TP Chercher - remplacer Christophe Viroulaud Terminale - NSI Algo 28

chercher-remplacer.zip sur site (livre la guerre des mondes)

TP Chercher - remplacer

Christophe Viroulaud

Terminale - NSI

Algo 28

remplacer

TP Chercher -

La fonction chercher et remplacer est implémentée dans de nombreux logiciels : éditeurs de texte. IDE (Environment de Développement Intégré)...Il est ainsi possible de remplacer, en une fois. le nom d'une variable dans un programme ou le

La fonction *chercher et remplacer* est implémentée dans de nombreux logiciels : éditeurs de texte, IDE (Environment de Développement Intégré)...Il est ainsi possible de remplacer, en une fois, le nom d'une variable dans un programme ou le nom d'un personnage dans un livre.

Comment implémenter une fonction de recherche efficace?

TP Chercher remplacer



-Importer un texte

Activité 1 : 1. Télécharger et extraire le dossier compressé chercher-remplacer.zip Dans un programme Python, importer le contenu du fichier la-guerre-des-mondes-wells.txt Trouver le nombre de caractères du livre.

Importer un texte

Importer un texte

#### Activité 1 :

- 1. Télécharger et extraire le dossier compressé chercher-remplacer.zip
- 2. Dans un programme Python, importer le contenu du fichier la-guerre-des-mondes-wells.txt dans une variable livre.
- 3. Trouver le nombre de caractères du livre.

TP Chercher remplacer

Importer un texte





- -Avant de regarder la correction

- Analyser les messages d'erreur.
- Demander au professeur

- ► Prendre le temps de réfléchir,
- ► Analyser les messages d'erreur,
- ▶ Demander au professeur.

Importer un fichier texte - méthode :

livre = f.read()

### Correction

Importer un fichier texte

Importer un fichier texte - méthode 2

TP Chercher - remplacer

Importer un texte

echercher

echerche naïve estion de la casse

aluer l'efficacité gorithme de Boyer-Moore

mplacer



Not debarrer afficials de l'algorithme de Boye-Moore, il stributeurs de tituler une recherche salve. L'extende 2: L'étagles 1: L'étagles 1

### Recherche naïve

Afin d'observer l'efficacité de l'algorithme de Boyer-Moore, il est intéressant de tester une recherche naïve.

#### Activité 2 :

- Adapter la fonction recherche\_naive vue en cours pour qu'elle renvoie la liste des positions du motif dans le texte.
- 2. Tester la fonction sur *la guerre des mondes* avec le motif *guerre*.
- À l'aide d'un éditeur de texte ou d'un bloc-notes, compter le nombre d'occurrences du motif guerre.
   Comparer au résultat obtenu avec la fonction Python.

## TP Chercher - remplacer

Importer un texte

Kechercher

#### Recherche naïve

Gestion de la casse Évaluer l'efficacité

Algorithme de Boyer-Moo

Присег

Analyser les messages d'erreur,
 Demander au professeur.

Avant de regarder la correction



- ► Prendre le temps de réfléchir,
- ► Analyser les messages d'erreur,
- ► Demander au professeur.

### TP Chercher - remplacer

Importer un texte

Rechercher

Recherche naïve

### Correction

```
def recherche_naive(texte: str, motif: str) -> list:
       res = []
        # dernière position = taille(texte) - taille(
      motif)
       for i in range(len(texte)-len(motif)+1):
 4
            j = 0
            while (j < len(motif)) and \</pre>
 6
                     (motif[j] == texte[i+j]):
                j += 1
 8
            # correspondance sur toute la fenêtre
 9
            if j == len(motif):
10
                res.append(i)
11
12
        return res
```

On compte 21 occurrences pour 28 avec un éditeur de texte.

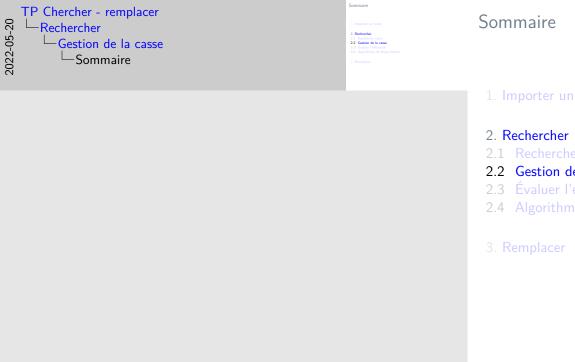
print(recherche\_naive(livre, "guerre"))

Recherche naïve

TP Chercher -

remplacer

10 / 33



- 2.1 Recherche naïve
- 2.2 Gestion de la casse
- 2.4 Algorithme de Boyer-Moore

TP Chercher -

remplacer

Gestion de la casse

11/33

Gestion de la casse

L'éditure de tote part regione les mois Guarre, GUERRE au
gener midifferement.

Actività 3:

1 Estes la forción es, atamaccia Cistere: sizo
- es ej combination de la combination de la
caracter les charges de la combination de la
caracter les charges de la caracter les c

### Gestion de la casse

L'éditeur de texte peut repérer les mots *Guerre, GUERRE ou guerre* indifféremment.

#### Activité 3:

- Écrire la fonction en\_minuscule(lettre: str)
   → str qui renvoie la version minuscule de la
   lettre s'il s'agit d'une lettre majuscule et le
   caractère inchangé sinon. La fonction ne devra
   pas utiliser la méthode native lower.
- Adapter la fonction recherche\_naive pour qu'elle compte les mots sans prendre en compte la casse.

# TP Chercher - remplacer

Importer un texte

techercher

Gestion de la casse Évaluer l'efficacité

gorithme de Boyer-Moore

empiacer

Analyser les messages d'erreur,
 Demander au professeur.

Avant de regarder la correction

### Avant de regarder la correction



- ▶ Prendre le temps de réfléchir,
- ► Analyser les messages d'erreur,
- ► Demander au professeur.

TP Chercher - remplacer

Importer un texte

chercher

Gestion de la casse Évaluer l'efficacité

Remplacer

```
1 def en_minuscule(lettre: str) -> str:
        11 11 11
 2
        renvoie la minuscule de lettre
 3
        ou le caractère inchangé si ce n'est pas une
 4
       lettre
        11 11 11
        dec = 32 # ord("a") - ord("A")
 6
        if ord(lettre) >= ord("A") and \
                ord(lettre) <= ord("Z"):</pre>
            return chr(ord(lettre)+dec)
10
        else:
            return lettre
11
```

TP Chercher - remplacer

mporter un texte

Rechercher

Recherche naïve

Gestion de la casse

valuer l'efficacité

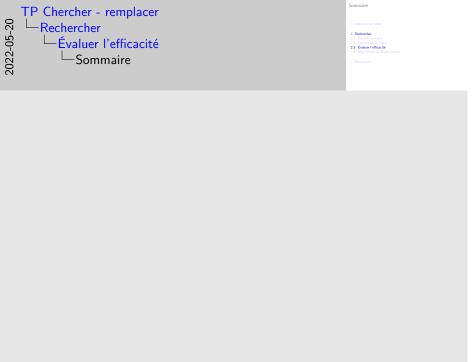
mplacer

```
def recherche_naive(texte: str, motif: str) -> list:
    11 11 11
    renvoie les positions du motif dans le texte
    11 11 11
    res = []
    # dernière position = taille(texte) - taille(motif)
    for i in range(len(texte)-len(motif)+1):
        j = 0
        while (j < len(motif)) and \
        (en minuscule(motif[j]) == en minuscule(texte[i+j])):
            j += 1
        # correspondance sur toute la fenêtre
        if j == len(motif):
            res.append(i)
    return res
```

Utilisation de la fonction en\_minuscule

TP Chercher -

remplacer



# Sommaire

- 2. Rechercher
- 2.1 Recherche naïve
- 2.3 Évaluer l'efficacité
- 2.4 Algorithme de Boyer-Moore

TP Chercher -

remplacer

Évaluer l'efficacité

16/33

Évaluer l'efficacité

Pour mesurer l'efficacité de l'algorithme, nous pouvons

chronométrer la durée d'exécution de la fonction. Cependant, il semble plus pertinent de compter le nombre de comparaisons effectuées. En effet, cette approche est

ficacité relative de deux algorithmes

Algorithme de Boyer-Moore

Remnlacer

### À retenir

Pour mesurer l'efficacité de l'algorithme, nous pouvons chronométrer la durée d'exécution de la fonction. Cependant, il semble plus pertinent de compter le nombre de comparaisons effectuées. En effet, cette approche est indépendante du matériel et permettra de comparer l'efficacité relative de deux algorithmes.

### Activité 4 :

- 1. Dans le programme principal, ajouter la variable NB\_COMPARAISONS initialisée à 0.
- Modifier la fonction recherche\_naive pour compter le nombre de comparaisons effectuées. La variable NB\_COMPARAISONS sera utilisée comme variable globale.

#### TP Chercher remplacer

Importer un texte

Rechercher

Recherche naïve

Évaluer l'efficacité

Algorithme de Boyer-Moore

Remplacer

Analyser les messages d'erreur
 Demander au professeur.

Avant de regarder la correction

### Avant de regarder la correction



- ► Prendre le temps de réfléchir,
- ► Analyser les messages d'erreur,
- ► Demander au professeur.

TP Chercher - remplacer

Importer un texte

Rechercher

Gestion de la casse Évaluer l'efficacité

Algorithme de Boyer-Moore

templacer

return res

```
def recherche naive(texte: str, motif: str) -> list:
    global NB COMPARAISONS
    res = []
    # dernière position = taille(texte) - taille(motif)
    for i in range(len(texte)-len(motif)+1):
        j = 0
        # comparaison de la première lettre
        NB COMPARAISONS += 1
        while (j < len(motif)) and \</pre>
        (en_minuscule(motif[j]) == en_minuscule(texte[i+j])):
            j += 1
            # comparaisons dans la fenêtre
            NB_COMPARAISONS += 1
        # correspondance sur toute la fenêtre
        if j == len(motif):
            res.append(i)
```

TP Chercher -

remplacer

```
| WM_CDMMAISDMS = 0
| print("sombre de caractères: ", les(livre))
| print(secherche_paire(livre,"gourre*))
| print("comparaisons: ",NM_CDMMAAISDMS)
```

2 [35, 340, 577, 859, 918, 1984, 7343, 7517, 8599, 67998, 110250, 166464, 229800, 241073, 264050, 272256, 302198, 330291, 33738, 333770, 334412, 372854, 376022, 302191, 303202, 401899, 415041, 415120]
2 cceparaisens: 435048

On a plus de comparaisons que de nombre de caractères.

#### Correction

```
NB_COMPARAISONS = 0
print("nombre de caractères: ", len(livre))
print(recherche_naive(livre, "guerre"))
print("comparaisons: ",NB COMPARAISONS)
```

```
nombre de caractères: 433983
[35, 340, 577, 859, 958, 1954, 7343, 7517, 8099,
67998, 110280, 146464, 229890, 241073, 264650,
272295, 326198, 333691, 333738, 333770,
334412, 372834, 376022, 392191, 393202,
401899, 415041, 415120]
comparaisons: 438048
```

On a plus de comparaisons que de nombre de caractères.

TP Chercher - remplacer

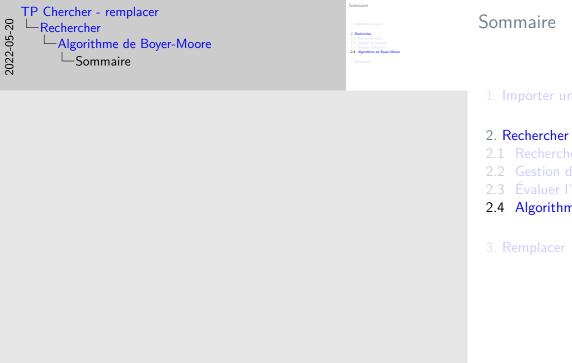
Importer un texte

Rechercher

Gestion de la casse Évaluer l'efficacité

Algorithme de Boyer-Moore

nplacer



Sommaire

2.1 Recherche naïve

2.4 Algorithme de Boyer-Moore

TP Chercher -

remplacer

Algorithme de Boyer-Moore



22 / 33

Importer un texte

Rechercher

1 Michigen malve

2 Gaktion de la casse

3 Algorithme de Boyer-Moore

Remplacer

#### Activité 5:

- 1. Reprendre les fonctions de l'algorithme de Boyer-Moore vu en classe.
- 2. Adapter la fonction pretraitement\_decalages pour qu'elle gère la casse.
- 3. Adapter la fonction decalage\_fenetre pour qu'elle gère la casse.
- 4. Adapter la fonction **compare** pour qu'elle gère la casse.
- 5. Modifier la fonction boyer\_moore pour qu'elle renvoie la liste des positions du motif dans le texte.
- 6. Modifier une des fonctions pour compter le nombre de comparaisons effectuées.

TP Chercher - remplacer

Importer un texte

Rechercher

Recherche naïve Gestion de la casse

Algorithme de Boyer-Moore

empiacer

Avant de regarder la correction

 Analyser les messages d'erreur. Demander au professeur

TP Chercher -

remplacer

Algorithme de Bover-Moore



- ► Prendre le temps de réfléchir,
- ► Analyser les messages d'erreur,
- ▶ Demander au professeur.

TP Chercher - remplacer

inporter un texte

echercher

Gestion de la casse Évaluer l'efficacité Algorithme de Boyer-Moore

```
def pretraitement_decalages(motif: str) -> dict:
    decalages = dict()
    # on s'arrête à l'avant dernière lettre du motif
    for i in range(len(motif)-1):
        # len(motif)-1 est la position de la dernière lettre
        decalages[en_minuscule(motif[i])] = len(motif)-1-i
    return decalages
```

def decalage\_fenetre(decalages: dict, taille: int,
 lettre: str) => int:

lettre = en\_minuscule(lettre)
for cle, val in decalages.items():
 if cle == lettre:
 return val

```
mplacer
```

```
def decalage_fenetre(decalages: dict, taille: int,
    lettre: str) -> int:
lettre = en_minuscule(lettre)
for cle, val in decalages.items():
    if cle == lettre:
        return val
    # si la lettre n'est pas dans le dico (= le motif)
    return taille
```

```
TP Chercher - remplacer
```

echercher

# byer\_morefuter: sir, metric sir) >> list: ret = [] decalage = pretrational, decalages/metri) decalage = pretrational, decalages/metri) white 1 <= local(metric)-(metric): from morefuters = metric) from morefuters = metric)

i += len(notif)

return res

decale = decalage\_fenetre(decalages, len(notif),

texte[i\*lem(motif)=1])

```
def boyer moore(texte: str, motif: str) -> list:
    res = []
    decalages = pretraitement decalages(motif)
                                                                Bover-Moore
    i = 0
    while i <= len(texte)-len(motif):</pre>
        # si on trouve le motif
        if compare(texte, i, motif):
            res.append(i)
            # on recommence à la fin du motif trouvé
            i += len(motif)
        else:
            # sinon on décale
            decale = decalage_fenetre(decalages, len(motif),
                                 texte[i+len(motif)-1])
            i += decale
    # si on sort de la boucle, on n'a rien trouvé
    return res
```

TP Chercher -

remplacer

```
of compare(texte: str, position: int, motif: str) -> bool:
  global NB_COMPARAISONS
  for i in range(len(motif)=1, -1, -1):
      if not( en minuscule(texte[en cours]) ==
         en cours -= 1
```

On peut compter les comparaisons dans la fonction compare

```
def compare(texte: str, position: int, motif: str) -> bool:
    global NB_COMPARAISONS
    # position de la dernière lettre de la fenêtre
    en cours = position + len(motif)-1
    # parcours de la fenêtre à l'envers
    for i in range(len(motif)-1, -1, -1):
        # compare au moins la dernère lettre de la fenêtre
        NB COMPARAISONS += 1
        if not( en minuscule(texte[en cours]) ==
                en minuscule(motif[i]) ):
```

On peut compter les comparaisons dans la fonction compare.

return False

en cours -= 1

else:

return True

TP Chercher -

remplacer



### Remplacer

Il est maintenant possible de remplacer toutes les occurrences du motif dans le texte.

#### Activité 6 :

 Écrire la fonction remplacer(livre: str, motif: str, remplacement: str) → str qui remplace le motif dans le livre par

**NB** : On pourra utiliser un slice pour récupérer un morceau du texte :

remplacement et renvoie le texte modifié.

- # renvoie la chaîne entre l'indice
   debut inclus et fin exclu
  texte[debut: fin]
- 2. Remplacer le mot guerre par paix.
- 3. Créer alors un fichier la-paix-des-mondes.txt du livre modifié.

TP Chercher - remplacer

orter un texte

ercher

erche naïve ion de la casse

Évaluer l'efficacité Algorithme de Boyer-Moc

Algorithme de Boyer-Moord Remplacer

Avant de regarder la correction

 Analyser les messages d'erreur. Demander au professeur

- ► Prendre le temps de réfléchir,
- ► Analyser les messages d'erreur,
- ▶ Demander au professeur.

TP Chercher remplacer

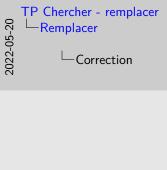
Remplacer

#### 

#### Correction

On TP Chercher - remplacer

```
def remplacer(livre: str, motif: str, remplacement: str) ->
   str:
    11 11 11
    remplace 'motif' par 'remplacement' dans 'livre'
    Returns:
        str: livre modifié
    11 11 11
    positions = boyer_moore(livre, motif)
    livre_modifie = ""
    debut = 0
    for fin in positions:
        livre_modifie += livre[debut: fin] + remplacement
        # recommence à la fin du motif dans livre
        debut = fin + len(motif)
    return livre_modifie
```



3 fichier.write(modifie)

4 fichier.close()

modifie = remplacer(livre, "guerre", "paix") fichier = open("la-paix-dez-mondez.txt", "w", encoding="utf8")

Création du livre

fichier.write(modifie) fichier.close()

```
Remplacer
1 modifie = remplacer(livre, "guerre", "paix")
2 fichier = open("la-paix-des-mondes.txt", "w",
```

encoding="utf8")

Création du livre

TP Chercher remplacer