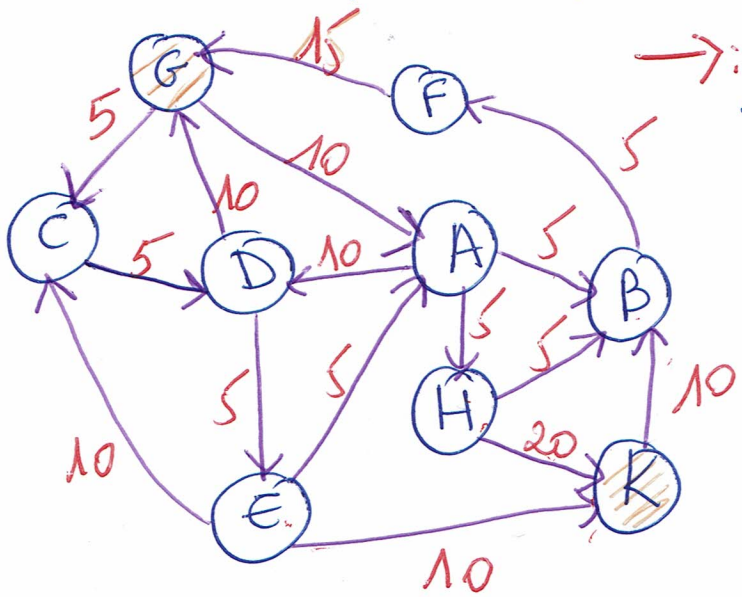
Méthode de recherche d'une solution dans un espace d'état

Recherche aveugle — En largeur d'abord (Breadth first)

On veut aller d'une ville à une autre ville.
Voilà le graphe qui représente les différents chemins

1) les chiffres représentent les coûts de transition d'une ville à une autre (d'un état à un autre).
2) les flèches représentent les transitions
3) les villes = Etats sont représentés par des lettres

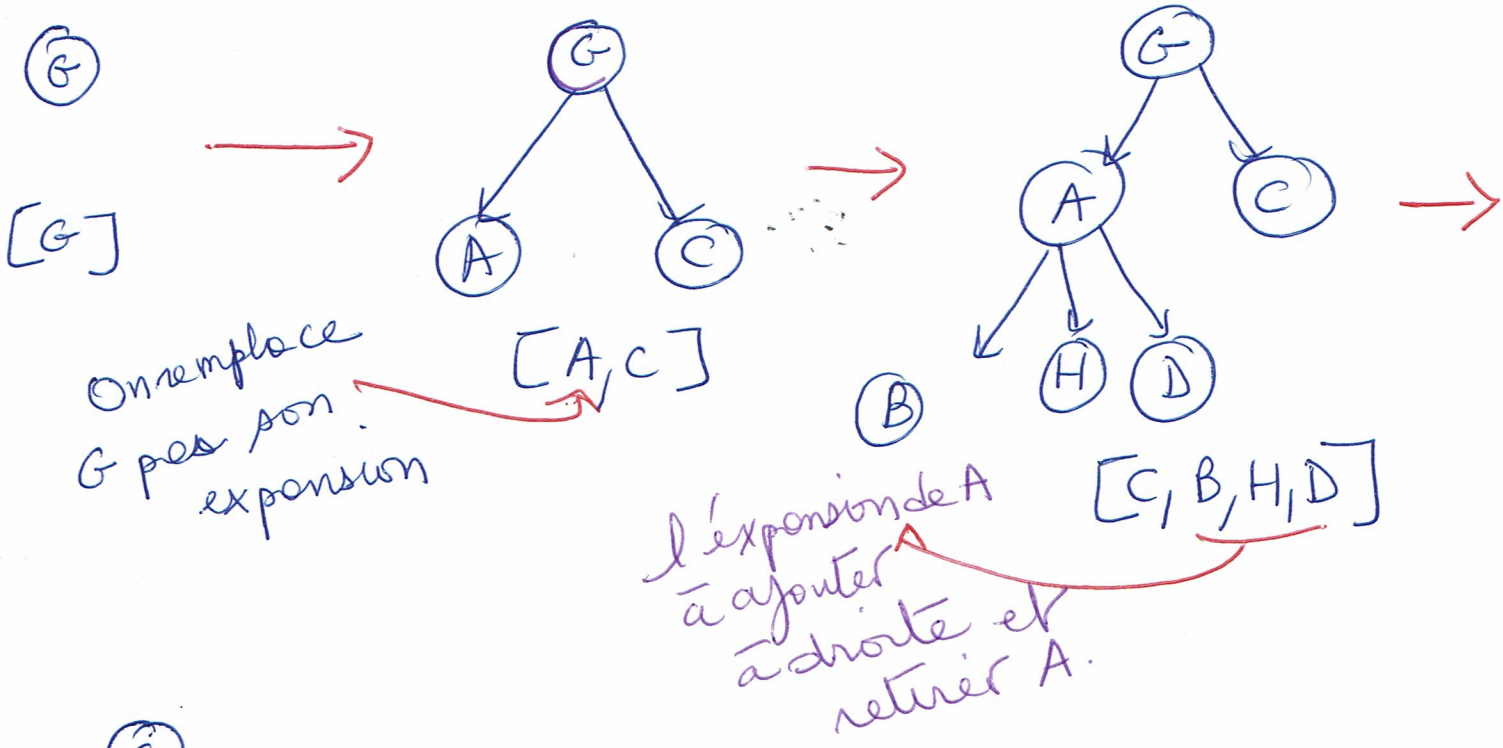


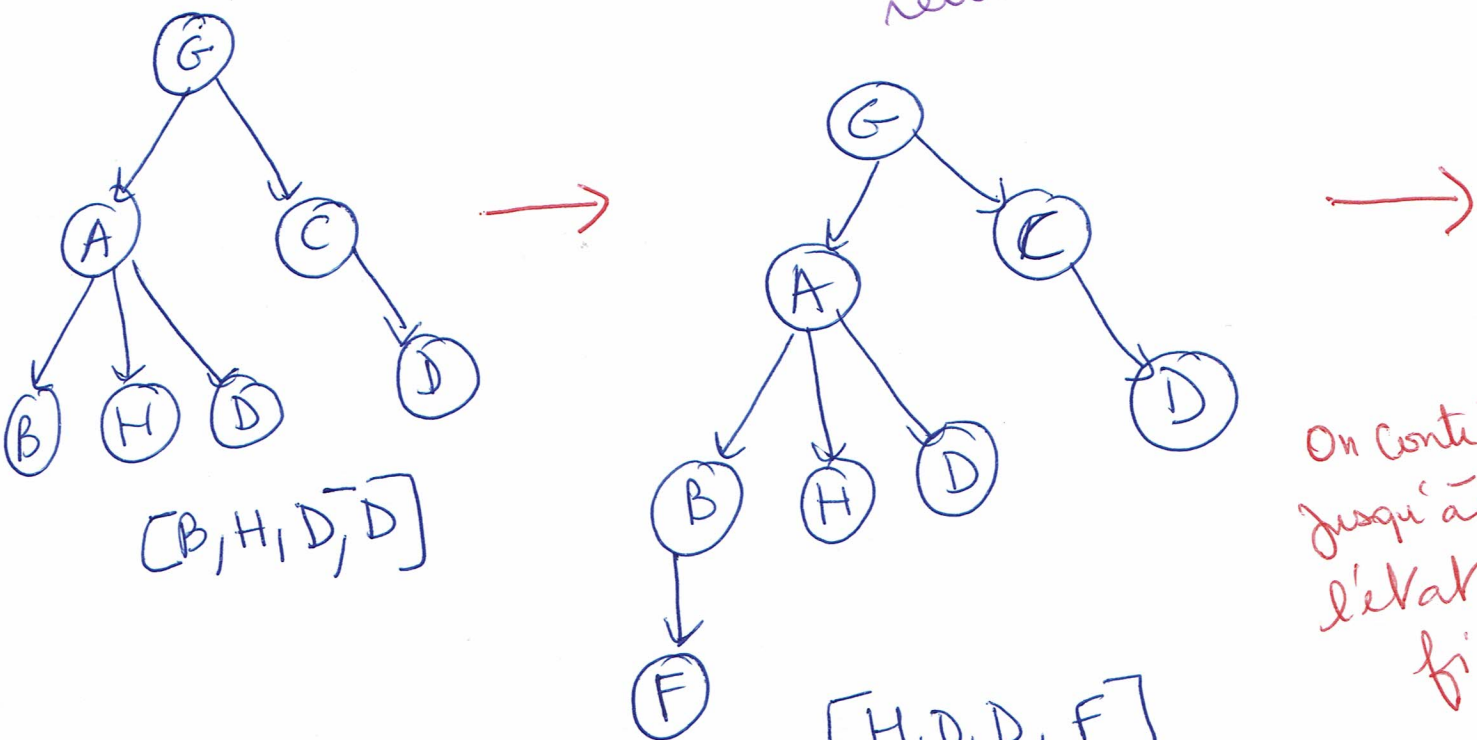
→ : représente le passage d'un état à un autre.

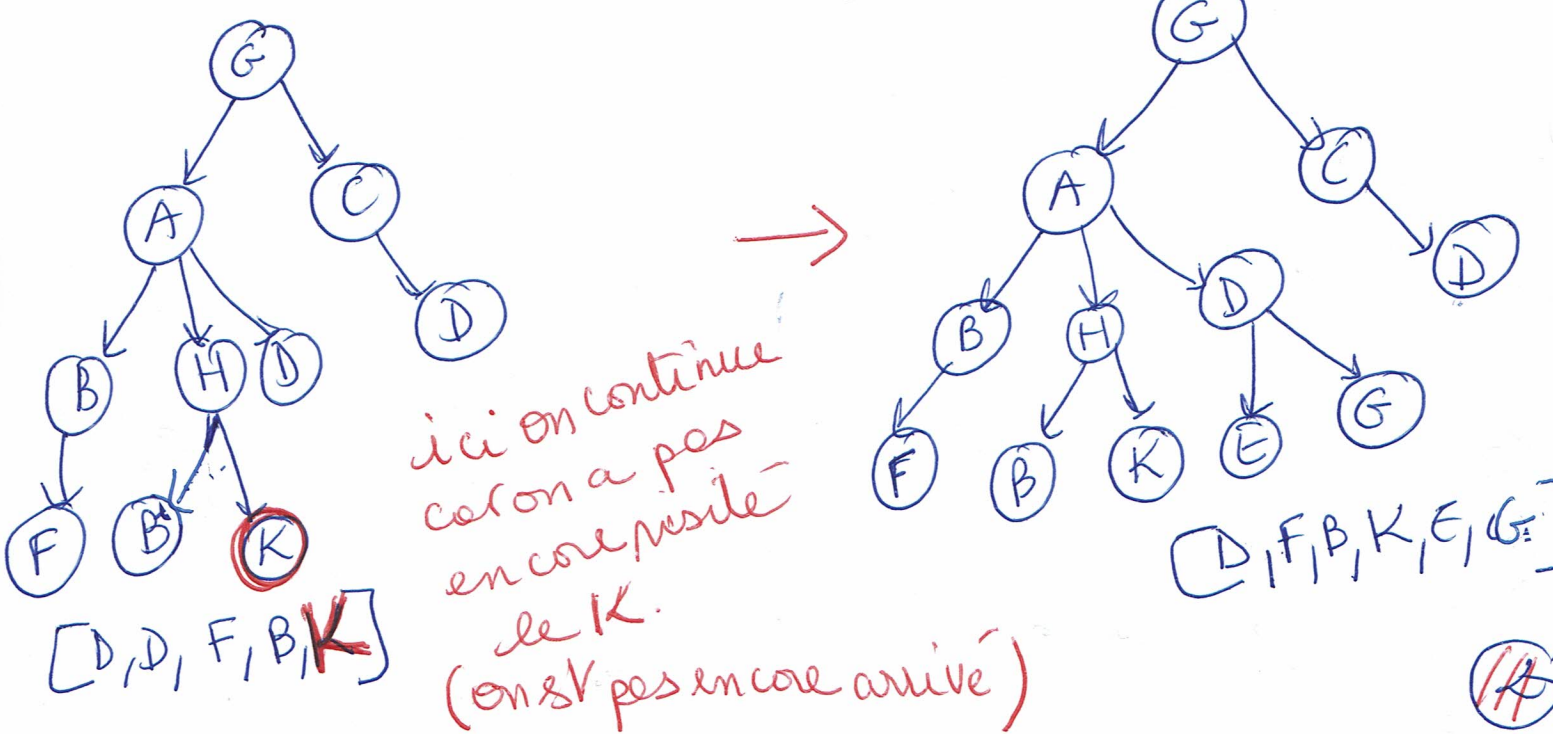
On veut aller de la ville G à la ville K

- G : Etat initial
- K : Etat final

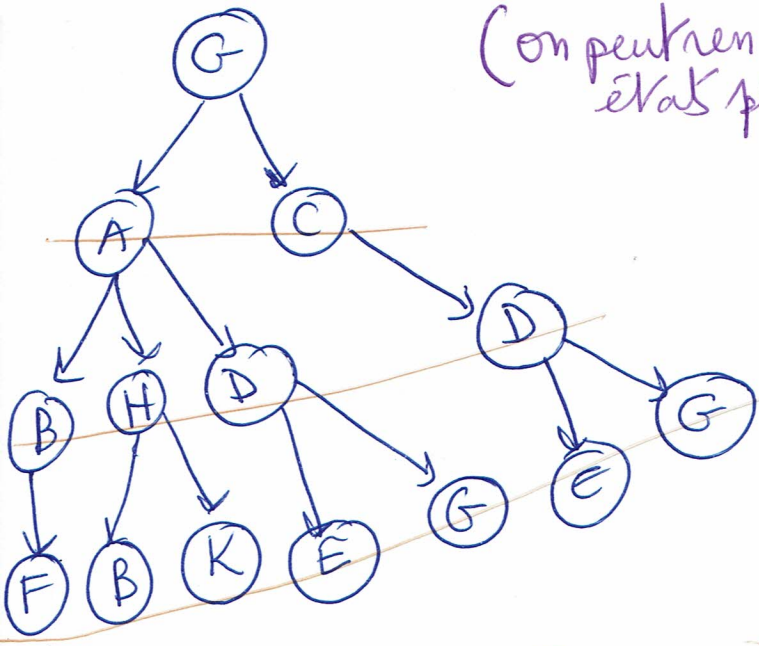
Cet algorithme repose sur l'exploration de chaque niveau en largeur ensuite passer à un niveau inférieur (largeur d'abord)
C'est la méthode de recherche la plus simple, elle est utile si l'espace est petit. La solution si elle existe, elle n'est pas optimale l'inconvénient elle encombre la mémoire par l'enregistrement des états.


G
[G]
G
A
C
[A, C]
On remplace G par son expansion
G
A
C
B
H
D
[C, B, H, D]
l'expansion de A à ajouter à droite et retirer A.

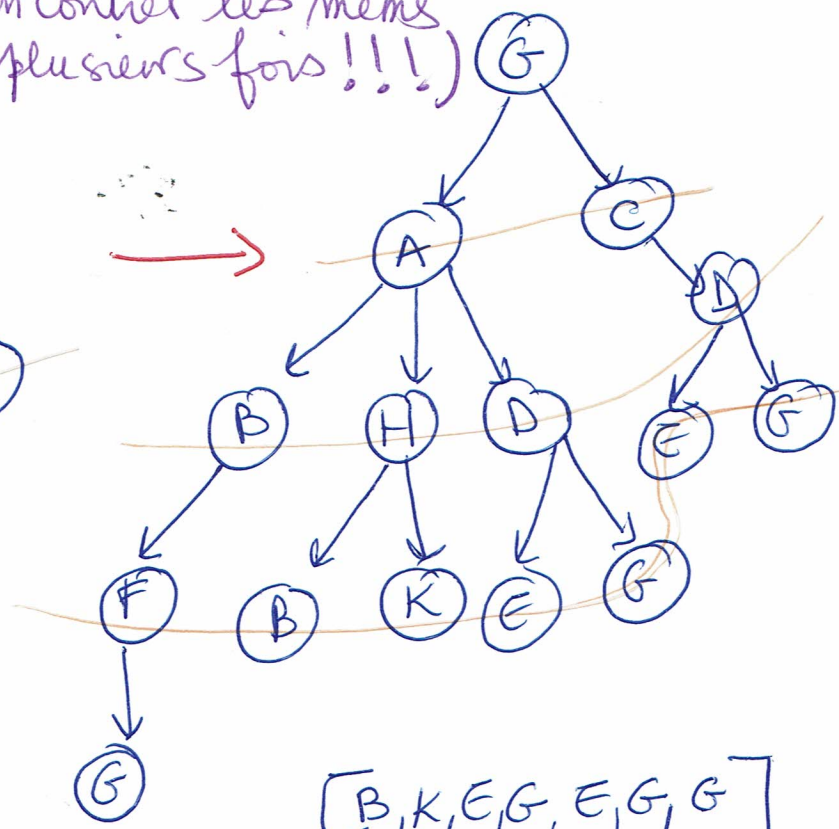

G
A
C
B
H
D
D
[B, H, D, D]
G
A
C
B
H
D
D
F
[H, D, D, F]
On continue jusqu'à l'état final


G
A
C
B
H
D
D
F
B
K
[D, D, F, B, K]
ici on continue car on a pas encore visité le K.
(on s'y pas encore arrivé)
G
A
C
B
H
D
D
F
B
K
E
G
[D, F, B, K, E, G]

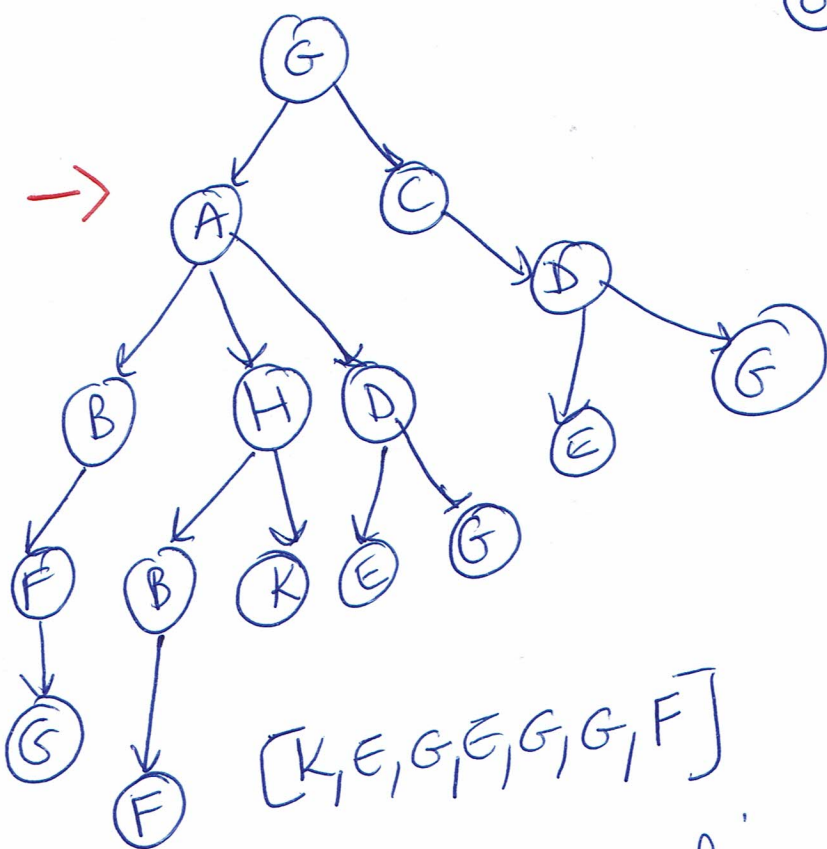
(on peut rencontrer les mêmes états plusieurs fois !!!)



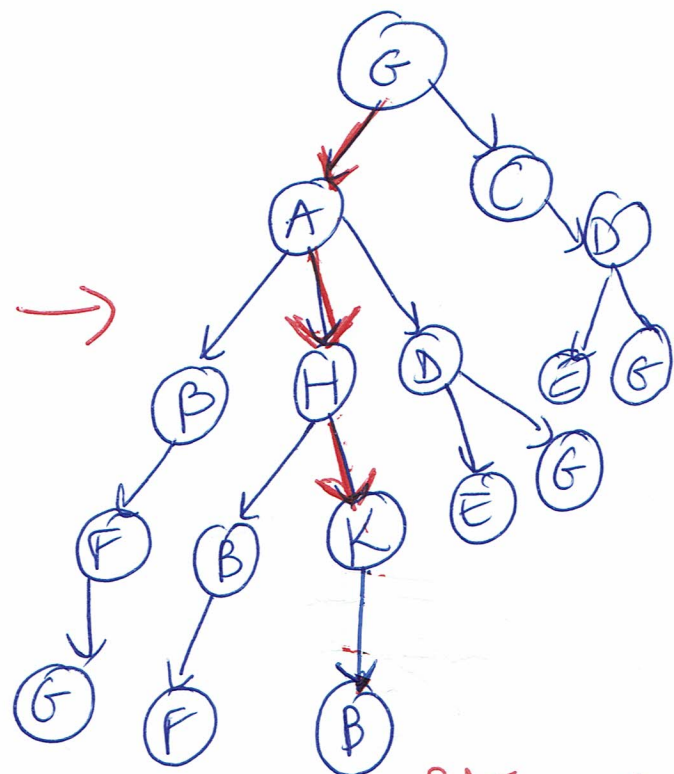
[F, B, K, E, G, E, G]



[B, K, E, G, E, G, G]



[K, E, G, E, G, G, F]



On continue à chaque fois jusqu'à trouver une solution, une fois trouvée; on s'arrête

ici on s'arrête car on est arrivé à K, on peut le remplacer.

les états entre [ ] sont utiles pour la programmation.

Donc le chemin est: G → A → H → K

Cette solution n'est pas optimale car les coûts sont différents