Intelligence artificielle (IF06M100) • TD 2

Algorithmes de recherche en IA

Les problèmes déterministes et complètement observables

- Un problème déterministe et complètement observable est défini par :
  1. un **état initial**
     - par exemple, "à Arad"
  2. un **ensemble d'actions** ou une **fonction de transition**, $succ(x)$:
     - par exemple, $succ(Arad) = \{Zerind, Timisoara, Sibiu\}$
  3. un **test de terminaison** pour savoir si le but est atteint
     - explicite, e.g., "à Bucharest"
     - implicite, e.g., "vérifier mat au échec"
  4. un **coût** (additif)
     - ça peut être la somme des distances, le nombre d'actions exécutées, etc.
     - par exemple, $c(x, a, y) \geq 0$ est le coût de l'action a qui permet de passer de l'état x à l'état y
- Une solution est une séquence d'actions partant de l'état initial et menant à l'état but.

Exercice 1

Donnez l'état initial, le but, la fonction successeur et la fonction de coût pour chacun des problèmes suivants.

Question 1

Vous devez colorier une carte de façon à ce que les pays adjacents ne soient pas de la même couleur, et en sachant que vous avez à votre disposition 4 couleurs distinctes

Etat initial	Aucune pays coloré
Le but	Tous les pays colorés et aucun pays adjacent n'a la même couleur
La fonction successeur	Attribuer une couleur à un pays.
La fonction de coût	Nombre d'affectations

Exercice 1

Donnez l'état initial, le but, la fonction successeur et la fonction de coût pour chacun des problèmes suivants.

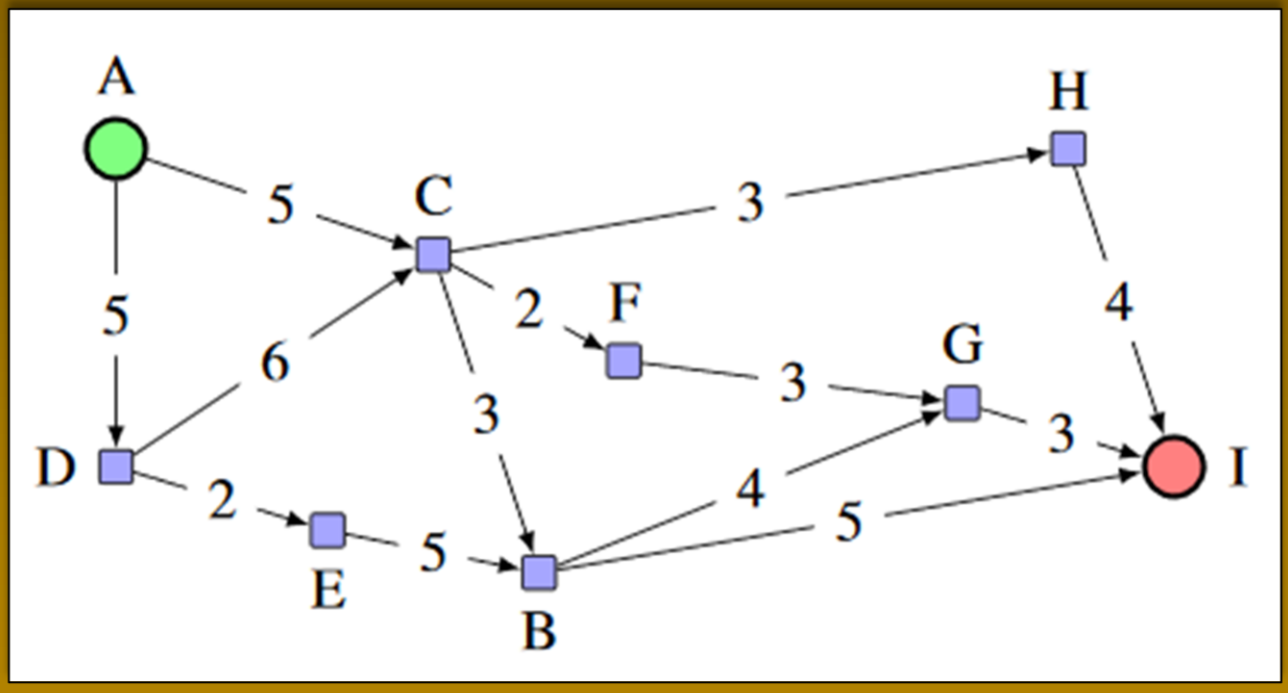
Question 2

Un singe mesurant 1 mètre se trouve dans une pièce de 3 mètres de hauteur. Une banane est suspendue au plafond de cette pièce, et le singe aimerait bien avoir cette banane. La pièce contient également 2 caisses qu'il peut déplacer et sur lesquelles il peut monter, chaque caisse mesurant 1 mètre

Etat initial	Comme décrit dans le texte Un singe mesurant 1 mètre se trouve dans une pièce de 3 mètres de hauteur. Une banane est suspendue au plafond de cette pièce. La pièce contient également 2 caisses, chaque caisse mesurant 1 mètre.
Le but	Le singe arrive à avoir la banane
La fonction successeur	Montez sur une caisse ; Descendez de la caisse ; Pousser la caisse d'un endroit à un autre, marcher d'un endroit à un autre ; saisir la banane (si debout sur une caisse)
La fonction de coût	Nombre d'actions

Exercice 2

Considérez la carte (orientée) suivante.
L'objectif est de trouver un chemin allant de A à I.



Donner l'ordre de parcours des nœuds pour les algorithmes suivants.

- **Si vous avez le choix entre deux nœuds, vous développerez en priorité le premier dans l'ordre alphabétique.**
  On suppose que nous pouvons éviter les répétitions : un état contenu dans un nœud déjà développé ne le sera plus.

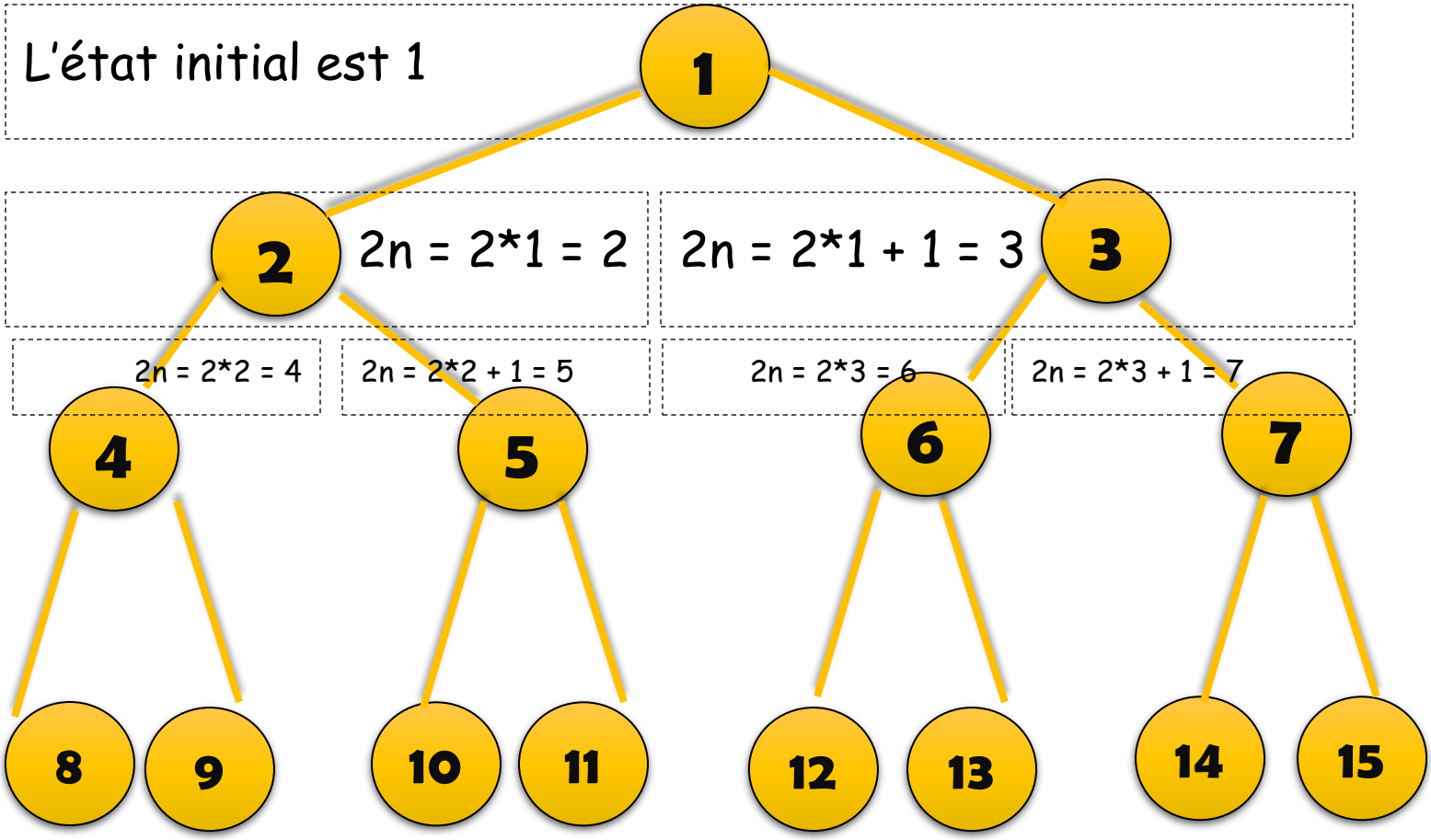
1. largeur d'abord
2. coût uniforme
3. profondeur d'abord

Exercice 3

Considérez un espace de recherche dans lequel l'état initial est 1 et la fonction successeur pour un nœud n retourne deux états contenant les entiers $2n$ et $2n+1$.

Question 1

Dessiner la partie de l'espace de recherche contenant les nœuds de 1 à 15



Exercice 3

Considérez un espace de recherche dans lequel **l'état initial est 1** et **la fonction successeur** pour un nœud n retourne deux états contenant les entiers $2n$ et $2n+1$.

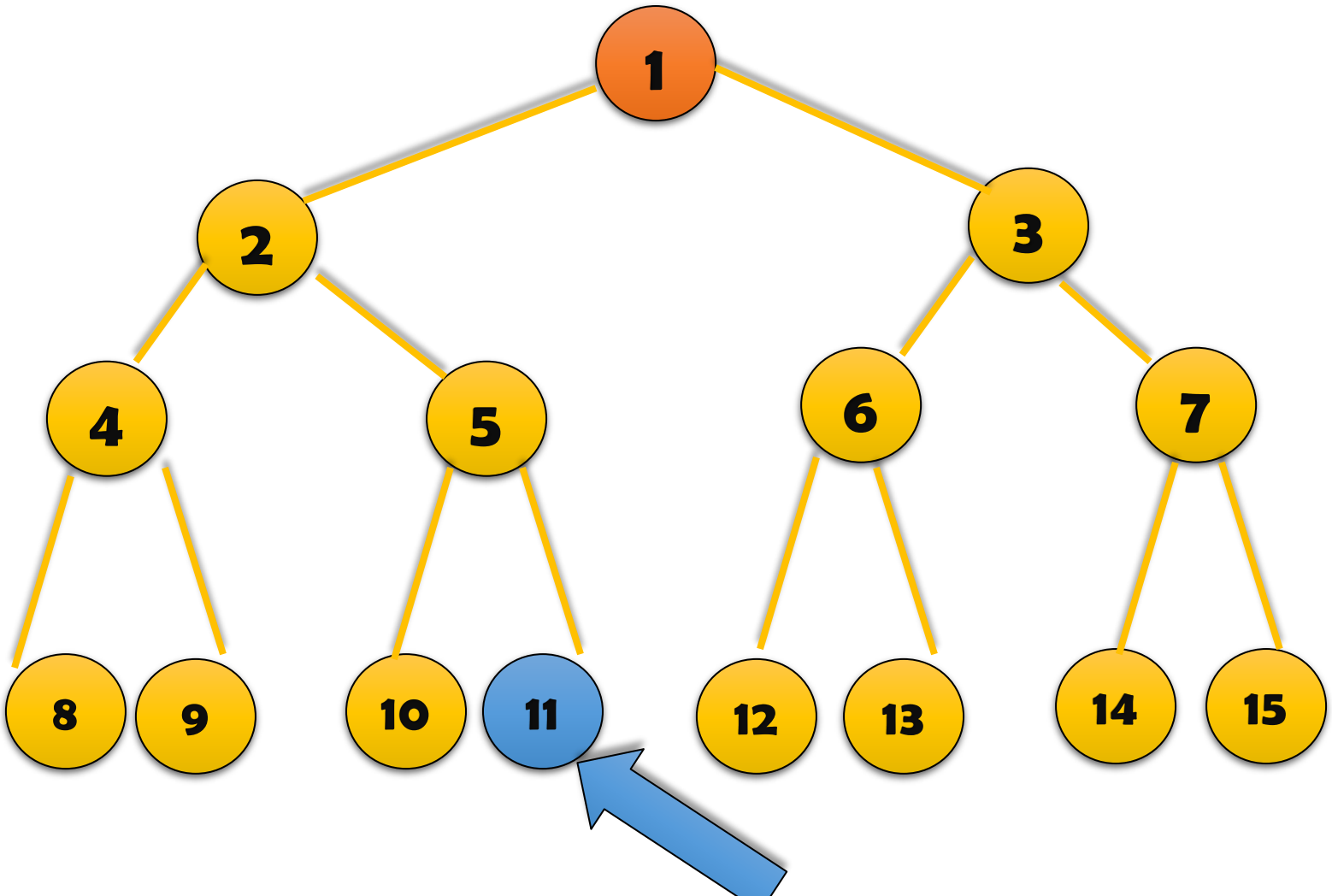
Question 2

Supposez que le but soit 11,

et que vous considérez l'espace de recherche complet.

Donner l'ordre de parcours des nœuds pour les algorithmes :

(a) largeur d'abord



Recherche en largeur d'abord (BFS)

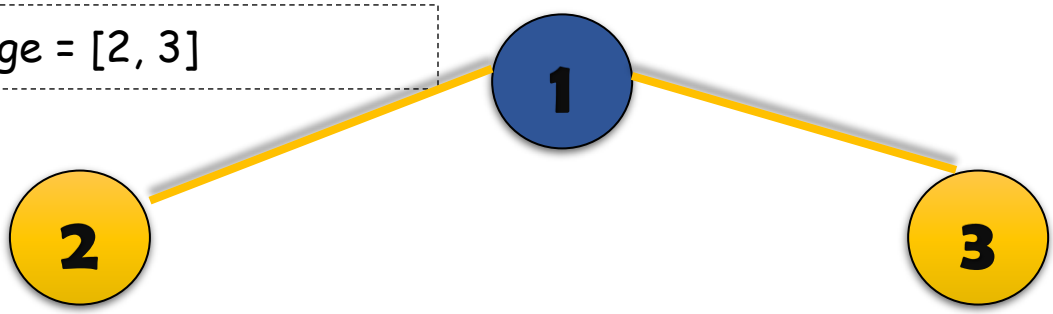
- `Insert-Fn` ajoute les successeurs en fin de liste *(c'est une file)*

fringe = [1]

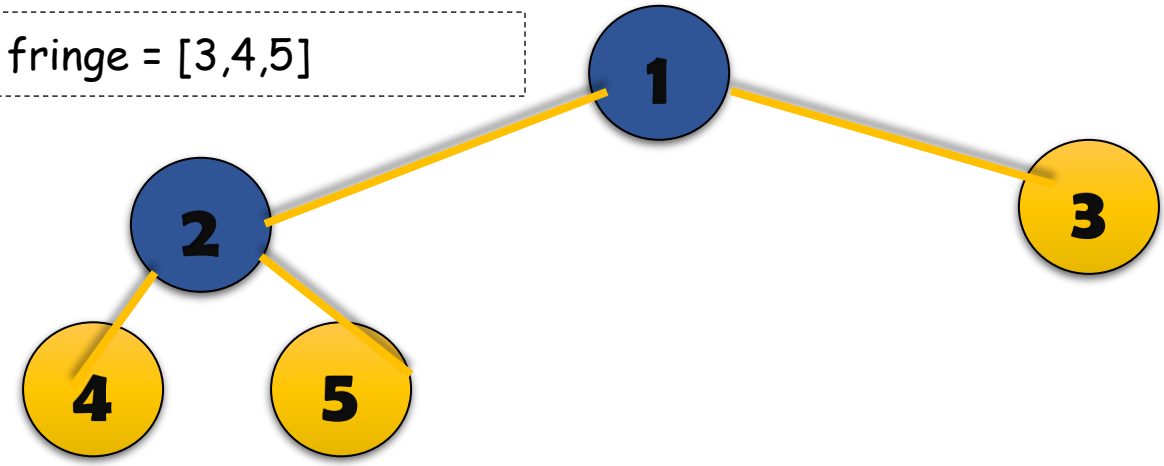
Une file qui représente l'ensemble des nœud a développer

L'état initial est 1

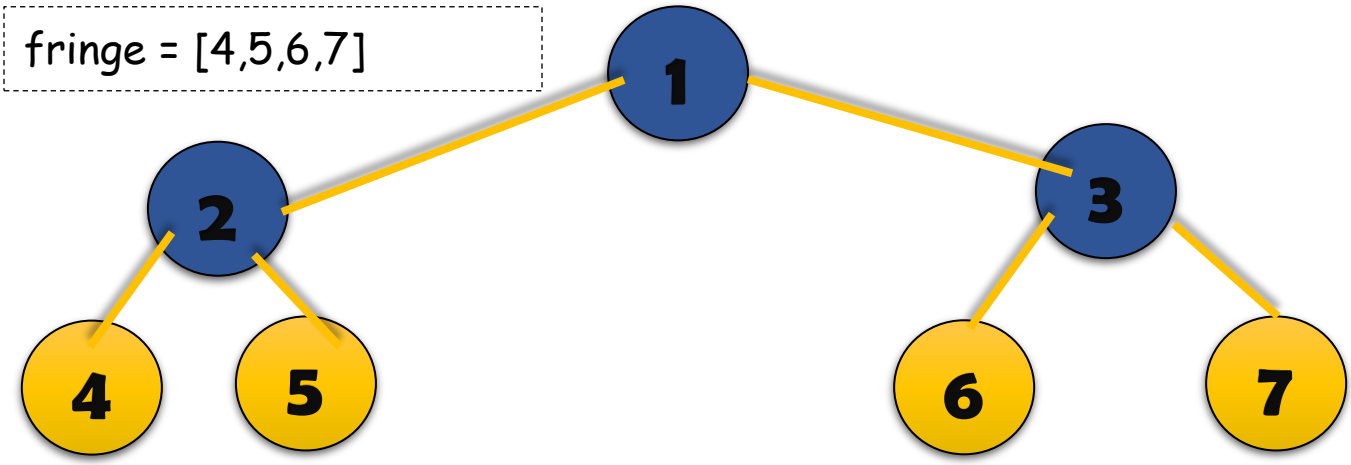
fringe = [2, 3]

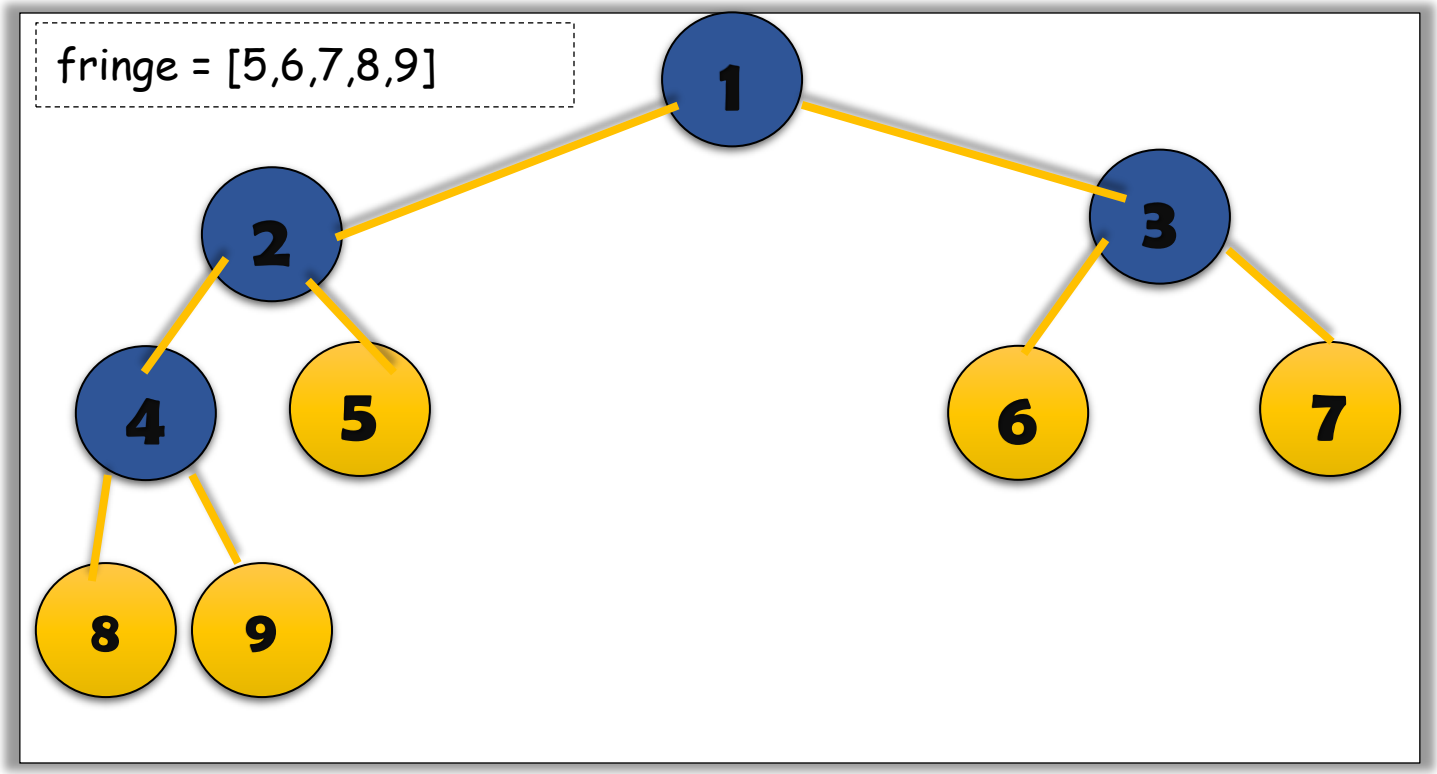


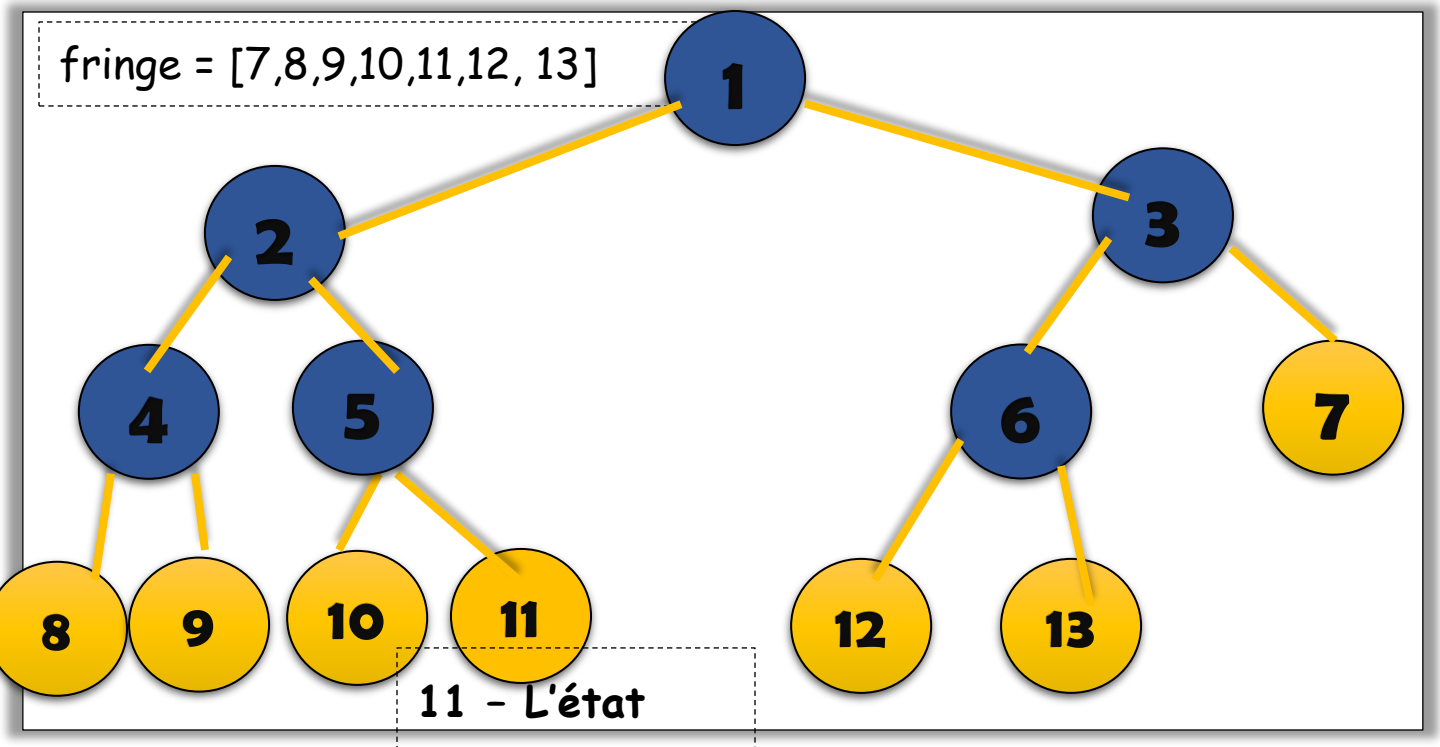
fringe = [3,4,5]



fringe = [4,5,6,7]

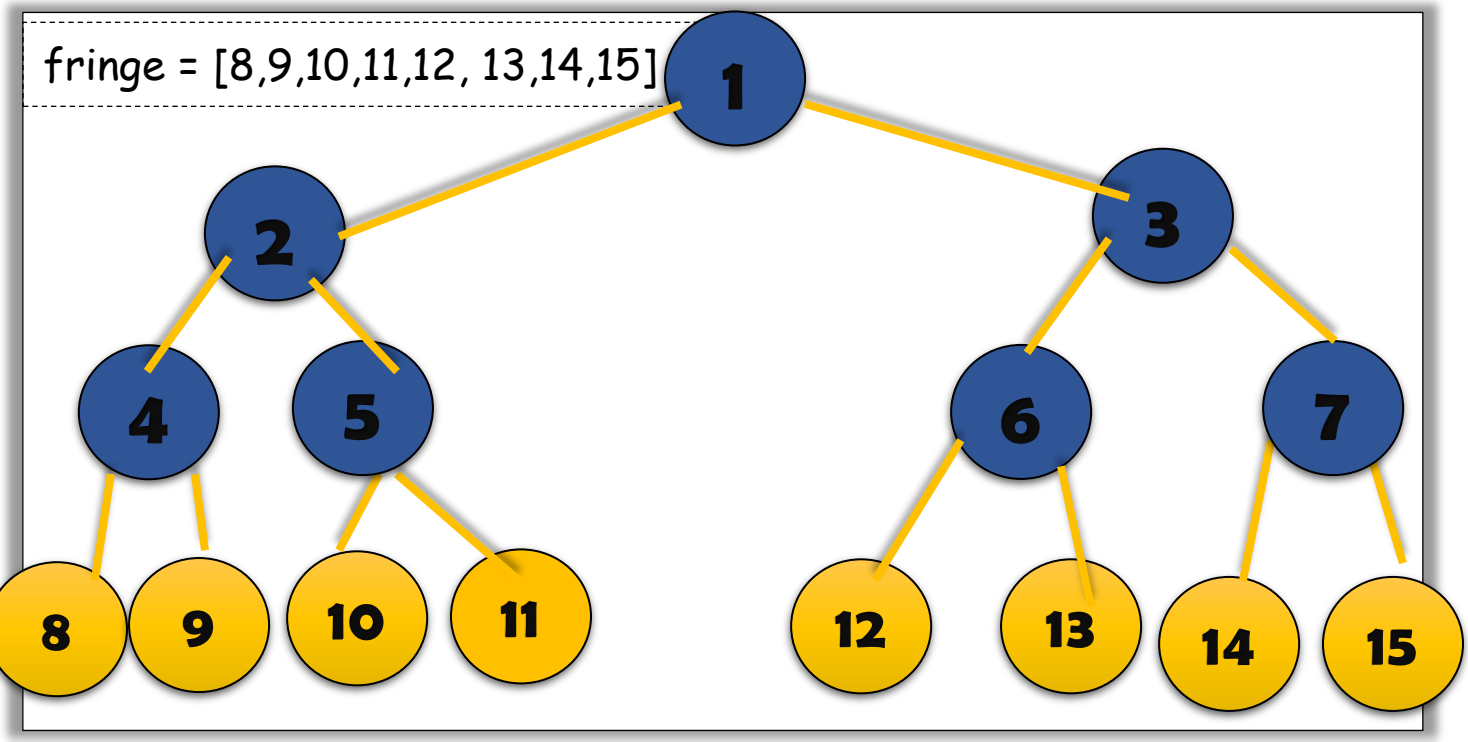


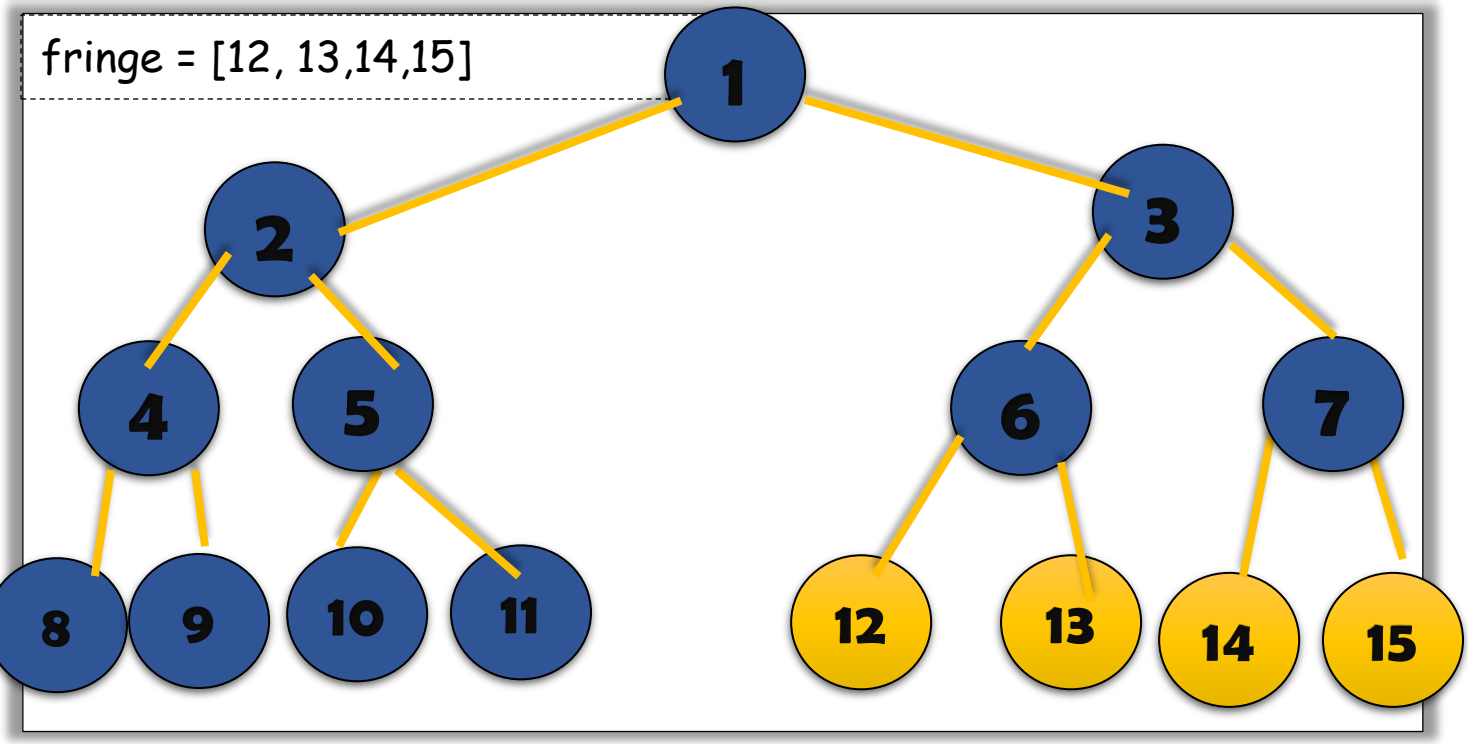




11 - L'état final

on n'a pas encore vérifier que 11 était bien l'état but donc je ne m'arrête pas parce que Je n'ai pas encore développé ces nœuds.





1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

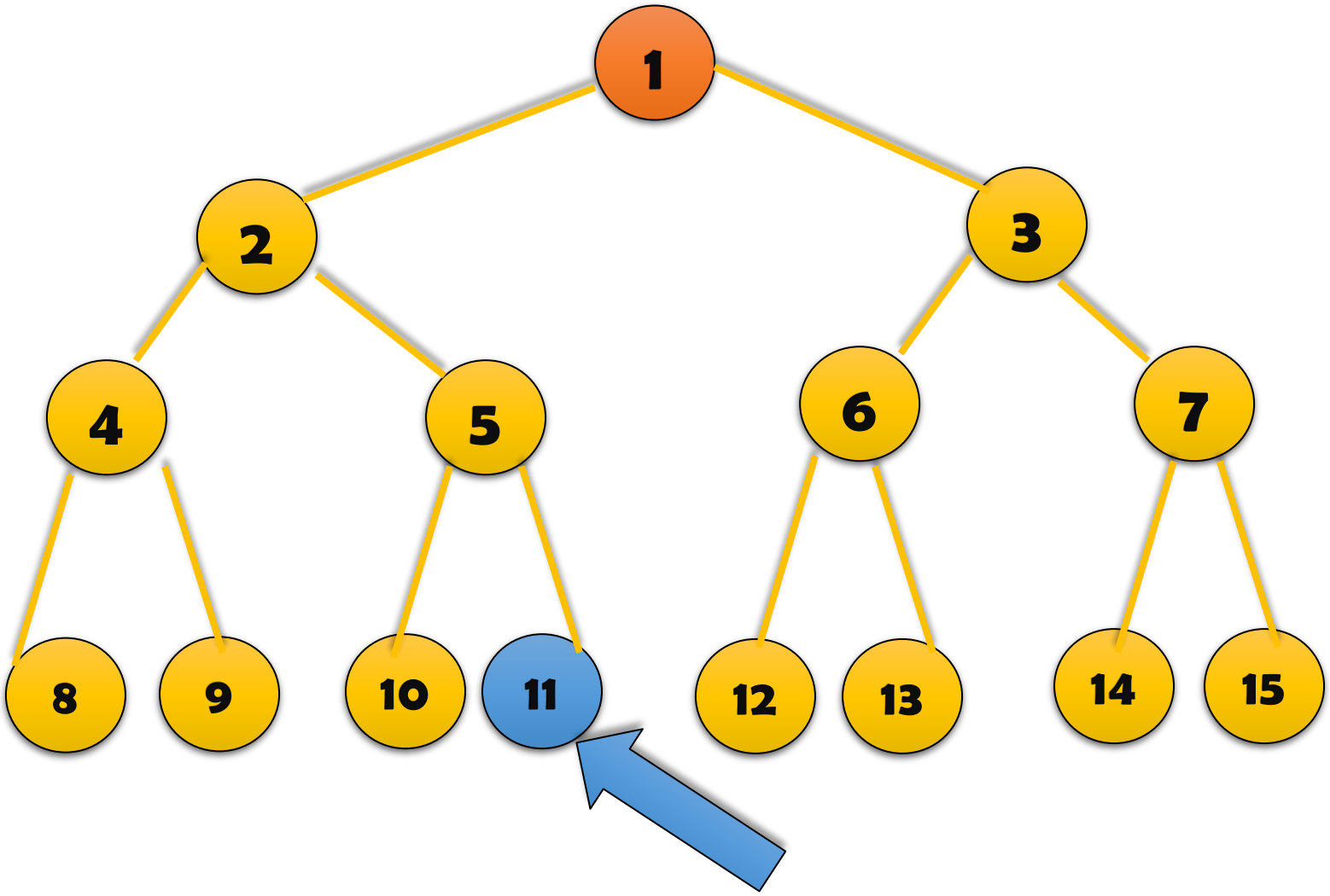
Question 2

Supposez que le but soit 11,

et que vous considérez l'espace de recherche complet.

Donner l'ordre de parcours des nœuds pour les algorithmes :

(b) profondeur d'abord



Recherche en profondeur d'abord (DFS)

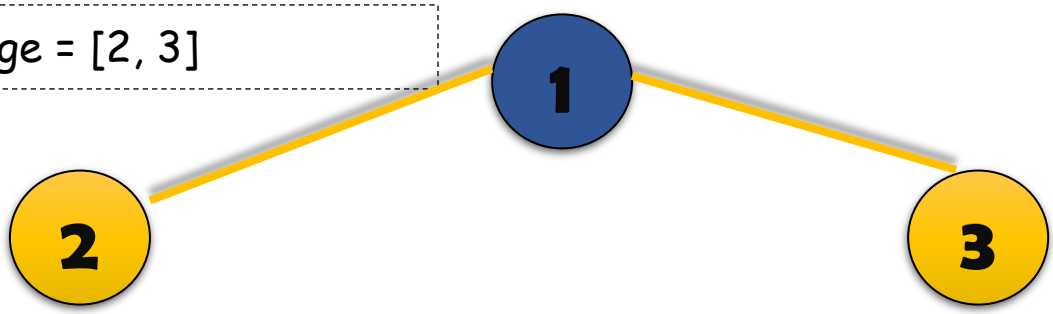
- `Insert-Fn` ajoute les successeurs en début de liste *(c'est une pile)*

fringe = [1]

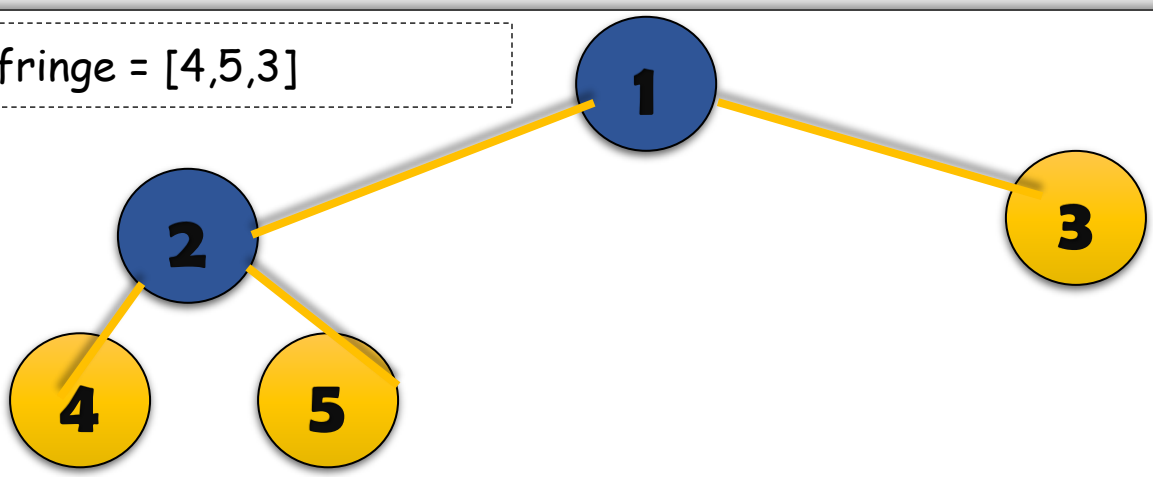
Une pile qui représente l'ensemble des nœud à développer

L'état initial est 1

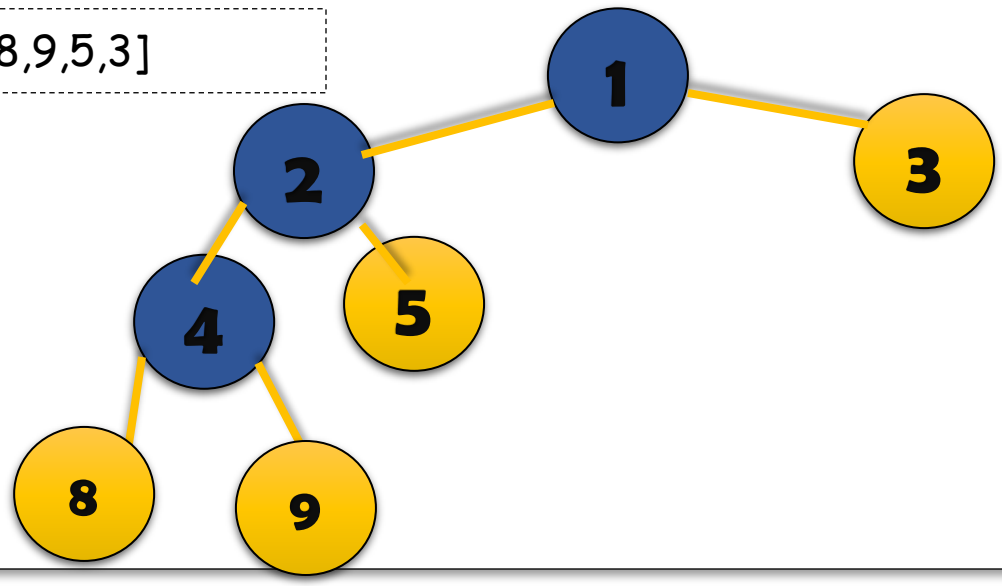
fringe = [2, 3]



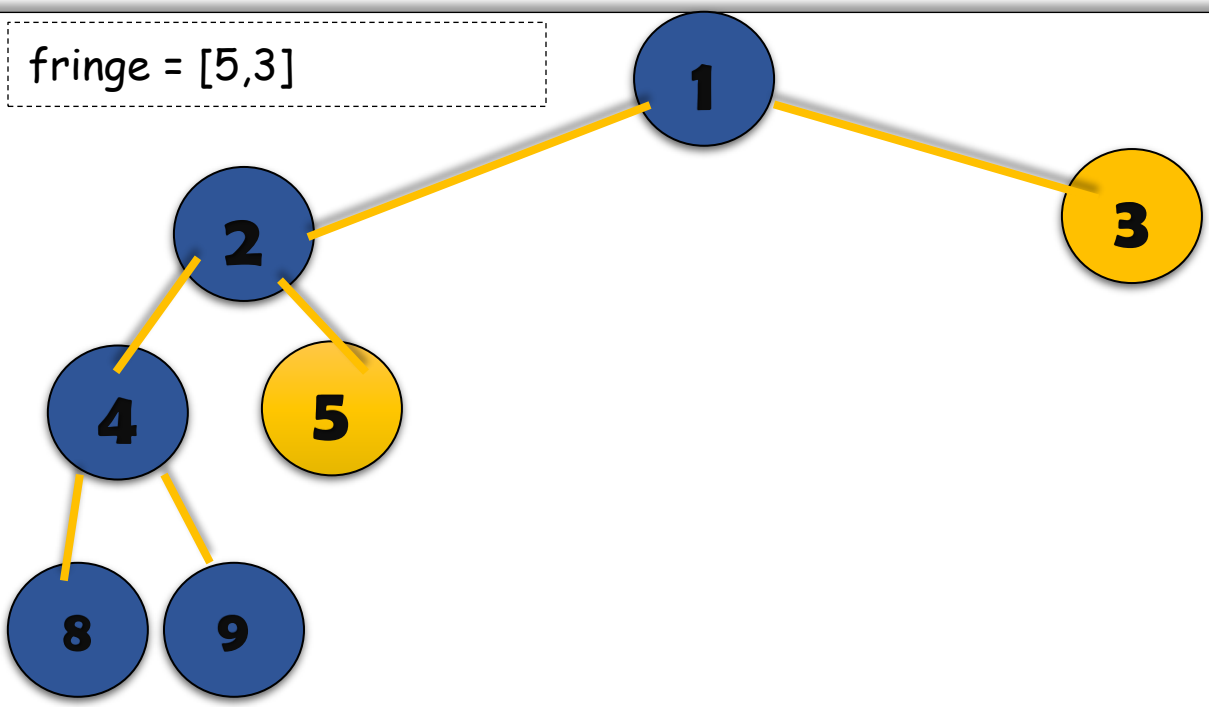
fringe = [4,5,3]



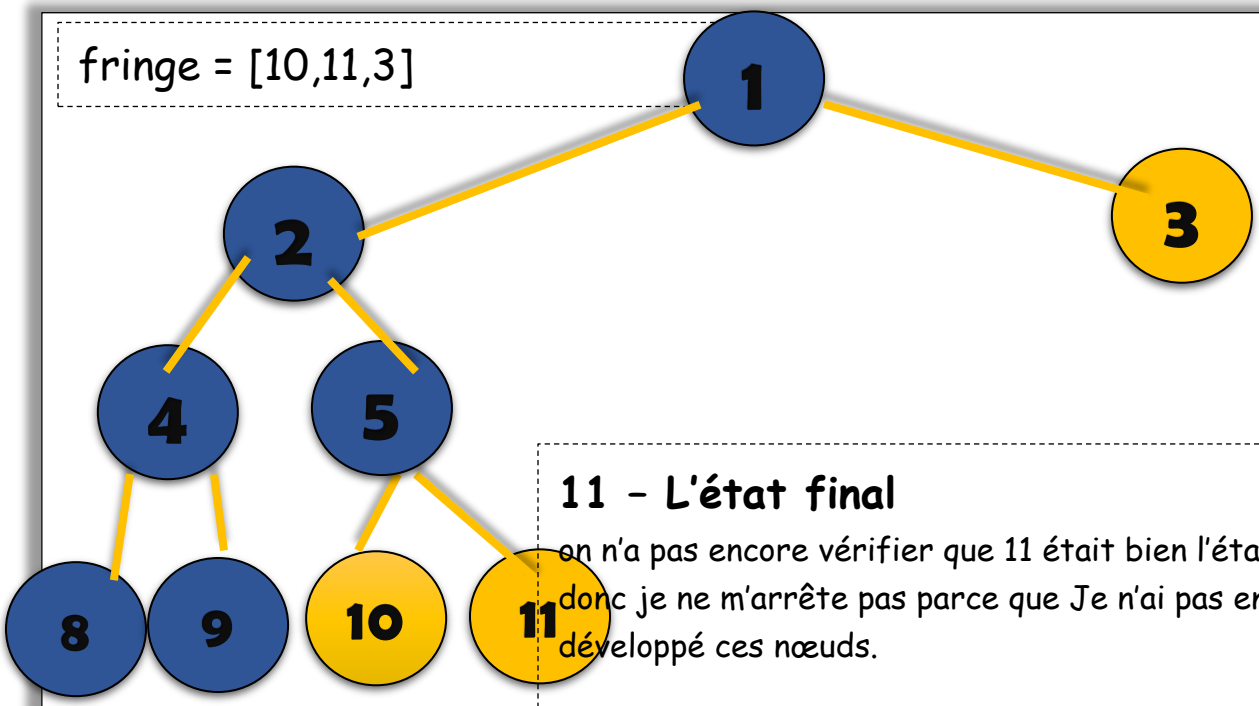
fringe = [8,9,5,3]



fringe = [5,3]

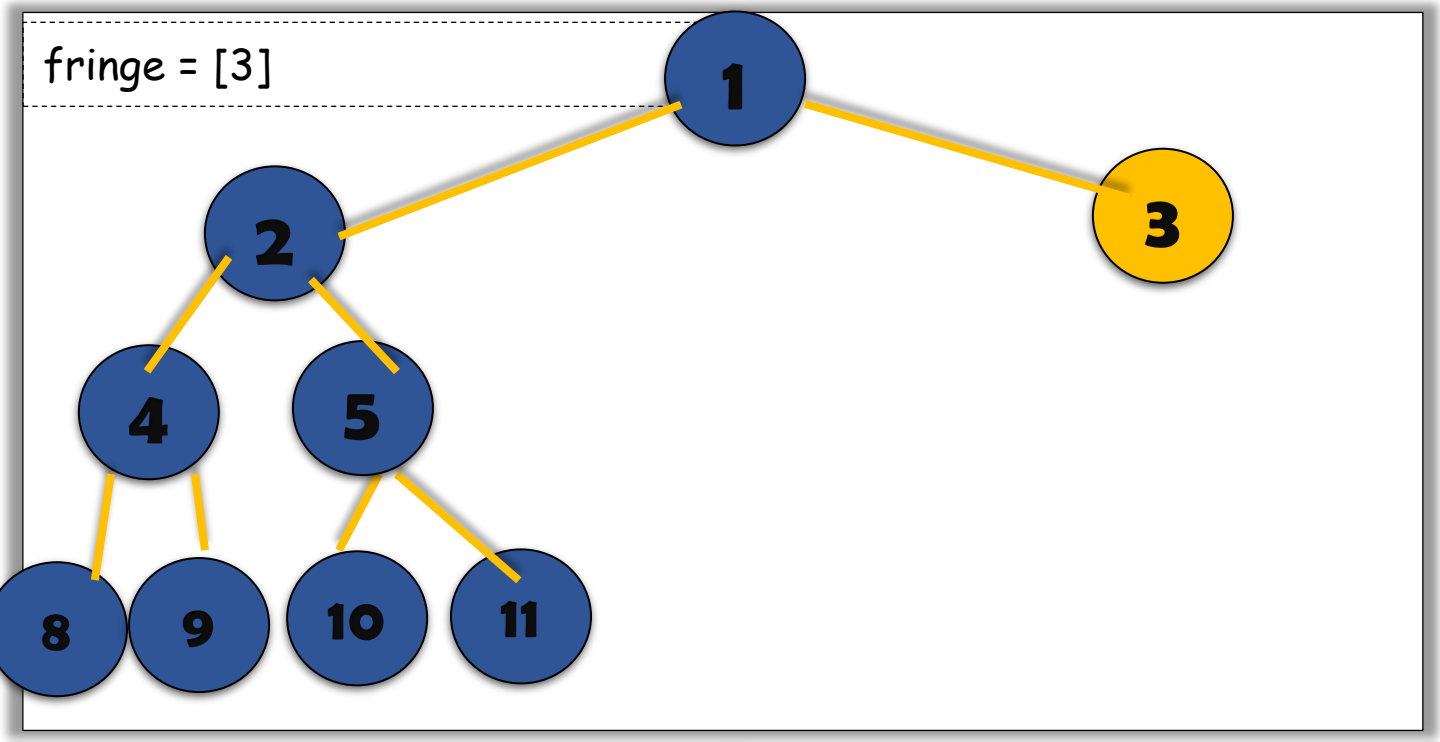


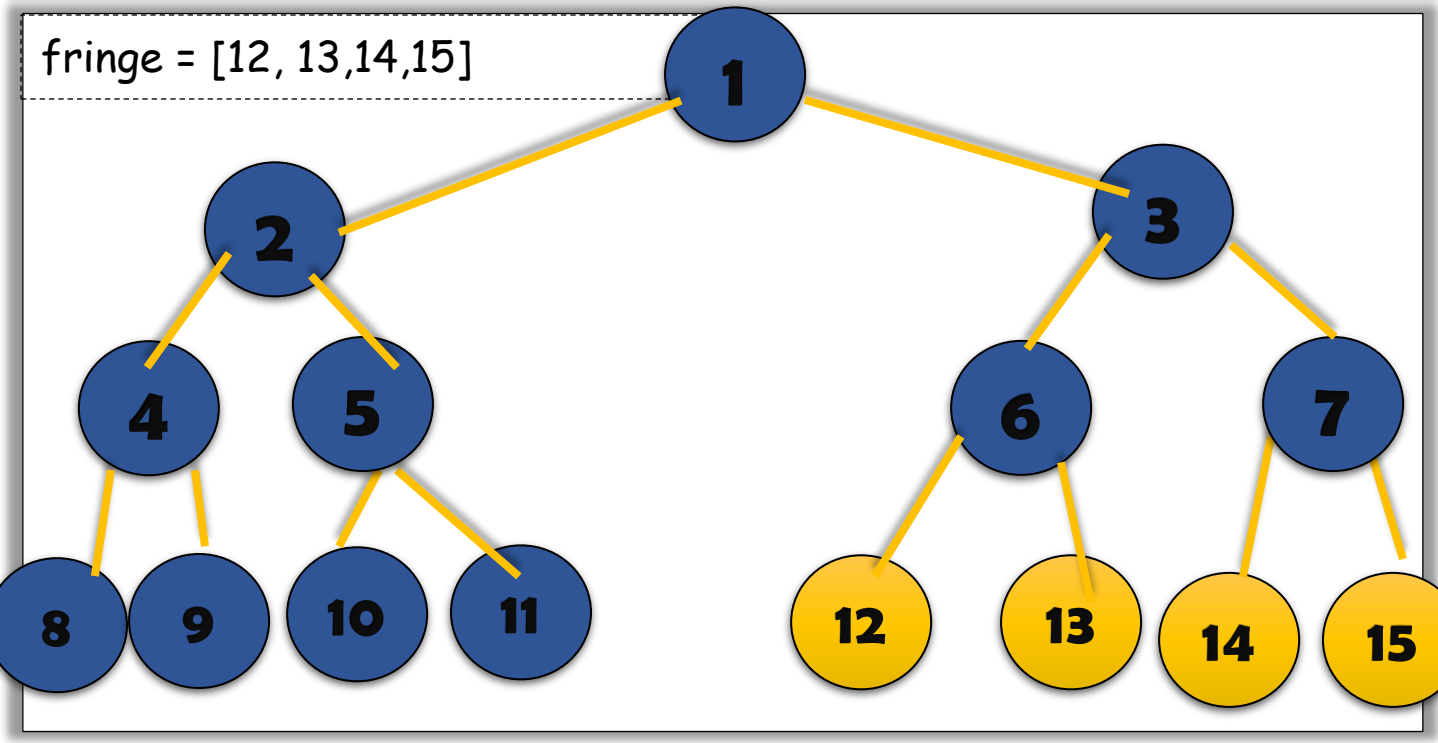
fringe = [10,11,3]



11 – L'état final

on n'a pas encore vérifier que 11 était bien l'état but donc je ne m'arrête pas parce que Je n'ai pas encore développé ces nœuds.


fringe = [3]
1
2
3
4
5
8
9
10
11


fringe = [12, 13,14,15]
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

1,2,4,8,9,5,10,11

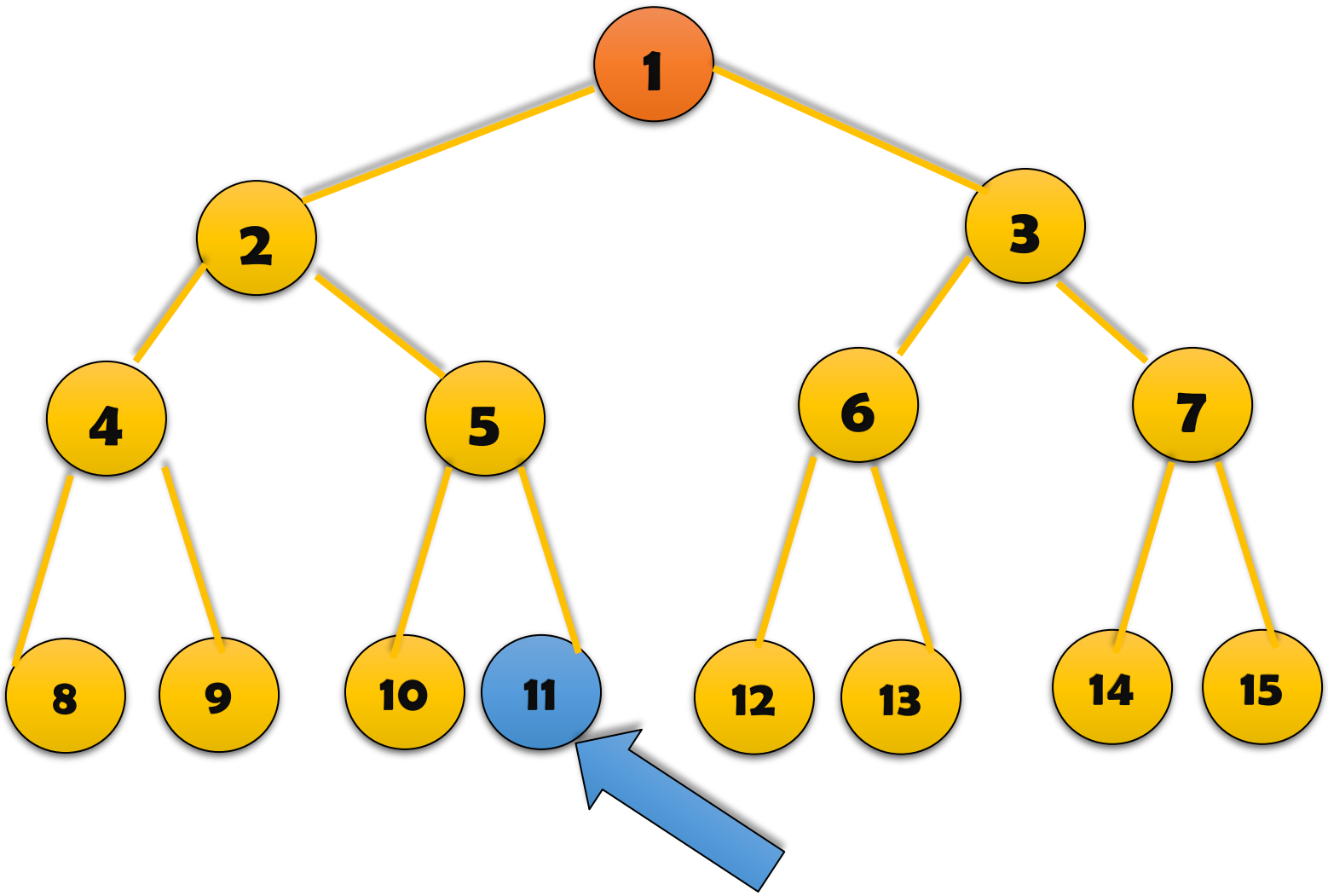
Question 3

Supposez que le but soit 11,

et que vous considérez l'espace de recherche complet.

Donner l'ordre de parcours des nœuds pour les algorithmes :

(c) profondeur itérative



Recherche en profondeur itérative

- Profondeur limitée, mais en essayant toutes les profondeurs : 0, 1, 2, 3, ...
- Evite le problème de trouver une limite pour la recherche profondeur limitée
- Combine les avantages de largeur d'abord (complète et optimale), mais a la complexité en espace de profondeur d'abord

Si le coup des actions est 1.

```
function ITERATIVE-DEEPENING-SEARCH( problem) returns a solution, or failure
    inputs: problem, a problem
    for depth ← 0 to ∞ do
        result ← DEPTH-LIMITED-SEARCH( problem, depth)
        if result ≠ cutoff then return result
```

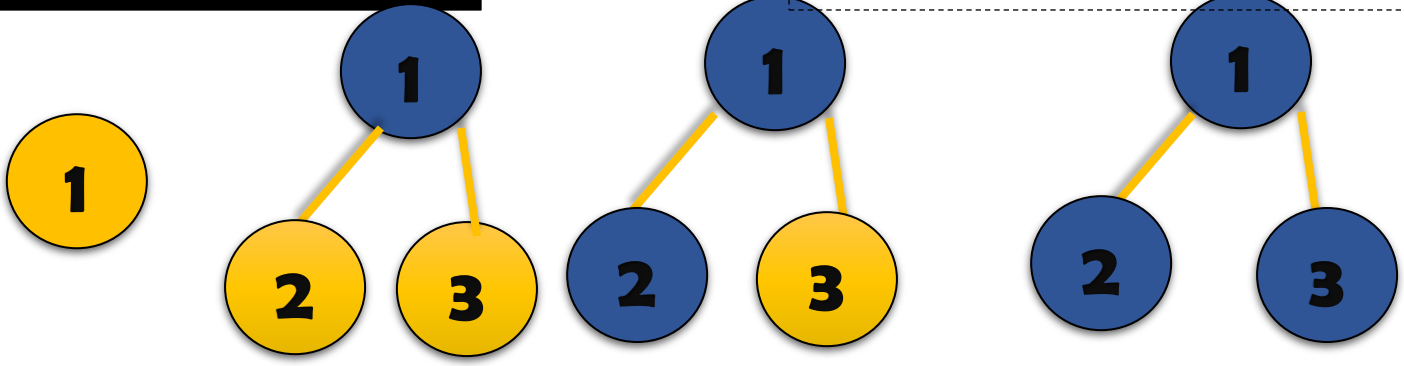
Recherche en profondeur itérative : l = 0

Nœuds développes : [1]

Recherche en profondeur itérative : l = 1

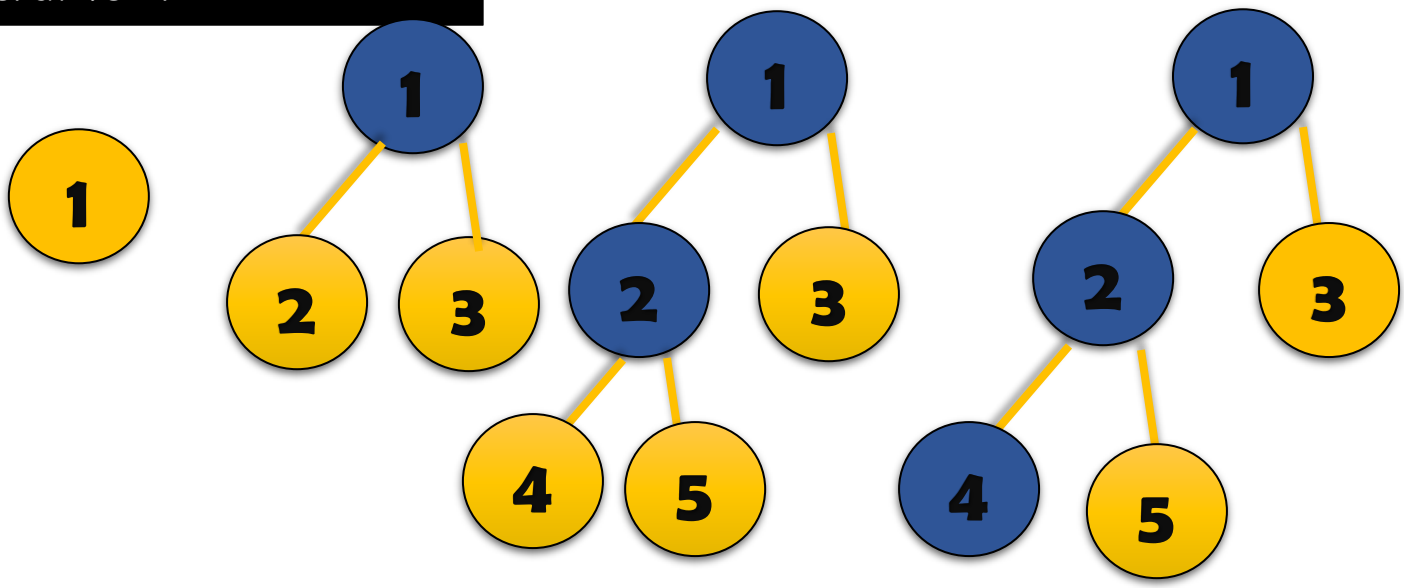
Nœuds développes : [1 ;1,2,3]

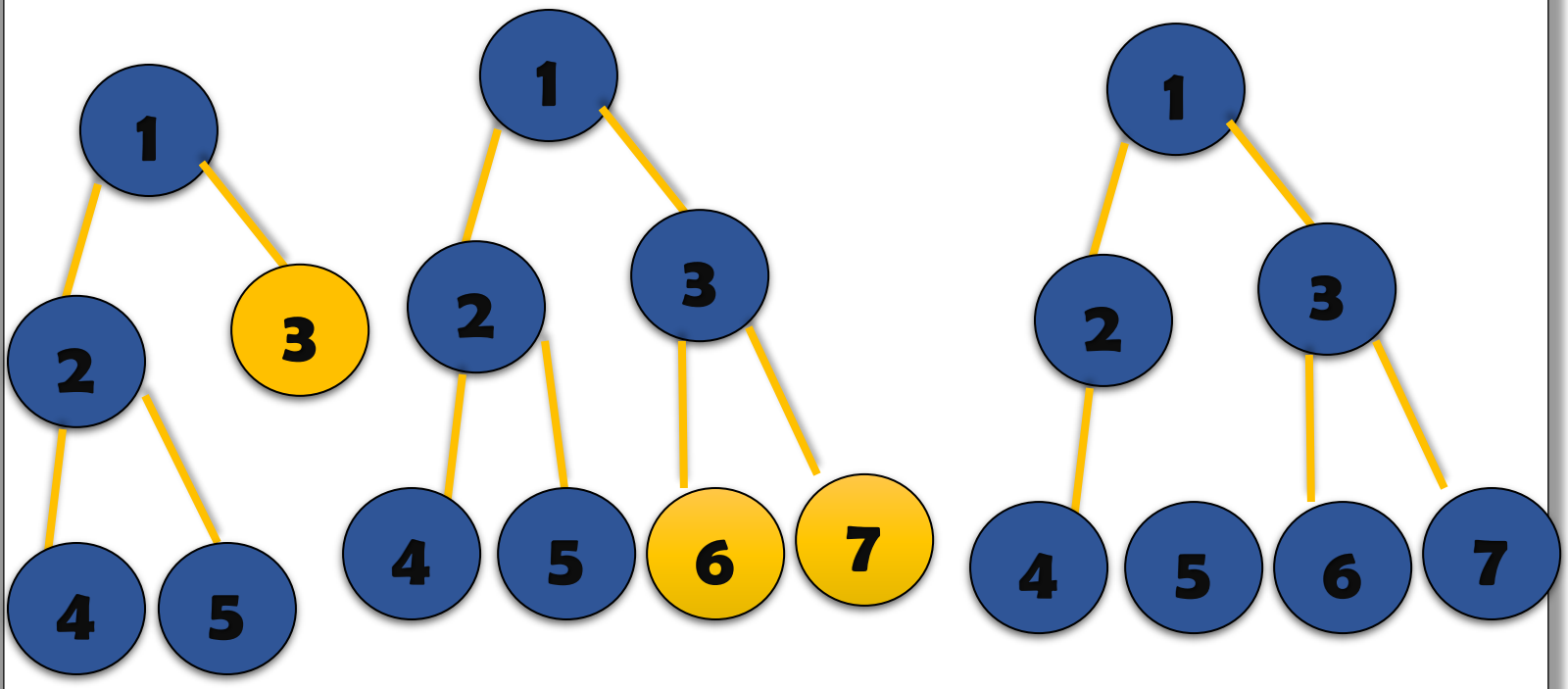
On a redéveloppé 1



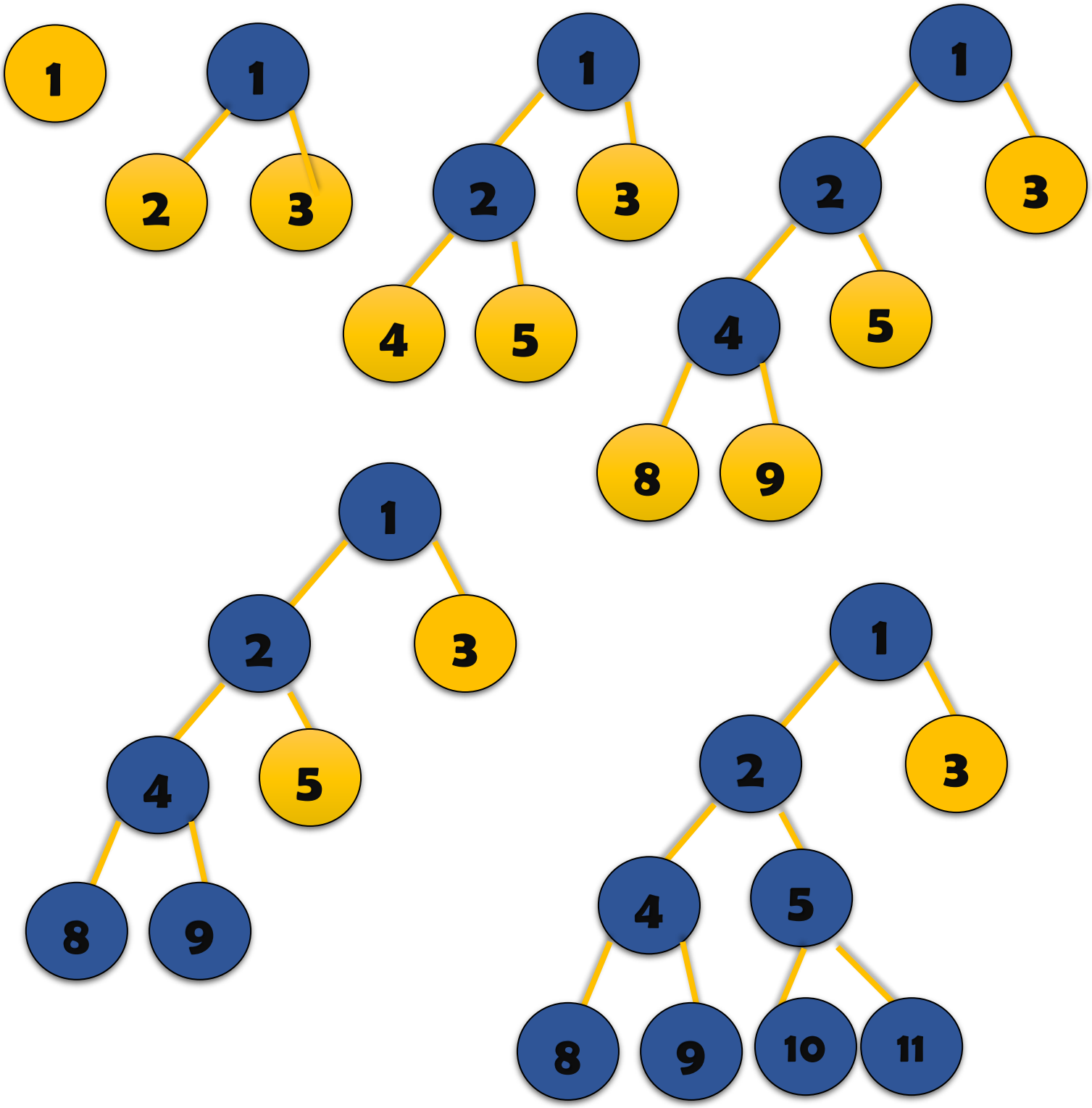
Recherche en profondeur itérative : l = 2

Nœuds développes : [1 ;1,2,3 ; 1,2,4,5,3,6,7]





Recherche en profondeur itérative : l = 3



Nœuds développes : [1 ;1,2,3 ; 1,2,4,5,3,6,7 ; 1, 2, 4, 8 , 9 , 5 , 10 ,11]

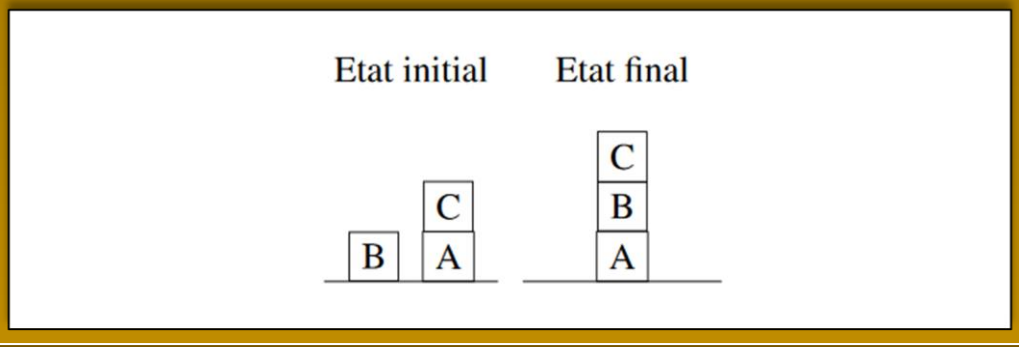
$$\underbrace{1}_{l=0} ; \underbrace{1,2,3}_{l=1} ; \underbrace{1,2,4,5,3,6,7}_{l=2} ; \underbrace{1,2,4,8,9,5,10,11}_{l=3}$$

Exercice 4

On est dans la situation initiale donnée sur la figure suivante.

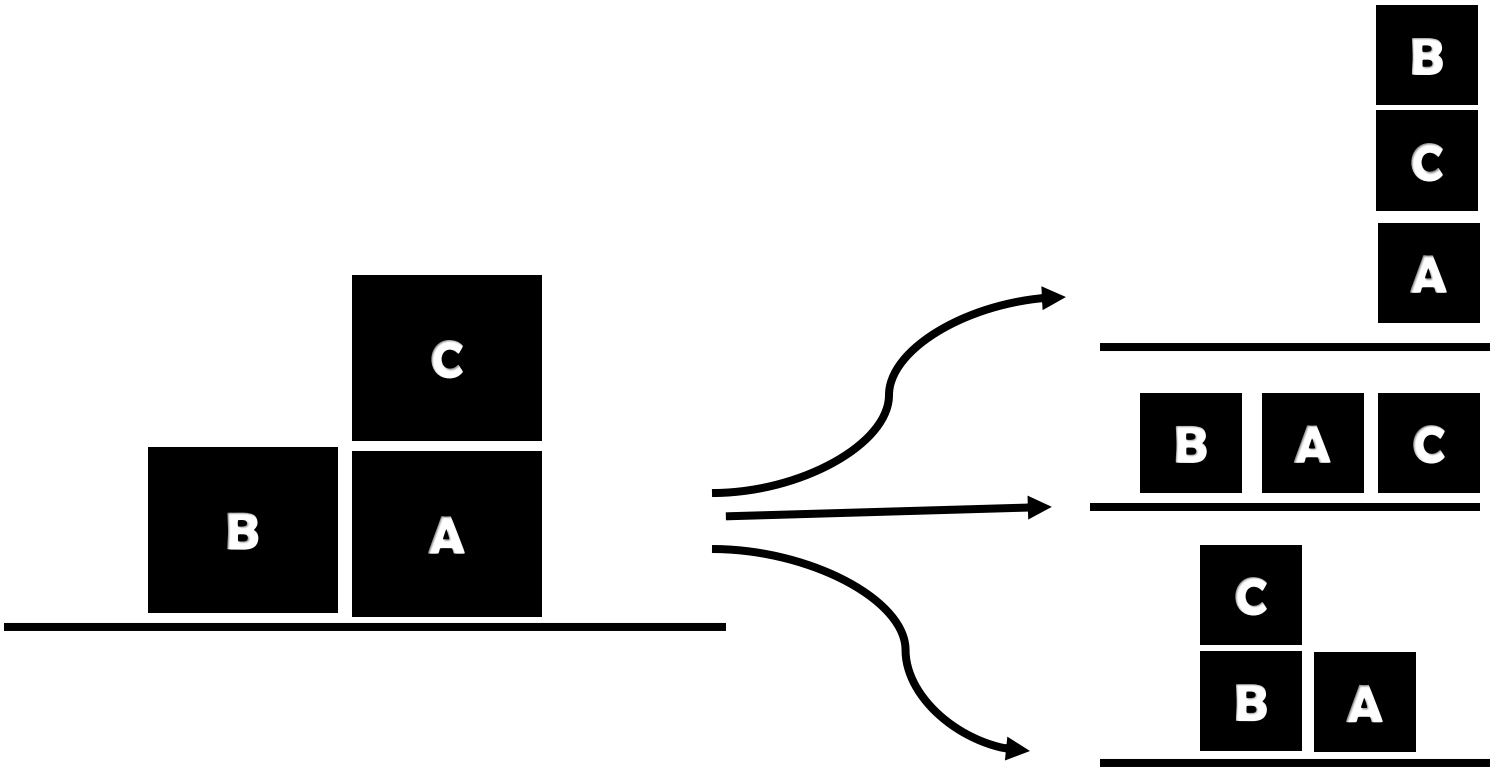
Sur une table sont posés trois cubes, les cubes A et B à même la table et le cube C sur le cube A. On suppose que la position des cubes sur la table les uns par rapport aux autres et indifférencié (que B soit à droite ou à gauche de A ne fait aucune différence).

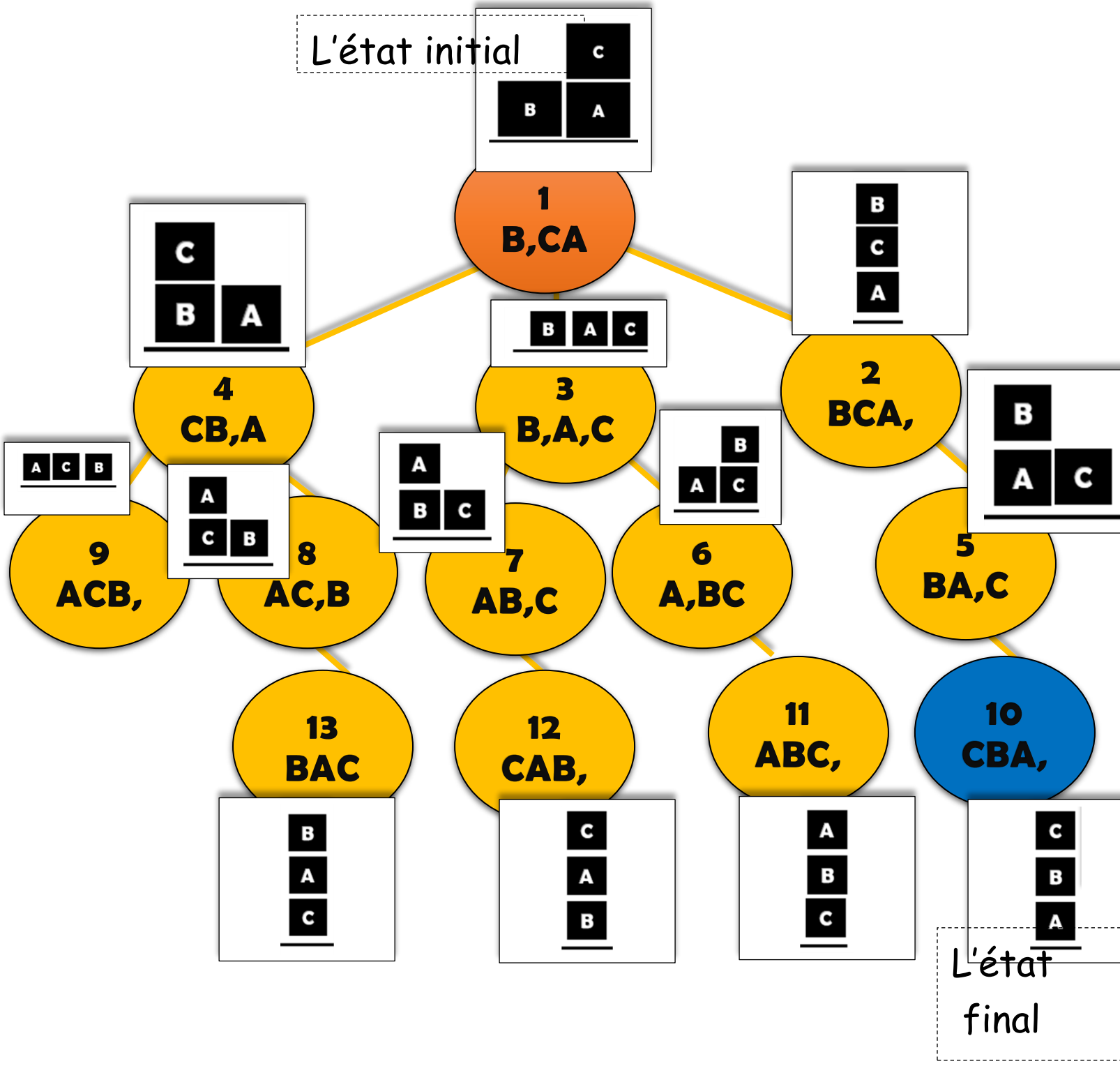
On veut atteindre l'état final. On a juste le droit de soulever un cube qui n'est pas recouvert par un autre cube et de le reposer ailleurs.



Question 1

Dessiner l'espace de recherche de ce problème.



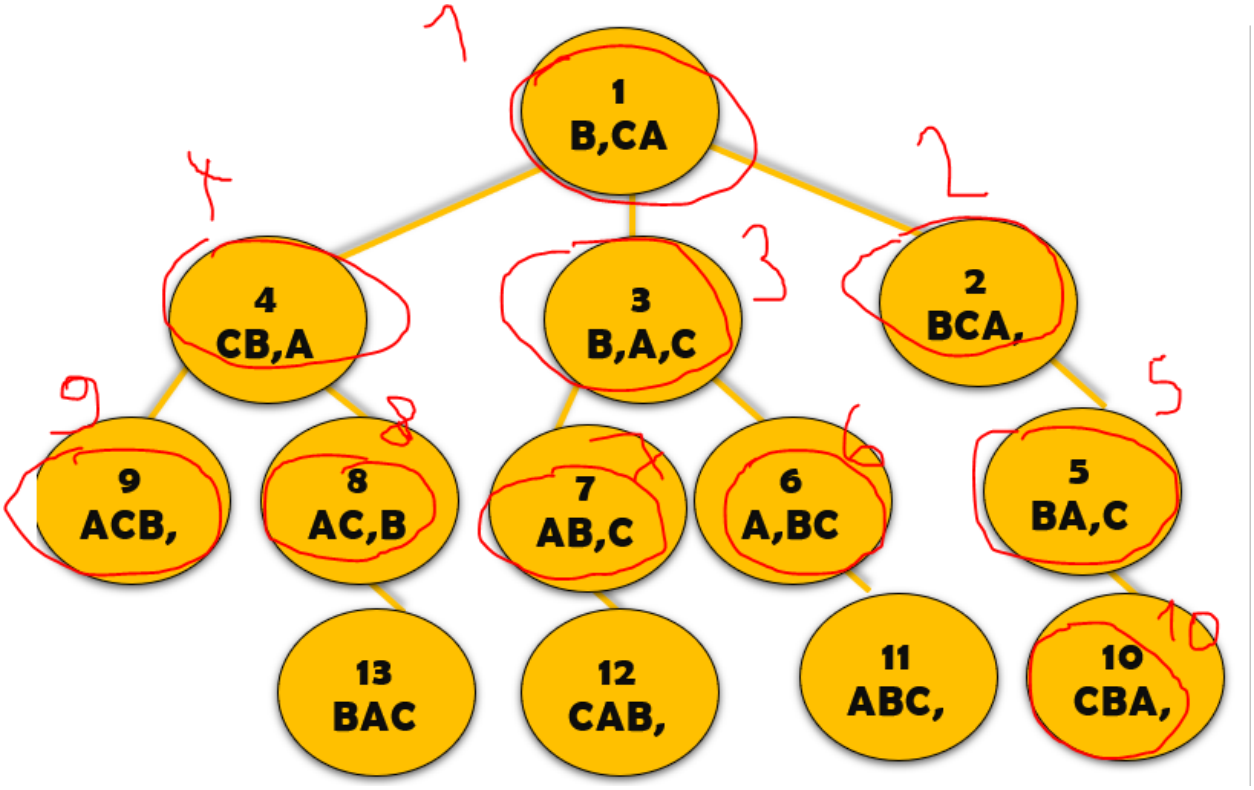

L'état initial
C
B A
1
B,CA
C
B A
4
CB,A
B A C
3
B,A,C
B
C
A
2
BCA,
A C B
9
ACB,
A
C B
8
AC,B
A
B C
7
AB,C
B
A C
6
A,BC
B
A C
5
BA,C
B
A
C
13
BAC
C
A
B
12
CAB,
A
B
C
11
ABC,
C
B
A
10
CBA,
L'état final

Exercice 4

Question 2

Donner l'ordre de parcours des nœuds pour les algorithmes (on suppose que nous pouvons éviter les répétitions : un état contenu dans un nœud déjà développé ne le sera plus) :

(a) largeur d'abord



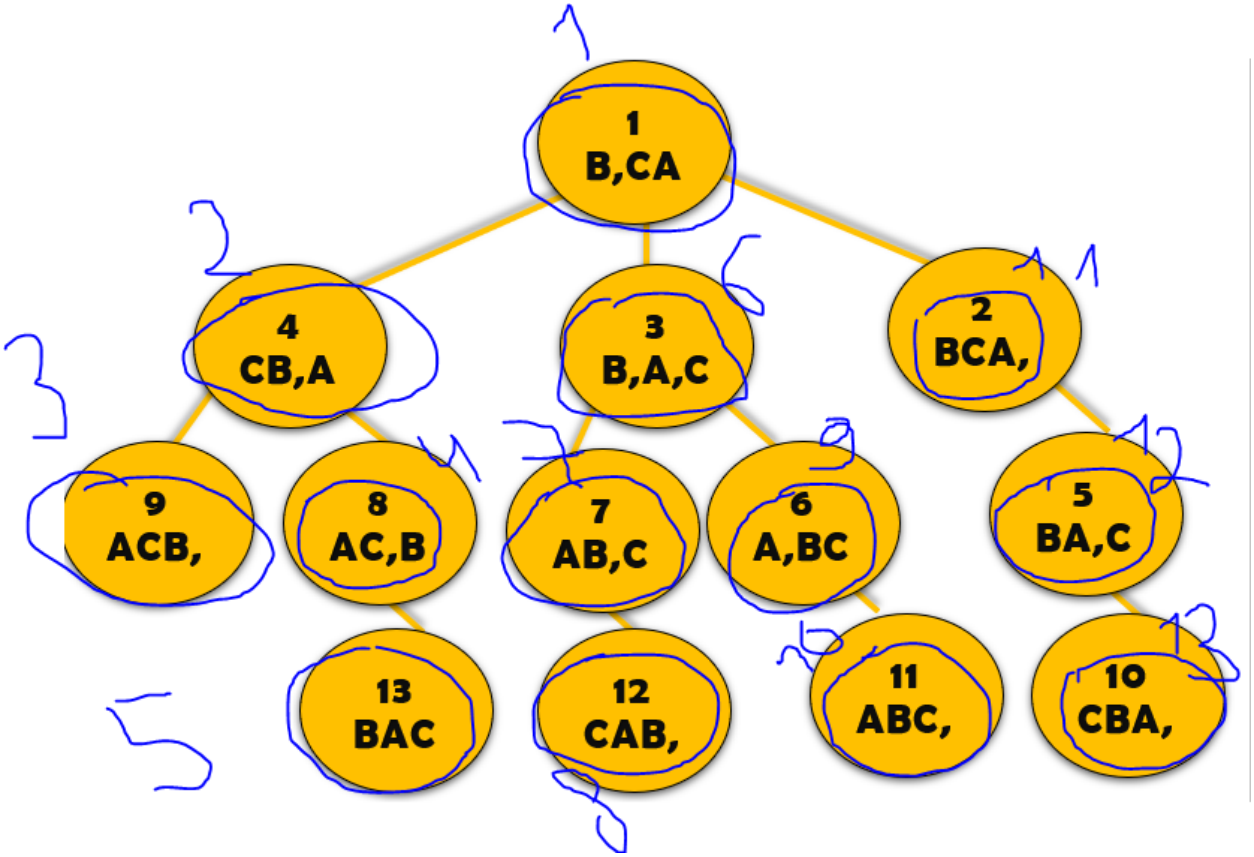
1,2,3,4,5,6,7,8,9,10

Exercice 4

Question 2

Donner l'ordre de parcours des nœuds pour les algorithmes (on suppose que nous pouvons éviter les répétitions : un état contenu dans un nœud déjà développé ne le sera plus) :

(b) profondeur d'abord



1,4,9,8,13,3,7,12,6,11,2,5,10 [?]

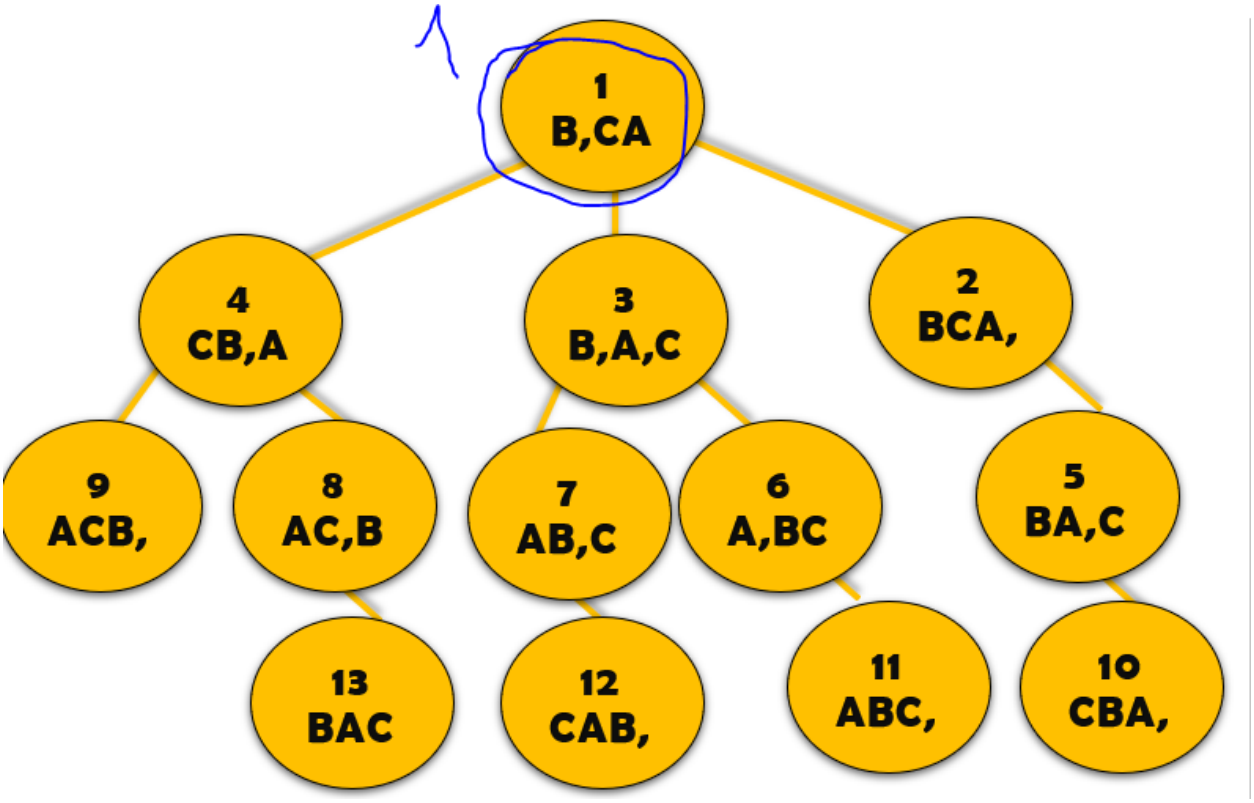
Exercice 4

Question 2

Donner l'ordre de parcours des nœuds pour les algorithmes (on suppose que nous pouvons éviter les répétitions : un état contenu dans un nœud déjà développé ne le sera plus) :

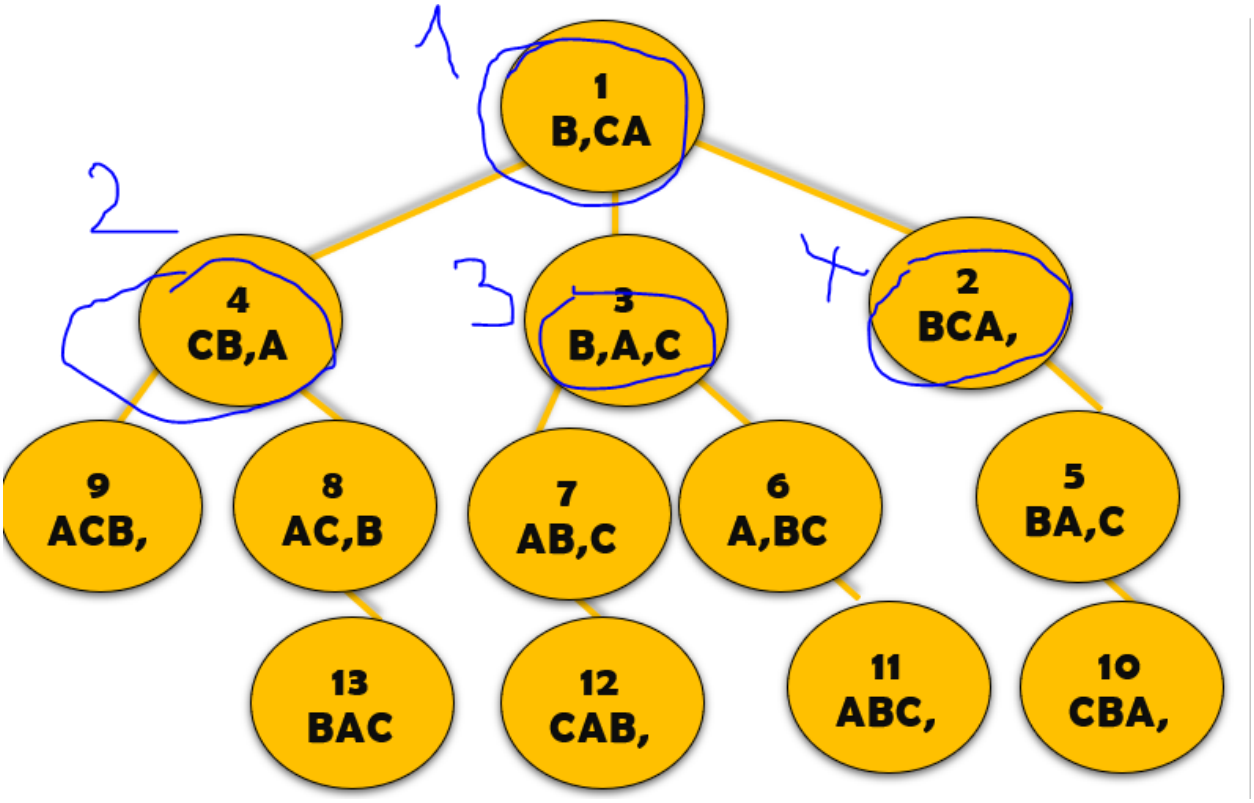
(c) profondeur itérative

$$l = 0$$



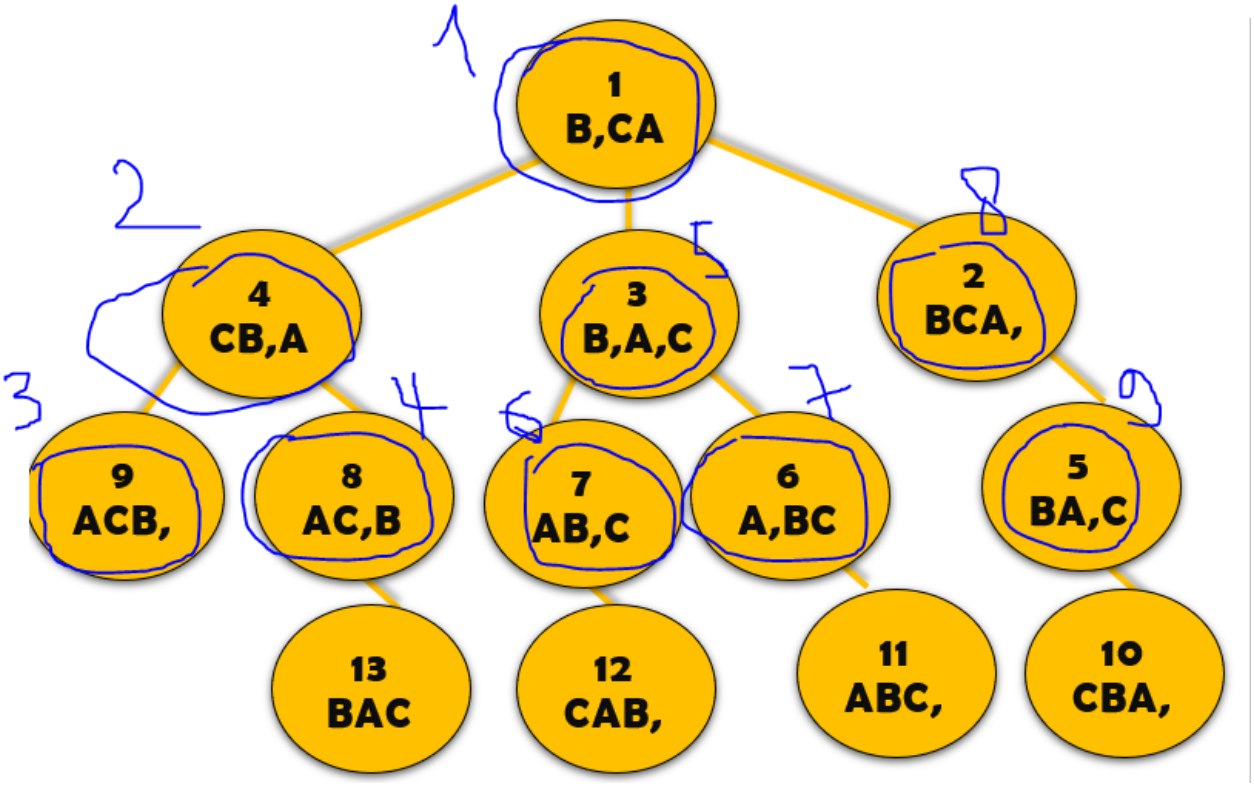
1

$l = 1$


1
B,CA
4
CB,A
3
B,A,C
2
BCA,
9
ACB,
8
AC,B
7
AB,C
6
A,BC
5
BA,C
13
BAC
12
CAB,
11
ABC,
10
CBA,

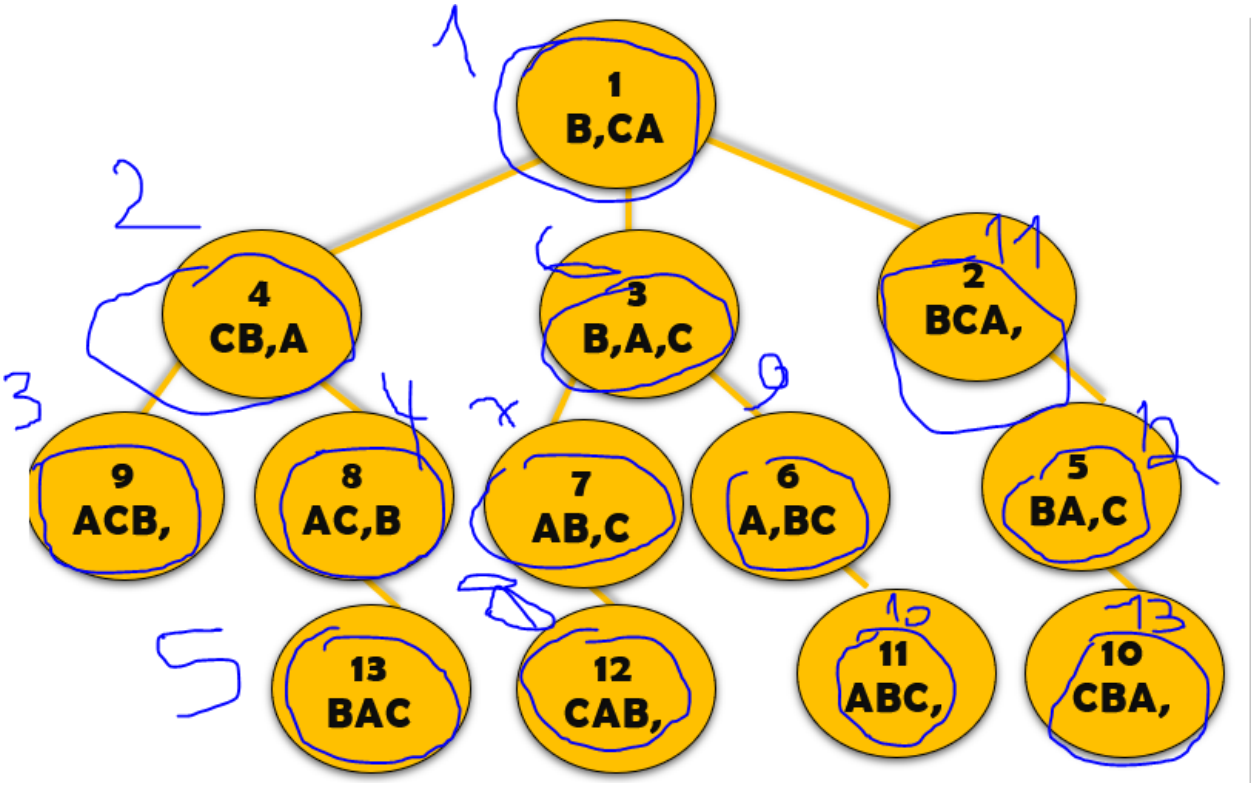
1,4,3,2

$l = 2$



1,4,9,8,3,7,6,2,5 [?]

$l = 3$



1,4,9,8,13,3,7,12,6,11,2,5,10 [?]

$$\underbrace{1}_{l=0} ; \underbrace{1,4,3,2}_{l=1} ; \underbrace{1,4,9,8,3,7,6,2,5}_{l=2}$$

$$; \underbrace{1,4,9,8,13,3,7,12,6,11,2,5,10}_{l=3}$$

a

f