TD : parcours arborescent

1 Constitution du parcours

Exercice 1 *Écrivez un programme récursif qui énumère tous les sous ensembles de l'ensemble des entiers naturels inférieurs à n.*

```
   def exo1 (n: int=6, prefix: List[int]=[]) -> None:  ...
```

```
   python /.../exo1.py
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 1, 0] ...
```

Exercice 2 *Écrivez un programme itératif qui énumère tous les sous ensembles de l'ensemble des entiers naturels inférieurs à n.*

```
   def exo2 (n: int=6) -> None:  ...
```

```
   python /.../exo2.py
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 1, 0] ...
```

Exercice 3 *Modifiez le programme de l'exercice 1 de façon à ce qu'il affiche également les nœuds internes de l'arbre sous forme de questions. Vérifiez qu'il y a bien 2^n feuilles et $2^n - 1$ nœuds internes.*

```
   def exo3 (n: int=6, prefix: List[int]=[]) -> None:  ...
```

```
   python /.../exo3.py
[0] ou [1] ?
[0, 0] ou [0, 1] ?
[0, 0, 0] ou [0, 0, 1] ?
[0, 0, 0, 0] ou [0, 0, 0, 1] ?
[0, 0, 0, 0, 0] ou [0, 0, 0, 0, 1] ?
[0, 0, 0, 0, 0, 0] ou [0, 0, 0, 0, 0, 1] ?
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1] ...
```

Dans toute la suite de l'énoncé on considérera la graphe suivant :

```
graph_1: Dict[int, List[int]] =
{0: [1, 3], 1: [0, 4], 2: [3, 5], 3: [0, 2, 4, 5], 4: [1, 3, 5], 5: [2, 3, 4]}
```

Exercice 4 *Modifiez le programme de l'exercice 1 de façon à ce qu'il n'affiche que les ensembles stables (qui ne contiennent aucune arête. Combien compte-t-il de nœuds internes ?*

```
def exo4 (n: int=6, prefix: List[int]=[], graph: Dict[int, List[int]]=graph_1) ->
None:  ...
```

```
python /.../exo4.py
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 0] ...
```

2 Pruning

Exercice 5 *Modifiez le programme de l'exercice 1 de façon à ce que non seulement il n'affiche que des ensembles stables, mais qu'il cesse d'explorer le sous-arbre dès lors qu'il y a une arête. Vérifiez qu'on passe de 63 à 32 nœuds internes (par exemple en affichant ceux-ci comme dans l'exercice 3).*

```
def exo5 (n: int=6, prefix: List[int]=[], graph: Dict[int, List[int]]=graph_1) ->
None:  ...
```

```
python /.../exo5.py
[0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 0] ...
```

Exercice 6 *Desssinez (à la main ou via un logiciel) l'arbre parcouru par cet algorithme.*

Exercice 7 *Modifiez le programme de l'exercice 5 de façon à ce qu'il n'affiche que les stables de taille maximale.*

```
def exo7 (graph: Dict[int, prefix: List[int], n: int, List[int]]=graph_1,
stable_max: List[List[int]]): -> None:  ...
```

```
python /.../exo7.py
[[1, 0, 1, 0, 1, 0]]
```

Exercice 8 *(Plus difficile) Modifiez le programme de l'exercice 7 de façon à ce qu'il cesse de chercher dans un sous-arbre si il n'est pas possible de faire mieux que le maximum déjà enregistrer. Vérifiez que* ***si on explore les 1 avant les 0****, cela réduit le nombre de nœuds internes à 14 !*