Exercice 1:

1. — racine : A

— 8 feuilles

— taille : 16

- profondeur : 3

2. — parcours en largeur : A B C D E F N G H I J K L O P M

— parcours en profondeur : A B C F K L N D G O P H I M E J

Exercice 2:

— Windows : taille = 22; hauteur = 2

— Linux : taille = 35; hauteur = 3

Exercice 3:

1. liste

```
dictionnaire = ['arbre', 'arbres', 'arbitre', 'arbitrent', '
    arbitrer', 'arbitres', 'arbitrez', 'arbitrons', 'binaire', '
    binaires', 'binette', 'binettes', 'bio', 'empile', 'empilent' '
    empiler', 'empiles', 'empilez', 'empilons', 'exact', 'exacte', '
    exactes', 'exacts']
```

23 éléments dans la liste et 22 tests pour trouver exactes.

2. arbre

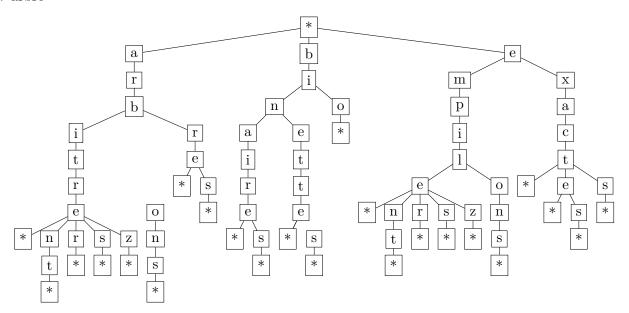


Figure 1 – Dictionnaire

8 tests pour trouver exactes.

Exercice 4:

1. Arbre (presque complet)



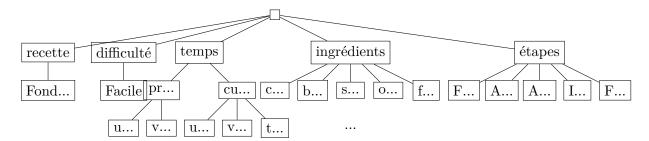


FIGURE 2 – Recette du fondant au chocolat

- 2. Ouvrir le fichier recette.py. L'arbre a été construit avec la classe Noeud.
- 3. BFS

```
1
   def BFS(arbre: Noeud)->list:
       visites = []
2
       f = File()
3
       f.enfiler(arbre)
4
       while not f.est_vide():
5
6
          en_cours = f.defiler()
          for noeud in en_cours.fils:
7
              visites.append(noeud.valeur)
8
              f.enfiler(noeud)
9
10
       return visites
11
```

4. affichage

```
def affiche(arbre: Noeud, decalage: str = "", affichage: str = "")
    ->str:
    """
    affiche l'arbre avec un décalage de 4 espaces pour chaque niveau
    """
    affichage += decalage + str(arbre.valeur)+"\n"
    for f in arbre.fils:
        affichage = affiche(f, decalage+" ", affichage)
    return affichage
```

