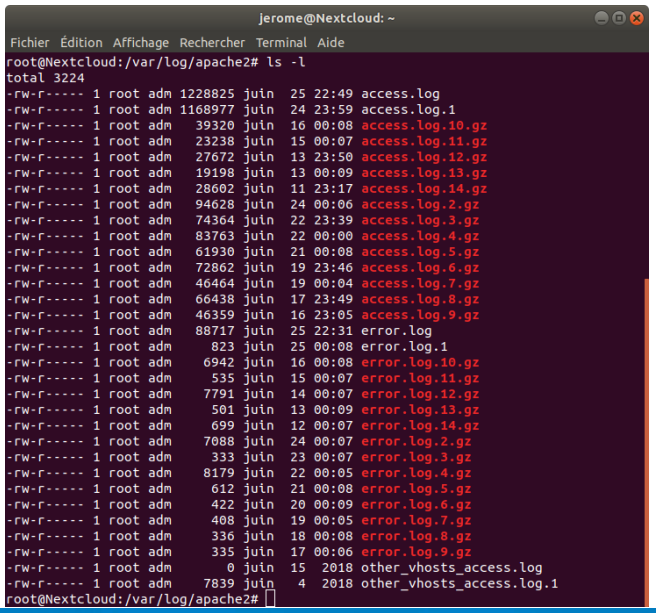
Recherche d'occurrences

Pierre Duret, Stéphane Rodriguez, Jérôme Signouret

27/06/2019



LaTeX

Table des matières

Première partie
Le travail à faire

1 Principe

On cherche à trouver un motif dans un fichier comportant un grand nombre d'entrées.

On commencera par rechercher un motif constitué d'un seul élément puis on généralisera à un motif fait de plusieurs éléments.

On calculera puis on expérimentera le "nombre d'opérations" que l'on réalise lors de la recherche et sa variation en fonction de la taille des données.

On pourra essayer d'introduire la notion de performance d'un algorithme.

2 Accroche possible

On souhaite analyser un fichier de log d'un serveur pour savoir s'il y a eu des tentatives de connections illicites et si elles ont échoué ou réussi.

2.1 Ce que l'on sait faire en seconde

Un fichier *log* contient des messages enregistrés par le système ou les applications. Cela permet d'améliorer les configs, déboguer, comprendre le fonctionnement...

Un fichier *log* est un fichier texte dont les caractères sont encodés par défaut en UTF-8 sur nos systèmes. Il sont est comme des fichiers *csv*, une ligne par enregistrement (jusqu'au caractère <fin de ligne>) les données étant séparées par des champs spéciaux (virgule, tabulation, espace, point-virgule).

— Ouvrir le fichier *access.txt* avec le tableur, choisir le caractère de séparation "espace" à l'import.
— Utiliser les fonctionnalités du tableur pour dénombrer le nombre de fois où apparaît le motif "GET"

3 La partie du programme NSI première concernée

Contenus	Capacités attendues	Commentaires
Parcours séquentiel d'un tableau	Écrire un algorithme de recherche d'une occurrence sur des valeurs de type quelconque.	On montre que le coût est linéaire.

4 Activité : Rechercher un caractère

4.1 Trouvez le premier

Vous devez écrire un algorithme de fonction qui renvoie la position du premier caractère recherché.
On considère qu'une chaîne de caractères est un tableau dont les éléments sont chacun un des caractères constituant la chaîne.
La chaîne de caractère est contenue dans la variable *chaine*.

4.2 Trouvez les tous

Vous devez écrire un algorithme de fonction qui renvoie la position de chaque caractère recherché.

4.3 Comptez le nombre de lignes

Vous devez écrire un algorithme de fonction qui renvoie le nombre de caractère recherché et trouvé.
En choisissant correctement le caractère recherché, on obtient le nombre de lignes dans le fichier.

4.4 Combien de caractères lus ?

En reprenant votre algorithme précédent, vous devez utiliser une variable qui compte le nombre de fois que vous avez effectué une lecture pour trouver le caractère recherché.
Cette fonction doit renvoyer le nombre de caractères lus.

4.5 Lien entre le nombre de caractères lus et la taille de la chaîne

On se pose la question de savoir si l'algorithme est plus ou moins efficace. C'est à dire : le nombre de lectures va-t-il doubler quand la taille de la chaîne va doubler ?
Proposez une méthode qui permet de répondre à cette question en TP informatique.

5 Le cours

5.1 Objectifs de la partie cours

— Présentation de l'objectif de l'activité globale
— maîtriser les 3 structures de bases (si, pour, tant que)

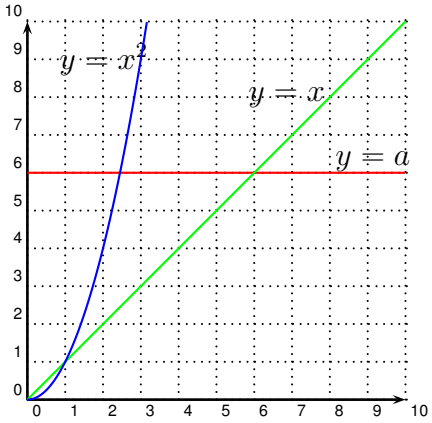
— Introduction de critères de comparaison d'algorithme (taille mémoire et/ou temps de calcul)
— Être capable reconnaître des évolutions en fonction des données à traiter.
— Être capable de choisir la (ou les) donnée(s) à analyser.
— Être capable d'analyser la complexité (théorique).
— Être capable d'analyser la complexité (pratique).
— Introduire la notion de cas (min, max, moyen).

5.2 Présentation de l'objectif de l'activité globale

Critères de comparaison d'algorithme (taille mémoire et/ou temps de calcul)

5.2.1 Courbes

— $y = a$: coût constant
— $y = x$: coût linéaire ou proportionnel
— $y = x^2$: coût quadratique



5.2.2 Partie théorique

P0

1. Combien de fois « Au revoir » va-t-il s'afficher ?
2. Combien de fois « On ne peut pas prendre la racine d'un réel négatif ! » va-t-il s'afficher ?
3. Combien de fois « La racine de ... est .... va-t-il s'afficher ?

```
1 from math import sqrt
2 x = 2
3 if x >= 0:
4     y = sqrt(x)
5     print("La racine de {:.2f} est {:.3f}".format(x, y))
6 else:
7     print("On ne peut pas prendre la racine d'un reel negatf !")
8 print("\nAu revoir")
```

P1

1. Combien de fois « Au revoir » va-t-il s'afficher ?
2. Combien de fois « On ne peut pas prendre la racine d'un réel négatif ! » va-t-il s'afficher ?

3. Combien de fois « La racine de ... est .... va-t-il s'afficher ?

```
from math import sqrt
from random import *

x = float(input())
if x >= 0:
    y = sqrt(x)
    print("La racine de {:.2f} est {:.3f}".format(x, y))
else:
    print("On ne peut pas prendre la racine d'un reel negatf !")
print("\nAu revoir")
```

P2

1. Combien de fois « Au revoir » va-t-il s'afficher ?
2. Combien de fois « On ne peut pas prendre la racine d'un réel négatif ! » va-t-il s'afficher ?
3. Combien de fois « La racine de ... est .... va-t-il s'afficher ?

```
from math import sqrt
from random import *

x = float(randint(-10,10))
if x >= 0:
    y = sqrt(x)
    print("La racine de {:.2f} est {:.3f}".format(x, y))
else:
    print("On ne peut pas prendre la racine d'un reel negatf !")
print("\nAu revoir")
```

P3

1. Quelles sont les données ?
2. Combien de fois « L'élément cherche n'est pas dans la sequence. » va-t-il s'afficher ?
3. Combien de fois « Fin » va-t-il s'afficher ?

P4

1. Combien de fois « L'élément cherche n'est pas dans la sequence. » va-t-il s'afficher ?
2. Combien de fois « Fin » va-t-il s'afficher ?

```
from easygui import integerbox, msgbox

uneSequence = list(input())
print(uneSequence)

cible = integerbox("Entrez un entier :","")
for i in uneSequence:
    print(int(i))
    if int(i) == cible:
        break
    else:
```

```
12          msgbox("L'element cherche n'est pas dans la sequence.")
13  msgbox("Fin")
```

P5

1. Combien de fois « facteur trouvé : ..... » va-t-il s'afficher ?

```
1  # -*- coding: utf-8 -*-
2  from easygui import integerbox, msgbox
3  d=2
4  n=14
5  while n>1:
6      while n%d==0:
7          n = n//d
8          print("facteur trouvé:", d)
9      d += 1
```

P6

1. Combien de fois « facteur trouvé : ..... » va-t-il s'afficher ?

```
1  from easygui import integerbox, msgbox
2  d=2
3  n=integerbox("Entrez un entier :","")
4  while n>1:
5      while n%d==0:
6          n = n//d
7          print("facteur trouvé:", d)
8      d += 1
```

P0 bis

1. Sur le programme P0, modifier le programme afin de donner la réponse à la question : Combien de fois « La racine de ... est .... va-t-il s'afficher ?

5.2.3 Partie pratique

Même question mais les élèves devront répondre aux questions en faisant des tests et devront insérer des compteurs pour avoir directement les réponses.

6 Activité : Rechercher un motif

6.1 Trouvez le premier

Un motif est une chaîne de caractères que l'on souhaite trouver dans un fichier.
Vous devez écrire un algorithme de fonction qui renvoie la position du premier motif recherché.
On considère qu'une chaîne de caractères est un tableau dont les éléments sont chacun un des caractères constituant la chaîne.
La chaîne de caractère est contenue dans la variable *chaine*.

6.2 Trouvez les tous

1. Vous devez écrire / compléter / analyser un algorithme de fonction qui renvoie le nombre d'occurrences du motif recherché dans un fichier.
2. Vous devez chercher à évaluer le coût en nombre d'opérations de l'exécution de la recherche.
3. Vous devez observer les variations éventuelles du coût :
   — en fonction de la taille du fichier
   — en fonction de la longueur du motif

6.3 Combien de caractères lus ?

En reprenant votre algorithme précédent, vous devez utiliser une variable qui compte le nombre de fois que vous avez effectué une lecture pour trouver le motif recherché.
Cette fonction doit renvoyer le nombre de caractères lus.

6.4 Lien entre le nombre de caractères lus et la taille de la chaîne

On se pose la question de savoir si l'algorithme est plus ou moins efficace. C'est à dire : le nombre de lectures va-t-il doubler quand la taille de la chaîne va doubler ?
Proposez une méthode qui permet de répondre à cette question en TP informatique.

6.5 Lien entre le nombre de caractères lus et la taille du motif

On se pose la question de savoir si l'algorithme est plus ou moins efficace. C'est à dire : le nombre de lectures va-t-il doubler quand la taille de la chaîne va doubler ?
Proposez une méthode qui permet de répondre à cette question en TP informatique.

7 Exercices

7.1 Compter le nombre de caractère "e"

À vous d'écrire l'algorithme permettant de trouver le nombre de caractères "e" présent dans le fichier.
Vous pourrez essayer de coder en Python votre algorithme.

7.2 Compteur de performance

À vous d'ajouter un compteur qui s'incrémente à chaque affectation dans l'algorithme suivant.

```
1 fonction maFonction(laliste):
2    Pour i de 0 à len(laliste) - 1
3       x = laliste[i]
4       j = i
5       Tant que j > 0 et laliste[j - 1] > x
6          laliste[j] = laliste[j - 1]
7          j = j - 1
```

```
        laliste[j] = x
liste = [2,6,8,1,0,5]
maFonction(liste)
```

7.3 Évolution de l'algorithme précédent

Comment se comporte l'algorithme précédent quand la longueur de liste s'allonge ? Reprendre la méthode vue dans l'exercice 4.5.

Deuxième partie
Le travail fait

1 Activité : Rechercher un caractère

1.1 Trouvez le premier

Vous devez écrire un algorithme de fonction qui renvoie la position du premier caractère recherché.
On considère qu'une chaîne de caractères est un tableau dont les éléments sont chacun un des caractères constituant la chaîne.
La chaîne de caractère est contenue dans la variable *chaine*.

```
fonction rechercher(recherche,chaine):
   Pour i variant de 0 à len(chaine):
      si chaine[i]==recherche:
         retourner i

rechercher('a',chaine)
```

1.2 Trouvez les tous

Vous devez écrire un algorithme de fonction qui renvoie la position de chaque caractère recherché.

```
fonction rechercher(recherche,chaine):
   liste = []
   Pour i variant de 0 à len(chaine):
      si chaine[i]==recherche:
         liste.ajouter(i)
   retourner liste

rechercher('a',chaine)
```

1.3 Comptez le nombre de lignes

Vous devez écrire un algorithme de fonction qui renvoie le nombre de caractère recherché et trouvé.

En choisissant correctement le caractère recherché, on obtient le nombre de lignes dans le fichier.

```
fonction rechercher(recherche,chaine):
   nombre = 0
   Pour i variant de 0 à len(chaine):
      si chaine[i]==recherche:
         nombre = nombre + 1
      retourner nombre

rechercher('\n',chaine)
```

1.4 Combien de caractères lus ?

En reprenant votre algorithme précédent, vous devez utiliser une variable qui compte le nombre de fois que vous avez effectué une lecture pour trouver le caractère recherché.
Cette fonction doit renvoyer le nombre de caractères lus.

```
fonction rechercher(recherche,chaine):
   nombre = 0
   compteur = 0
   Pour i variant de 0 à len(chaine):
      compteur = compteur + 1
      si chaine[i]==recherche:
         nombre = nombre + 1
   retourner compteur

rechercher('\n',chaine)
```

1.5 Lien entre le nombre de caractères lus et la taille de la chaîne

On se pose la question de savoir si l'algorithme est plus ou moins efficace. C'est à dire : le nombre de lectures va-t-il doubler quand la taille de la chaîne va doubler ?
Proposez une méthode qui permet de répondre à cette question en TP informatique.

On espère que les élèves proposeront de compter le nombre de lecture pour différentes taille de chaîne.
Pour la mise en oeuvre, on peut penser à utiliser un fichier texte relativement long dont on ne copierai qu'une partie des données dans une chaîne pour créer des couples (taille,compteur) que l'on affiche avec *matplotlib*.

2 Le cours

P0

1. Combien de fois « Au revoir » va-t-il s'afficher ? *Une fois*
2. Combien de fois « On ne peut pas prendre la racine d'un réel négatif ! » va-t-il s'afficher ? *1*
3. Combien de fois « La racine de ... est .... va-t-il s'afficher ?

0

P1

1. Combien de fois « Au revoir » va-t-il s'afficher ? *Une fois*
2. Combien de fois « On ne peut pas prendre la racine d'un réel négatif ! » va-t-il s'afficher ? *1*
3. Combien de fois « La racine de ... est .... va-t-il s'afficher ? *0*

P2

1. Combien de fois « Au revoir » va-t-il s'afficher ? *Une fois*
2. Combien de fois « On ne peut pas prendre la racine d'un réel négatif ! » va-t-il s'afficher ? *moy= ? ? min=0 max=1*
3. Combien de fois « La racine de ... est .... va-t-il s'afficher ? *moy= ? ? min=0 max=1*

P3

1. Quelles sont les données ? *Une fois*
2. Combien de fois « L'élément cherche n'est pas dans la sequence. » va-t-il s'afficher ? *moy= 0,5 min=0 max=1*
3. Combien de fois « Fin » va-t-il s'afficher ? *moy= 0,5 min=0 max=1*

P4

1. Combien de fois « L'élément cherche n'est pas dans la sequence. » va-t-il s'afficher ? *Moy= ? ? ? min=0 max=n avec n =nombre d'éléments dans la séquence*
2. Combien de fois « Fin » va-t-il s'afficher ? *1*

P5

1. Combien de fois « facteur trouvé : ..... » va-t-il s'afficher ?*Deux fois*

P6

1. Combien de fois « facteur trouvé : ..... » va-t-il s'afficher ? *? ? ? Tout dépend de n mini=0 moyen= ? ? ? maxi=dépend de n*

P0 bis

1. Sur le programme P0, modifier le programme afin de donner la réponse à la question : Combien de fois « La racine de ... est .... va-t-il s'afficher ? *Voir P0-1.py*

```
from math import sqrt
x = 2
compteur=0
if x >= 0:
    y = sqrt(x)
    print("La racine de {:.2f} est {:.3f}".format(x, y))
else:
    print("On ne peut pas prendre la racine d'un reel negatf !")
    compteur+=1
print("\nAu revoir")
print("réponse à la question ",compteur)
```

3 Activité : Rechercher un motif

3.1 Trouvez le premier

Un motif est une chaîne de caractères que l'on souhaite trouver dans un fichier.
Vous devez écrire un algorithme de fonction qui renvoie la position du premier motif recherché.
On considère qu'une chaîne de caractères est un tableau dont les éléments sont chacun un des caractères constituant la chaîne.
La chaîne de caractère est contenue dans la variable *chaine*.

3.2 Trouvez les tous

1. Vous devez écrire / compléter / analyser un algorithme de fonction qui renvoie le nombre d'occurrences du motif recherché dans un fichier.
2. Vous devez chercher à évaluer le coût en nombre d'opérations de l'exécution de la recherche.
3. Vous devez observer les variations éventuelles du coût :
   — en fonction de la taille du fichier
   — en fonction de la longueur du motif

```
 -- algorithme naif
fonction recherche(motif,file)
ouvrir file
-- (precondition) verifier file non vide
parcourir toutes les lettres du fichier:
   si lettreCourante == premiere lettre du motif:
      chercher si les lettres suivantes "valident" le motif
      si oui :
         occurence +=1
 -- (postcondition)
 --    tous les lettes du fichier ont été parcourues
 --    toutes les occurences ont été trouvées...
```

```
fonction rechercher(recherche,chaine):
   nombre = 0
   Pour i variant de 0 à len(chaine):
      si chaine[i]==recherche:
         nombre = nombre + 1
         retourner nombre

rechercher('e',chaine)
```

```
#!/usr/bin/python3

# fonction de recherche motif dans file
def chercheMotif(motif, file):

# lecture du fichier
    try:
        with open(file,"r") as f:
            contenu=f.read()
        assert{contenu != ''}
```

```
# si assertion fausse ecrire "fichier vide"
    except AssertionError :
        print("Le fichier est vide")

# initialisations
    i = 0
    j = 0
    compteur = 0
    cout = 0

# parcourrir toutes les lettres du fichier
    while i < len(contenu):
        cout += 1
        # si lettreCourante == premiere lettre du motif:
        if  contenu[i] == motif[j]:
            # chercher si les lettres suivantes "valident" le motif
            while j< len(motif):
                cout += 1
                if contenu[i+j] != motif[j]:
                    j = 0
                    break
                j += 1
            if j == len(motif):
                compteur += 1
                j = 0
        i += 1
    return (compteur, cout)



# appel
chaine = input('Saisis la chaine que tu cherches : ')
fichier = 'access.txt'
compteur, cout = chercheMotif(chaine,fichier)
print ('nombre d\'occurences : ', compteur)
print ('cout : ', cout)
```

Deuxième version

```
#!/usr/bin/python3

# fonction de recherche de motif dans un fichier texte
def chercheMotif(motif, file):

# lecture du fichier
    try:
        with open(file,"r") as f:
            contenu=f.read()
        assert{contenu != ''}
# si assertion fausse ecrire "fichier vide"
    except AssertionError :
        print("Le fichier est vide")

# initialisations
```

```
    i = 0
    compteur = 0
# parcours du contenu
    for i in  range(len(contenu)-len(motif)):
        j = 0
        trouve = True
        while  trouve and j < len(motif):
            if contenu[i+j] == motif[j]:
                j +=1
            else:
                trouve = False
        if j == len(motif):
            compteur +=1
    return (compteur)

# appel
chaine = input('Saisis la chaine que tu cherches : ')
fichier = 'access.txt'
print ('nombre d\'occurences : ', chercheMotif(chaine,fichier))
```

4 Exercices

4.1 Compter le nombre de caractère "e"

À vous d'écrire l'algorithme permettant de trouver le nombre de caractères 'à' présent dans le fichier.
Vous pourrez essayer de coder en Python votre algorithme.

4.2 Compteur de performance

À vous d'ajouter un compteur qui s'incrémente à chaque affectation dans l'algorithme suivant.

```
fonction maFonction(laliste):
   Pour i de 0 à len(laliste) - 1
      x = laliste[i]
      j = i
      Tant que j > 0 et laliste[j - 1] > x
         laliste[j] = laliste[j - 1]
         j = j - 1
      laliste[j] = x
liste = [2,6,8,1,0,5]
maFonction(liste)
```

4.3 Évolution de l'algorithme précédent

Comment se comporte l'algorithme précédent quand la longueur de liste s'allonge ?
Reprendre la méthode vue dans l'exercice 4.5.

Lors du TP, on pourra fournir une fonction qui crée une liste de n entiers mélangés.

```
def listeLongueurVariable("nomdufichier",taille):
    f = open(nomdufichier,'rb')
    chaine = f.read(taille)
    f.close()
    return chaine
machaine = listeLongueurVariable("access.log",1000)
```

Troisième partie

Remarques, allons plus loin

Pour aller plus loin, on peut faire appel à d'autres algorithmes :

- Algorithme à l'aide d'automates finis
- Algorithme de Morris-Pratt

Un lien : http ://www.lsv.fr/ haddad/coursalgorithmique.pdf