

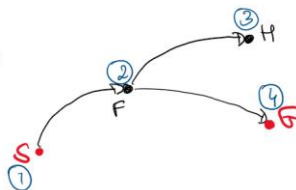
TP_3

exo 1

①

on commence à l'état S ou veut atteindre G

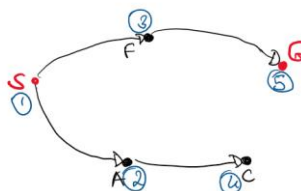
DFS



- ① $[] + [A, F] = [A, F]$
- ② $[A] + [A, G, H] = [A, A, G, H]$
- ③ $[A, A, G] + [] = [A, A, G]$

- $$\begin{aligned} T(S) &= [A, F] \\ T(A) &= [A, F] \\ T(F) &= [A, G, H] \\ T(H) &= [] \\ T(G) &= [D, G] \\ T(D) &= [G] \end{aligned}$$

BFS



- ① $[] + [A, F] = [A, F]$
 - ② $[F] + [C, D] = [F, C]$
 - ③ $[C] + [A, G, H] = [A, G, H]$
 - ④ $[G, H] + [D, G] = [G, H, D]$
- ou ne met pas le doublon
déjà exploré

② ① état

	S	A	C	D	F	G	H
h	10	5	4	3	4	0	2

Heuristique : ① \forall noeud V , $h(V) \geq 0$ ✓
 ② soit V' le noeud à atteindre $\Rightarrow h(V') = 0$ ✓

c'est une heuristique, est-elle admissible?

admissible : ① soit $h^*(V)$ la fonction qui retourne le coût réel du noeud V à la solution

état

	S	A	C	D	F	G	H
h	10	5	4	3	4	0	2
h*	11	3	5	3	5	0	

$$\Rightarrow \forall \text{noeud } V, 0 \leq h(V) \leq h^*(V) \quad \checkmark$$

\rightarrow oui elle est admissible

② greedy best first search

greedy prend la meilleure décision possible à chaque nœud v , sans compter avec les étapes déjà faites,

soit $f(v)$ la fonction d'évaluation à nœud v ,

$$\leadsto f(v) = h(v)$$

A^*

fonctionne comme greedy, mais prend en compte le coût total depuis l'origine

$$\leadsto f(v) = h(v) + g(v)$$

où $g(v)$ est le coût depuis origine à v

③

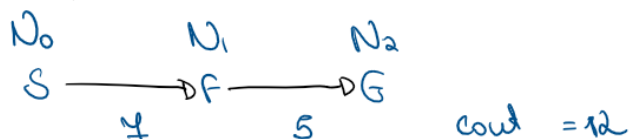
Greedy

① crée la racine de l'arbre de recherche N_0 avec l'état S et $f(N_0) = 10$, ensuite il atteint les nœuds suivants successivement:

② $N_0 \rightarrow F \leadsto f(F) = 4$
 $N_0 \rightarrow A \leadsto f(A) = 5$
 N_0 crée N_1 avec l'état $F \leadsto f(N_1) = 4$

③ $N_1 \rightarrow H \leadsto f(H) = 2$
 $N_1 \rightarrow G \leadsto f(G) = 0$
 $N_1 \rightarrow A \leadsto f(A) = 5$
 N_1 crée N_2 avec l'état $G \leadsto f(N_2) = 0$

Solution :



A*

① crée la racine de l'arbre de recherche N_0 avec l'état S et $f(N_0) = 10 + 0$, ensuite il atteint les noeuds suivants successivement:

② $N_0 \rightarrow A \rightarrow f(A) = h(A) + g(A) = 5 + 2 = 7$
 $N_0 \rightarrow F \rightarrow f(F) = h(F) + g(F) = 4 + 7 = 11$

N_0 crée N_1 avec l'état $A \rightarrow f(N_1) = 7$

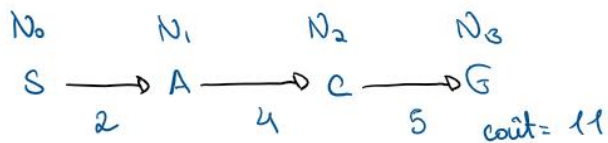
③ $N_1 \rightarrow F \rightarrow f(F) = h(F) + g(F) = 4 + 10 = 14$
 $N_1 \rightarrow C \rightarrow f(C) = h(C) + g(C) = 4 + 6 = 10$

N_1 crée N_2 avec l'état $C \rightarrow f(N_2) = 10$

④ $N_2 \rightarrow D \rightarrow f(D) = h(D) + g(D) = 3 + 9 = 12$
 $N_2 \rightarrow G \rightarrow f(G) = h(G) + g(G) = 0 + 11 = 11$

N_2 crée N_3 avec l'état $G \rightarrow f(N_3) = 11$

Solution:



④ cf code python