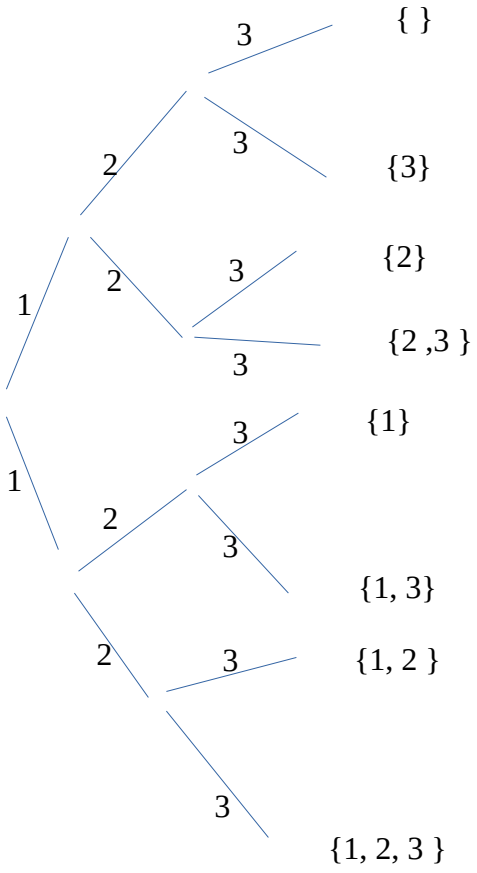
INTRODUCTION

La notion d'Ensembles est un outil très important des Mathématiques discrètes. Pour un ensemble E donné, son ensemble des parties peut être construit à l'aide du graphe des parties.

Dans ce rapport nous nous attarderons sur la construction du programme informatique en Langage Python qui calcule le rang d'un élément donné dans l'ensemble des parties d'un ensemble E; nous présenterons aussi la méthode algorithmique et le programme python permettant de faire le travail inverse c'est à dire trouver l'élément de l'ensemble des parties correspondant à un rang donné. Pour les explications relatives à la construction du programme, nous utiliserons l'ensemble F de 3 éléments, avec F = {1, 2, 3}. Pour tester notre programme, nous utiliserons l'ensemble E de 2^{24} = 65536 éléments, avec E = {1, 2, 3, …., 2^{24}}.

I. PRÉSENTATION DU GRAPHE DES PARTIES ET EXPLICATIONS

- Considérons l'ensemble F de 3 éléments, avec F = {1, 2, 3}. Alors P(F) désigne l'ensemble des parties de F et P(F) contient 2^n ou n désigne la taille de l'ensemble F. Alors le cardinal de P(F) est |P(F)| = 2^3 = 8 éléments. Utilisons alors le graphe des parties pour lister les éléments de P(F).

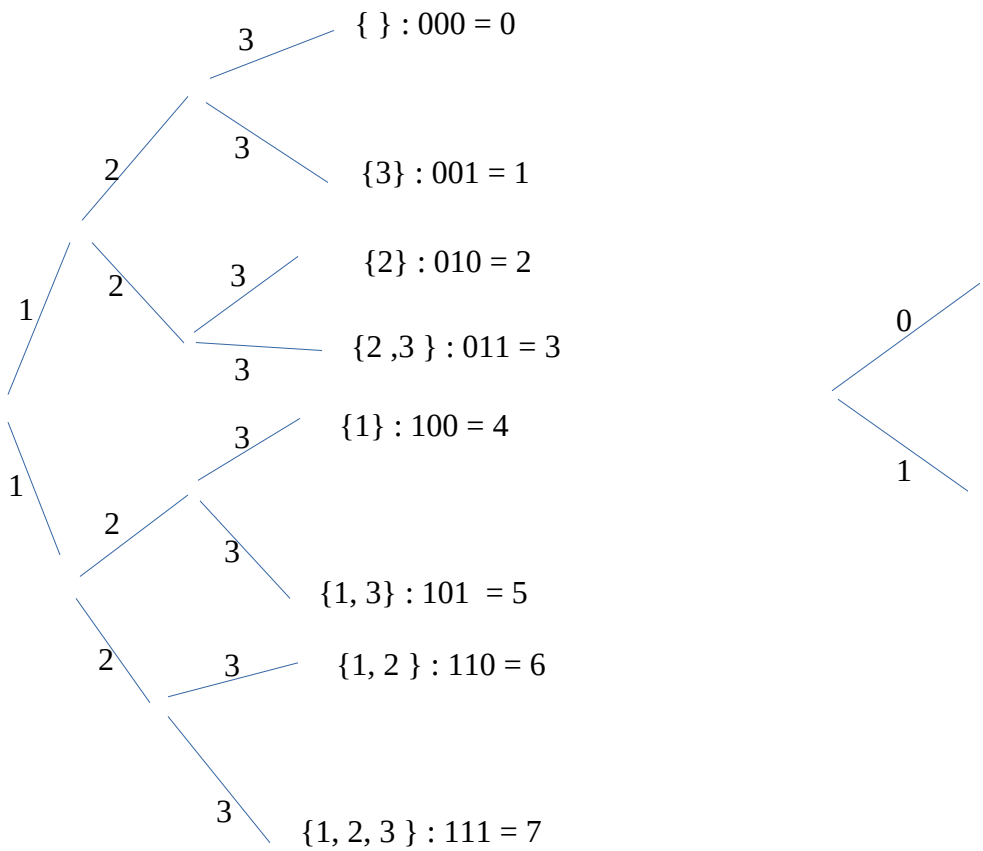


GRAPHES DES PARTIES DE F

On peut alors dire que: P(F) = {{ }, {3}, {2}, {2, 3}, {1}, {1, 3}, {1, 2}, {1, 2 , 3} }, avec { } au rang 1, {3} au rang 2, {2} au rang 3 , {2, 3} au rang 4 , {1} au rang 5 , {1, 3} au rang 6 , {1, 2} au rang 7, {1, 2, 3} au rang 8

- **Explications de l'idée derrière le code:**

Conformément à la logique présentée par la figure ci dessous représentant le graphe des parties de F, les branches portent les éléments de F et les feuilles portent les éléments de P(F). Ici chaque élément de P(F) est encodée grâce au 0 et 1: on encode alors toutes les branches supérieures avec «0» et toutes les branches inférieures avec «1». On obtient alors grâce à ce procédé d'encodage tous les éléments de P(F) représentés par des nombres binaires. On peut dès lors constater que l'équivalent en décimal plus un, d'un élément de P(F) encodé nous donne exactement la position de cet élément dans l'ensemble des parties P(F) .



Exemple: Soit la position 5 passé en paramètre. Trouvons l'élément de P(F) à la position 5:

- Vérifier la condition: $0 < rang <= 2^n$, ou n=|F|
- si cette condition est vérifiée: rang = rang – 1 => rang = 4
    - convertit rang = 4 en binaire rang = $4=(100)_2$
    - pour i allant de (nombre de bit de rang) à 1
        si rang[i]=1 alors ajouter F[n] à élément
        n = n-1
    - retourner élément = {1}

Soit l'élément A = {1, 2}. Trouvons son rang dans P(F)

- Vérifier si tous les éléments de A sont bien présents dans F
- si cette condition est vérifiée:
    pour i allant de 1 à n:
        si F[i] présent dans A alors ajouter «1» dans Rang
        sinon ajouter «0» dans Rang
    Rang = 110
- convertir rang en décimal: Rang = 6
- retourner Rang = Rang + 1 = 7

II. PSEUDO CODE

- **Détermination du rang d'un Élément dans l'ensemble des parties:**

  Récupérer (élément)

  vérifier(élément)

  générer(nombre binaire)

  nombre décimal = convertir_en_décimal (nombre binaire)

  rang = (nombre décimal + 1)

  retourner (rang)

- **Détermination d'un Élément de l'ensemble des parties à partir de son rang:**

  Récupérer (rang)

  vérifier(rang)

  rang = rang – 1

  nombre binaire = convertir_en_binaire(rang)

  générer(élément)

  retourner (élément)

III. PROGRAMME PYTHON

- **Détermination du rang d'un Élément dans l'ensemble des parties:**

```
def binary_to_decimal(binary_num: str):
    decimal = int(binary_num, 2)
    return decimal

def element_to_position(elements_list, lenght, element):
    binary_num = ""
    for i in element:
        if not i in elements_list:
            return f"BAD ELEMENT: {i} not present in the general set.\n"
    for i in elements_list:
        if i in element:
            binary_num += "1"
```

```
        else:
            binary_num += "0"
    return binary_to_decimal(binary_num)+1
```

- **Détermination d'un Élément de l'ensemble des parties à partir de son rang:**

```
def decimal_to_binary(number: int):
    binary_num = bin(number)
    binary_num = binary_num[2:]
    return binary_num

def position_to_element(elements_list, lenght: int, position: int):
    if(position <= 2**lenght and position>0):
        position -= 1
        binary_num = decimal_to_binary(position)
        element=[]
        j = lenght
        for i in range(len(binary_num)-1, -1,-1):
            if(binary_num[i]=="1"):
                element.insert(0, elements_list[j])
            j -= 1
        element = tuple(element)
        return element
    else:
        return "POSITION INVALIDE"
```

CONCLUSION

Ce TP a permis d'implémenter de manière efficace le concept de «Ranking» sur les éléments de l'ensemble des parties d'un ensemble E donné. La méthode d'encodage binaire qui a été utilisée est très efficace, car elle ne cherche pas à construire l'ensemble des parties. La construction de l'ensemble des parties pourrait s'avérer très difficile pour un ensemble E de 2^{16} éléments ($|P(E)| = 2^{2^{16}}$ éléments) en raison d'une complexité algorithmique plus accrue.