Partie 1

Activité : recherche dichotomique

1.1 Zero d'une fonction

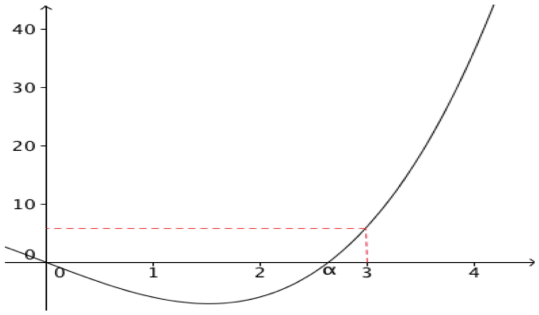

FIGURE 1 – f(x) = x**3 - 7x*

1.2 Recherche dans une liste de mots

1.2.1 Recherche séquentielle

```
%%timeit
def recherche_mot(X,mots):
    """
    recherche un mot dans une liste et renvoie l'indice si le mot est trouvée, -1 sinon
    Params :
    -------------------
    X : str, mot à trouver
    mots : list, une liste de mots, dans un ordre quelconque.
    Sortie :
    ------
    j : int, indice dans la liste
    Principe :
    --------
    on parcourt la liste avec une boucle non bornée, tant que X n'est pas trouvé dans la liste
    on augmente la valeur de j à chaque nouvelle itération
    """
    j = 0
    n = len(mots)

    while j<n and X!=mots[j]:
        j += 1
    if j==n : return -1
    return j

recherche_mot('tracts',mots)
# affiche
1.88 ms ± 46.7 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

1.2.2 Recherche dichotomique

```
%%timeit
def recherche_dicho_mot(X,mots):
```

```
    """recherche dans une liste de mots un mot X
    Params:
    ------
    mots : list, contient des mots triés
    X : str, mot à trouver
    Return :
    --------
    milieu (indice de mot dans la liste) si X est présent dans la liste
    -1 sinon"""
    # on initialise les indices début et fin aux extrémités de la liste
    gauche = 0
    droite = len(mots)
    trouve = False

    while gauche <= droite and not trouve:
        # On se place au milieu de la liste
        milieu = (gauche + droite) // 2 # il, s'agit d'une division entière

        if mots[milieu] == X:
            #print(élément, "trouvé à l'indice:", milieu , liste[milieu])
            trouve = True
            # on arrête la boucle
            #début = fin - 1
        elif mots[milieu] < X:
            gauche = milieu + 1
        else:
            droite = milieu - 1
    #print(élément, "non trouvé")
    if not trouve : return -1
    return milieu

recherche_dicho_mot('tracts',mots)
# affiche
2.48 µs ± 45.9 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

Partie 2

Document : comparaison des fonctions de n

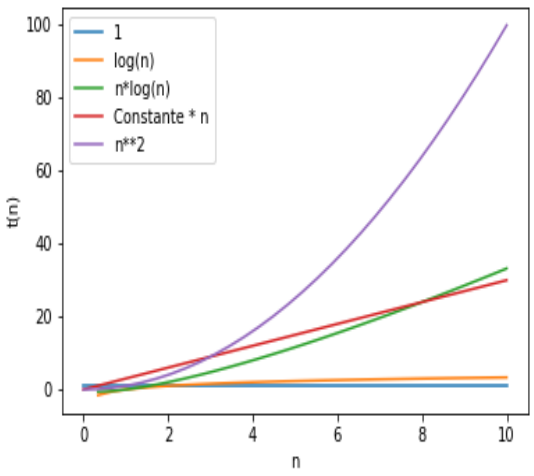

FIGURE 2 – n**2 domine les autres fonctions

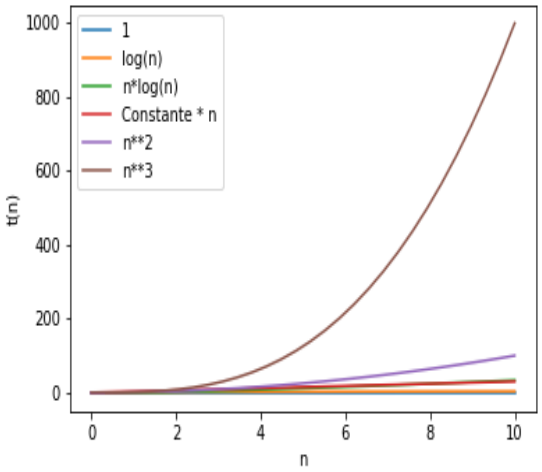

FIGURE 3 – n**3 domine les autres fonctions

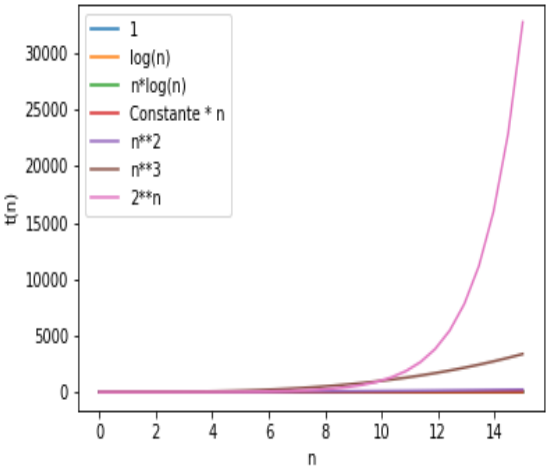

FIGURE 4 – 2**n domine les autres fonctions