

Fonction chercher et remplacer Christophe Viroulaud

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

Fonction *chercher et remplacer*

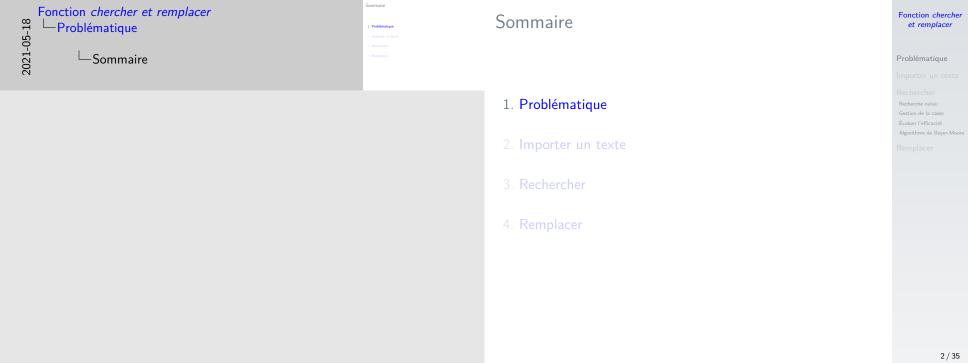
Fonction *chercher et remplacer*

Christophe Viroulaud

Terminale - NSI

Fonction chercher et remplacer

1/35



-Problématique

Problématique

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?

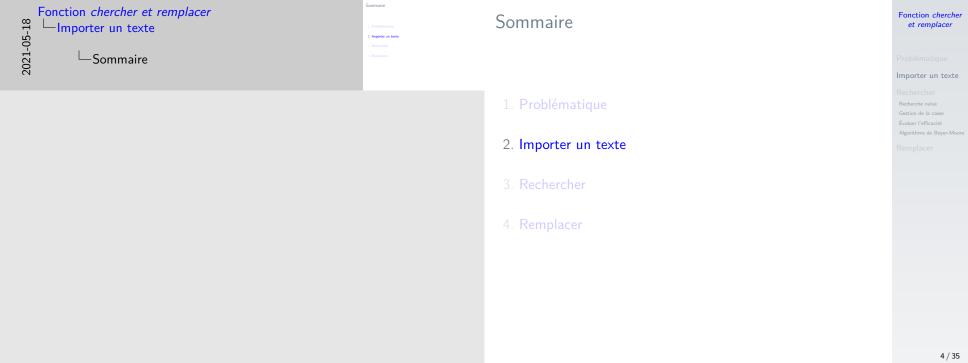
Problématique

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. ment implémenter une fonction de recherche efficace

Fonction chercher et remplacer

Problématique

3 / 35



└─Importer un texte

Importer un texte

Importer un texte

Activité 1 :

 Télécharger et extraire le dossier compressé charcher-emplacer.zip

 Dans un programme Python, importer le contenu du fichier la-auerre-des-mondes-wells text dans une

Trouver le nombre de caractères du livre.

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.

Fonction chercher et remplacer

Problématique

Importer un texte

Recherche naïve
Gestion de la casse

-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.

Fonction chercher et remplacer

Importer un texte

-Correction

f = open("la-guerre-des-mondes-wells.txt") f.close() print(len(livre)) Importer un fichier texte with open("la-guerre-des-mondes-wells.txt") as f:

Importer un fichier texte - méthode 2

Correction

Importer un texte

Fonction chercher

et remplacer

```
livre = f.read()
f.close()
print(len(livre))
```

Importer un fichier texte

```
with open("la-guerre-des-mondes-wells.txt") as f:
   livre = f.read()
print(len(livre))
```

f = open("la-guerre-des-mondes-wells.txt")

Importer un fichier texte - méthode 2



Afin d'observer l'efficacité de l'algorithme de Bover-Moore, il Adapter la fonction recherche_maive vue en cours pour qu'elle renvoie la liste des positions d 2. Tester la fonction sur la guerre des mondes avec le A 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

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.
- 3. 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.

Fonction chercher et remplacer

Recherche naïve

-Avant de regarder la correction

Avant de regarder la correction



- 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.

Fonction chercher et remplacer

Recherche naïve

```
Correction

1 def refused__main(uses siz_medit siz_) => 0s:
2 ms = 0 ms = main(use)__main(uses siz_medit siz_) => 0s:
4 for is respective(use)__main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__siz_main(use)__s
```

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
          i = 0
          while (i < len(motif)) and (motif[i] == texte[i+j]):
 6
             i += 1
          if j == len(motif): # correspondance sur toute la fen
 8
       être
             res.append(i)
 9
10
       return res
```

```
print(recherche_naive(livre, "guerre"))
```

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

Problematique

Fonction chercher

et remplacer

.

techercher

Recherche naïve

Gestion de la casse

Évaluer l'efficacité

Algorithme de Boyer-Mod

11 / 35



Gestion de la casse

L'iddisse de teste part repérer les mots Guerre, CUERRE ou
gener indifférement

Activité 3:

1. Écrire la fonction es, atamaccule (Lettre: str)

— set qui remois la version ministaire de la

caractère inclassigé sinon. La fonction ne devan
pas cilient e ministre autre l'ever

2. Adapter la fonction resk-reche, autre que
qu'il compile la ministra serperiere on compile la

qu'il compile la ministra serperiere on compile la

qu'il compile la ministra serperiere on compile la

Gestion de la casse

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

Activité 3:

- 1. 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.

Fonction chercher et remplacer

Problematique

importer un texte

Recherche naïve

Gestion de la casse

Évaluer l'efficacité Algorithme de Boyer-Moore

Fonction *chercher et remplacer* -Rechercher -Gestion de la casse

-Avant de regarder la correction

Avant de regarder la correction



- 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.

Fonction chercher et remplacer

Gestion de la casse

2021-05-18

```
| dat on administrative or ) -> to:
| remove to interesting to a relative to the control of latter or relative to the control of the control
```

Correction

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

Fonction chercher et remplacer

Recherche naïve

Gestion de la casse Évaluer l'efficacité

gorithme de Boyer-Moore

2021-05-18

```
Correction

I add without parallel at 0, read 0 0 0 0 to 0.

The same in parallel at 0 and 0 and 0 to take 0 and 0 and
```

Correction

```
def recherche_naive(texte: str, motif: str) -> list:
       renvoie les positions du motif dans le texte
 3
 5
       res = ||
       # dernière position = taille(texte) - taille(motif)
       for i in range(len(texte)—len(motif)+1):
          i = 0
          while (j < len(motif)) and (en_minuscule(motif[j
       ) == en minuscule(texte[i+i]):
             i += 1
10
          if j == len(motif): # correspondance sur toute la
        fenêtre
             res.append(i)
13
       return res
```

Utilisation de la fonction en minuscule

Fonction chercher et remplacer

1 Toblematique

Rechercher

Gestion de la casse

Evaluer l'efficacité Algorithme de Boyer-Moc



Évaluer l'efficacité

Pour mesurer l'efficacité de l'algorithme, nous pouvons chronométrer la durée d'exécution de la fonction.

l'efficacité relative de deux algorithmes

Cependant, il semble plus pertinent de compter le nombre de comparaisons effectuées. En effet, cette approche est indépendante du matériel et permettra de comparer

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.

Fonction chercher et remplacer

Évaluer l'efficacité

Activité 4:

1. Dans le programme principal, ajouter la variable NB_COMPARATSONS initialisée à 0.

2. Modifier la fonction recherche_maive pour compre le nombre de comparaions effectuées. La variable NB_COMPARATSONS sera utilisée comme variable globale.

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.

Fonction chercher et remplacer

Problématique

mporter un texte

Rechercher Recherche naïve

Évaluer l'efficacité

gorithme de Boyer-Moore

Avant de regarder la correction



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.

Fonction chercher et remplacer

Problématiqu

Importer un texte

Rechercher

Gestion de la casse Évaluer l'efficacité

gorithme de Boyer-Moore

```
COPECION

(Fig. 1) The control of th
```

Correction

```
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):
      i = 0
      # comparaison de la première lettre
      NB COMPARAISONS += 1
      while (j < len(motif)) and (en_minuscule(motif[j]) ==
   en_minuscule(texte[i+j])):
         i += 1
         # comparaisons dans la fenêtre
         NB COMPARAISONS +=1
      if j == len(motif): # correspondance sur toute la fenêtre
         res.append(i)
   return res
```

Fonction chercher et remplacer

Problematique

iporter un texte

Gestion de la casse Évaluer l'efficacité

Algorithme de Boyer-Moo

пріасеі

Correction

1 NBL COMPAGADONS = 0
2 pixel("membra de assectione ".ine[dens))
3 pixel("membra de assectione ".ine[dens))
3 pixel("membra de assectione ".ine[dens))
1 pixel("comparadone," NBL COMPAGADONS)
1 membra de assectione 3.3383
2 (S. S. S. S. 77, 80, 50, 106, 13, 13, 13, 13, 100, 6, 1000
1 1000; 146, 2400; 2400; 2400; 2400; 2400; 2500;

On a plus de comparaisons que de nombre de caractères

Correction

- $1 NB_COMPARAISONS = 0$
- print("nombre de caractères: ", len(livre))
- 3 print(recherche_naive(livre, "guerre"))
- print("comparaisons: ",NB_COMPARAISONS)

nombre de caractères: 433983

2 [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] 3 comparaisons: 438048

On a plus de comparaisons que de nombre de caractères.

et remplacer

Fonction chercher

Problèmatique

Importer un texte

echercher

Recherche naïve Gestion de la casse

Évaluer l'efficacité



Problématique
Importer un tacte
Rechercher

Section de la case

Section de la case

Section de la case

Adjorithme 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.

Fonction chercher et remplacer

Problématique

mporter un texte

echercher

Évaluer l'efficacité

Algorithme de Boyer-Moore

Fonction *chercher et remplacer* -Rechercher -Algorithme de Boyer-Moore

-Avant de regarder la correction

Analyser les messages d'erreur

Demander au professeur

Avant de regarder la correction

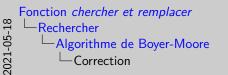
Avant de regarder la correction

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

Fonction chercher

et remplacer

Algorithme de Boyer-Moore





Correction

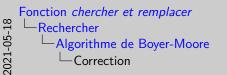
```
def pretraitement_decalages(motif: str) -> dict:
    decalages = dict()
    # on s'arrête à l'avant dernière lettre du motif
    for i in range(len(motif)-1):
        # len(motif)-1 est la position de la dernière
    lettre
    decalages[en_minuscule(motif[i])] = len(motif)
    -1-i
    return decalages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

Fonction chercher et remplacer

.

therche naïve

Algorithme de Boyer-Moore



```
Correction

| Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Correction | Corr
```

return val

si la lettre n'est pas dans le dico (= le motif)

```
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
```

Fonction chercher et remplacer

_ . . .

Recherche naïve Gestion de la casse

Évaluer l'efficacité Algorithme de Boyer-Moore

2021-05-18

in discussion of the contract of the contract

Correction

```
def compare(texte: str, position: int, motif: str) −> bool
       # 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):
          if not(en_minuscule(texte[en_cours]) ==
       en_minuscule(motif[i])):
             return False
          else:
             en_cours -= 1
       return True
10
```

Fonction chercher et remplacer

Importer un text

Rechercher

Gestion de la casse Évaluer l'efficacité

Algorithme de Boyer-Moore

empiacer

2021-05-18

Correction

```
def boyer_moore(texte: str, motif: str) -> list:
       res = []
       decalages = pretraitement decalages(motif)
 3
       i = 0
 4
       while i \le len(texte) - len(motif):
 5
          # si on trouve le motif
           if compare(texte, i, motif):
              res.append(i)
              # on recommence à la fin du motif trouvé
              i += len(motif)
10
          else:
              # sinon on décale
             decale = decalage_fenetre(decalages,
13
                                    len(motif),
14
                                    texte[i+len(motif)-1]
15
              i += decale
16
       # si on sort de la boucle, on n'a rien trouvé
17
18
       return res
```

Fonction chercher et remplacer

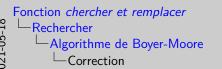
Problèmatique

mporter un texte

echercher

Gestion de la casse

Algorithme de Boyer-Moore





Correction

```
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 fenê
          NB\_COMPARAISONS += 1
          if not(en minuscule(texte[en_cours]) ==
       en_minuscule(motif[i])):
10
             return False
          else:
             en cours -=1
       return True
13
```

On peut compter les comparaisons dans la fonction compare.

Fonction chercher

et remplacer

Algorithme de Boyer-Moore



Fonction chercher et remplacer -Remplacer

-Remplacer

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

Activité 6

L Écrire la fonction remplacer(livre: str.

motif: str, remplacement: str) -> str qui remplace le motif dans le livre par remplacemen La fonction renvoie le texte modifié Remplacer le mot guerre par paix.

Créer alors un fichier la-paix-des-mondes.txt du

Remplacer

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

Activité 6 :

- 1. Écrire la fonction remplacer(livre: str, motif: str, remplacement: str) \rightarrow str qui remplace le motif dans le livre par remplacement. La fonction renvoie le texte modifié.
- 2. Remplacer le mot guerre par paix.
- 3. Créer alors un fichier la-paix-des-mondes.txt du livre modifié.

Fonction chercher et remplacer

A۱

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

Problématique

Importer un texte

Rechercher

Recherche naïve

Évaluer l'efficacité
Algorithme de Boyer-Moore

└**C**orrection

Correction

I add marghest/leve sit, medi sit, rangiacament sit)

> sit

> sit

for the product of the sit of

Correction

```
def remplacer(livre: str, motif: str, remplacement: str)
        −> str:
       remplace 'motif' par 'remplacement' dans 'livre'
 3
       Returns:
          str: livre modifié
       positions = boyer_moore(livre, motif)
       livre_modifie = ""
 9
       debut = 0
10
       for fin in positions:
          livre_modifie += livre[debut: fin] +
        remplacement
          # recommence à la fin du motif dans livre
13
          debut = fin + len(motif)
14
       return livre_modifie
15
```

Fonction chercher et remplacer

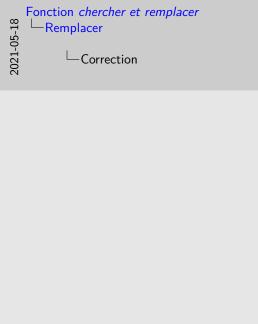
1 Toblematique

importer an text

Kecnercher

Gestion de la casse

Algorithme de Boyer-Mo



Correction

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

Création du livre

```
modifie = remplacer(livre, "guerre", "paix")
```

- fichier = open("la-paix-des-mondes.txt", "w")
- fichier.write(modifie)
- fichier.close()

Création du livre

Fonction chercher et remplacer

