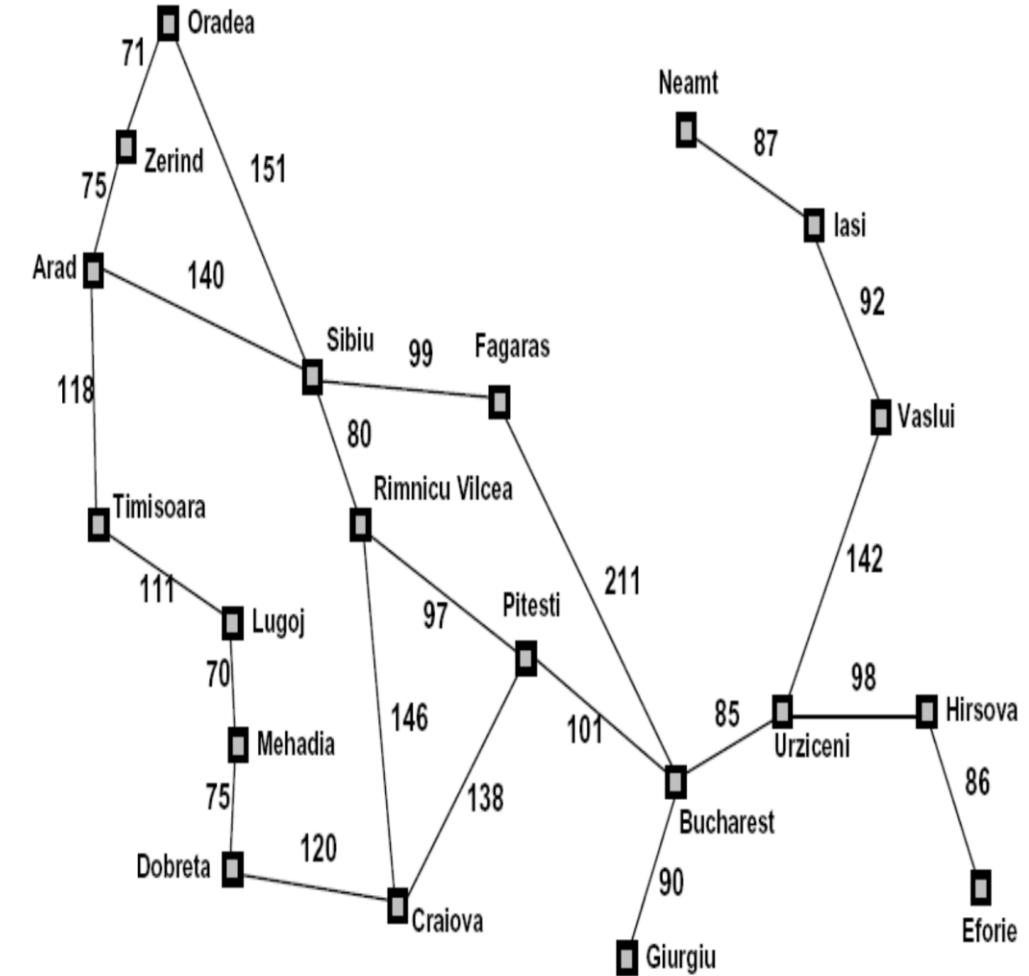
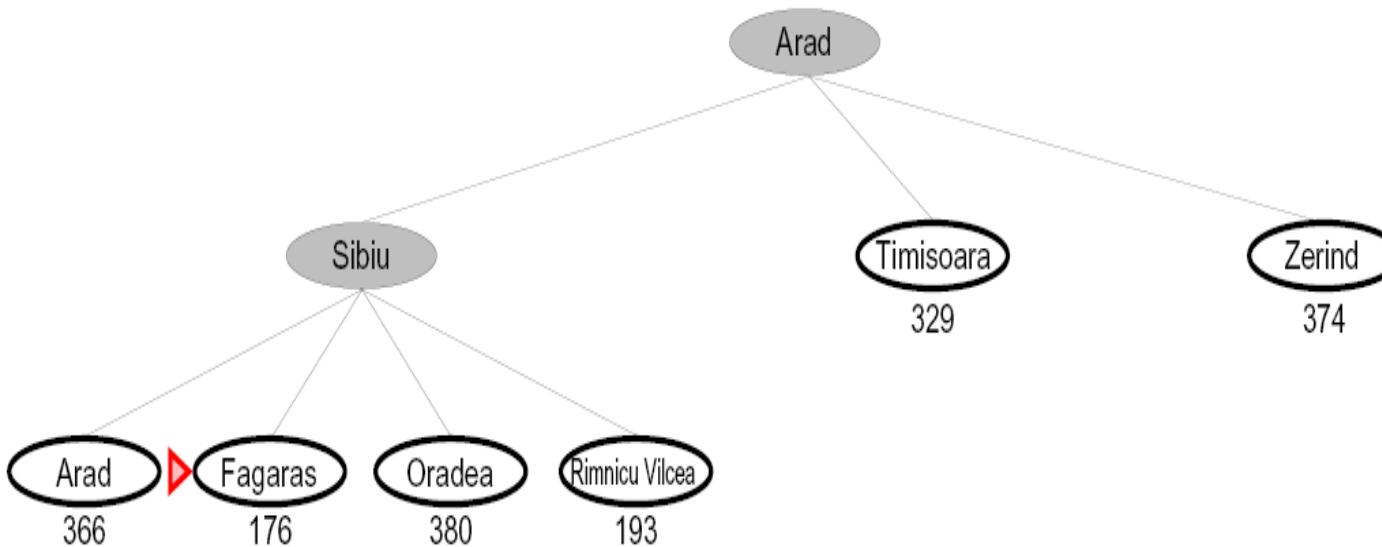
Greedy Search

- Mở rộng nút có vẻ gần nhất.....



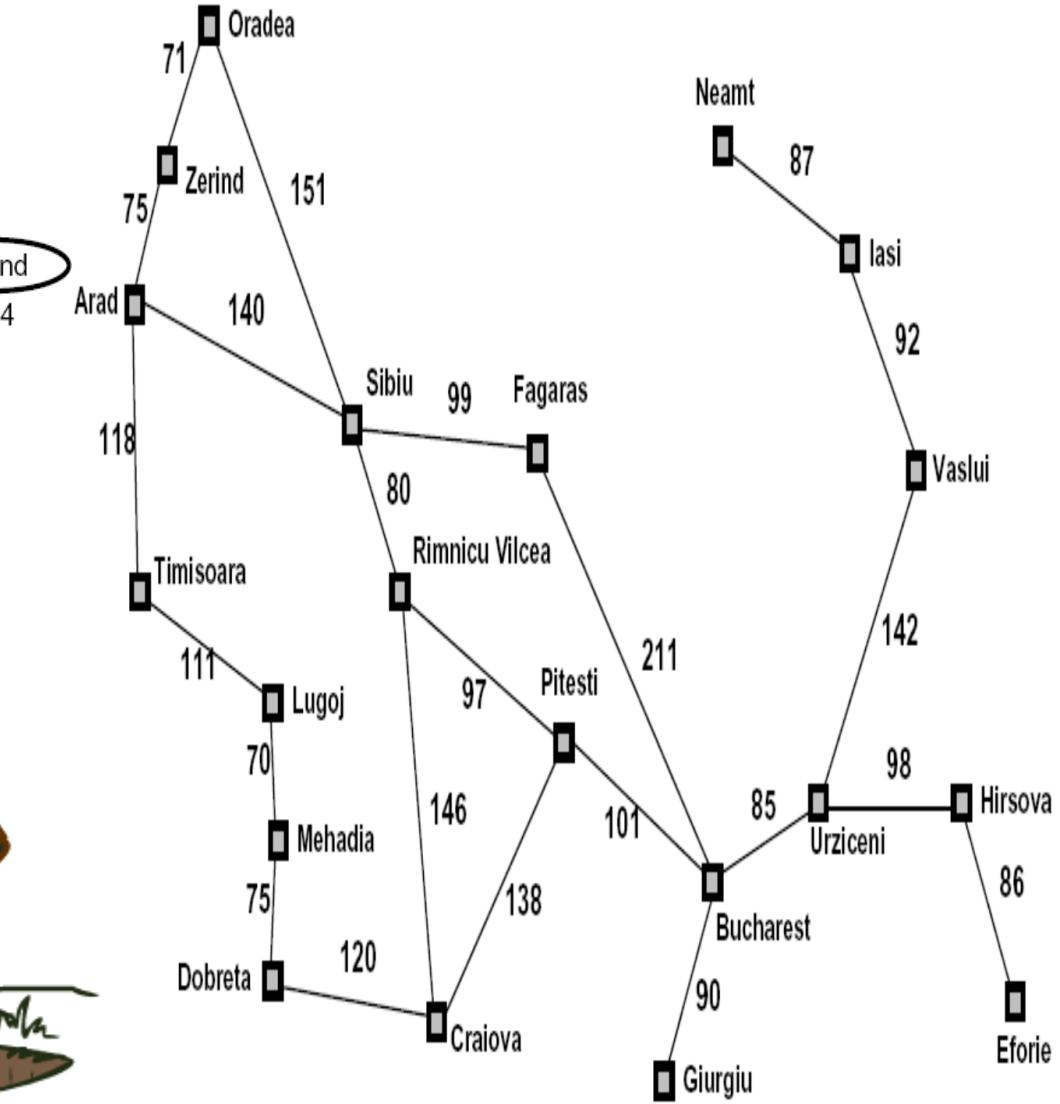
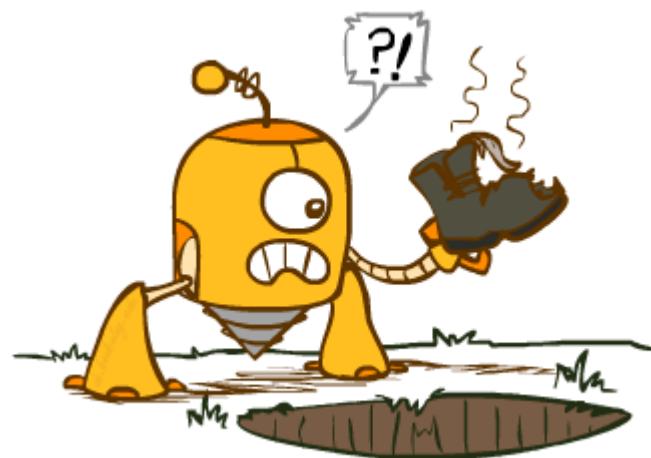
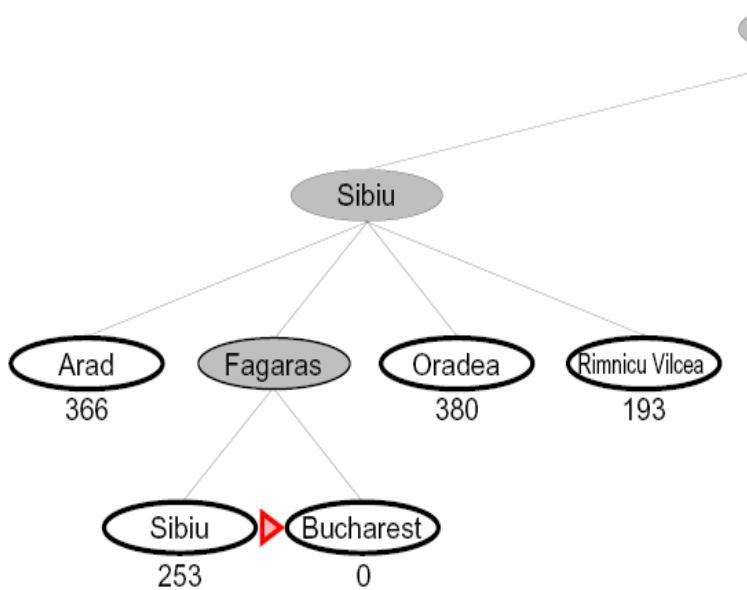
Greedy Search

- Mở rộng nút có vẻ gần nhất.....



Greedy Search

- Mở rộng nút có vẻ gần nhất.....



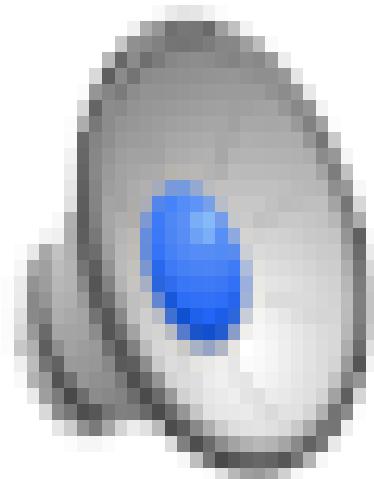
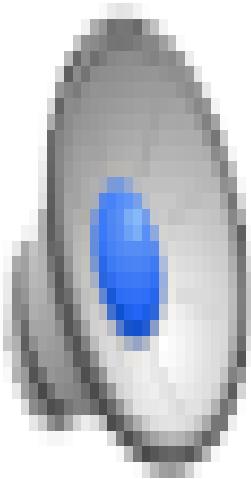
- Chúng ta đã tìm được con đường tốt nhất chưa???

Greedy Search

- Đặc điểm:
 - Không có tính đầy đủ do có khả năng tạo thành vòng lặp vô hạn ở một số nút.
 - Độ phức tạp về thời gian: trong trường hợp xấu nhất là $O(b^m)$.
 - b là số nhánh của cây/đồ thị
 - m là độ sâu của cây/đồ thị khi tìm ra Solution
 - Độ phức tạp về không gian: trong trường hợp xấu nhất là $O(b^m)$.
 - Thuật toán không tối ưu.

Greedy Search

- Video of Demo Greedy



A* Search



A* Search



UCS



Greedy



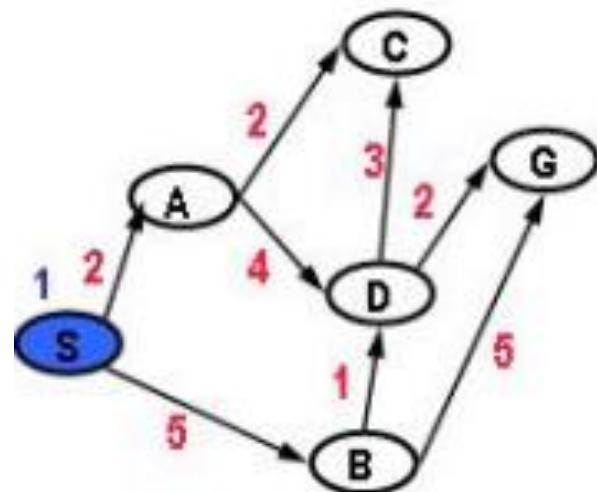
A*

A* Search

- Greedy Search không đảm bảo tìm ra đường đi ngắn nhất là do thuật toán chỉ quan tâm tới chi phí ước lượng từ một nút tới đích mà không quan tâm tới chi phí đã đi từ nút xuất phát tới nút đó.
- Ý tưởng A* Search: Sử dụng hàm đánh giá $f(N) = g(N) + h(N)$
 - $g(N)$ = chi phí từ node gốc (Start state) cho đến node 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:



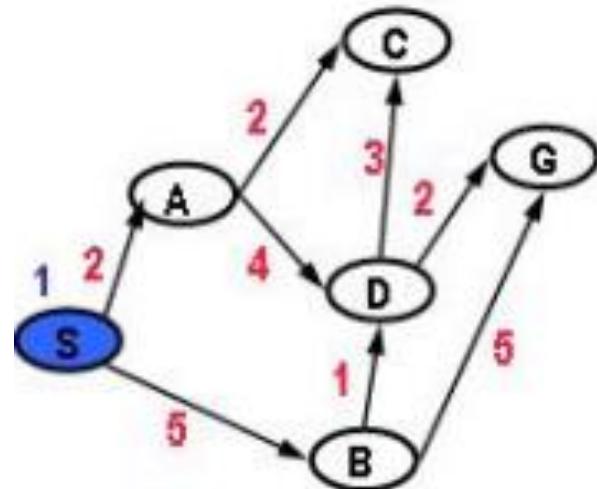
Heuristic Values

$$\begin{array}{lll} A=2 & C=1 & S=0 \\ B=3 & D=1 & G=0 \end{array}$$

Node hiện tại	Tập biên O
	<u>(0, S)</u>

A* Search

Ví dụ:



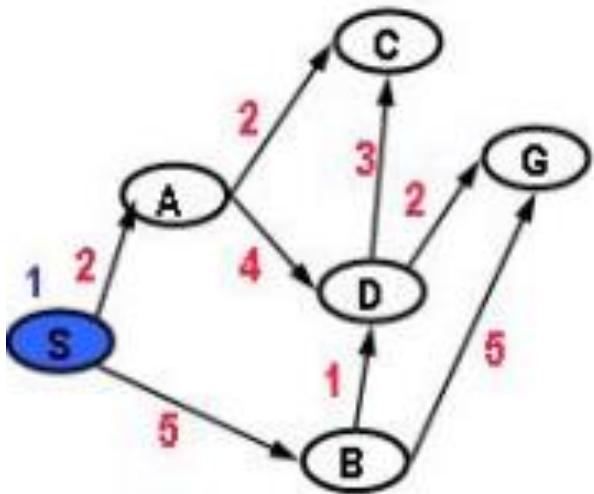
Heuristic Values

$$\begin{array}{lll} A=2 & C=1 & S=0 \\ B=3 & D=1 & G=0 \end{array}$$

Node được xét	Tập biên O
	(0, S)
	(0, S)

A* Search

Ví dụ:



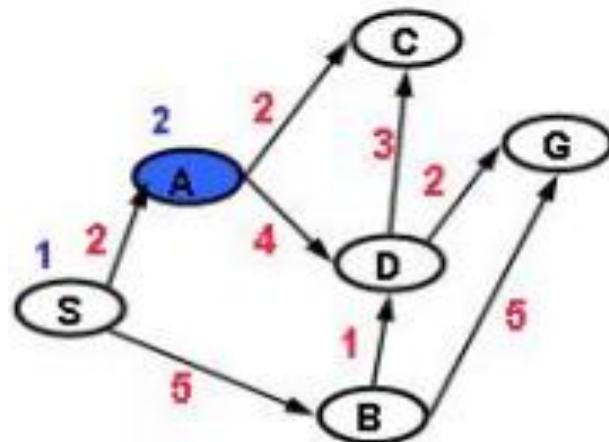
Heuristic Values

A=2 C=1 S=0
B=3 D=1 G=0

Node được xét	Tập biên O
	(0, S)
(0, S)	<u>(4,AS)</u> <u>(8,BS)</u>

A* Search

Ví dụ:



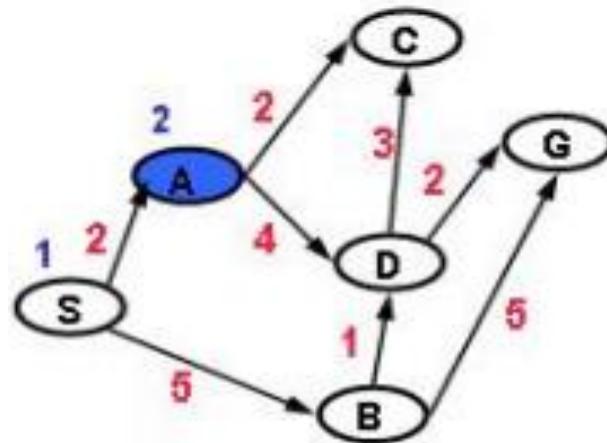
Heuristic Values

$$\begin{array}{lll} A=2 & C=1 & S=0 \\ B=3 & D=1 & G=0 \end{array}$$

Node được xét	Tập biên O
	(0, S)
(0, S)	(4,AS) (8,BS)
(4,AS)	

A* Search

Ví dụ:



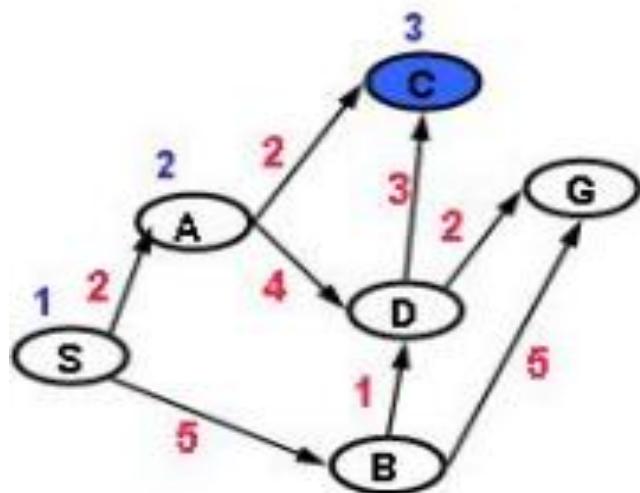
Heuristic Values

$$\begin{array}{lll} A=2 & C=1 & S=0 \\ B=3 & D=1 & G=0 \end{array}$$

Node được xét	Tập biên O
	(0, S)
(0, S)	(4,AS) (8,BS)
(4,AS)	<u>(5, CAS) (7,DAS) (8,BS)</u>

A* Search

Ví dụ:



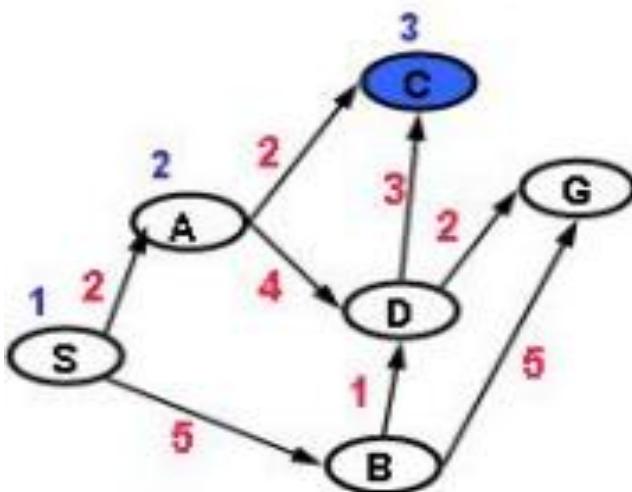
Heuristic Values

$$\begin{array}{lll} A=2 & C=1 & S=0 \\ B=3 & D=1 & G=0 \end{array}$$

Node được xét	Tập biên O
	(0, S)
(0, S)	(4,AS) (8,BS)
(4,AS)	(5, CAS) (7,DAS) (8,BS)
(5, CAS)	

A* Search

Ví dụ:



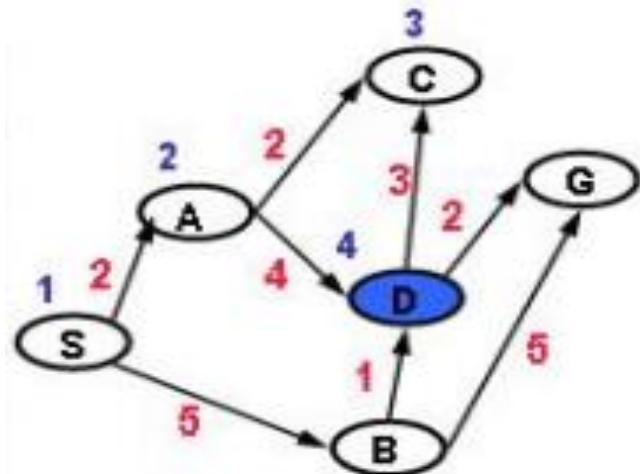
Heuristic Values

A=2	C=1	S=0
B=3	D=1	G=0

Node được xét	Tập biên O
	(0, S)
(0, S)	(4,AS) (8,BS)
(4,AS)	(5, CAS) (7,DAS) (8,BS)
(5, CAS)	(7,DAS) (8, BS)

A* Search

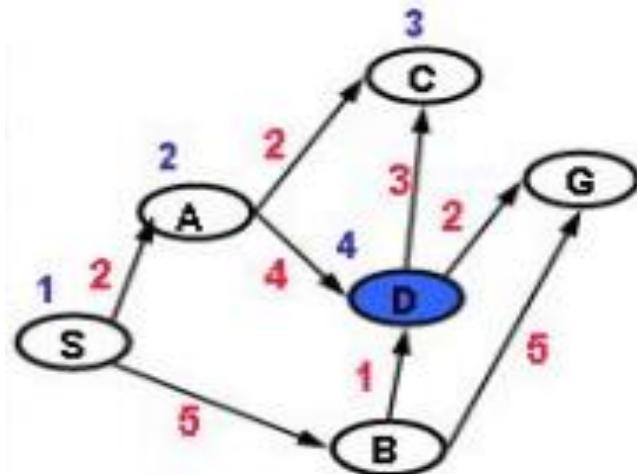
Ví dụ:



Node được xét	Tập biên O
(0, S)	(4,AS) (8,BS)
(4,AS)	(5, CAS) (7,DAS) (8,BS)
(5, CAS)	(7,DAS) (8, BS)
(7,DAS)	

A* Search

Ví dụ:



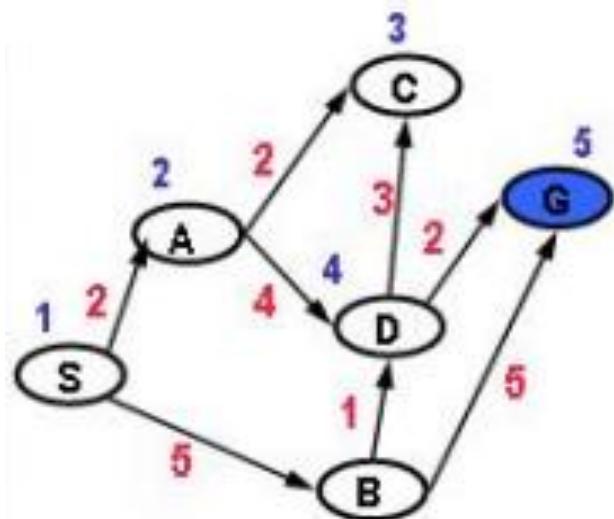
Heuristic Values

$$\begin{array}{lll} A=2 & C=1 & S=0 \\ B=3 & D=1 & G=0 \end{array}$$

Node được xét	Tập biên O
	(0, S)
(0, S)	(4,AS) (8,BS)
(4,AS)	(5, CAS) (7,DAS) (8,BS)
(5, CAS)	(7,DAS) (8, BS)
(7,DAS)	<u>(8,GDAS)</u> (10,CDAS) (8,BS)

A* Search

Ví dụ:



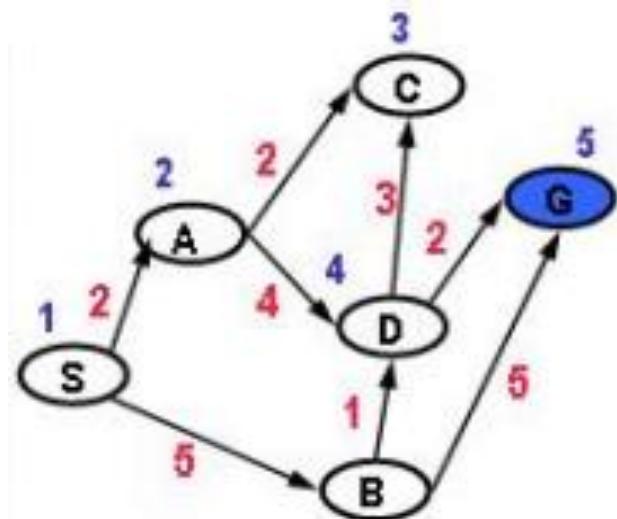
Heuristic Values

A=2 C=1 S=0
B=3 D=1 G=0

Node được xét	Tập biên O
(0, S)	(0, S)
(0, S)	(4,AS) (8,BS)
(4,AS)	(8,BS) (5, CAS) (7,DAS)
(5, CAS)	(7,DAS) (8, BS)
(7,DAS)	(8,GDAS) (10,CDAS) (8,BS)
(8,GDAS)	

A* Search

Ví dụ:



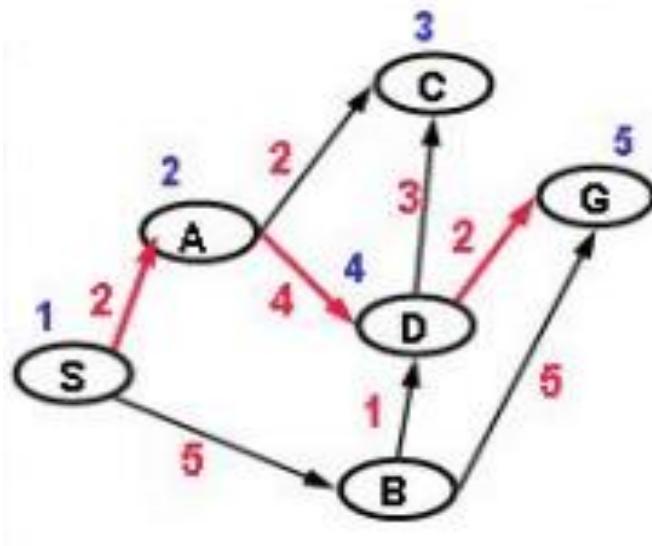
Heuristic Values

$$\begin{array}{lll} A=2 & C=1 & S=0 \\ B=3 & D=1 & G=0 \end{array}$$

Node được xét	Tập biên O
(0, S)	(0, S)
(0, S)	(4,AS) (8,BS)
(4,AS)	(8,BS) (5, CAS) (7,DAS)
(5, CAS)	(7,DAS) (8, BS)
(7,DAS)	(8,GDAS) (10,CDAS) (8,BS)
(8,GDAS)	(10,CDAS) (8,BS)

A* Search

Ví dụ:



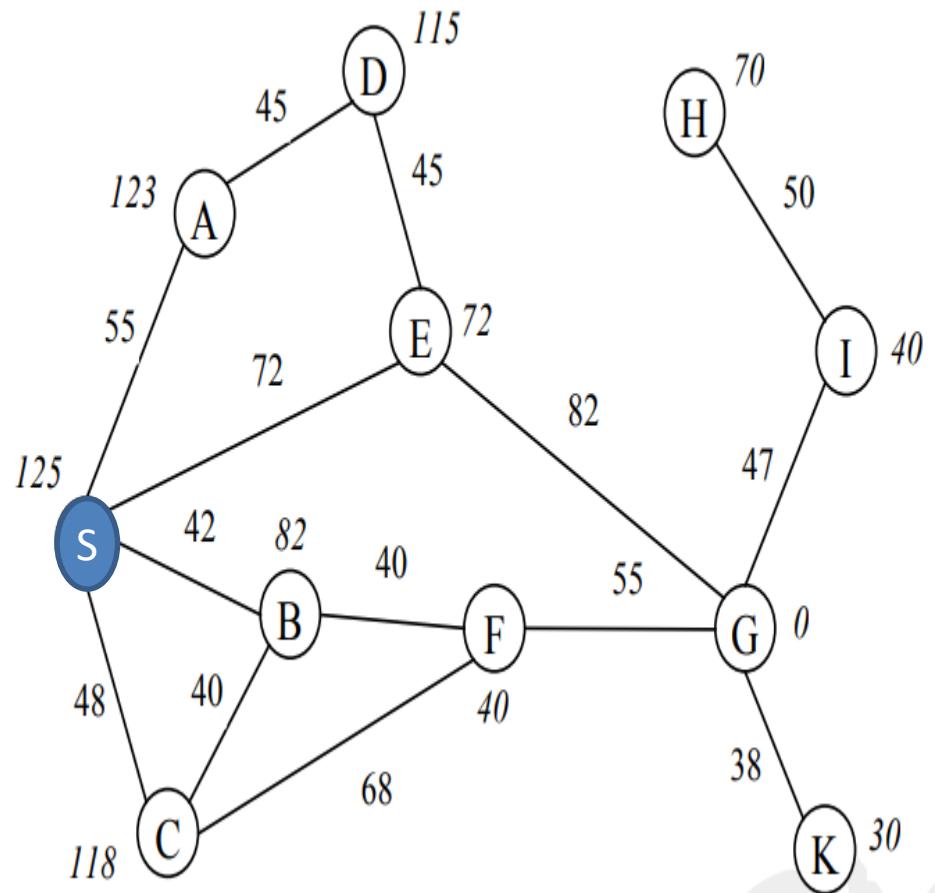
Heuristic Values

$$\begin{array}{lll} A=2 & C=1 & S=0 \\ B=3 & D=1 & G=0 \end{array}$$

Node được xét	Tập biên O
(0, S)	(0, S)
(0, S)	(4,AS) (8,BS)
(4,AS)	(8,BS) (5, CAS) (7,DAS)
(5, CAS)	(7,DAS) (8, BS)
(7,DAS)	(8,GDAS) (10,CDAS) (8,BS)
(8,GDAS)	(10,CDAS) (8,BS)

A* Search

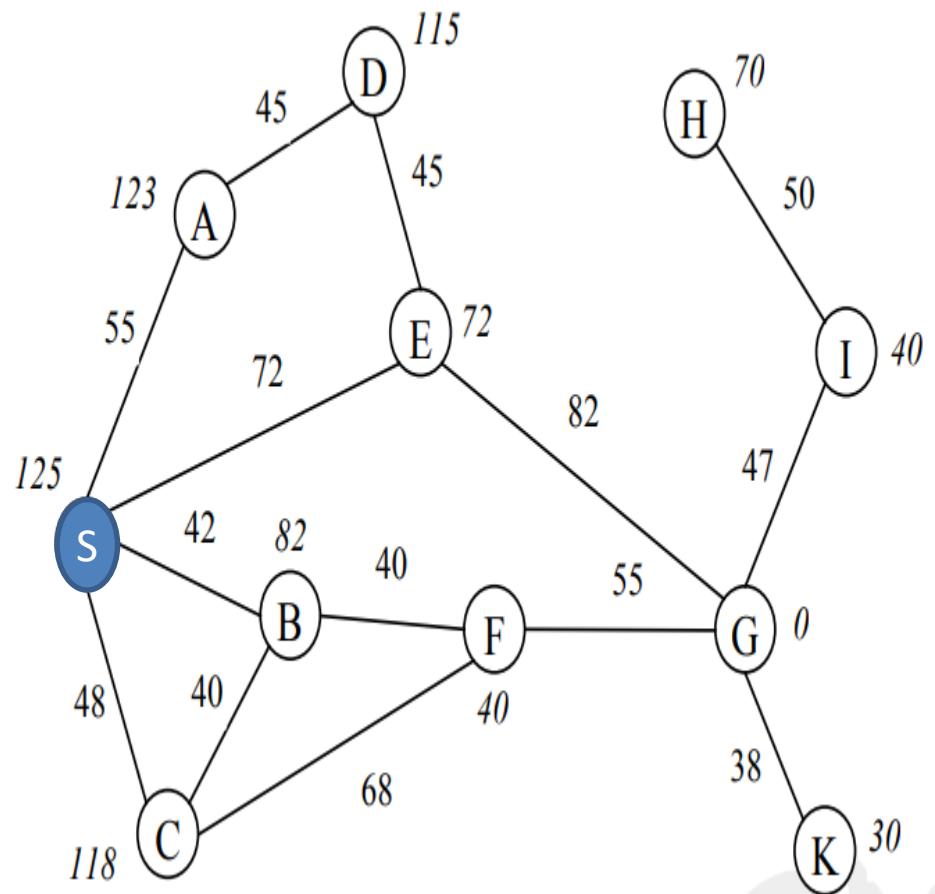
Ví dụ:



Node được xét	Tập biên O
	<u>(125, S)</u>

A* Search

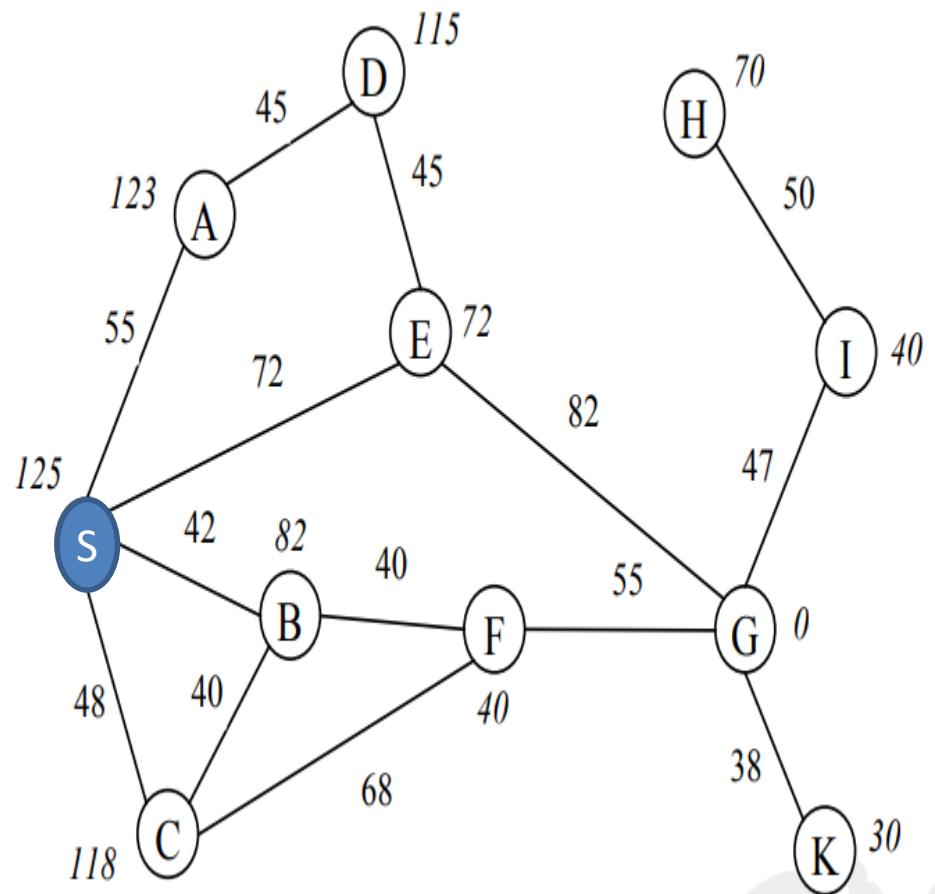
Ví dụ:



Node được xét	Tập biên O
	(125, S)
(125, S)	

A* Search

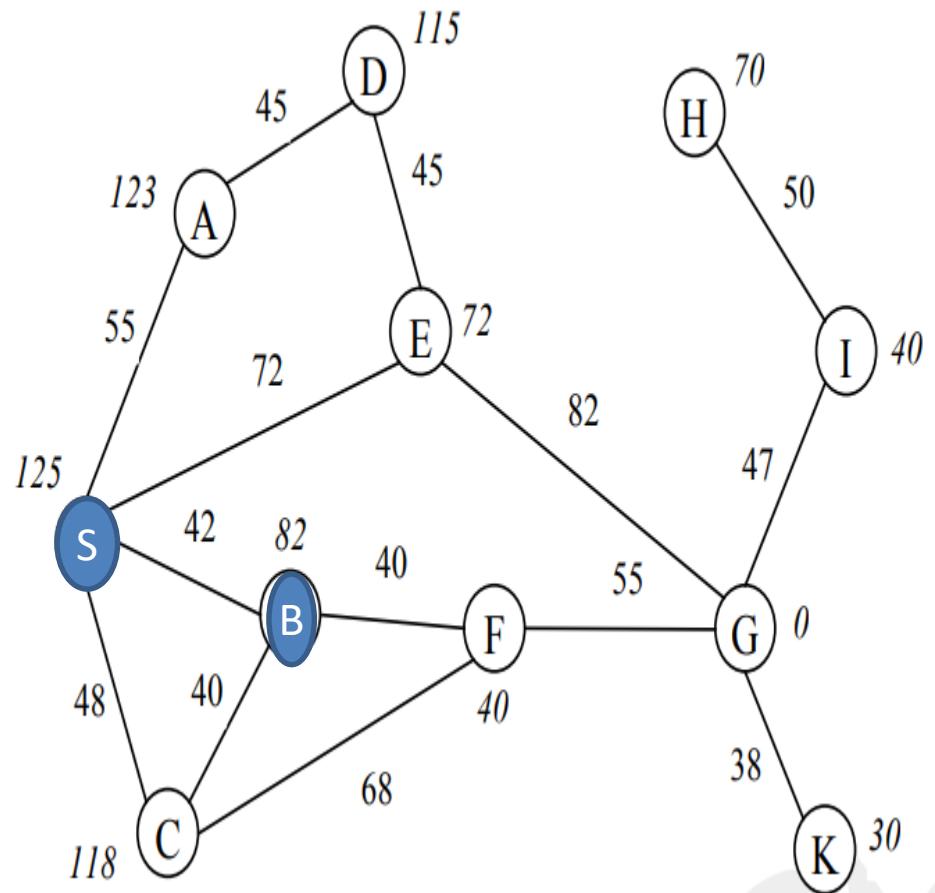
Ví dụ:



Node được xét	Tập biên O
	(125, S)
(125, S)	<u>(178,AS)</u> <u>(124,BS)</u> <u>(166,CS</u> <u>(144,ES)</u>

A* Search

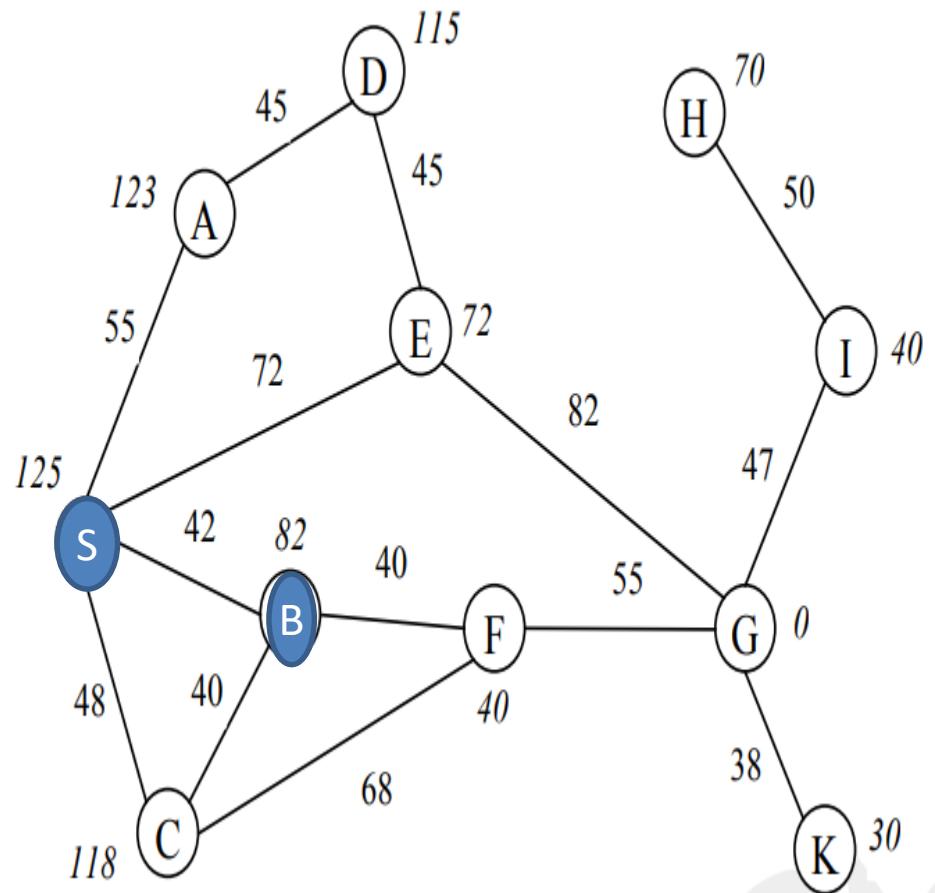
Ví dụ:



Node được xét	Tập biên O
	(125, S)
(125, S)	(178,AS) (124,BS) (166,CS) (144,ES)
(124,BS)	

A* Search

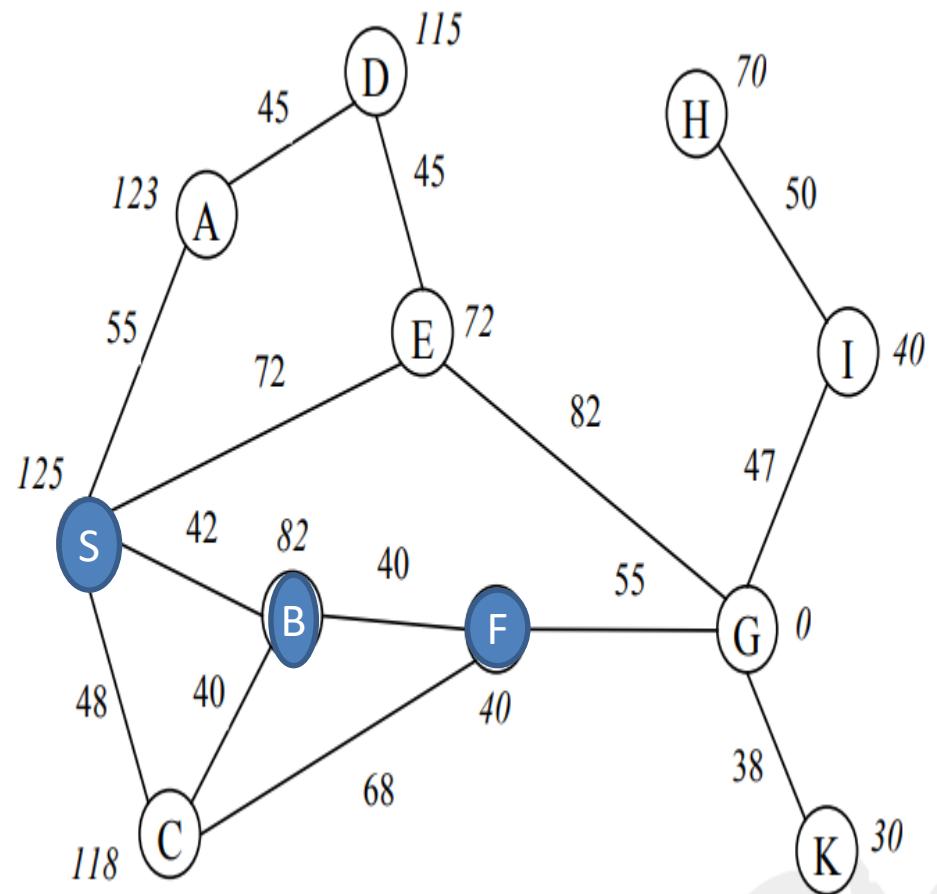
Ví dụ:



Node được xét	Tập biên O
	(125, S)
(125, S)	(178,AS) (124,BS) (166,CS) (144,ES)
(124,BS)	(178,AS) (166,CS) (144,ES) <u>(200,CBS)</u> <u>(122,FBS)</u>

A* Search

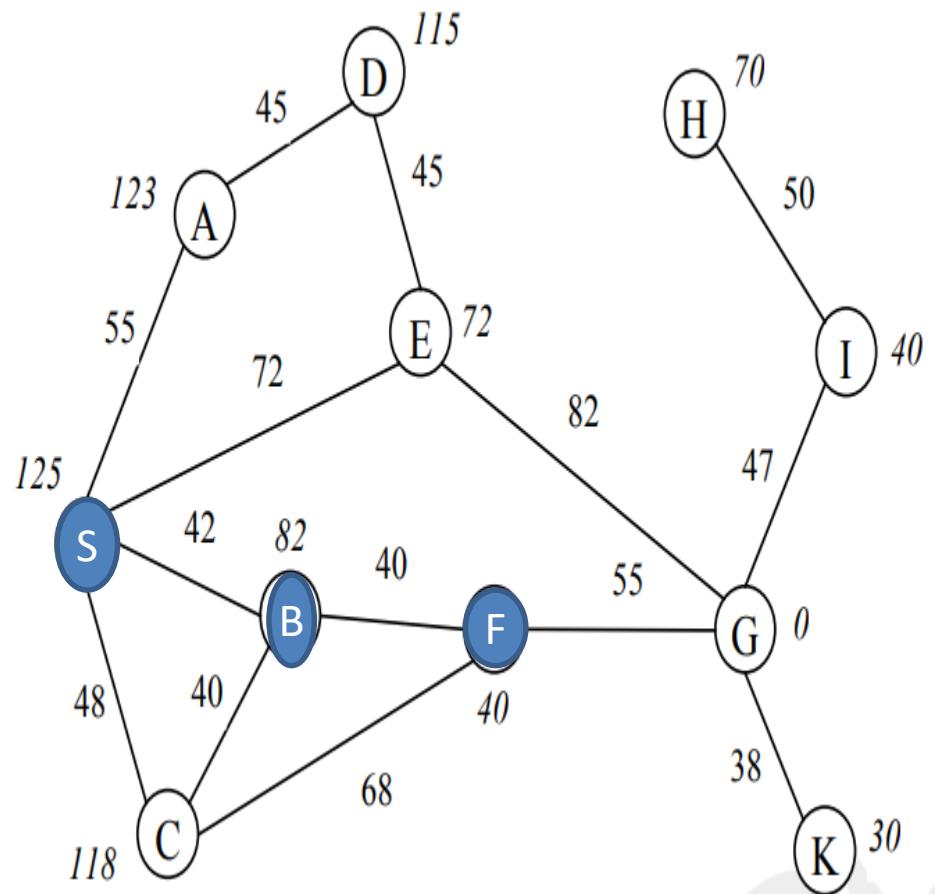
Ví dụ:



Node được xét	Tập biên O
	(125, S)
(125, S)	(178,AS) (124,BS) (166,CS) (144,ES)
(124,BS)	(178,AS) (166,CS) (144,ES) (200,CBS) (122,FBS)
(122,FBS)	

A* Search

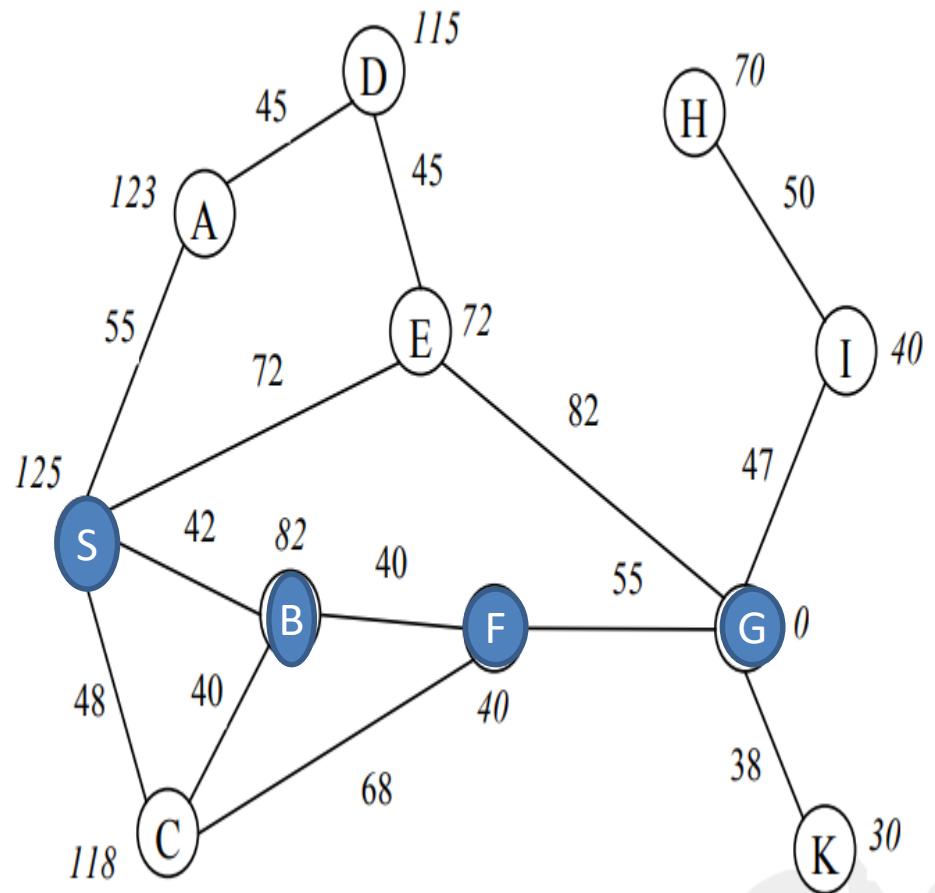
Ví dụ:



Node được xét	Tập biên O
	(125, S)
(125, S)	(178,AS) (124,BS) (166,CS) (144,ES)
(124,BS)	(178,AS) (166,CS) (144,ES) (200,CBS) (122,FBS)
(122,FBS)	(178,AS) (166,CS) (144,ES) (200,CBS) <u>(137,G,FBS)</u> (268,C,FBS)

A* Search

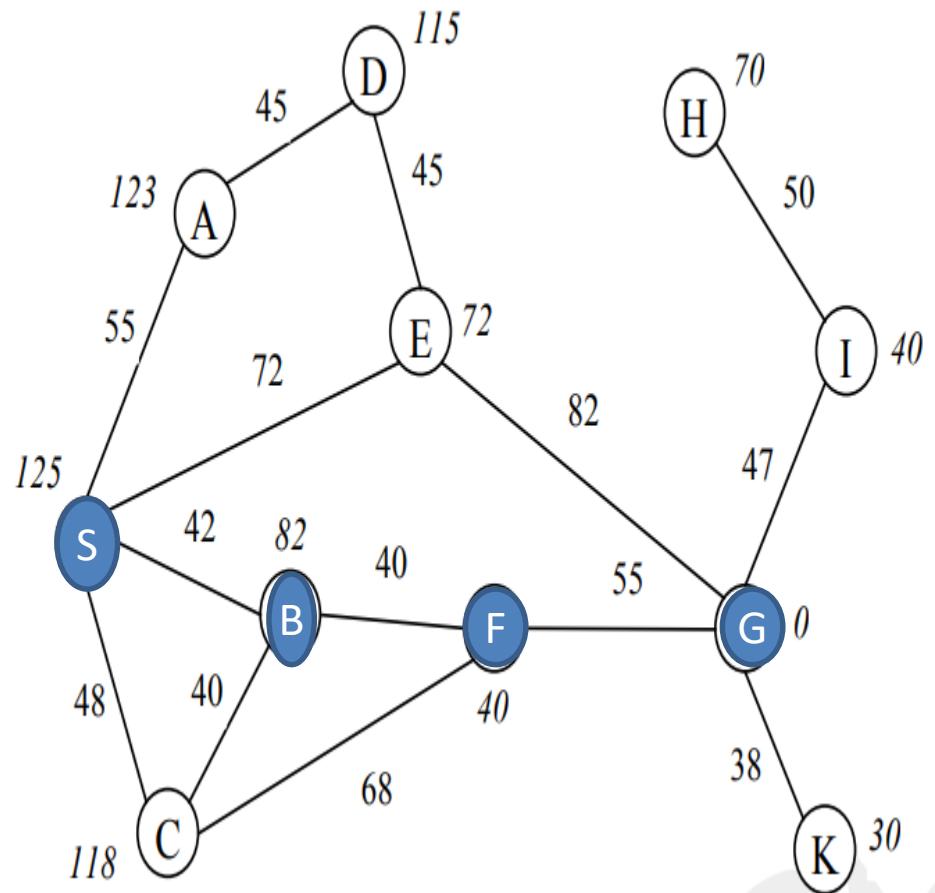
Ví dụ:



Node được xét	Tập biên O
	(125, S)
(125, S)	(178,AS) (124,BS) (166,CS) (144,ES)
(124,BS)	(178,AS) (166,CS) (144,ES) (200,CBS) (122,FBS)
(122,FBS)	(178,AS) (166,CS) (144,ES) (200,CBS) (137,GFBS) (268,CFBS)
(137,GFBS)	

A* Search

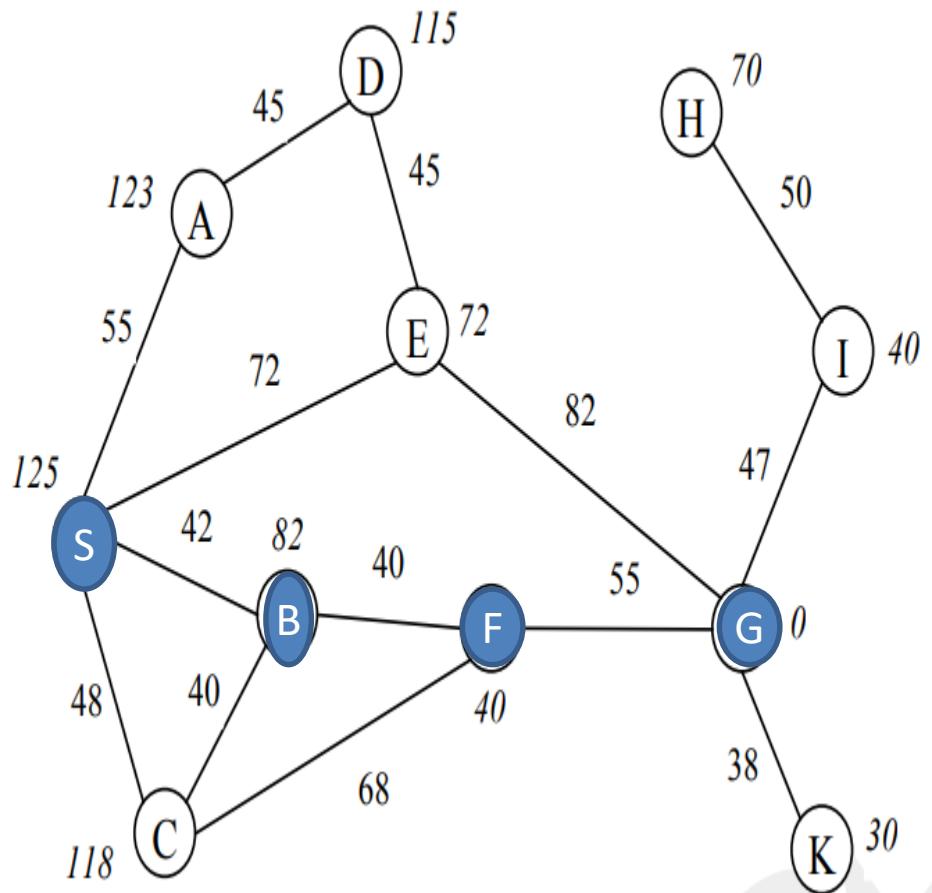
Ví dụ:



Node được xét	Tập biên O
(125, S)	
(125, S)	(178,AS) (124,BS) (166,CS) (144,ES)
(124,BS)	(178,AS) (166,CS) (144,ES) (200,CBS) (122,FBS)
(122,FBS)	(178,AS) (166,CS) (144,ES) (200,CBS) (137,GFBS) (268,CFBS)
(137,GFBS)	(178,AS) (166,CS) (144,ES) (200,CBS) (268,CFBS)

A* Search

Ví dụ:



Node được xét	Tập biên O
	(125, S)
(125, S)	(178,AS) (124,BS) (166,CS) (144,ES)
(124,BS)	(178,AS) (166,CS) (144,ES) (200,CBS) (122,FBS)
(122,FBS)	(178,AS) (166,CS) (144,ES) (200,CBS) (137,GFBS) (268,CFBS)
(137,GFBS)	(178,AS) (166,CS) (144,ES) (200,CBS) (137,GFBS) (268,CFBS)

Chúng ta đã tìm ra con đường (S->B->F->G) với chi phí bằng 137

A* Search

$A^*(O, S, G, P, h)$

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

While(O không rỗng) do

1. Lấy nút N có $f(N)$ nhỏ nhất ra khỏi O
2. Nếu n thuộc G , return(đường đi tới N)
3. Với mọi $M \in P(n)$
 - i. $f(M) = g(M) + h(M)$
 - ii. Thêm M vào O cùng giá trị $f(M)$

Return: không tìm được đường đi

A* Search

- Đặc điểm:
 - Thuật toán cho kết quả tối ưu nếu hàm heuristic h là hàm chấp nhận được.
 - Thuật toán đầy đủ, trừ trường hợp có vô số các nút với hàm f có giá trị rất nhỏ nằm giữa node xuất phát và node đích.
 - Độ phức tạp: trong trường hợp xấu nhất, khi hàm heuristic không có nhiều thông tin, độ phức tạp tính toán và yêu cầu bộ nhớ của A* đều là $O(b^m)$.
 - Trong tất cả các thuật toán tìm kiếm tối ưu sử dụng cùng hàm heuristics thì thuật toán A* có độ phức tạp tính toán nhỏ nhất, tức là yêu cầu sinh ra ít nút nhất trước khi tìm ra lời giải.

Như thế nào là hàm heuristic chấp nhận được???

A* Search

- Hàm heuristic $h(n)$ được gọi là chấp nhận được khi: $h(n) \leq h^*(n)$
 - $h^*(n)$ là giá thành đường đi thực tế từ n đến node đích
 - hàm $h(n)=0$ với mọi n , là hàm chấp nhận được
- Các hàm heuristic:
 - Được xây dựng tùy thuộc vào bài toán cụ thể.
 - Có thể có rất nhiều hàm heuristic khác nhau cho cùng một loại bài toán.
 - Chất lượng hàm heuristic ảnh hưởng rất nhiều đến quá trình tìm kiếm.
 - Trong tất cả các thuật toán tìm kiếm tối ưu sử dụng cùng hàm heuristics thì thuật toán A* có độ phức tạp tính toán nhỏ nhất, tức là yêu cầu sinh ra ít nút nhất trước khi tìm ra lời giải.

A* Search

- Ví dụ:

3	1	6
5		4
2	7	8

Trạng thái hiện thời

	1	2
3	4	5
6	7	8

Trạng thái đích

- Trong bài toán 8 ô, có thể xây dựng một số hàm heuristic. Dưới đây, ta sẽ xem xét hai trong số các hàm như vậy.

A* Search

- Ta có thể sử dụng hai hàm heuristic sau:
 - $h_1(n)$: số ô đặt sai chỗ (ví dụ: $h_1(u)=5$)
 - $h_2(n)$: tổng khoảng cách Manhattan giữa vị trí hiện thời của mỗi ô tới vị trí đúng của ô đó. (ví dụ: $h_2(u) = 1 + 4 + 1 + 2 + 1$
 - Khoảng cách Manhattan được tính bằng số ít nhất các dịch chuyển theo hàng hoặc cột để đưa một ô tới vị trí của nó trong trạng thái đích.
 - $h_3(n)?????$

3	1	6
5		4
2	7	8

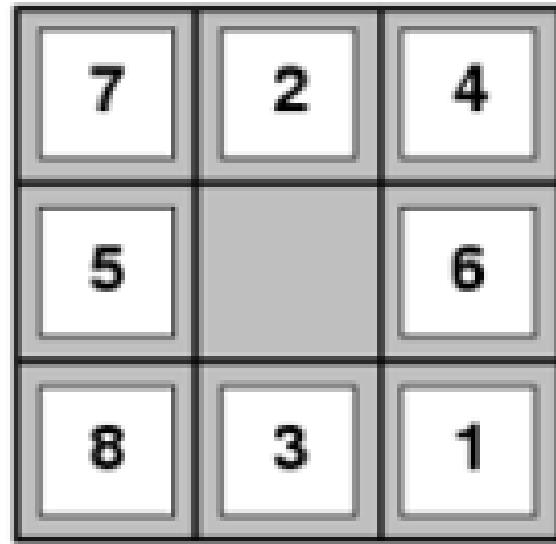
Trạng thái hiện thời

	1	2
3	4	5
6	7	8

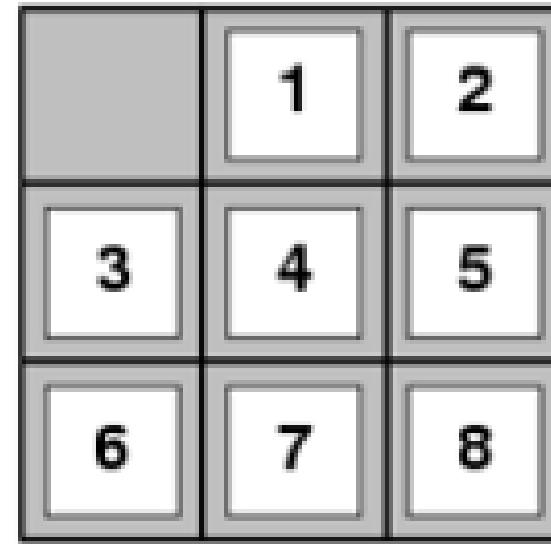
Trạng thái đích

A* Search

- Các ước lượng chấp nhận được



Start State



Goal State

- $h_1(S)=?????$
- $h_2(S)=?????$
- $h_4(S)=?????$

(Số ô sai, khoảng cách và số ô cản trên đường dịch chuyển

A* Search

- Các ước lượng chấp nhận được

- $h_1(u)$ = Số lượng ô sai vị trí.
- $h_2(u)$ = Tổng khoảng cách theo Manhattan Metric

(i.e., Số lượng ô từ ô hiện tại đến vị trí mong muốn)

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

- $h_1(S) = 8$

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

- $h_3(S) = 5+2+3+3+\dots$

(Số ô sai, khoảng cách và số ô cản trên đường dịch chuyển

A* Search

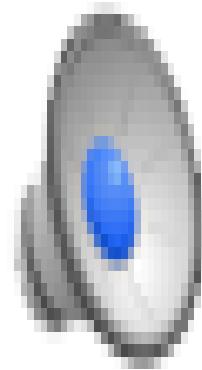
- Hàm heuristic trội:
 - Hàm tốt hơn, hàm nhanh dẫn tới kết quả hơn
 - Giả sử có hai hàm heuristic chấp nhận được $h_1(n)$ và $h_2(n)$.
 - Nếu $h_1(n) \leq h_2(n)$ với mọi n thì ta nói rằng $h_2(n)$ trội hơn so với $h_1(n)$
 - Trong trường hợp trong hai hàm $h_1(n)$ và $h_2(n)$ không có hàm trội hơn???

A* Search

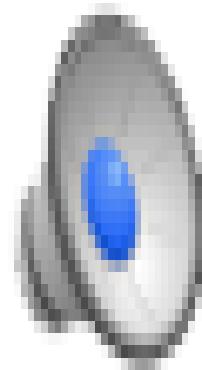
- Hàm heuristic trội:
 - Hàm tốt hơn, hàm nhanh dẫn tới kết quả hơn
 - Giả sử có hai hàm heuristic chấp nhận được $h_1(n)$ và $h_2(n)$.
 - Nếu $h_1(n) \leq h_2(n)$ với mọi n thì ta nói rằng $h_2(n)$ trội hơn so với $h_1(n)$
 - Trong trường hợp trong hai hàm $h_1(n)$ và $h_2(n)$ không có hàm trội hơn???
 - Có thể tạo ra hàm $h'(n) = \max(h_1(n), h_2(n))$ với mọi n .

So sánh giữa các thuật toán tìm kiếm

- Video of Demo Contours (Empty)



UCS

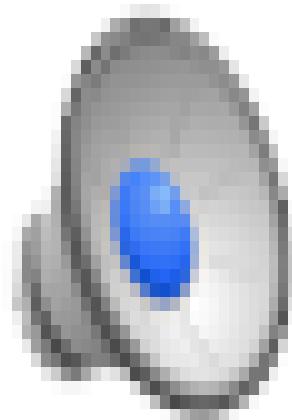


Greedy

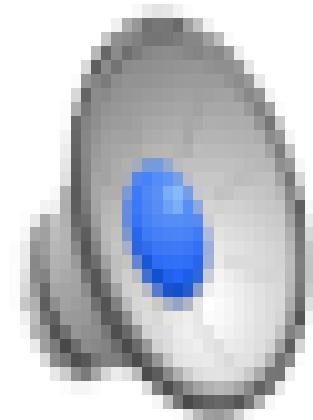
A^*

So sánh giữa các thuật toán tìm kiếm

- Video of Demo Contours (Pacman Small Maze)



Greedy



A*

So sánh giữa các thuật toán tìm kiếm



Greedy



Uniform Cost



A*

A* search

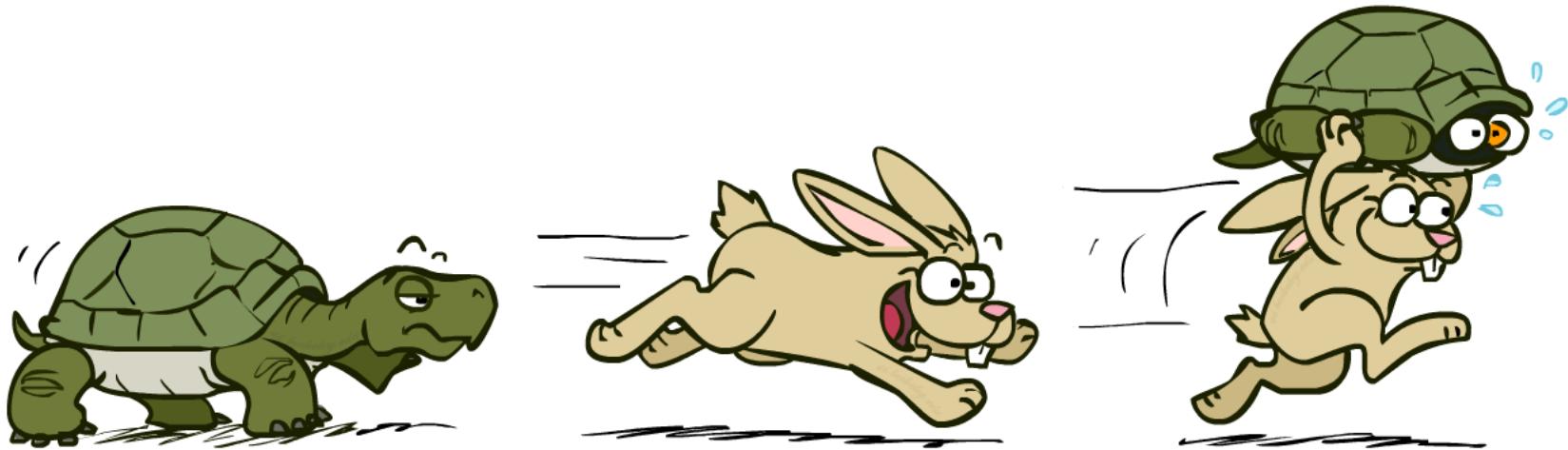
Các ứng dụng của A*

- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- ...



A* search

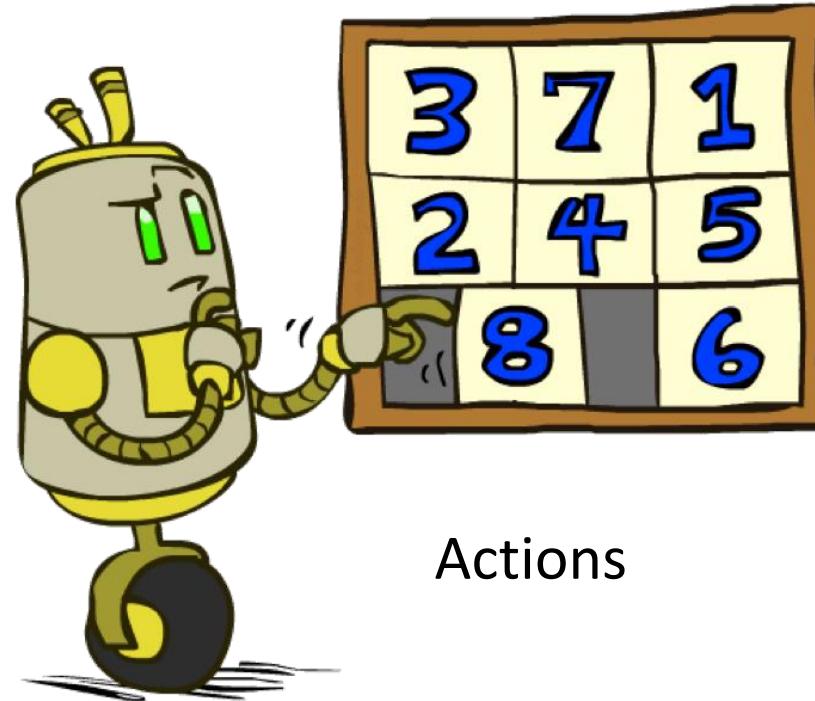
- Tóm tắt A*
 - A* sử dụng cả chi phí ngược và (ước tính) chi phí xuôi
 - A* là tối ưu với các phương pháp heuristic có thể chấp nhận được/nhất quán
 - Thiết kế heuristic là chìa khóa



Example: 8 Puzzle

7	2	4
5		6
8	3	1

Start State



Actions

	1	2
3	4	5
6	7	8

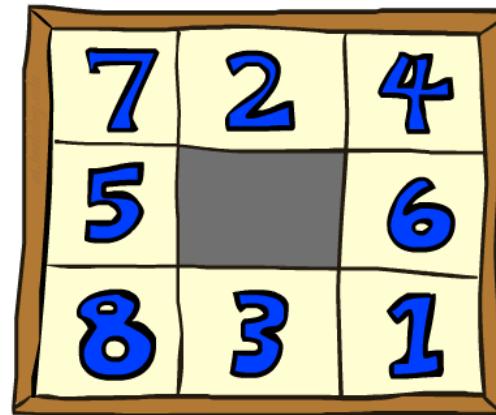
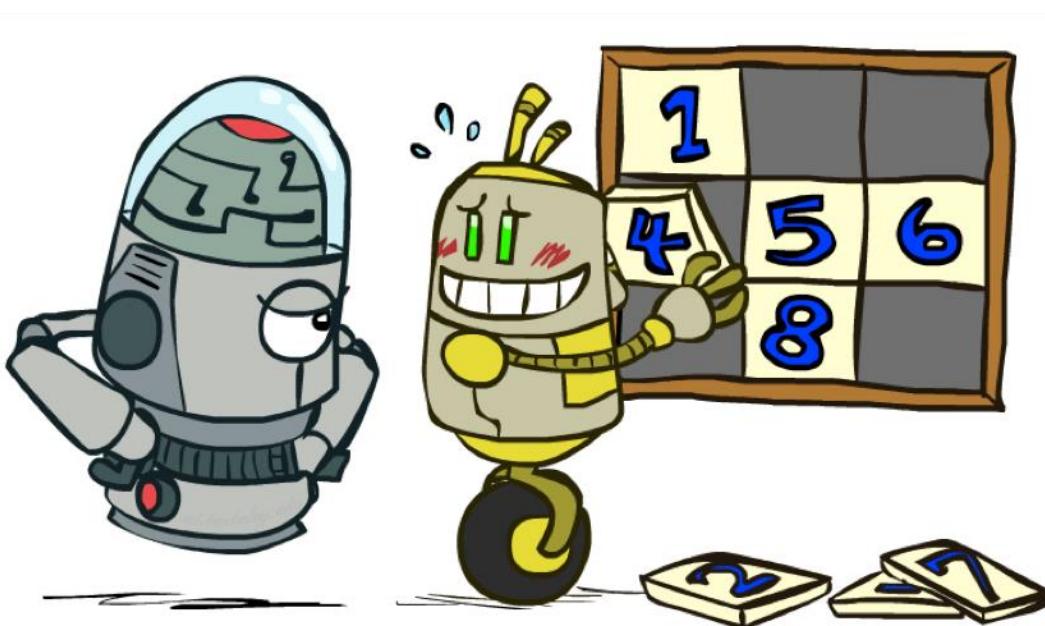
Goal State

- What are the states?
- How many states?
- What are the actions?
- How many successors from the start state?
- What should the costs be?

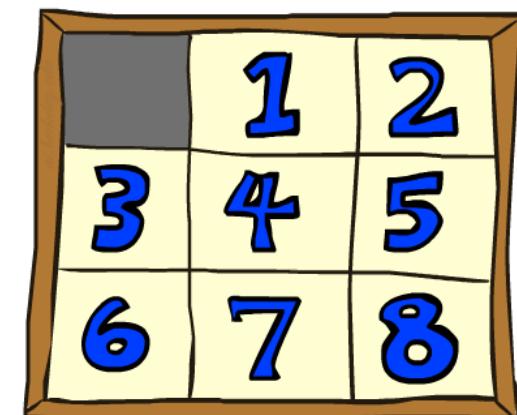
8 Puzzle heuristics

8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
- $h_1(\text{start}) = 8$
- This is a *relaxed-problem* heuristic



Start State



Goal State

Average nodes expanded
when the optimal path has...

	...4 steps	...8 steps	...12 steps
UCS	112	6,300	3.6×10^6
TILES	13	39	227

8 Puzzle heuristics

4	6	5
7	2	
1	3	8

3	1	8
7	4	2
6	5	

admissible?

tiles misplaced

Tile	n	Goal
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0
7	0	0
8	1	0
$h_1(n)$	7	
$h_1(goal)$	0	

8 Puzzle heuristics

4	6	5
7	2	
1	3	8

3	1	8
7	4	2
6	5	

admissible?

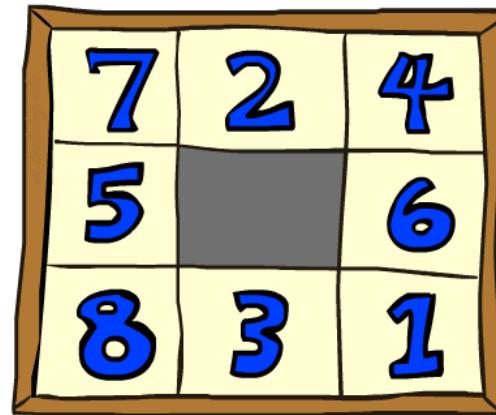
Manhattan distance

Tile	n	Goal
1	3	
2	1	
3	3	
4	2	
5	3	
6	3	
7	0	
8	2	
$h_2(n)$	17	
$h_2(goal)$	0	

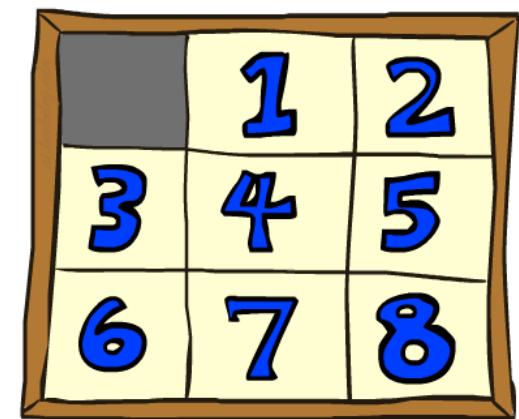
8 Puzzle heuristics

8 Puzzle II

- What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?
- Total *Manhattan distance*
- Why is it admissible?
- $h(\text{start}) = 3 + 1 + 2 + \dots = 18$



Start State



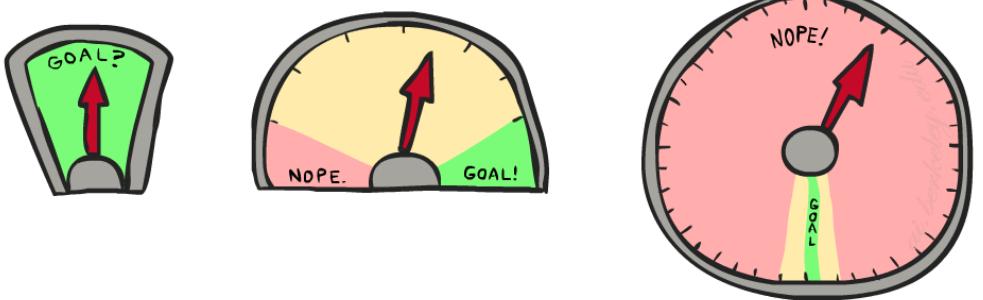
Goal State

Average nodes expanded when the optimal path has...			
	...4 steps	...8 steps	...12 steps
TILES	13	39	227
MANHATTAN	12	25	73

8 Puzzle heuristics

8 Puzzle III

- How about using the *actual cost* as a heuristic?
 - Would it be admissible?
 - Would we save on nodes expanded?
 - What's wrong with it?



- With A*: a trade-off between quality of estimate and work per node
 - As heuristics get closer to the true cost, you will expand fewer nodes but usually do more work per node to compute the heuristic itself

8 Puzzle with A*

- Solving 8 Puzzle with A* using tree search

3	1	8
4	2	
7	6	5

start

3	1	8
7	4	2
6	5	

goal

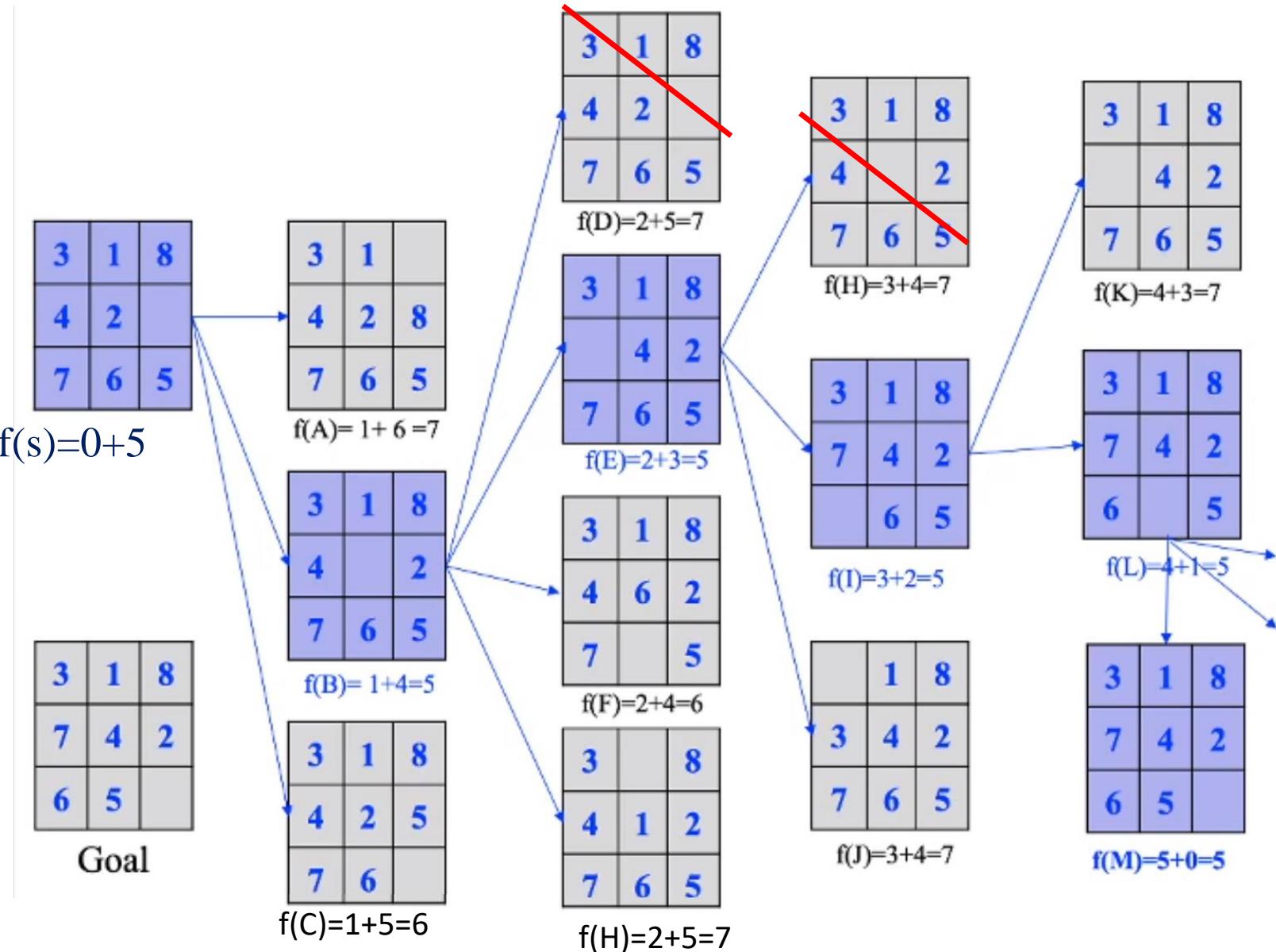
- Heuristic function $h(n)$ = the number of misplaced tiles.
- Steps of A*

Application of A*

8 Puzzle with A*

- Heuristic function
 $h(n)$ = the number of misplaced tiles.

A* Search orders by the sum: $f(n) = g(n) + h(n)$



Thanks for your attention!

Q&A

Bài tập vận dụng

- Sử dụng A* để tìm lời giải cho bài toán 8-puzzle với hàm h tính theo số miếng số sai vị trí, và g tính theo số bước dịch chuyển.

3	5	6
	7	8
4	2	1

start

3	5	6
1	2	4
8		7

goal

Bài tập vận dụng

Thực hiện tìm lời giải theo phương pháp duyệt IDS (Iterative Deepening search- sâu dần, với độ sâu mỗi mức là d=10) cho bài toán 8-puzzle, thể hiện bằng tree search (vẽ lên giấy) chụp đưa vào word=> PDF

Trạng thái đầu (**S**)

3	5	6
	7	8
4	2	1

Trạng thái đích (**G**)

3	5	6
1	2	4
8		7

Bài tập vận dụng

- Sử dụng A* để tìm lời giải cho bài toán 8-puzzle với hàm h tính theo khoảng cách Manhattan miếng số sai vị trí và g tính theo số bước dịch chuyển.

3	1	8
7	4	2
6	5	

start

4	6	5
7	2	
1	3	8

goal

Bài tập vận dụng

- Sử dụng A* để tìm lời giải cho bài toán 8-puzzle và xác định hàm h.



trạng thái xuất phát

2	6	5
	8	7
4	3	1

trạng thái đích

1	2	3
4	5	6
7	8	



1> DOWN	2> RIGHT	3> UP	4> RIGHT	5> UP
6> LEFT	7> LEFT	8> DOWN	9> RIGHT	10> RIGHT
11> DOWN	12> LEFT	13> LEFT	14> UP	15> RIGHT
16> RIGHT	17> UP	18> LEFT	19> DOWN	20> LEFT
21> UP	22> RIGHT	23> DOWN	24> DOWN	25> RIGHT
26> UP	27> LEFT	28> UP	29> RIGHT	30> DOWN
				31> DOWN

Thời gian thực hiện: 49.4436571598053