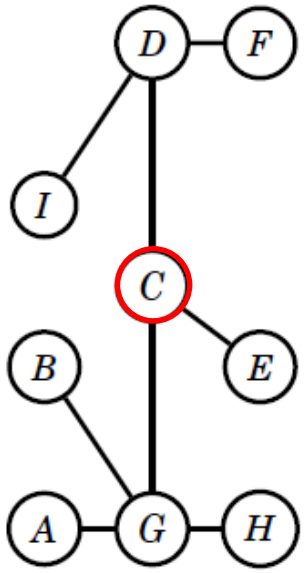
MPII
ITC
Séquence 11

GRAPHES (2)
Le parcours en largeur
Breadth-First Search (BFS)

0. RAPPEL NOTION D'ARBRE

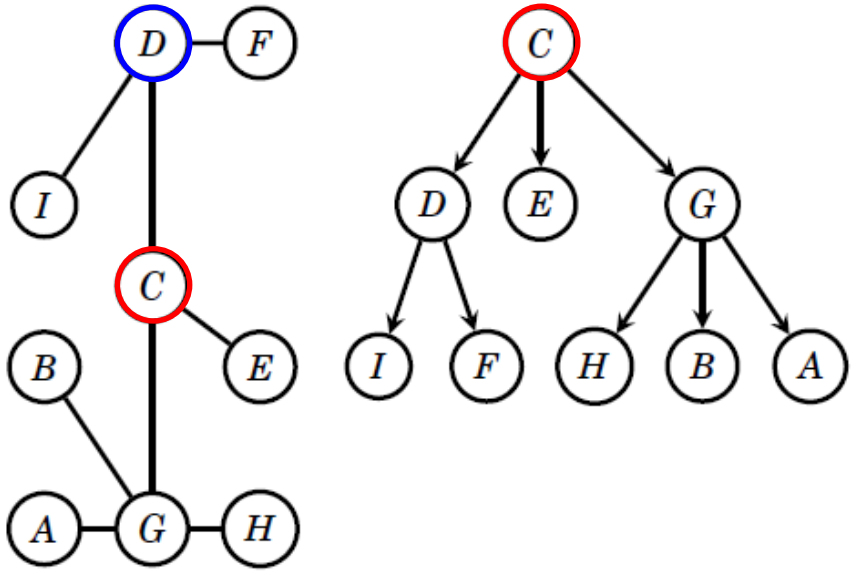
Les arbres sont un cas particulier de graphes : ce sont des **graphes simples non orientés connexes et acycliques**.
On peut **enraciner** l'arbre en choisissant un sommet comme racine.



Arbre simple

0. RAPPEL NOTION D'ARBRE

Les arbres sont un cas particulier de graphes : ce sont des **graphes simples non orientés connexes et acycliques**.
On peut **enraciner** l'arbre en choisissant un sommet comme racine.

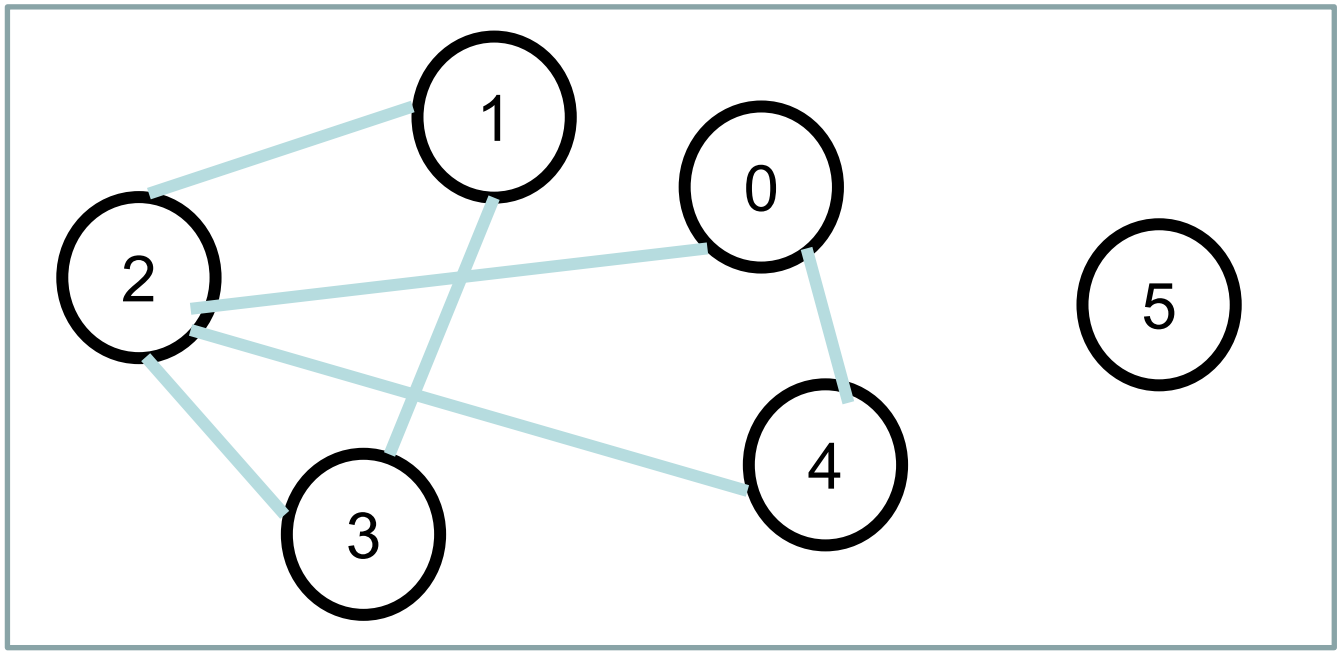


Arbre simple Arbre enraciné en C

Si le graphe n'est pas connexe mais reste acyclique, on parle de **forêt**, chaque composante connexe étant un arbre (= plusieurs arbres indépendants).

1. CONTEXTE DES PARCOURS DE GRAPHES

- **Parcourir un graphe**



➢ Visiter tous ses sommets et arêtes

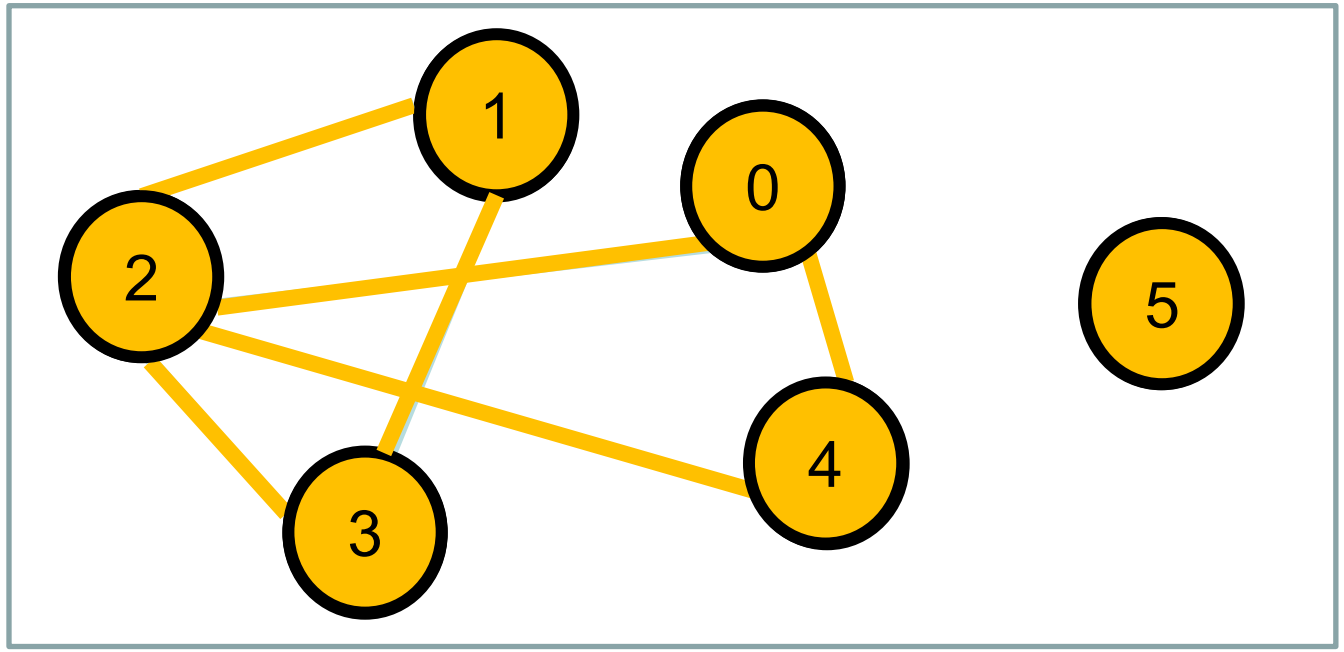
➢ En tirer des conclusions sur le graphe

- identification de composantes connexes;
- repérage de cycles ;
- construction d'un arbre couvrant;

☞ recherche d'un plus court chemin d'un sommet à un autre

1. CONTEXTE DES PARCOURS DE GRAPHES

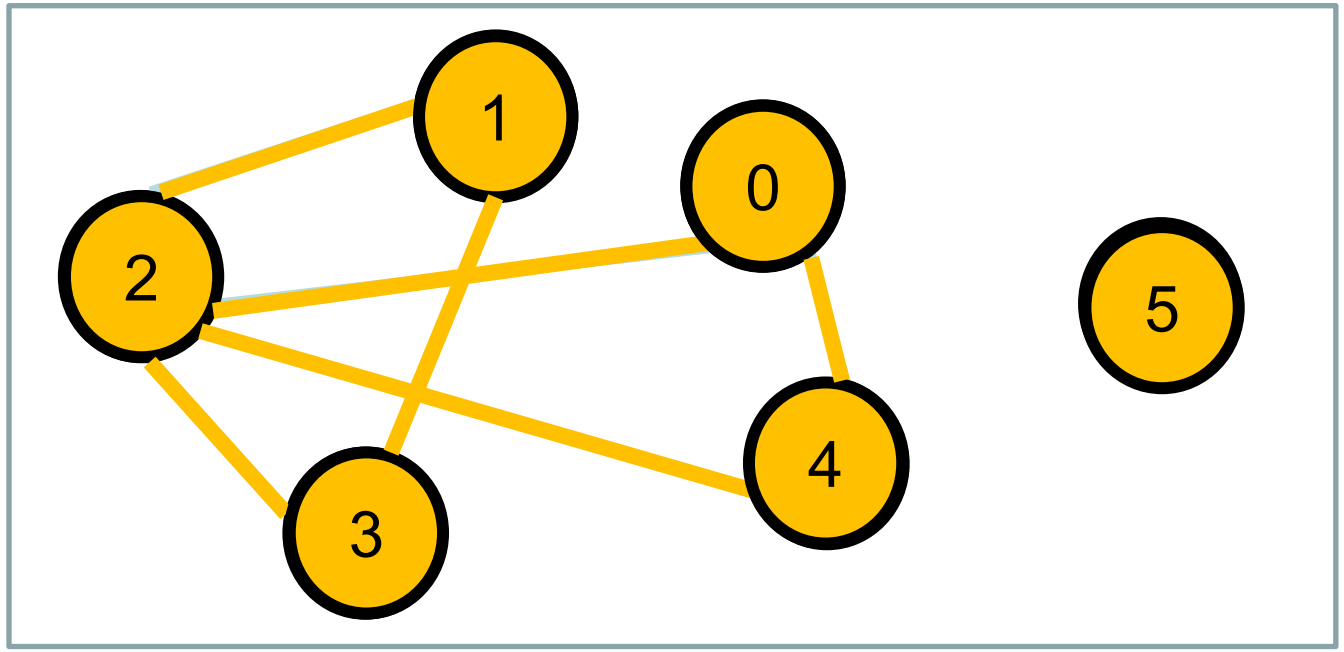
- **Comment parcourir un graphe**



➢ Au hasard

1. CONTEXTE DES PARCOURS DE GRAPHES

- **Comment parcourir un graphe**

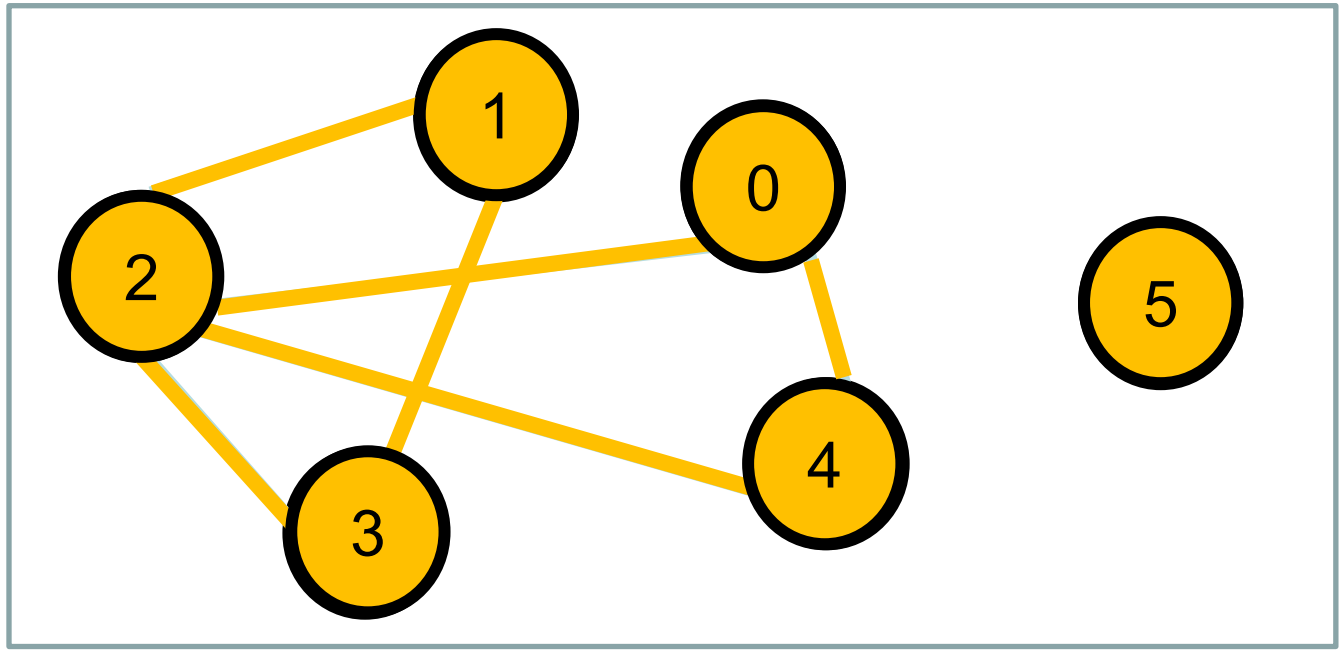


- Au hasard
- En se déployant régulièrement

Parcours en largeur

1. CONTEXTE DES PARCOURS DE GRAPHES

- **Comment parcourir un graphe**



- Au hasard
- En se déployant régulièrement

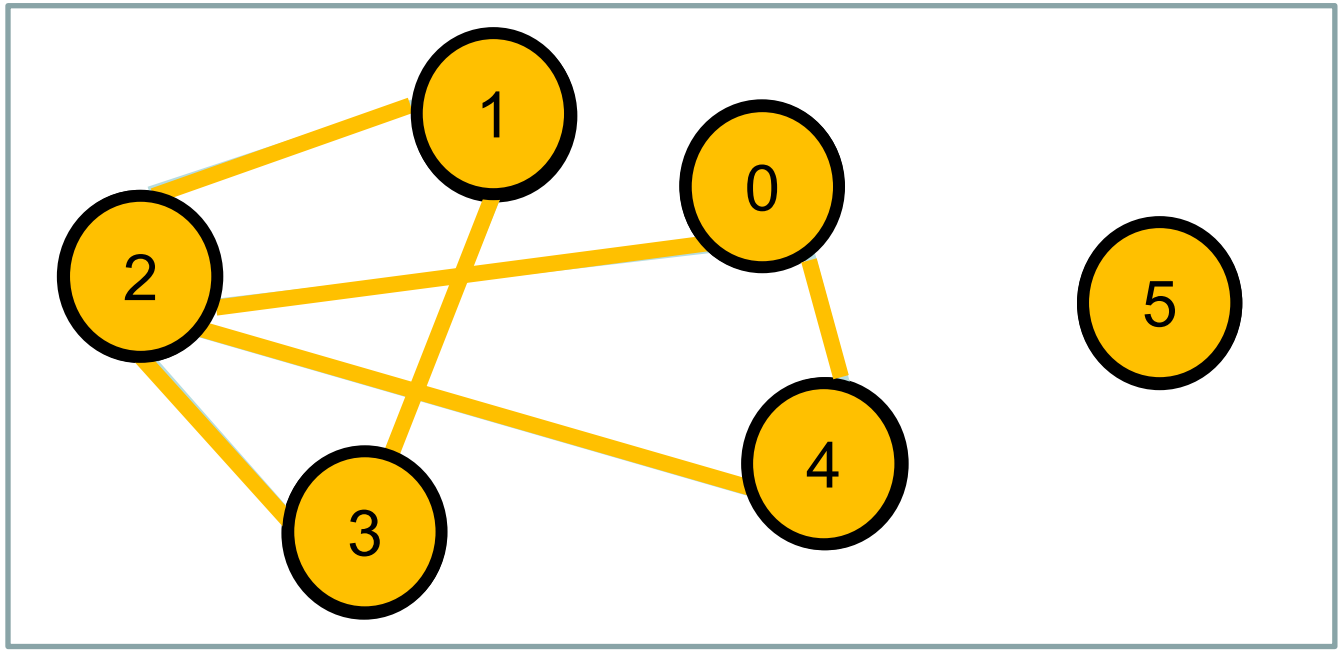
Parcours en largeur

- En fonçant par un chemin

Parcours en profondeur

1. CONTEXTE DES PARCOURS DE GRAPHES

- **Comment parcourir un graphe**



- Au hasard
- En se déployant régulièrement

  Parcours en largeur
- En fonçant par un chemin

  Parcours en profondeur
- En étudiant le chemin

  Algorithme de Dijkstra

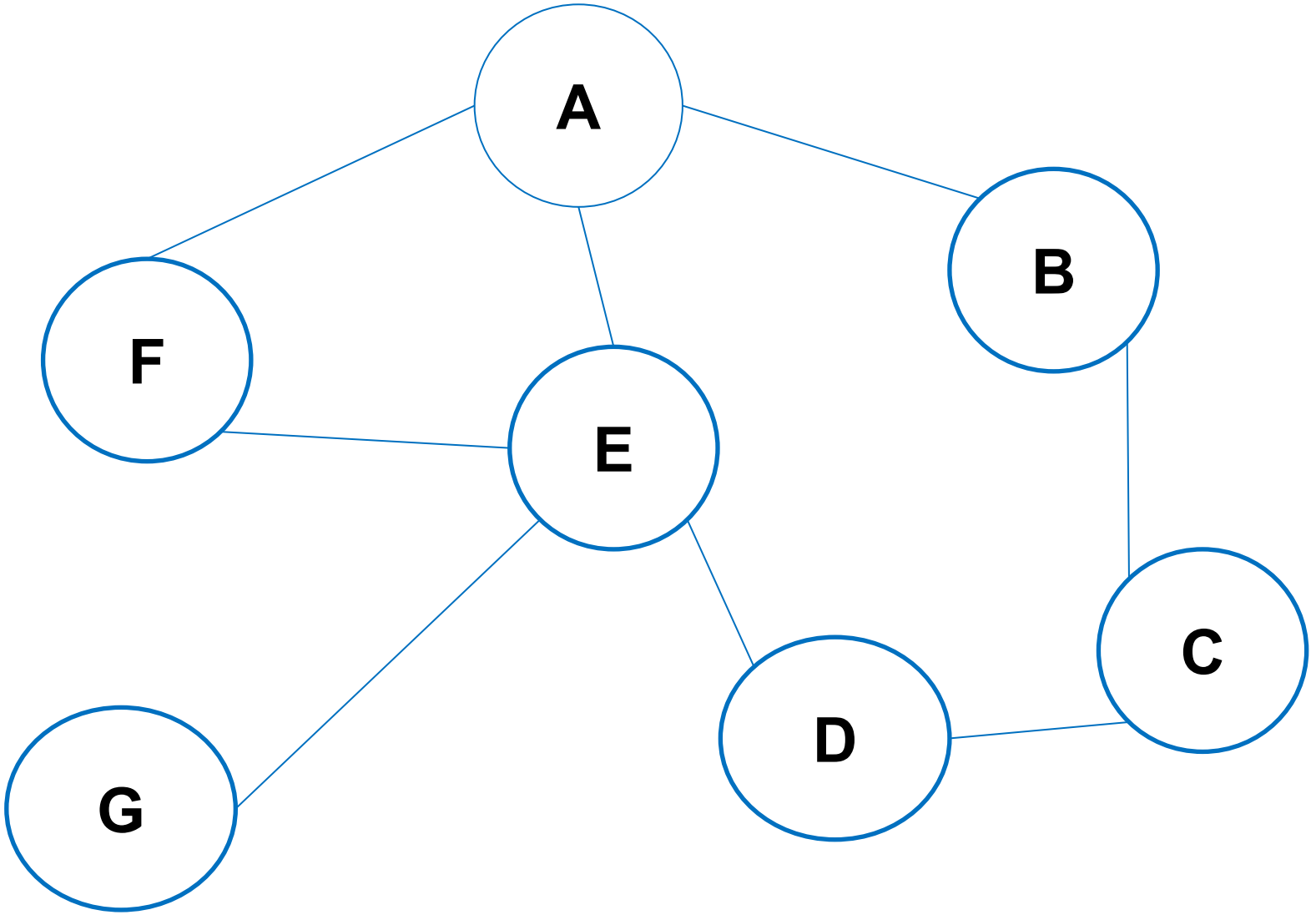
Dans tout le chapitre, on a:

- s ou s' des sommets, (s, s') un arc, {s, s'} une arête, et
- Le graphe G est décrit sous forme dictionnaire de listes d'adjacence.

2. PRINCIPE DU PARCOURS EN LARGEUR

- **Présentation**

Principe du déploiement régulier, effet de coulée



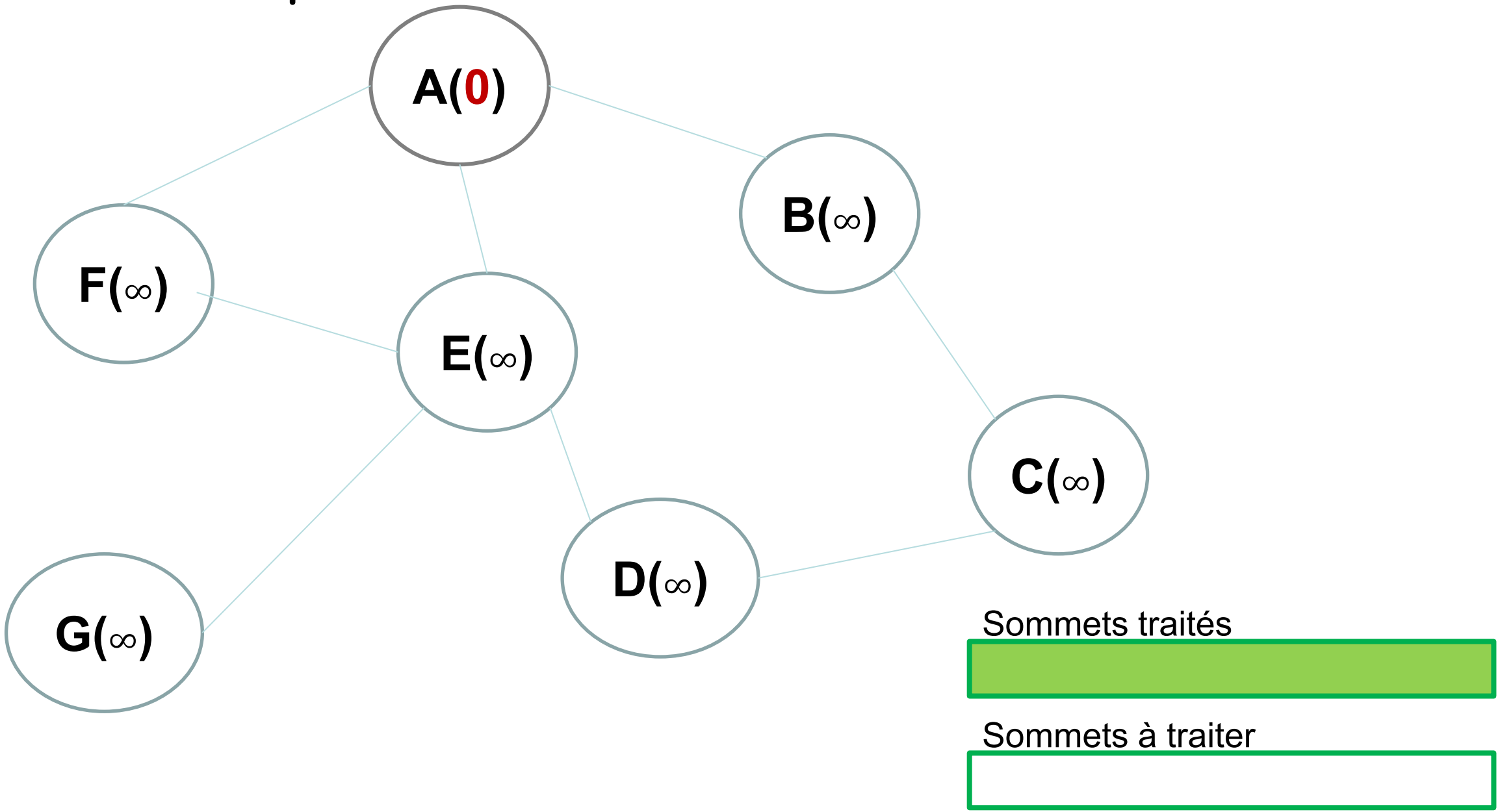
S Sommet non traité

S Sommet à traiter

S Sommet traité

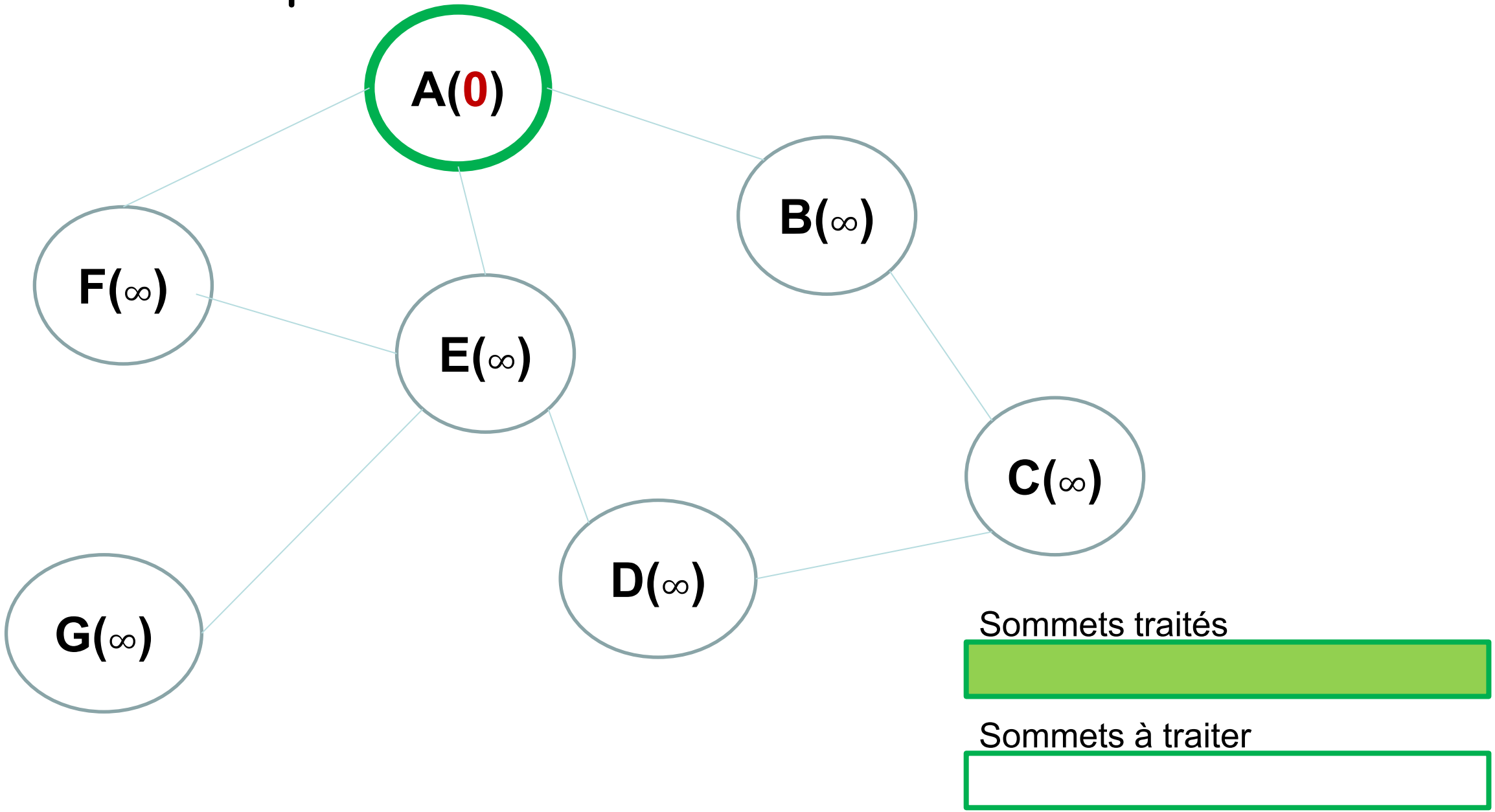
2. PRINCIPE DU PARCOURS EN LARGEUR

- **Illustration: 1- Départ**



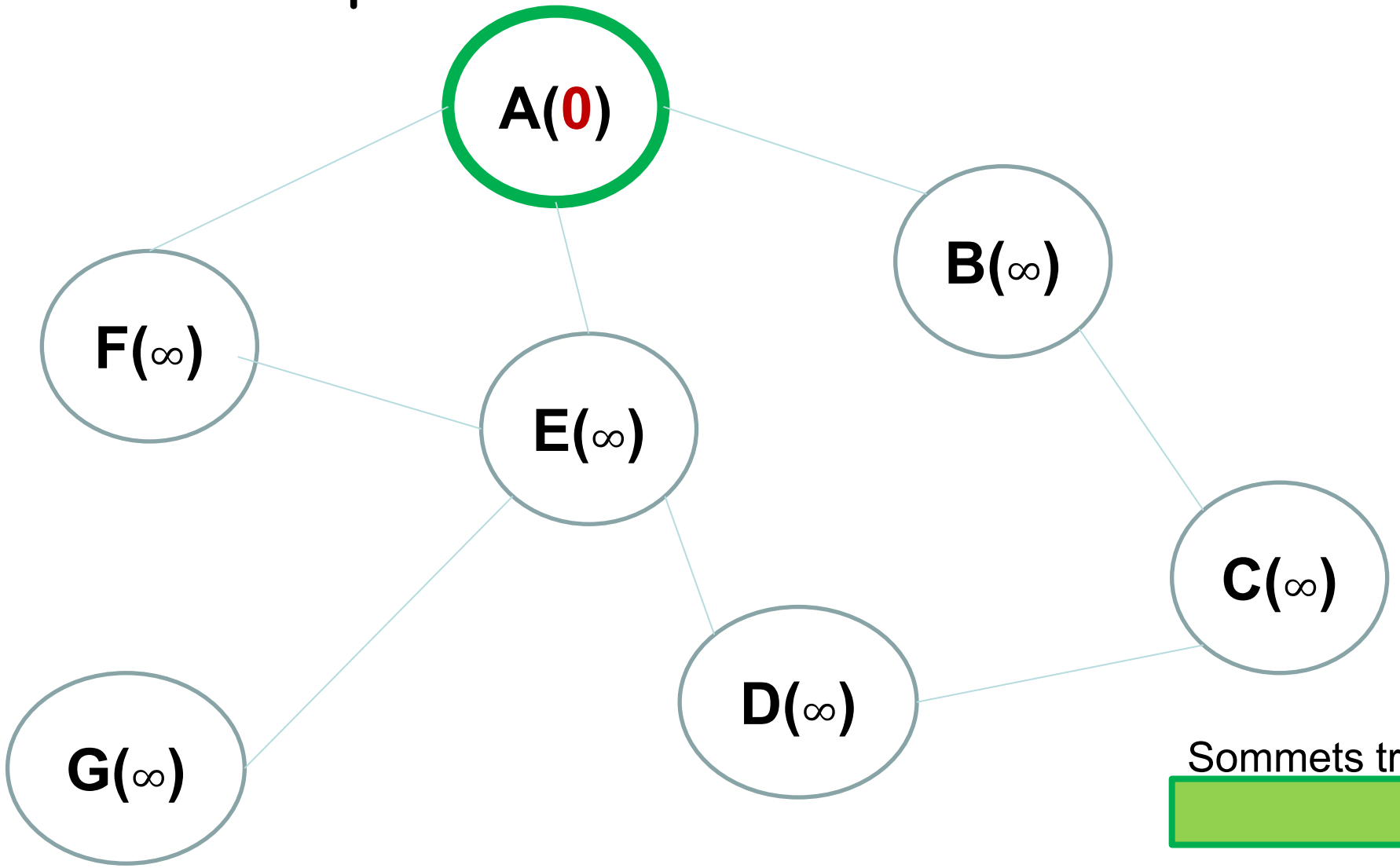
2. PRINCIPE DU PARCOURS EN LARGEUR

- **Illustration: 1- Départ**



2. PRINCIPE DU PARCOURS EN LARGEUR

- **Illustration: 1- Départ**



Sommets traités

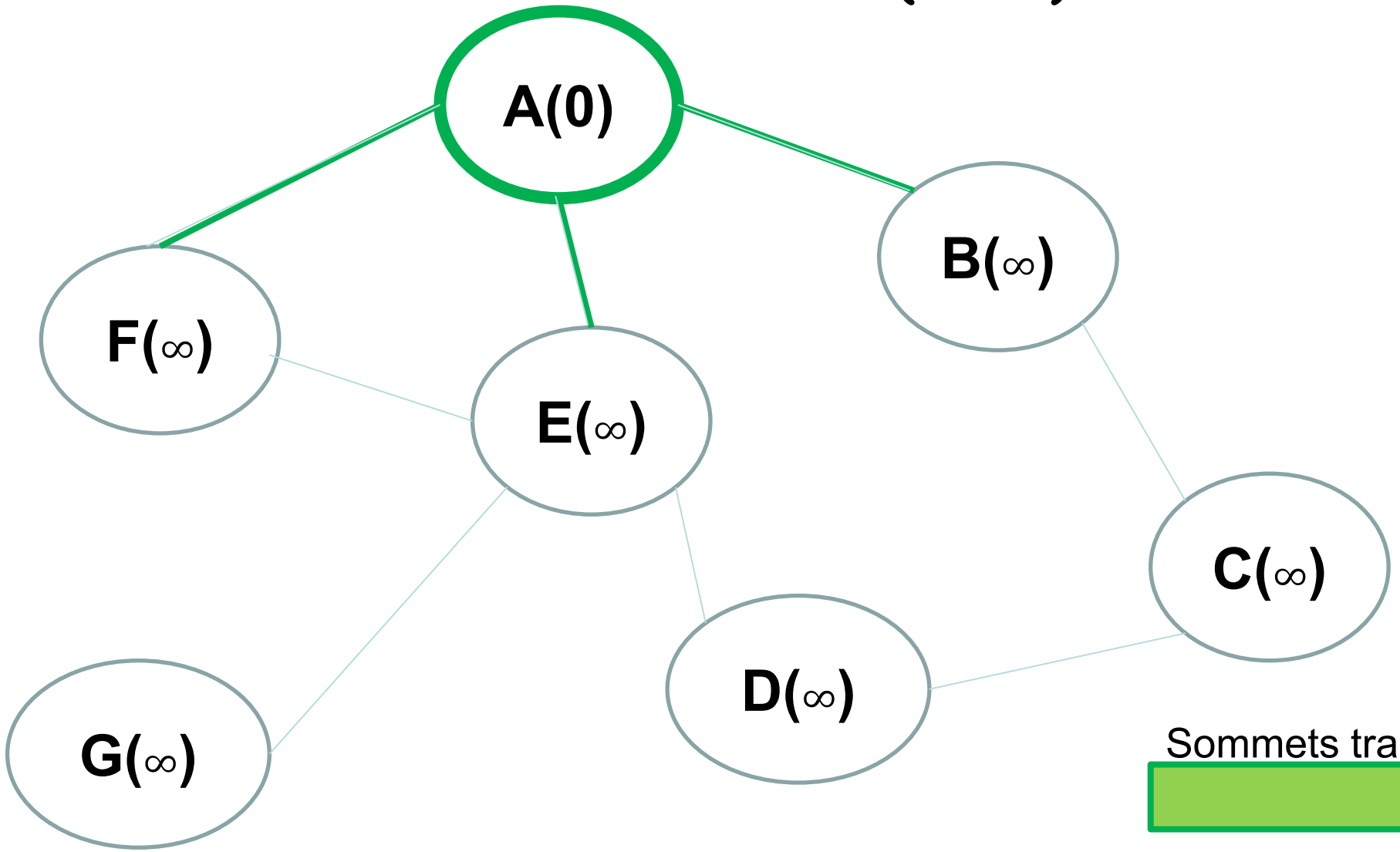
∅

Sommets à traiter

A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 2- Traitement du sommet A (début)



Sommets traités

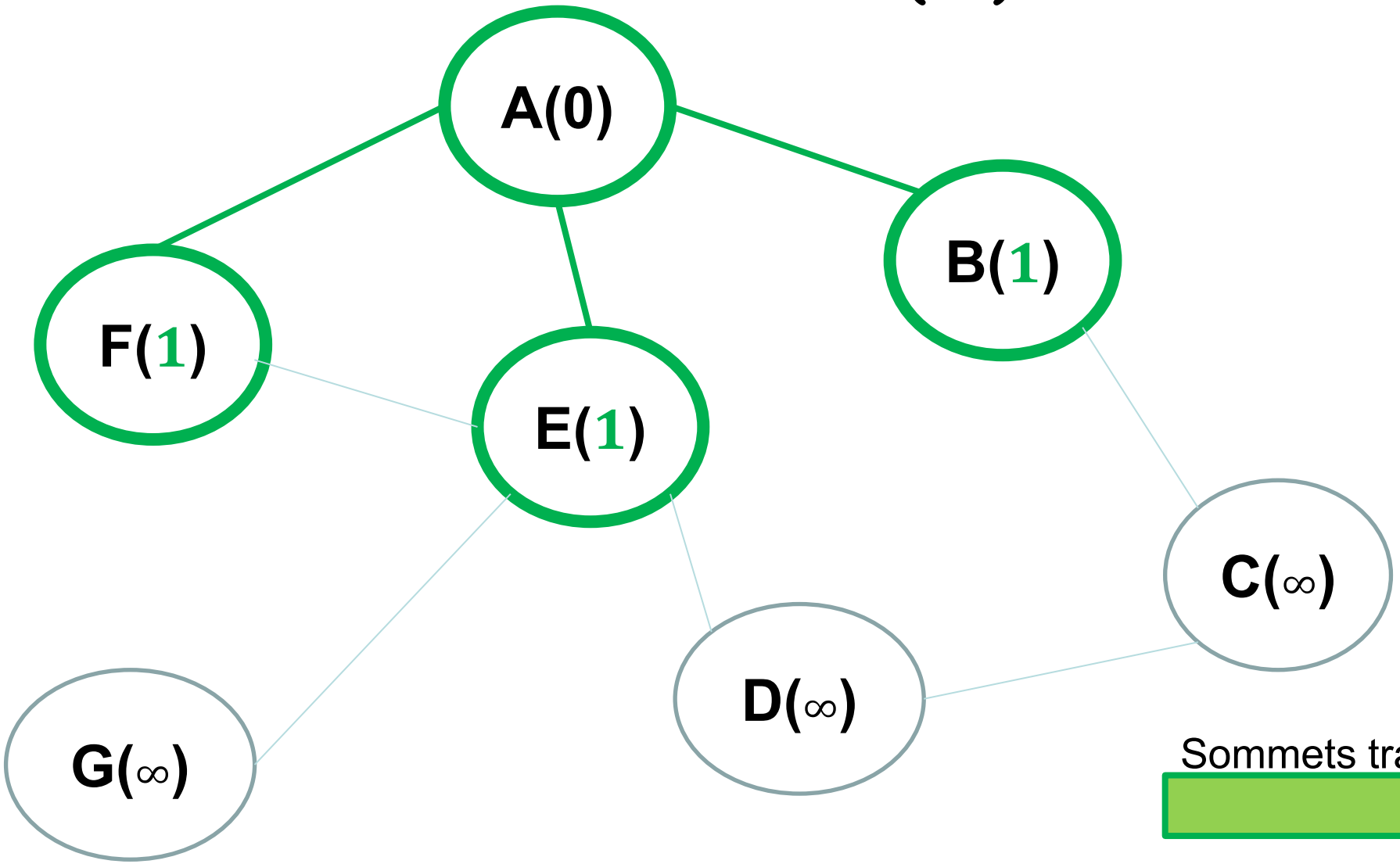
∅

Sommets à traiter

A

2. PRINCIPE DU PARCOURS EN LARGEUR

- **Illustration: 3- Traitement du sommet A (fin)**



Sommets traités

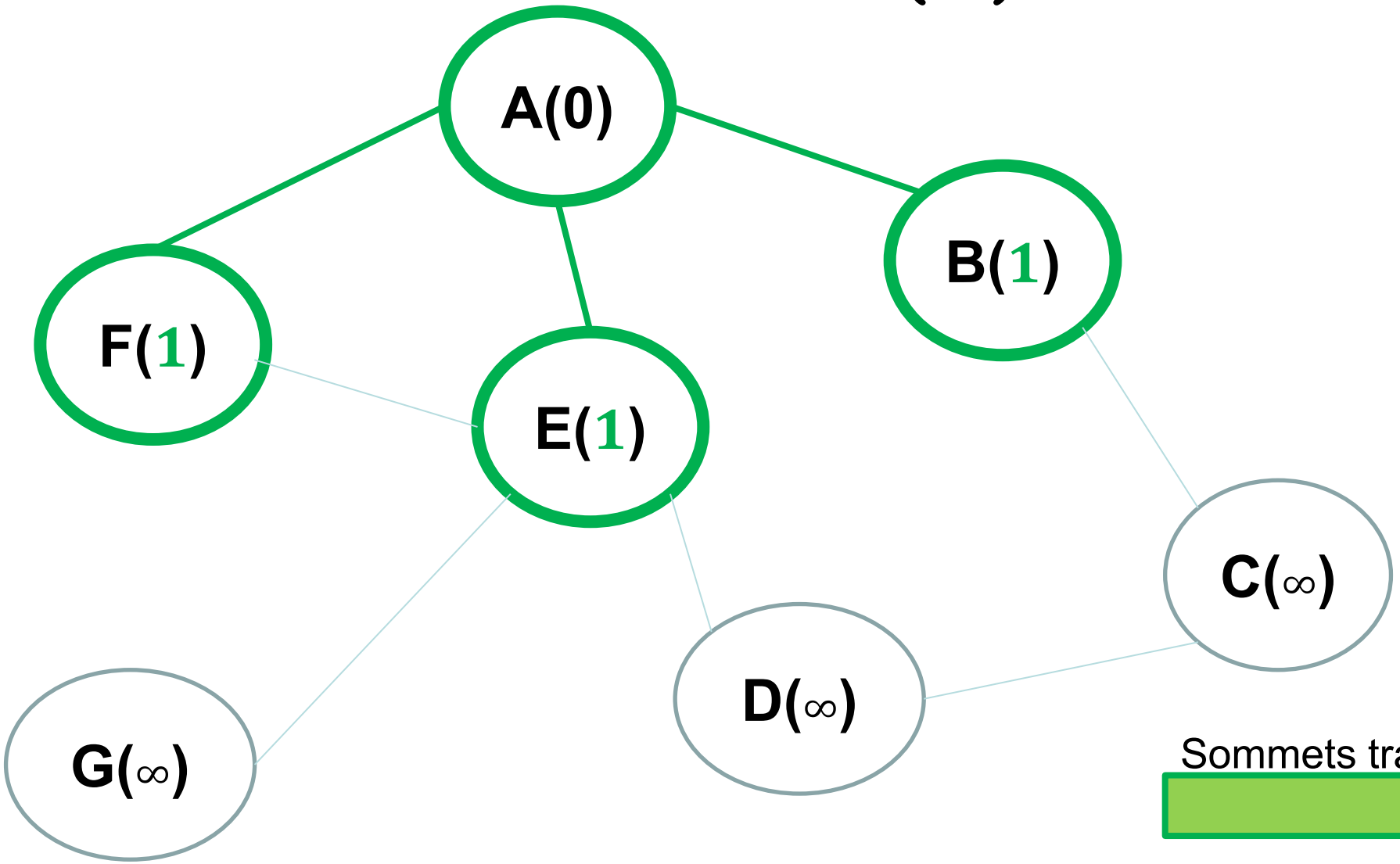
ø

Sommets à traiter

A

2. PRINCIPE DU PARCOURS EN LARGEUR

- **Illustration: 3- Traitement du sommet A (fin)**



Sommets traités

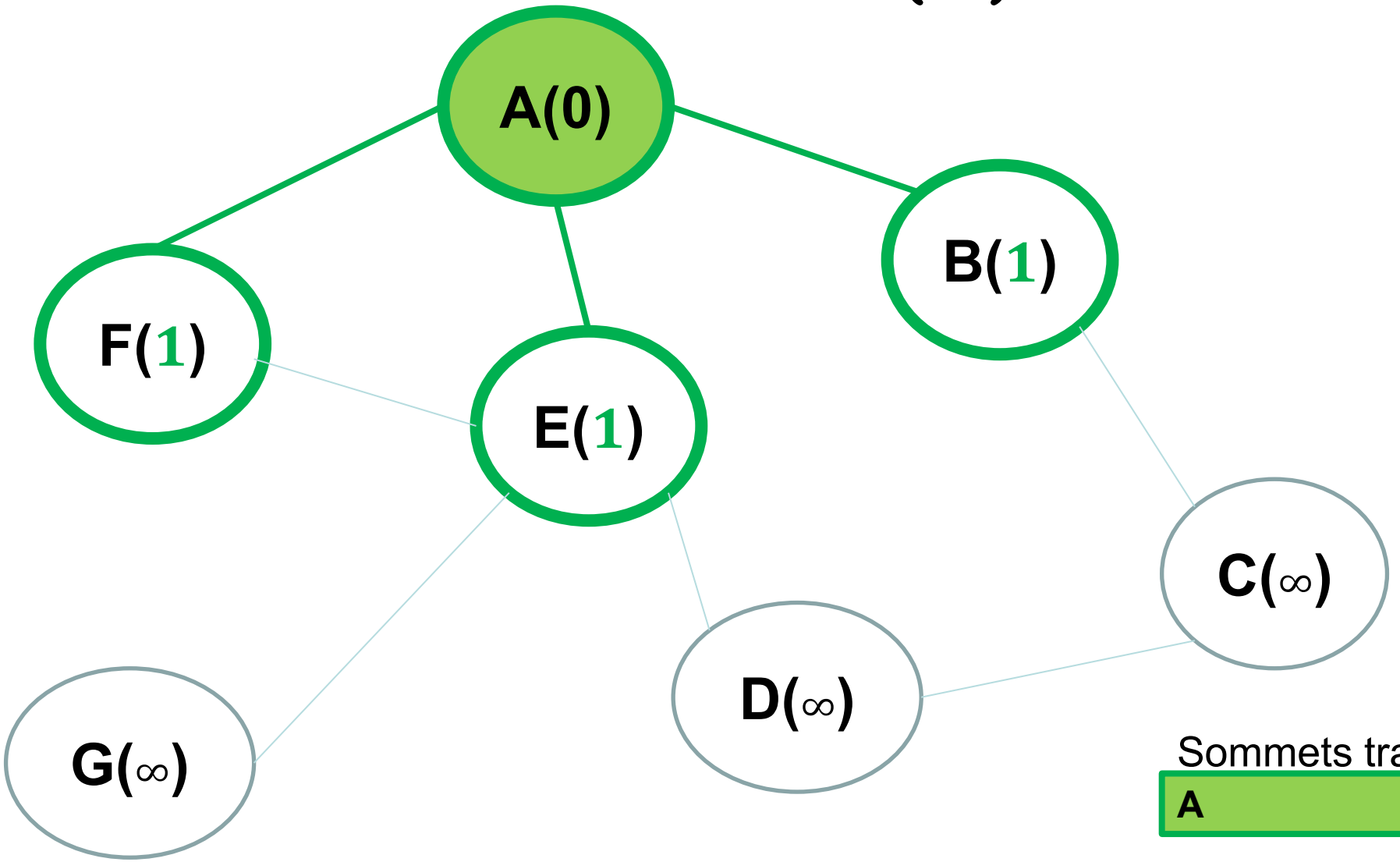
ø

Sommets à traiter

F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 3- Traitement du sommet A (fin)



Sommets traités

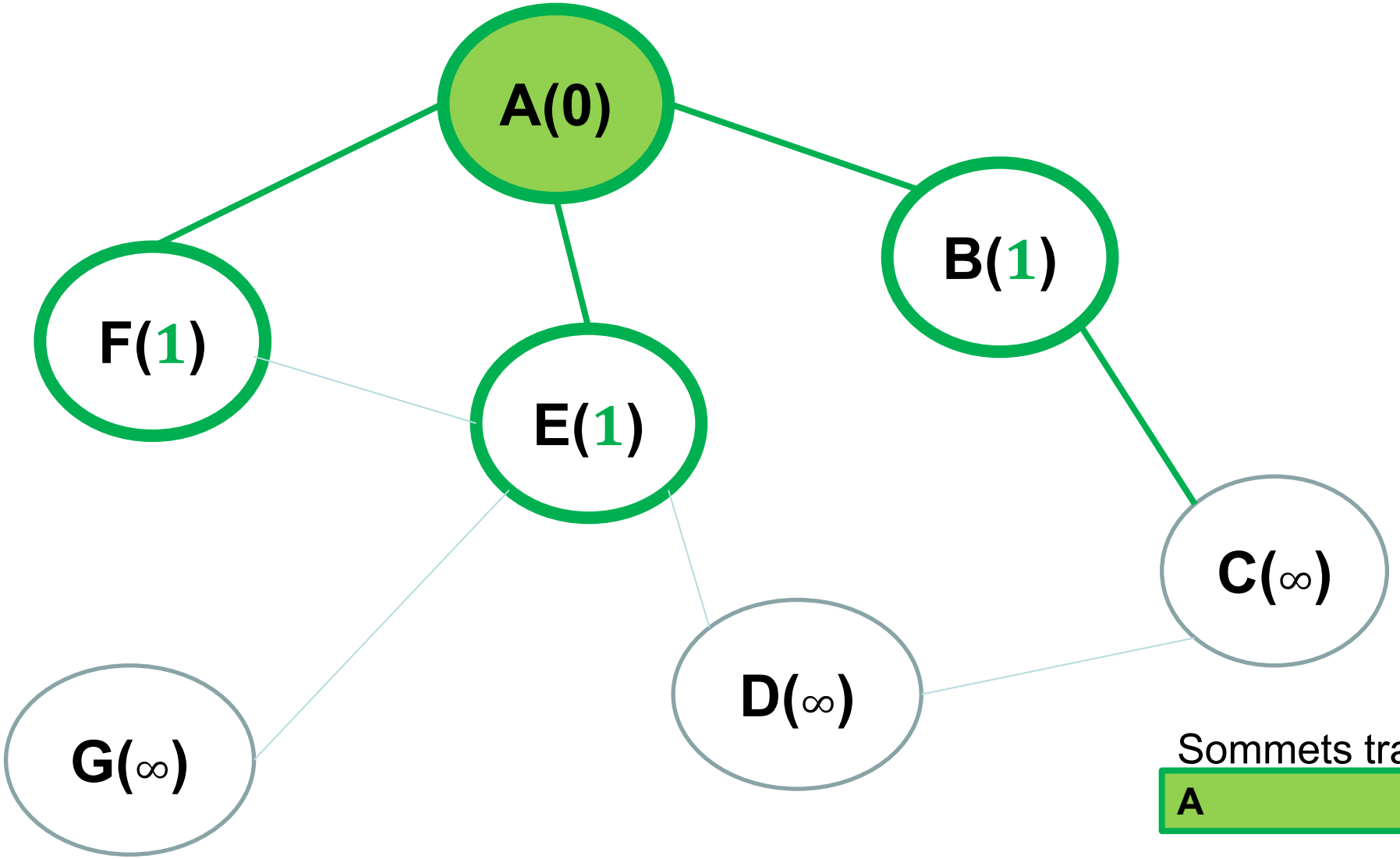
A

Sommets à traiter

F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 4- Traitement du sommet B



Sommets traités

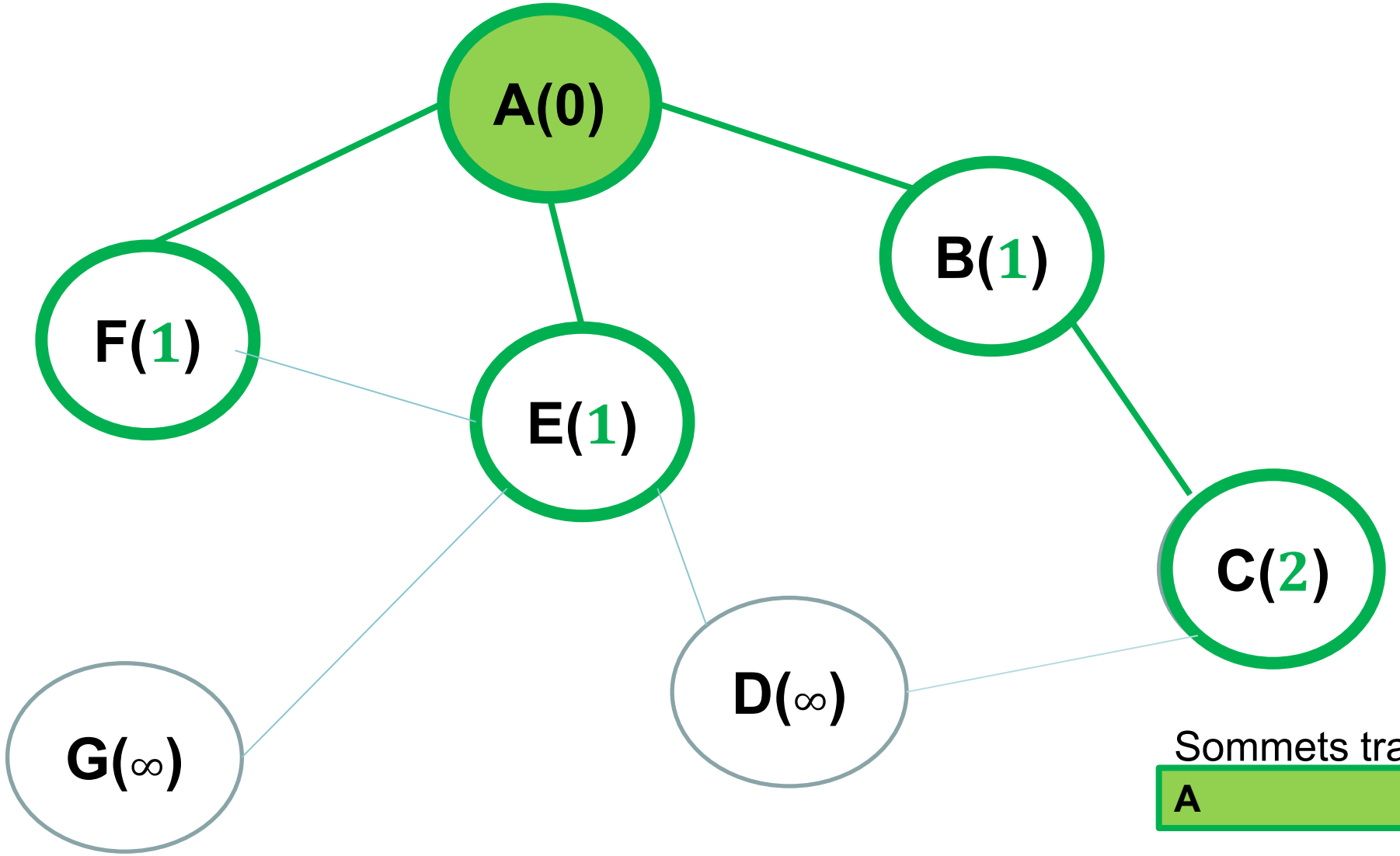
A

Sommets à traiter

F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 4- Traitement du sommet B



Sommets traités

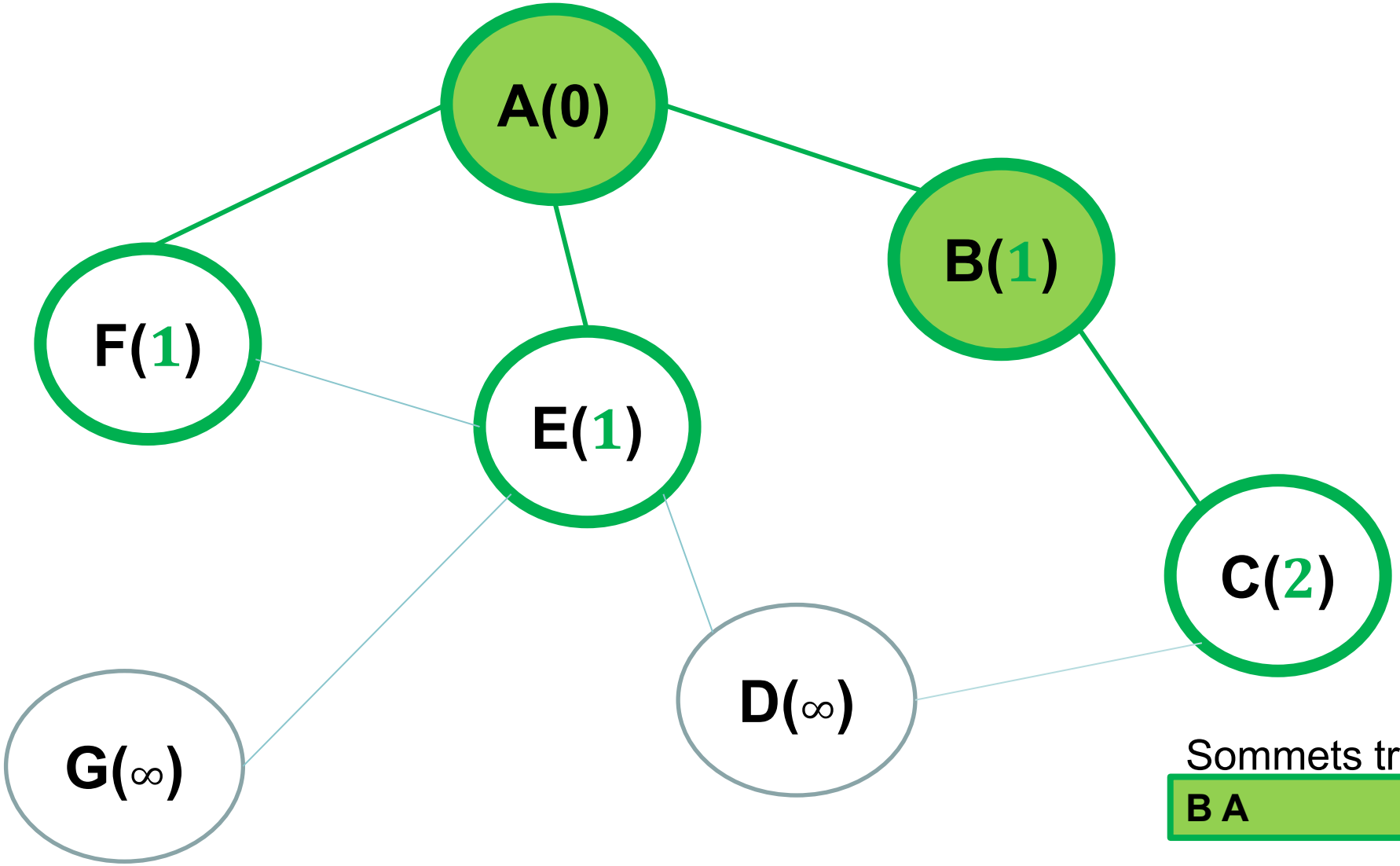
A

Sommets à traiter

C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- **Illustration: 4-Traitement du sommet B**



Sommets traités

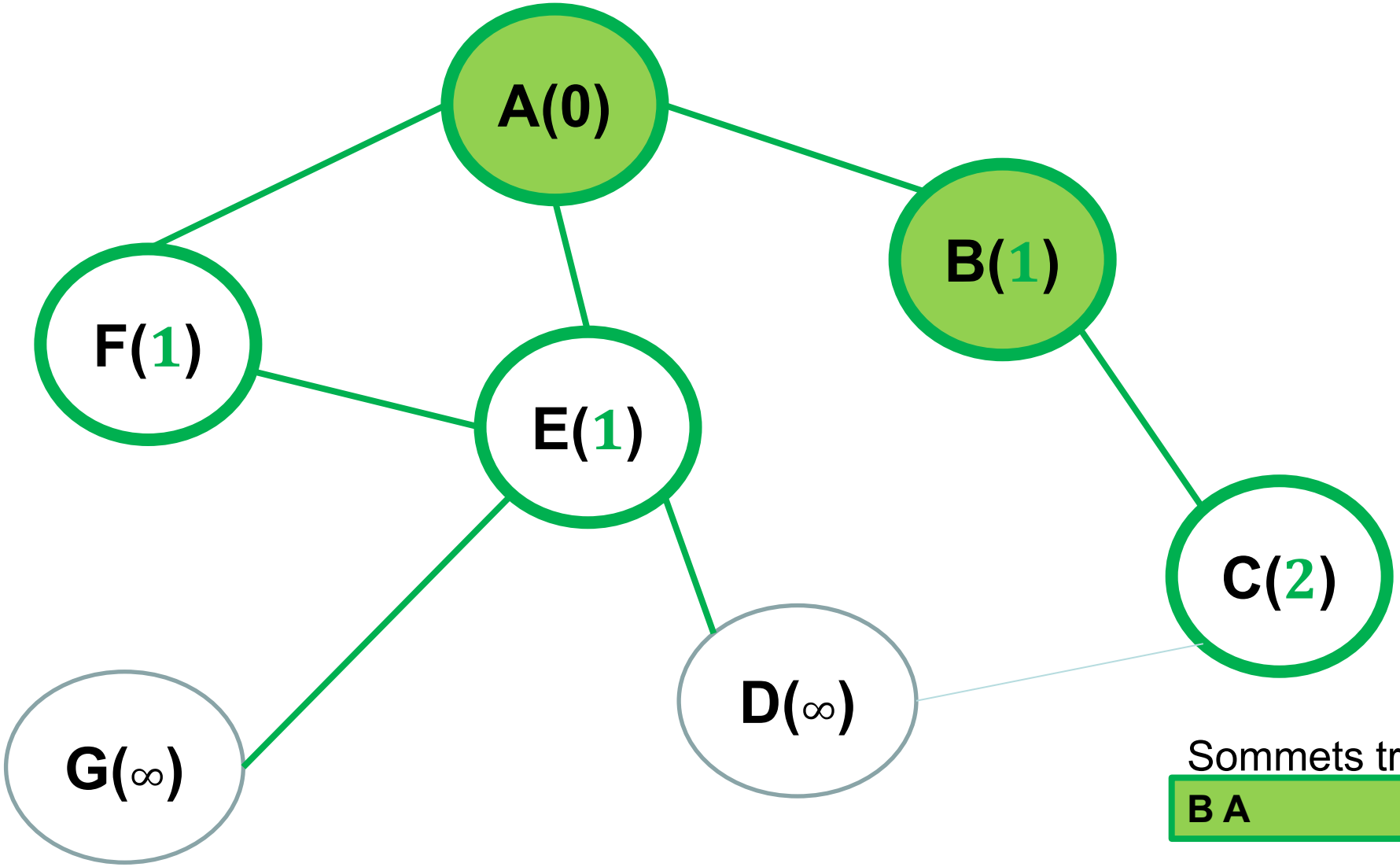
B A

Sommets à traiter

C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 5- Traitement du sommet E



Sommets traités

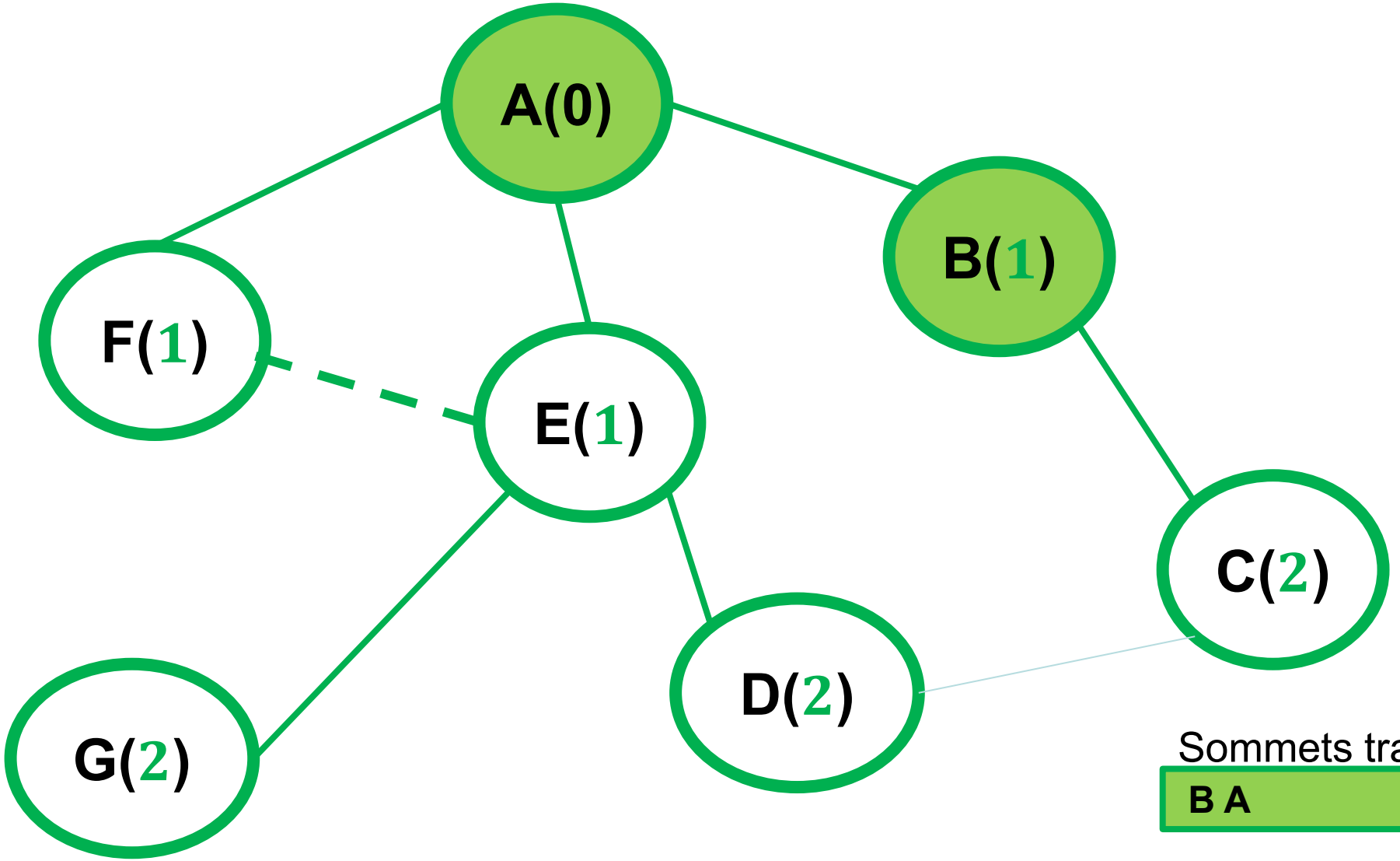
B A

Sommets à traiter

C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 5- Traitement du sommet E



Sommets traités

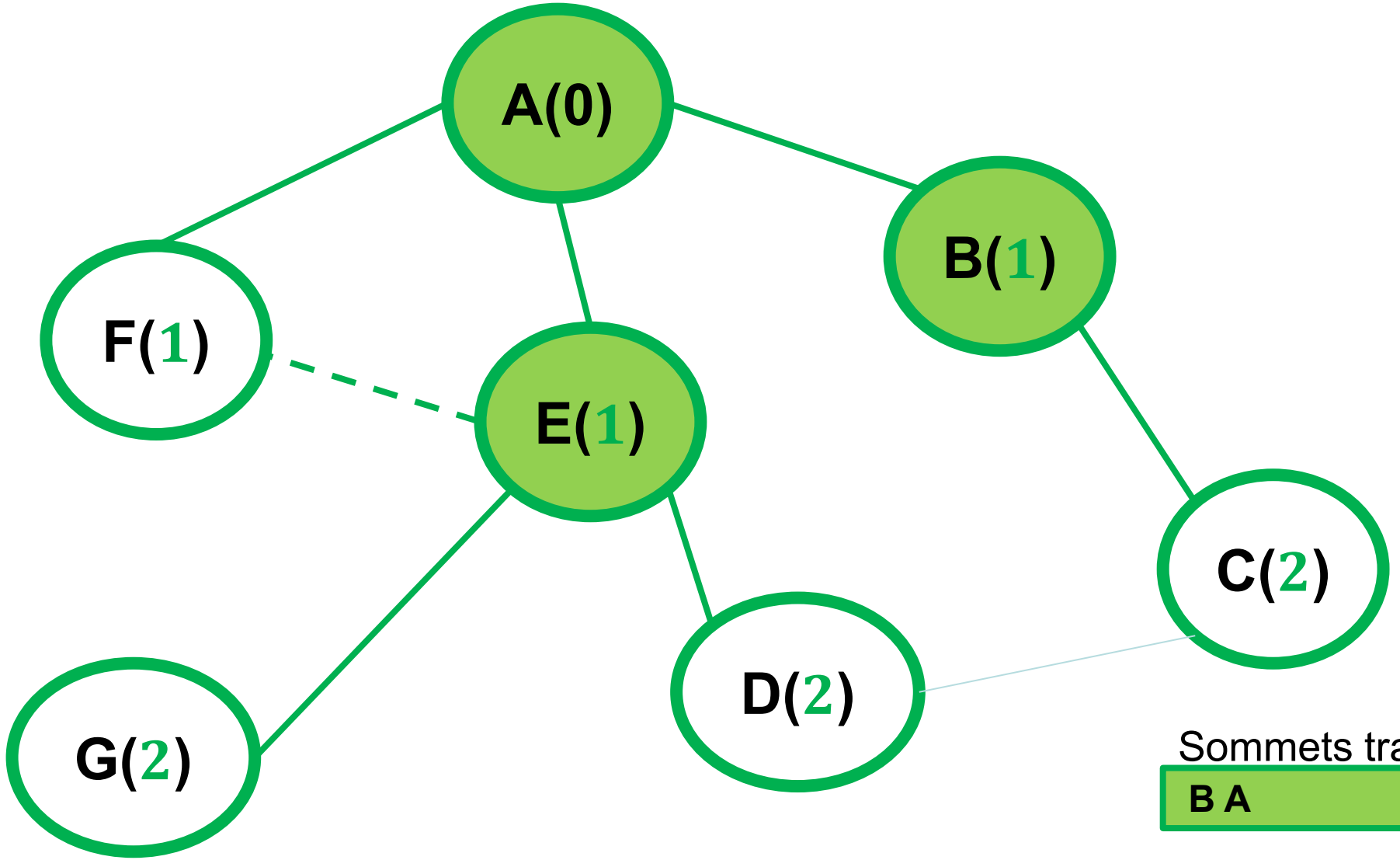
B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 5-Traitement du sommet E



Sommets traités

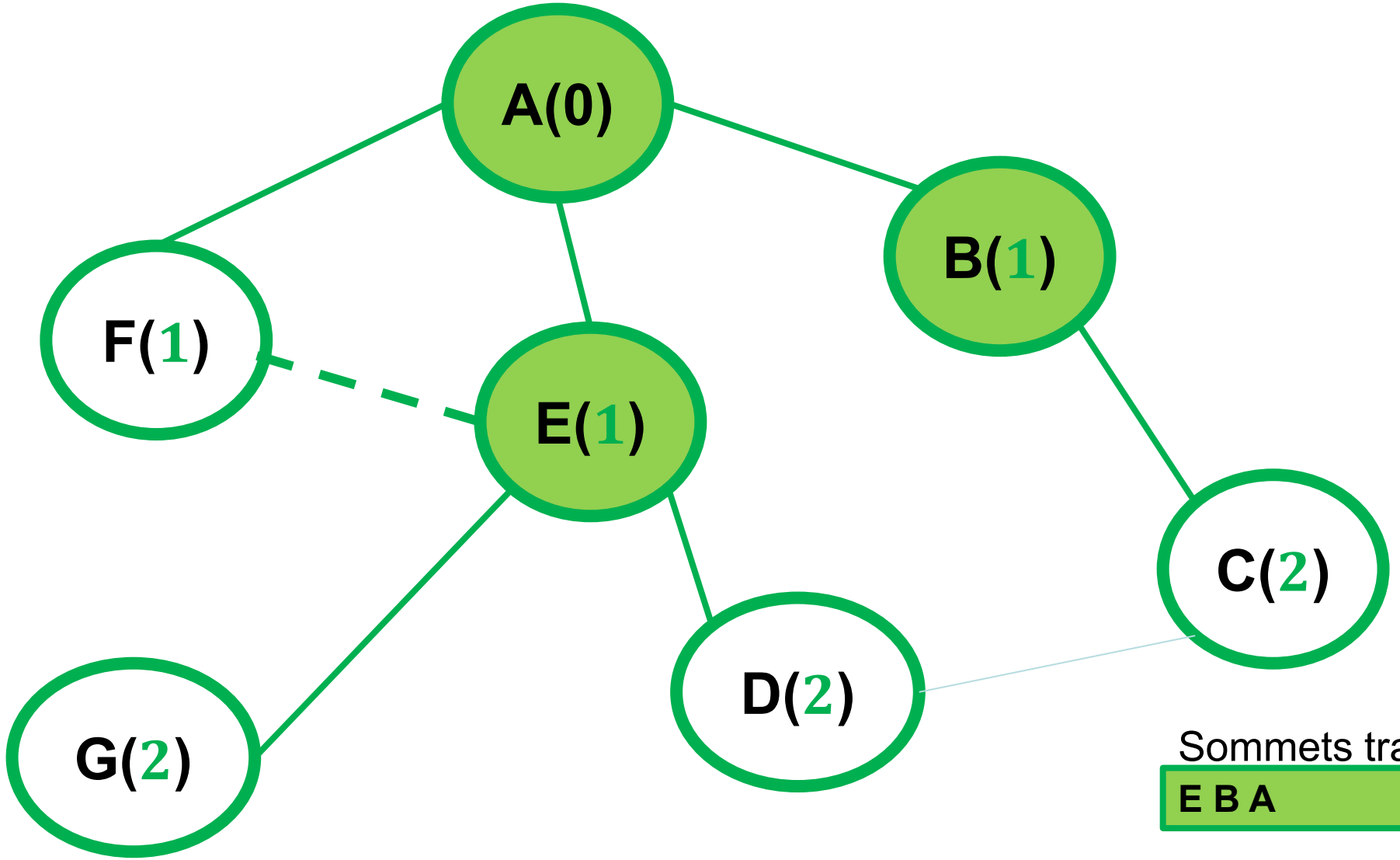
B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 6- Traitement du sommet F



Sommets traités

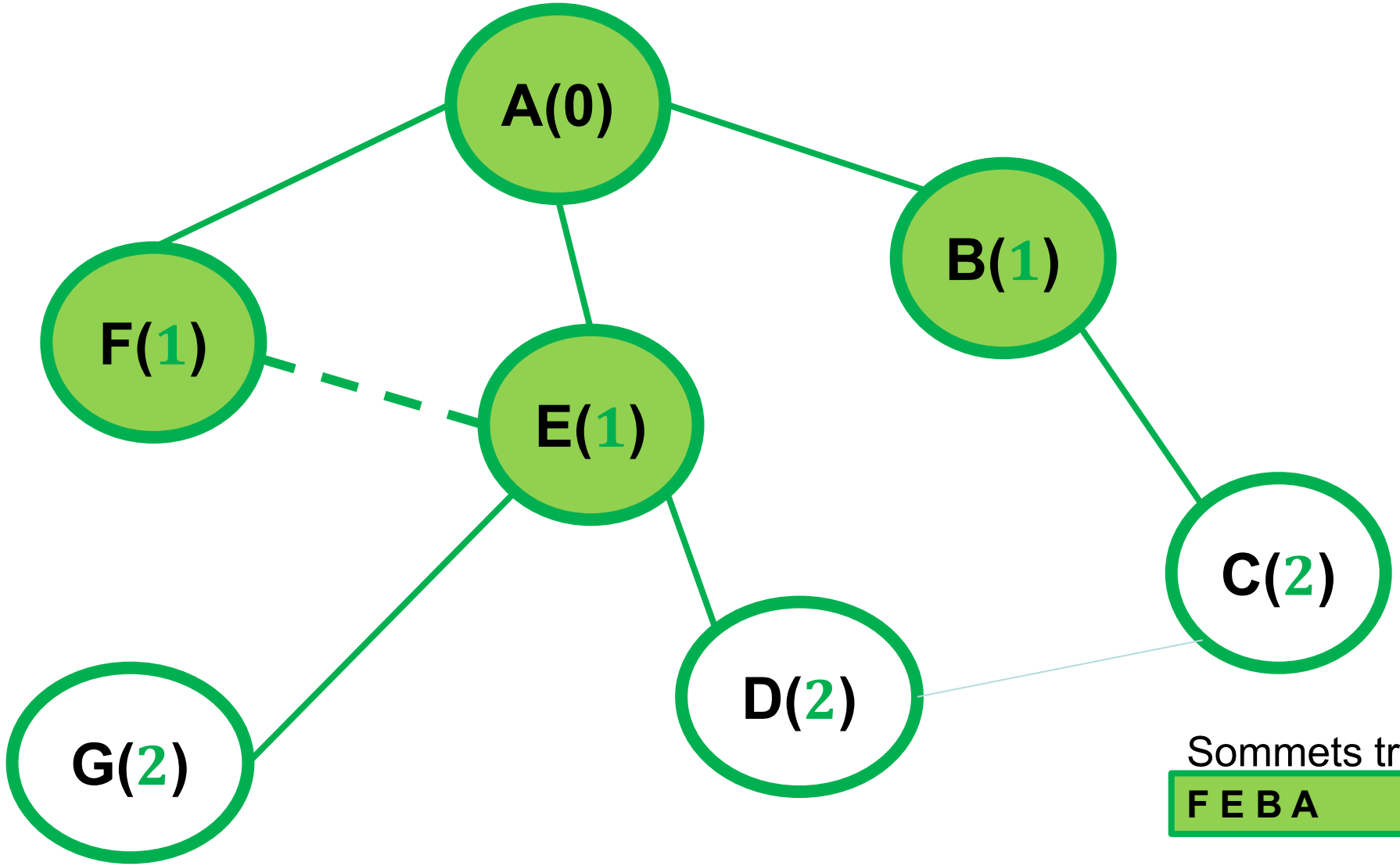
E B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 6- Traitement du sommet F



Sommets traités

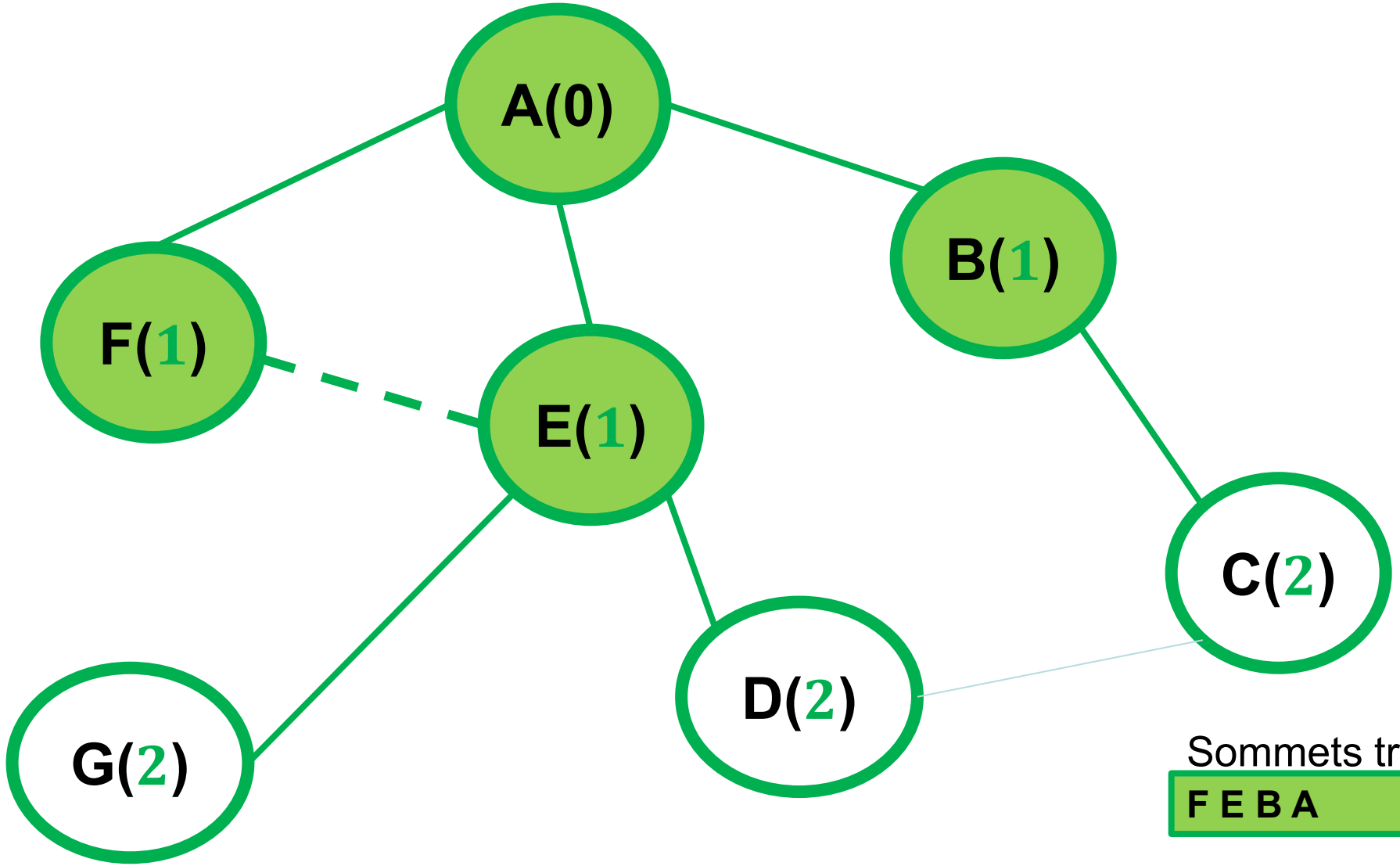
F E B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 7- Traitement du sommet C



Sommets traités

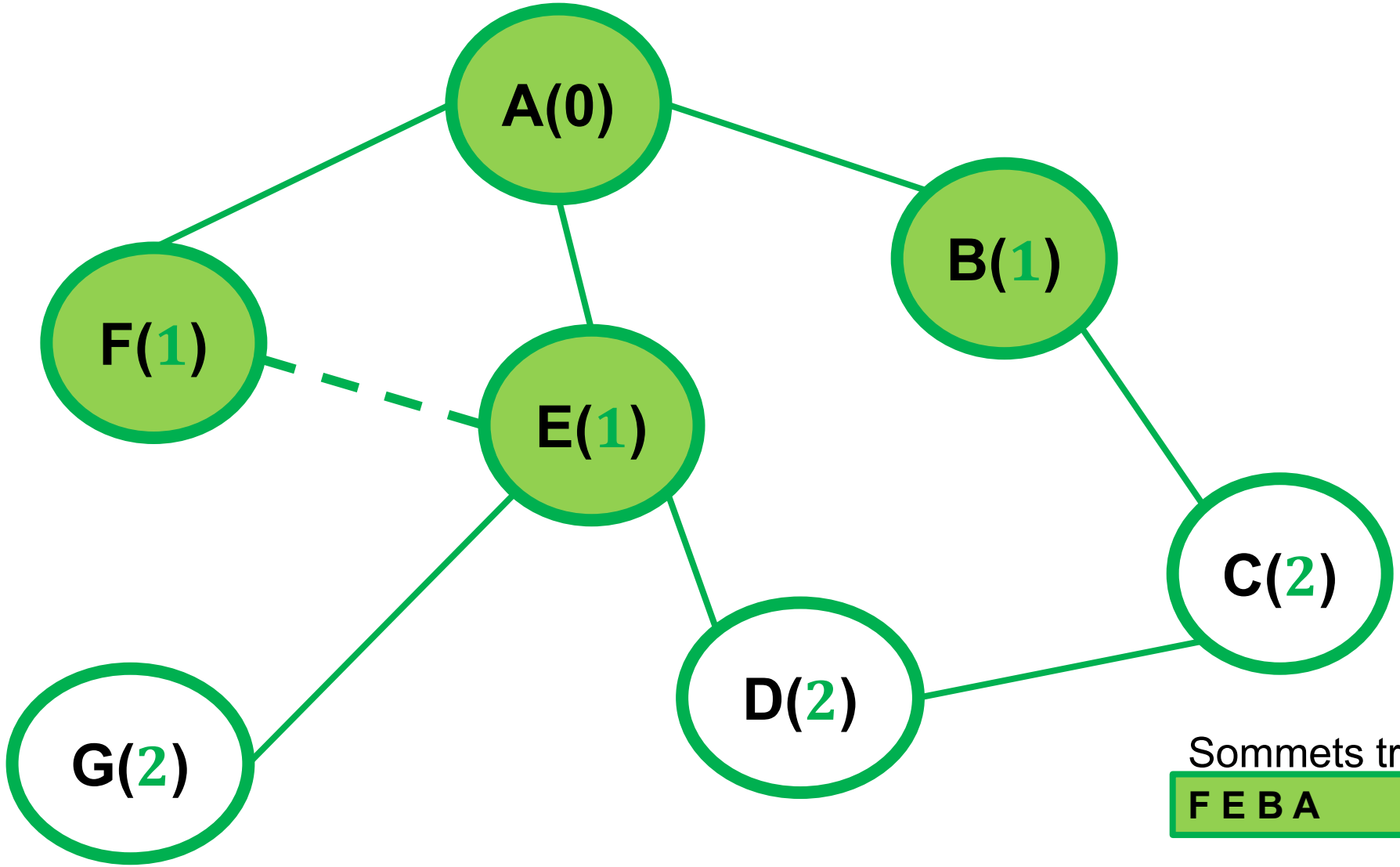
F E B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 7- Traitement du sommet C



Sommets traités

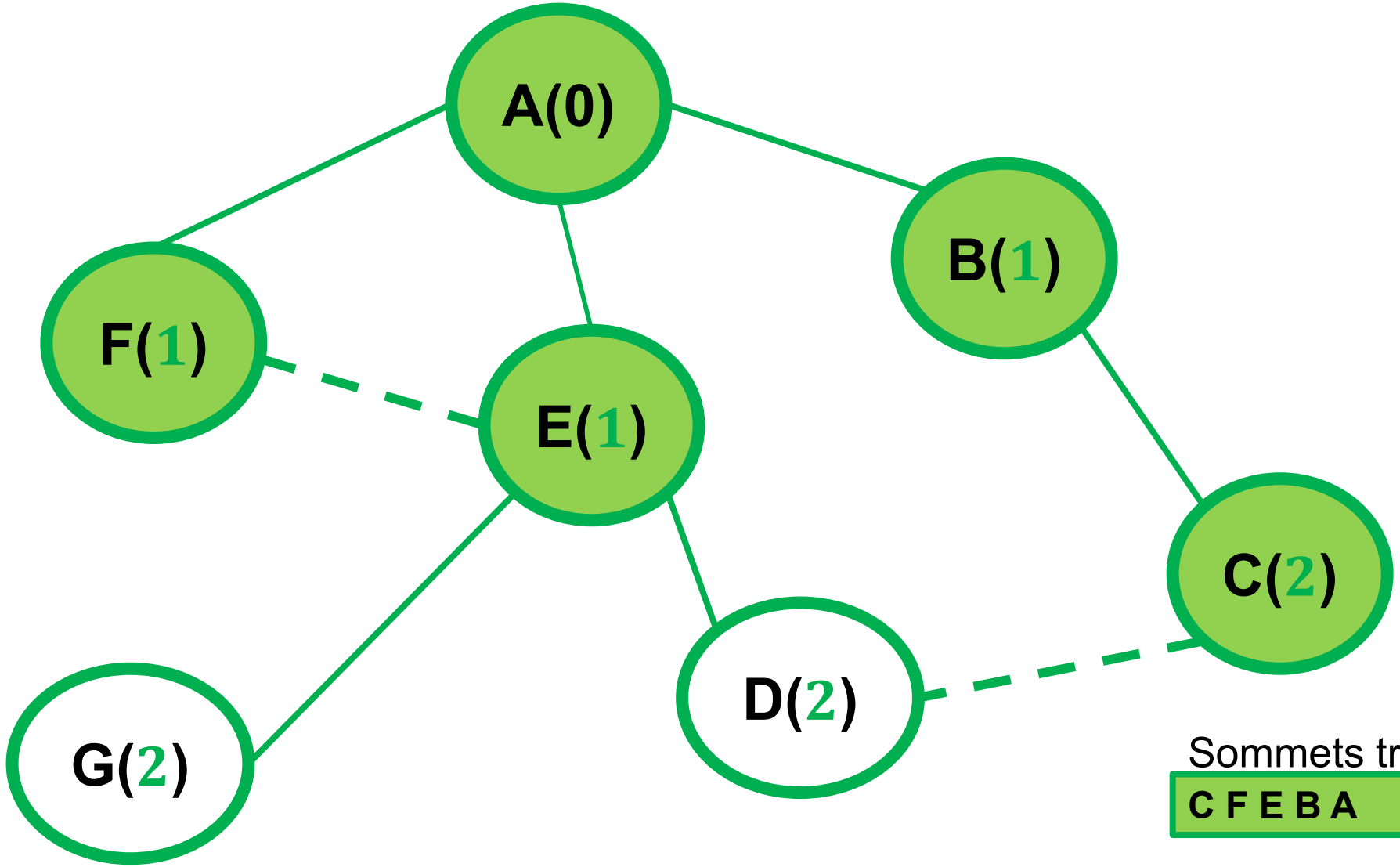
F E B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 7- Traitement du sommet C



Sommets traités

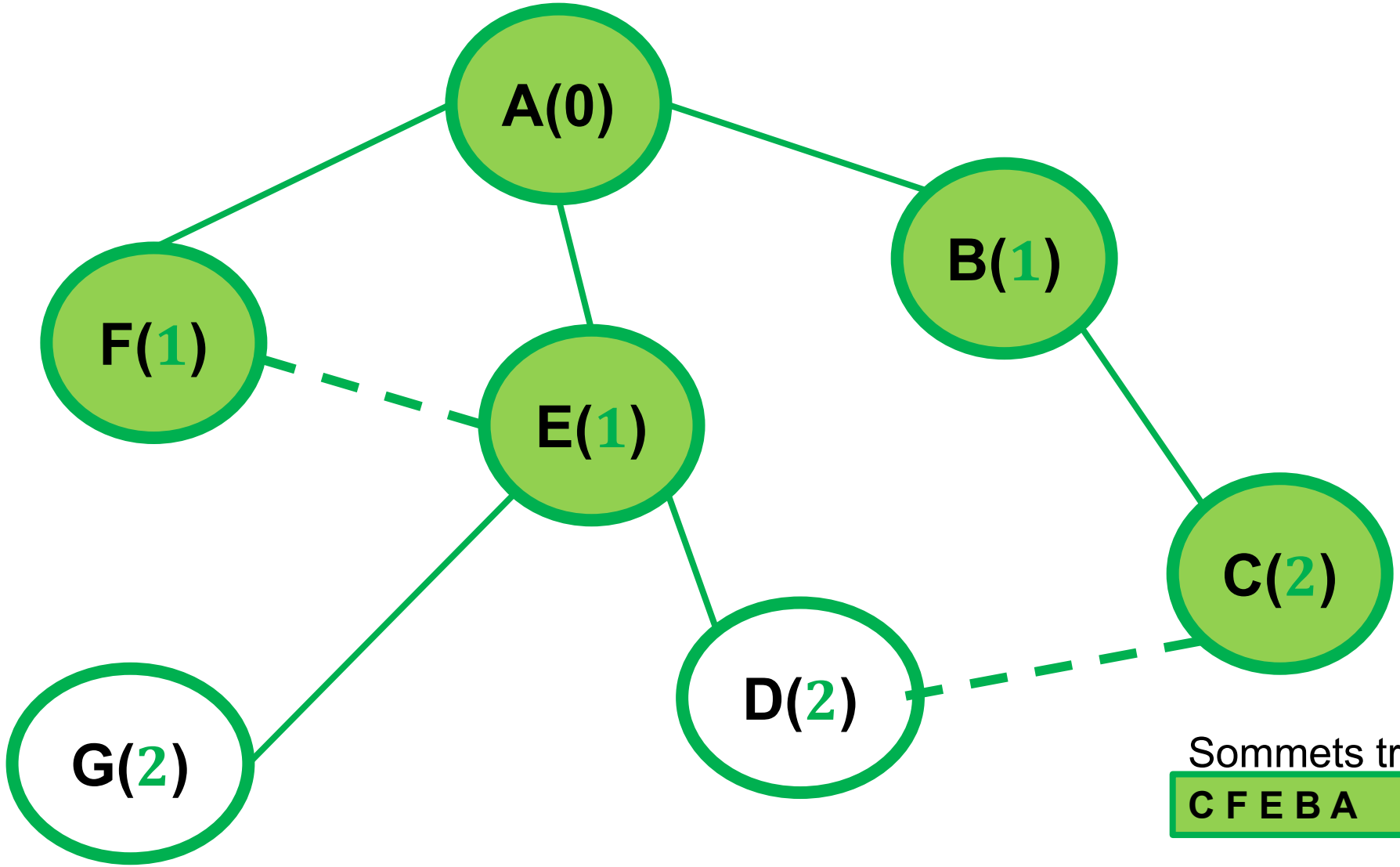
C F E B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 8- Traitement du sommet D



Sommets traités

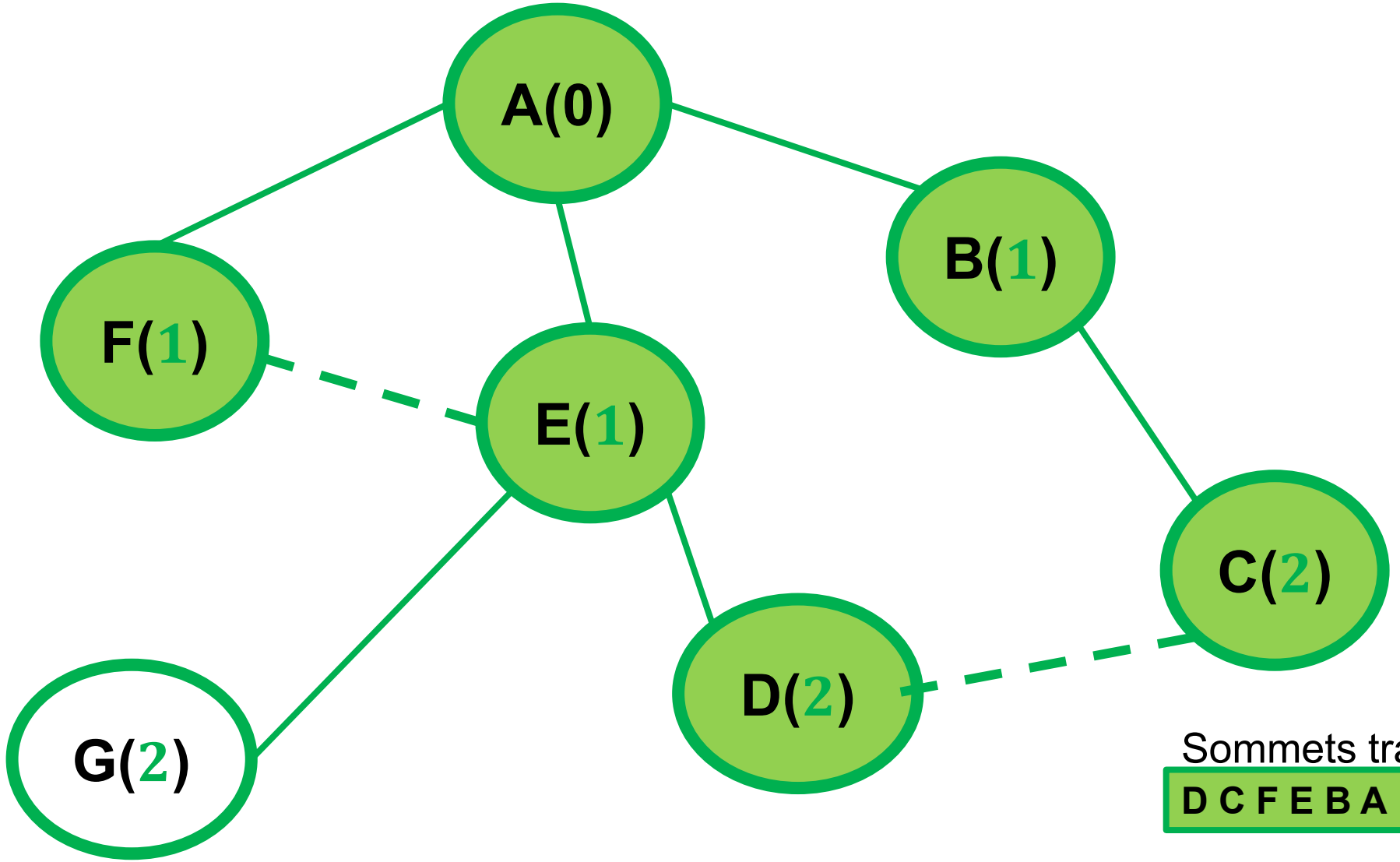
C F E B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 8- Traitement du sommet D



Sommets traités

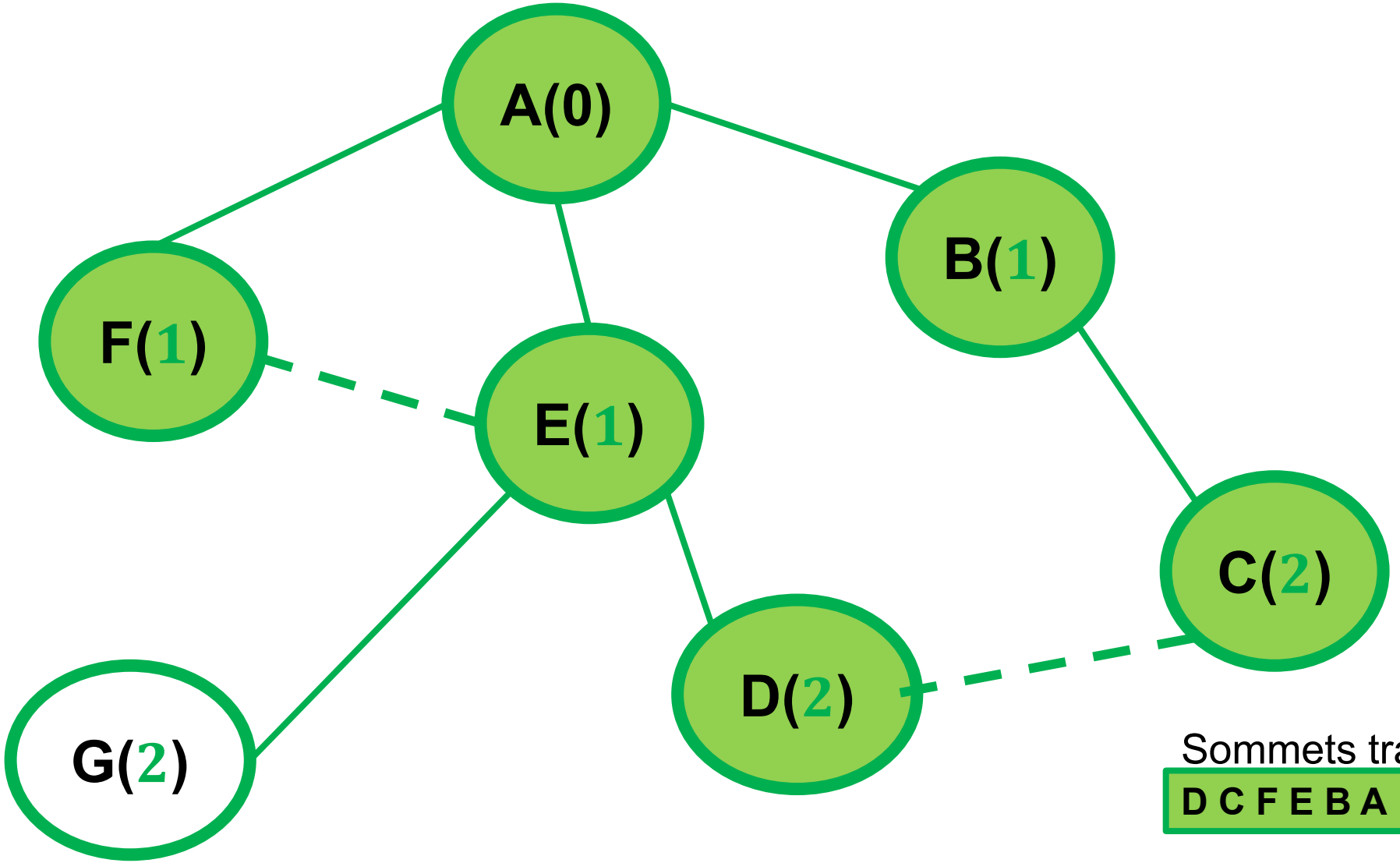
D C F E B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 9- Traitement du sommet *G*



Sommets traités

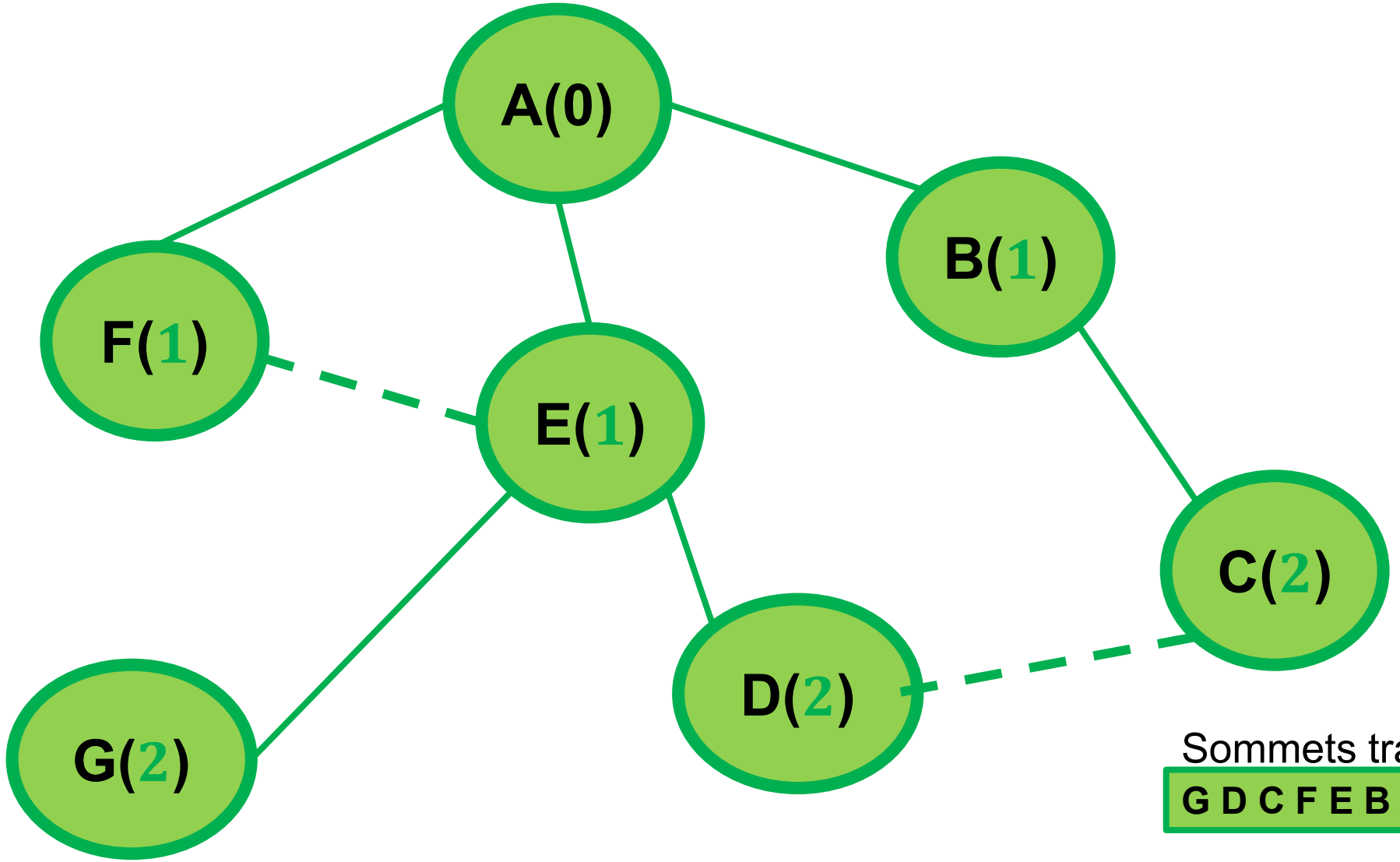
D C F E B A

Sommets à traiter

G D C F E B A

2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: 9- Traitement du sommet *G*



Sommets traités

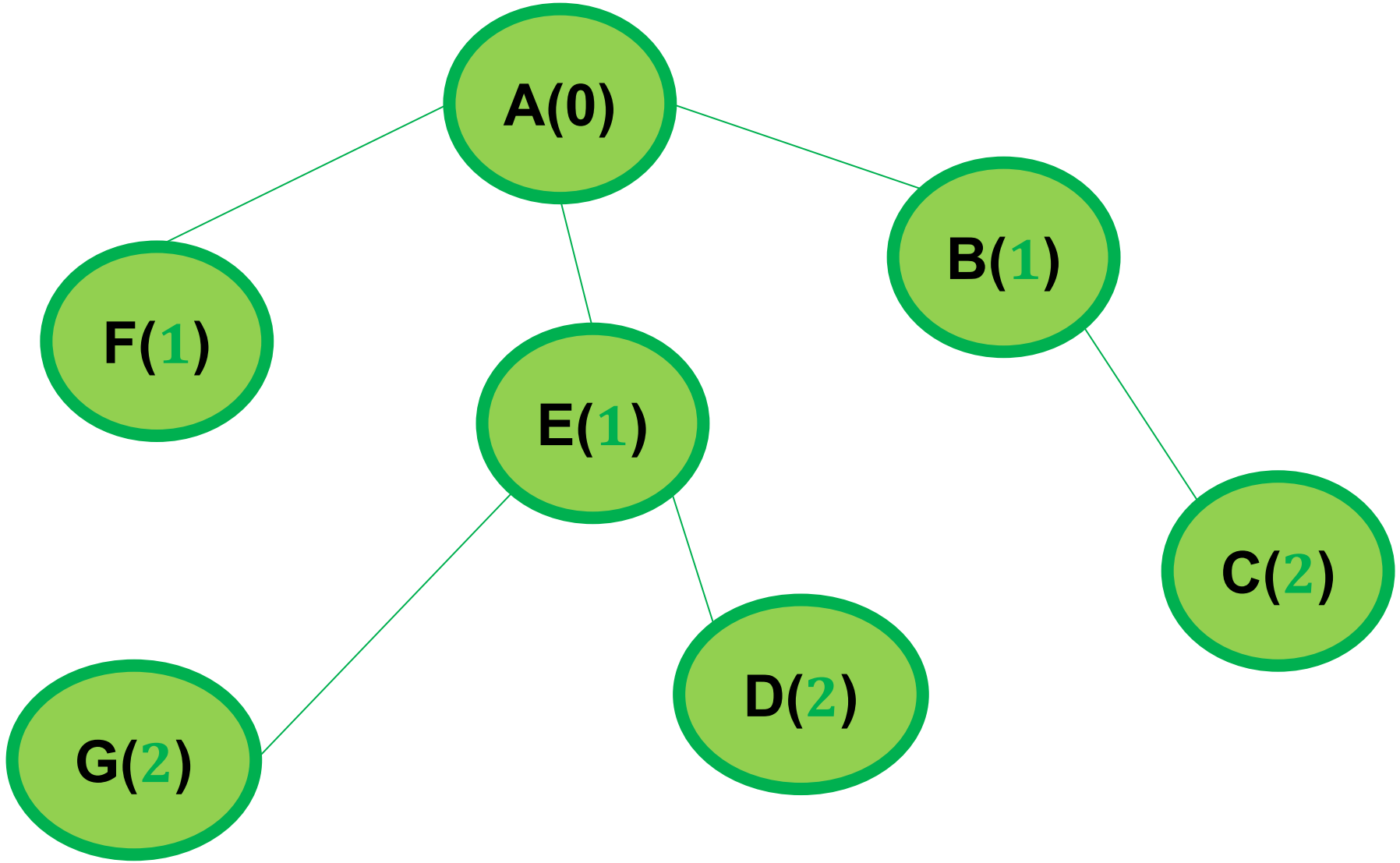
G D C F E B A

Sommets à traiter

∅ G D C F E B A

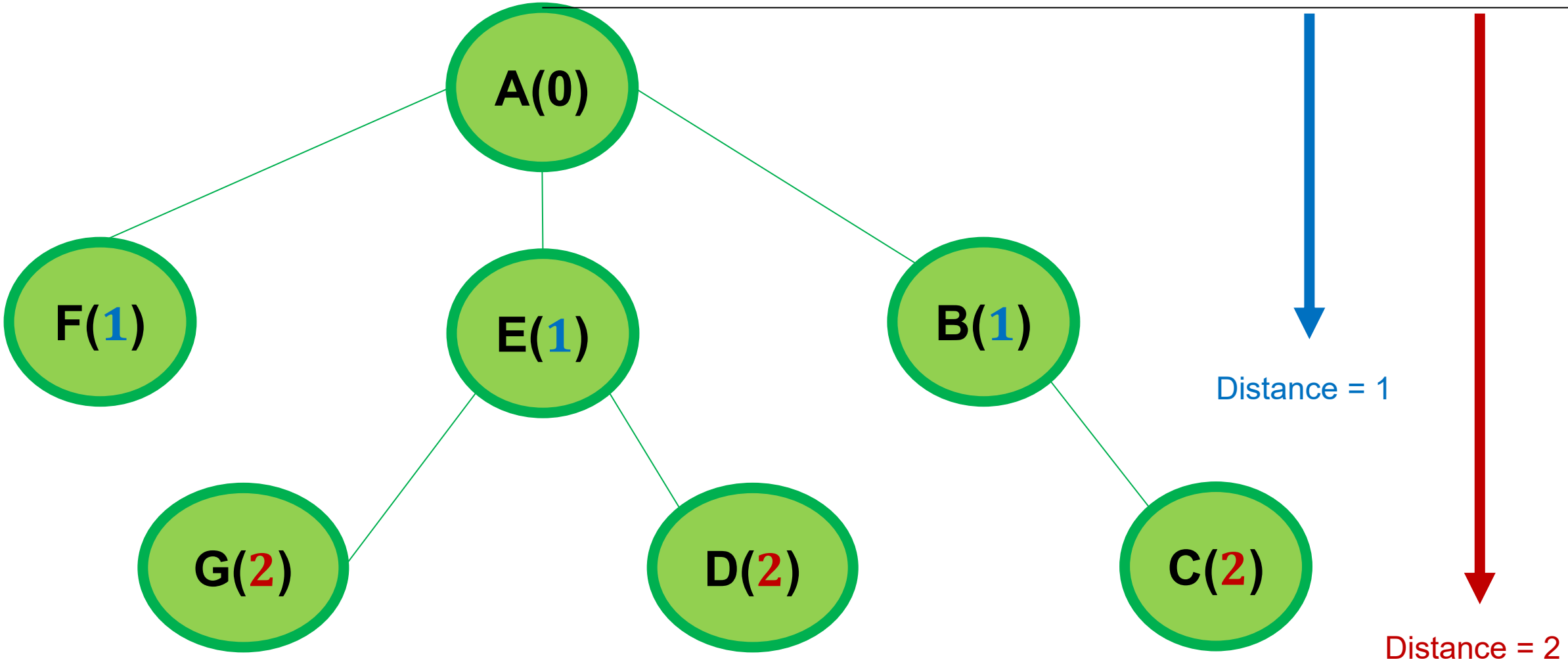
2. PRINCIPE DU PARCOURS EN LARGEUR

- Illustration: Création d'un arbre couvrant



2. PRINCIPE DU PARCOURS EN LARGEUR

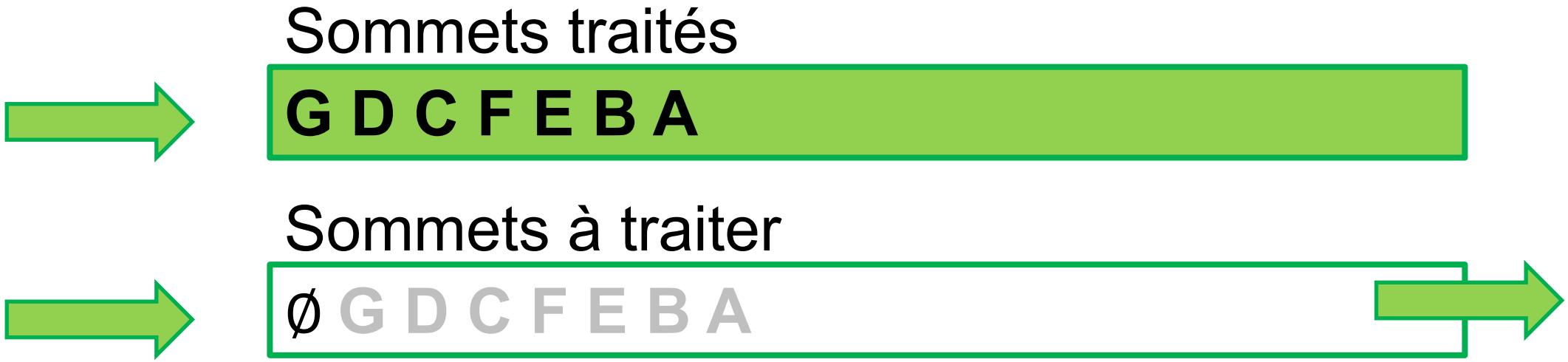
- Illustration: Création d'un arbre couvrant



☞ Lecture facilitée des distances au sommet de départ

2. PRINCIPE DU PARCOURS EN LARGEUR

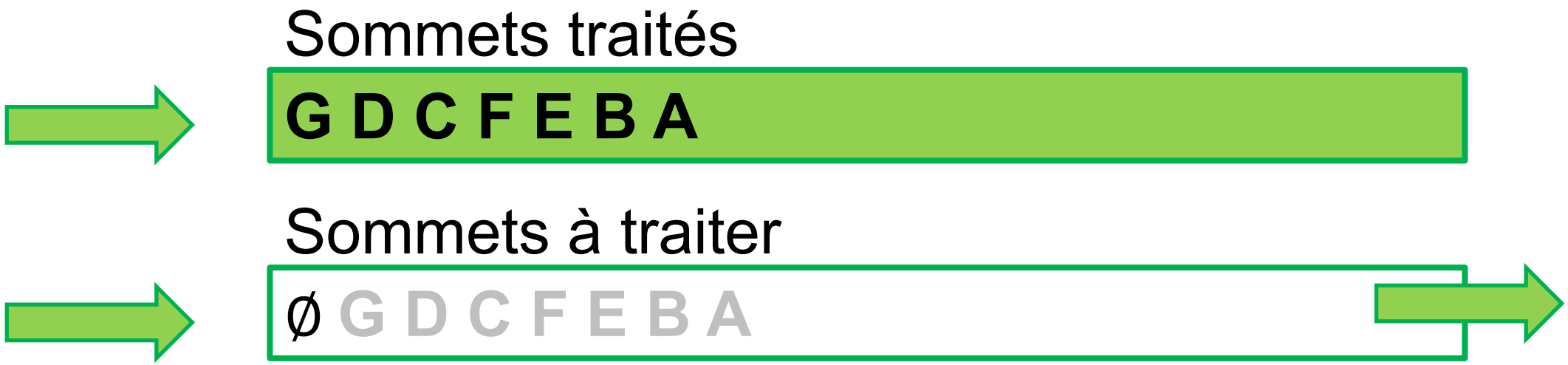
- **Notion de files**



First in, First out : FIFO

2. PRINCIPE DU PARCOURS EN LARGEUR

- **Notion de files**



— « Créer » : créer la file vide ;

— « Enfiler » : ajouter un élément **à la fin de la file**;

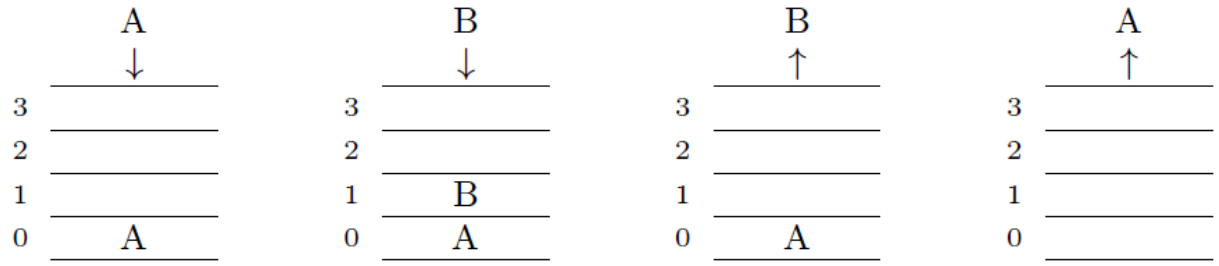
— « Défiler » : renvoyer le prochain élément de la file (premier de celle-ci), et le retirer de celle-ci ;

— « Tester si la file est vide » ;

ANNEXE module Queue

```
import queue as q
```

1.Pile LIFO (Last In First Out)

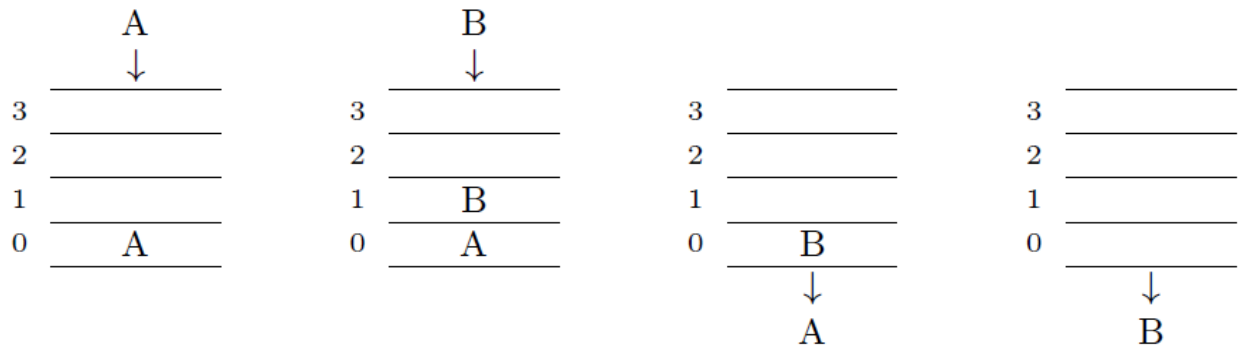


	Implémentation naïve avec le type list	méthodes du module queue
Création pile (vide)	`def creer_pile():` `  return [ ] # création pile vide`	`pile = q.LifoQueue()`
Empiler	`def empiler(elt, pile):` `  pile.append(elt)`	`pile.put(elt)` cette méthode (empiler) ajoute elt sur la pile
Dépiler	`def depiler(pile):` `  return pile.pop() # retire et renvoie le dernier element`	`pile.get()` cette méthode (désempiler) enlève le dernier élément de la pile et le renvoie
Test pile vide	`def pile_vide(pile):` `  return pile == [] # booléen`	`pile.empty()` La pile est-elle vide ?
Taille	`def hauteur(pile):` `  return len(pile)`	`pile.qsize()` cette méthode renvoie la taille de la pile

ANNEXE module Queue

```
import queue as q
```

2.File FIFO (First In First Out)



	Implémentation naïve avec le type list	méthodes du module queue

Création file (vide)	`def creer_file():`
	`  return [] # création file vide`
		`file = q.Queue()`
		ou en version simplifiée :
		`file = q.SimpleQueue()`

Emfiler	`def enfiler(elt, file):` `  file.append(elt)`	`file.put(elt)` cette méthode (enfiler) ajoute elt à la file
Défiler	`def defiler(file):` `  return file.pop(0) # retire et renvoie le premier element`	`file.get()` cette méthode (défiler) enlève premier elt de la file et le renvoie
Test file vide	`def file_vide(file):` `  return file == [] # booléen`	`file.empty()` La file est-elle vide ?
Taille	`def longueur(file):` `  return len(file)`	`file.qsize()` cette méthode renvoie la taille de la file

2. PRINCIPE DU PARCOURS EN LARGEUR

```
## Exemple utilisation module Queue pour les files
import queue as q
essaifile = q.Queue()  # initialise une file
essaifile.put('A')  # "enfile" un élément de valeur 1 ici en
fin de file
essaifile.put('B')  # "enfile" un élément de valeur 2
essaifile.put('C')  # "enfile" un élément de valeur 3
essaifile.put('D')  # "enfile" un élément de valeur 4
element_de_tete_de_file = essaifile.get()  # "défile"
l'élément de tête, ici 'A'
if not essaifile.empty():  # test de file vide
    print('file non vidée')
```

2. PRINCIPE DU PARCOURS EN LARGEUR

Initialiser une file_a_traiter avec le sommet de départ s;

Initialiser un dictionnaire dict_ sommets_decouverts avec False comme valeur pour chaque clé sauf pour le sommet de départ s (sommet du graphe étudié);

Initialiser une liste_ sommets_traites;

tant que file_a_traiter n'est pas vide **faire**

    sommet_courant ← défiler(file_a_traiter) ;

    pour chacun des voisins v du sommet_courant **faire**

        si v n'appartient pas au dictionnaire dict_sommets_decouverts **alors**

            Enfiler v à la file_a_traiter;

            Modifier la valeur boolenne de v à True dans dict_sommets_decouverts ;

        Fin si

    Ajouter le sommet_courant à la liste liste_ sommets_traites ;

    Fin pour

Fin tant que

Retourner liste_ sommets_traites

3. PROPRIÉTÉS

3-1- Complexité

Initialiser une file_a_traiter avec le sommet de départ s;

Initialiser un dictionnaire dict_ sommets_decouverts avec False comme valeur pour chaque clé sauf pour le sommet de départ s (sommet du graphe étudié);

Initialiser une liste_ sommets_traites;

tant que file_a_traiter n'est pas vide **faire**

    sommet_courant ← défiler(file_a_traiter) ;

    pour chacun des voisins v du sommet_courant **faire**

        si v n'appartient pas au dictionnaire dict_sommets_decouverts **alors**

            Enfiler v à la file_a_traiter;

            Modifier la valeur boolenne de v à True dans dict_sommets_decouverts ;

        Fin si

    Ajouter le sommet_courant à la liste liste_ sommets_traites ;

    Fin pour

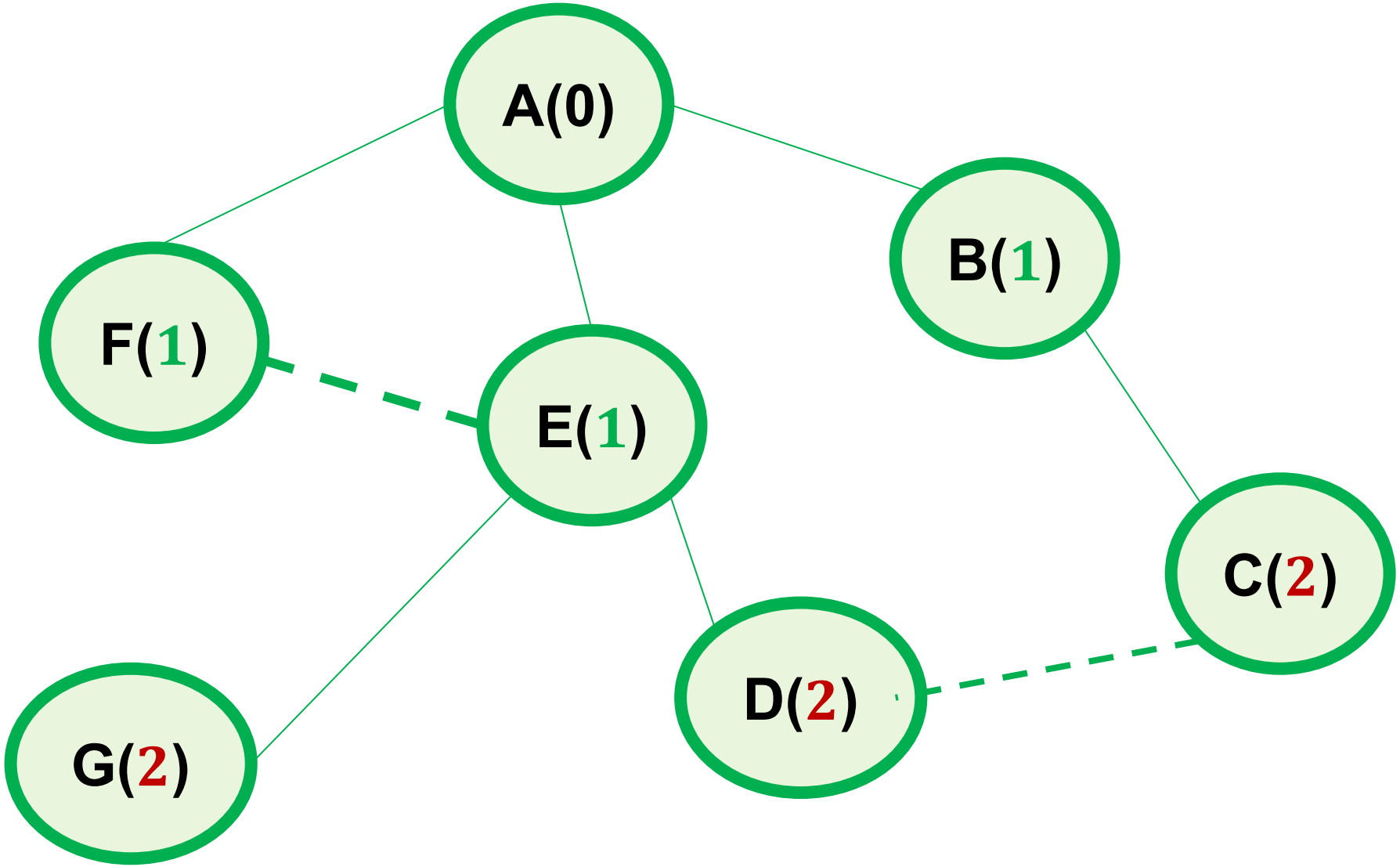
Fin tant que

Retourner liste_ sommets_traites

Sommet s parcouru au plus 1 fois,
Arête {s,s'} parcourue au plus 1 fois

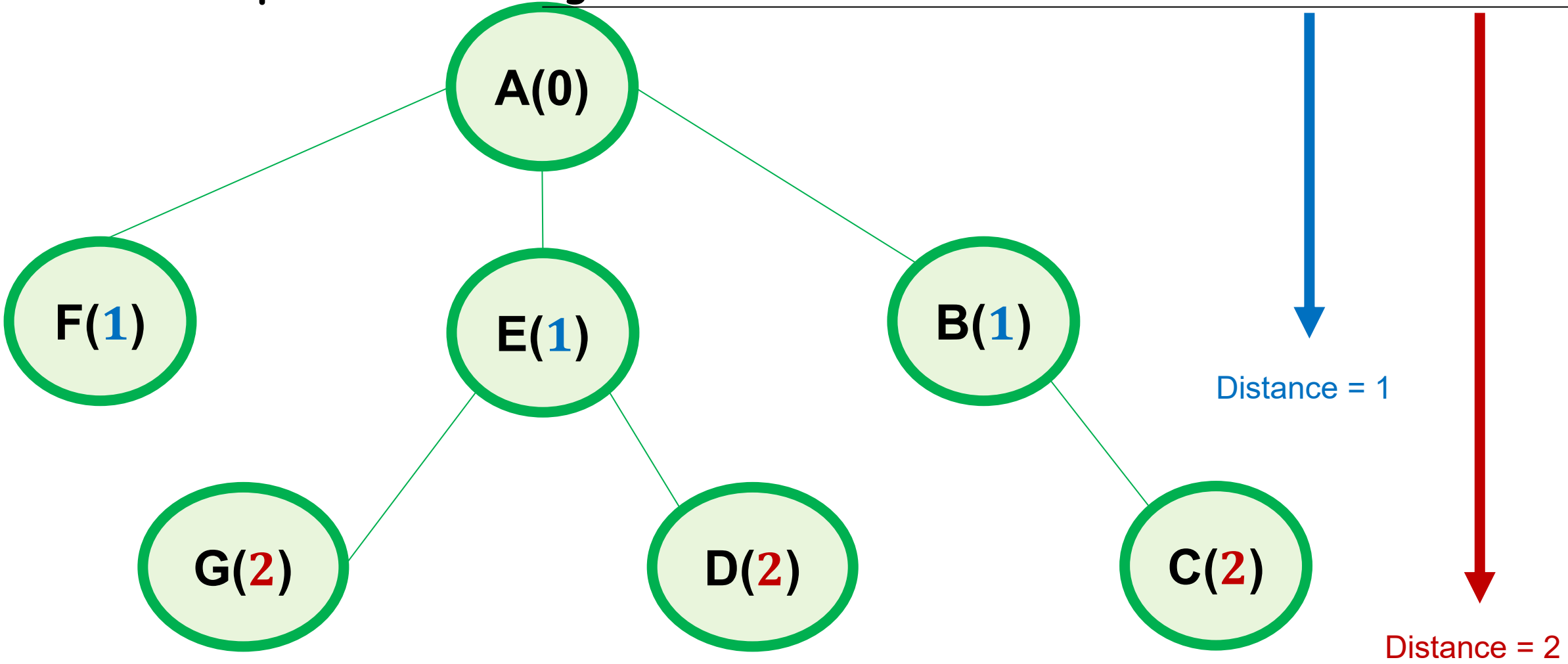
$$\mathcal{O}(n_{sommet} + n_{arête})$$

3-2- Plus court chemin



3. PROPRIÉTÉS

3-3- Arbre de parcours en largeur



sous-ensemble du graphe d'origine, qui conserve la propriété de plus court chemin

S10- APPLICATIONS Problème de parenthésage

On cherche à vérifier si une chaîne de caractères `ch_car` est bien parenthésée, c'est-à-dire si le nombre ainsi que l'ordre des parenthèses ouvrantes '(' et fermantes ')' sont corrects.

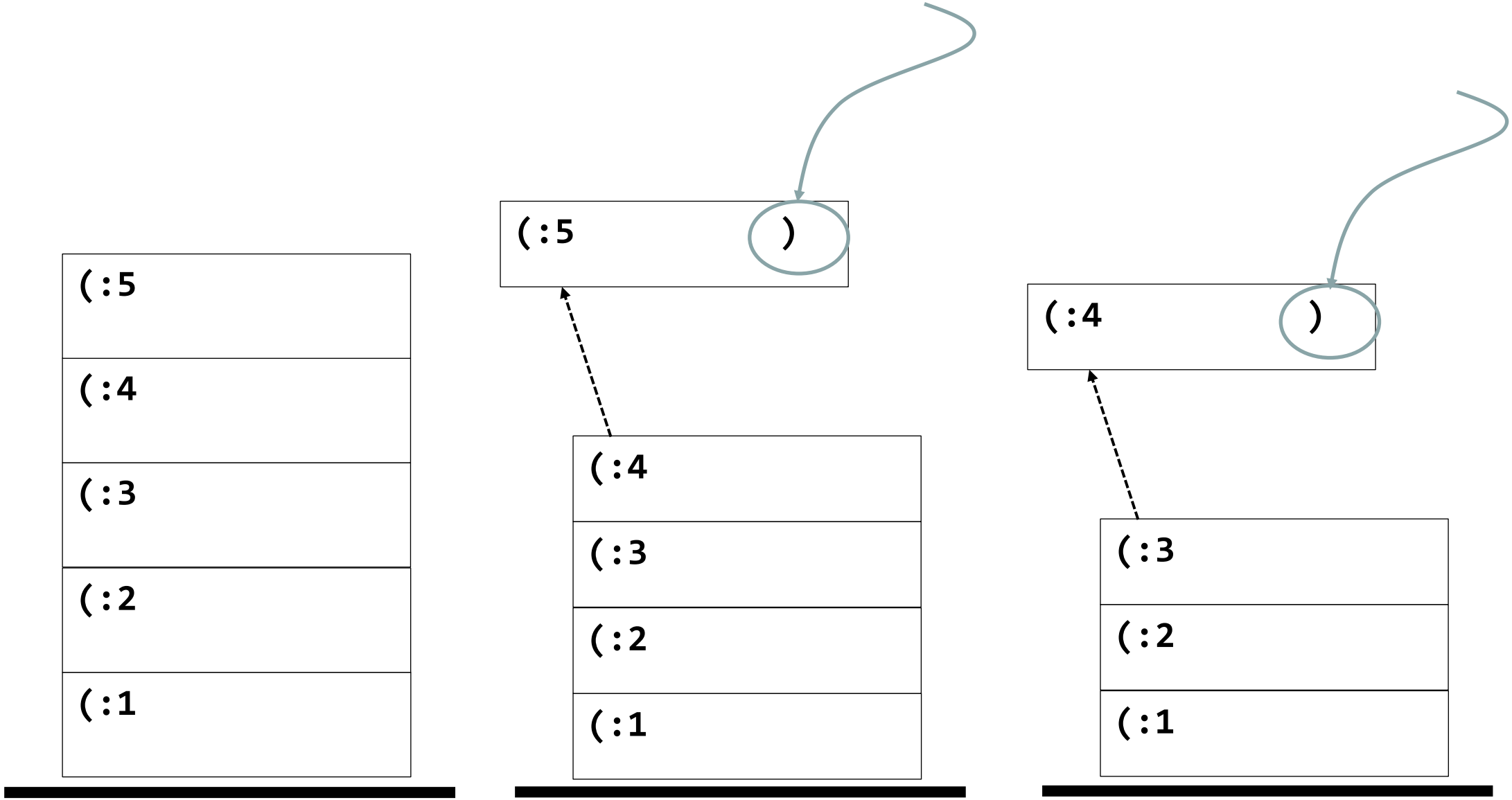
Lorsqu'on examine une parenthèse ouvrante, on empile cette parenthèse.
Lorsqu'on examine une parenthèse fermante, deux cas peuvent se produire:

- la pile est vide : alors on a plus de parenthèses fermantes qu'ouvrantes
  → ce qui signifie que le mot n'est pas bien parenthésé.
- la pile est non vide : on dépile alors l'élément en haut de la pile (une parenthèse ouvrante)

À la fin du traitement de la chaîne, on vérifie que la pile est vide. Si ce n'est pas le cas, le mot possède plus de parenthèses ouvrantes que fermantes
→ le mot n'est donc pas bien parenthésé.

Ecrire une fonction **parenthesage** qui admet comme argument une chaîne de caractères `ch_car` et qui retourne `True` si la chaîne est bien parenthésée et `False` sinon.

S10- APPLICATIONS Problème de parenthésage



S10- APPLICATIONS Problème de parenthésage

```
def parenthesage_1(ch_car):
    pile_comptage = []
    for caract in ch_car:
        if caract == '(':
            empiler(pile_comptage,caract)
        elif caract ==')’:
            depiler(pile_comptage)
    return pile_vide(pile_comptage)
```

```
def parenthesage_2(ch_car):
    pileO,pileF = [],[]
    for caract in ch_car:
        if caract == '(‘:
            empiler(pileO,caract)
    elif caract ==')’:
            empiler(pileF,caract)
    return len(pileO)==len(pileF)
```

4. APPLICATIONS

4.1. Outils

- degre_non_oriente
- degre_sortant
- degre_rentrant

4.2. Fonction de base

- convertir_matrice_dict
- convertir_matrice_pond
- voisins_dict

4.3. Parcours en largeur simple

- parcours_largeurDict

4. APPLICATIONS

```
g1_NO_d = {0:[3,4],1:[2,3,4],2:[1,4],3:[0,1], 4:[0,1,2]}
g1_NO_m = [[0,0,0,1,1],[0,0,1,1,1],[0,1,0,0,1],[1,1,0,0,0],[1,1,1,0,0]]

g2_NO_d = {0:[1,4],1:[0,2,3,4],2:[1], 3:[1,4], 4:[0,1,3]}
g2_NO_m = [[0,1,0,0,1],[1,0,1,1,1],[0,1,0,0,0],[0,1,0,0,1],[1,1,0,1,0]]

g3_O_d = {1:[2,4],2:[5,6],3:[2,6], 4:[5], 5:[3,6],6:[1,3]}

g4_O_d = {1:[2,6,7],2:[3,4],3:[1,5], 4:[6,7], 5:[1,4],6:[7], 7:[]}

g5_OP_m = [[0,1, 2,0, 0, 0], [0, 0, 12, 0, 0, 0], [0, 0, 0, 23, 24, 25], [0, 0, 0, 0, 34, 0],
[0, 0, 0, 0, 0, 45], [50, 0, 0, 0, 0, 0]]

g6_NO_d = {0:[2], 1:[2], 2:[0, 1, 3, 4], 3:[2, 4], 4:[2, 3]}
g6_NO_m = [[0,0,1,0,0], [0,0,1,0,0],[1,1,0,1,1],[0,0,1,0,1],[0,0,1,1,0]]
```