

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

TP Chercher - remplacer

Christophe Viroulaud

Terminale - NSI

Algo 28

TP Chercher - remplacer

Christophe Viroulaud

Terminale - NSI

Algo 28

La fonction *chercher et remplacer* est implémentée dans de nombreux logiciels : éditeurs de texte, IDE (Environnement 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 ?

La fonction *chercher et remplacer* est implémentée dans de nombreux logiciels : éditeurs de texte, IDE (Environnement 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 ?

Importer un texte

Rechercher

Recherche naïve

Gestion de la casse

Évaluer l'efficacité

Algorithme de Boyer-Moore

Remplacer

Sommaire

1. Importer un texte

2. Rechercher

3. Remplacer

Importer un texte

Rechercher

Recherche naïve

Gestion de la casse

Évaluer l'efficacité

Algorithme de Boyer-Moore

Remplacer

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.

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.



- ▶ Prendre le temps de réfléchir,
- ▶ 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.

```
1 f = open("la-guerre-des-mondes-wells.txt",  
2         encoding="utf8")  
3 livre = f.read()  
4 f.close()  
5 print(len(livre))
```

Importer un fichier texte

```
1 with open("la-guerre-des-mondes-wells.txt",  
2         encoding="utf8") as f:  
3     livre = f.read()  
4     print(len(livre))
```

Importer un fichier texte - méthode 2

Correction

```
1 f = open("la-guerre-des-mondes-wells.txt",  
2         encoding="utf8")  
3 livre = f.read()  
4 f.close()  
5 print(len(livre))
```

Importer un fichier texte

```
1 with open("la-guerre-des-mondes-wells.txt",  
2         encoding="utf8") as f:  
3     livre = f.read()  
4     print(len(livre))
```

Importer un fichier texte - méthode 2

Sommaire

1. Importer un texte

2. Rechercher

2.1 Recherche naïve

2.2 Gestion de la casse

2.3 Évaluer l'efficacité

2.4 Algorithme de Boyer-Moore

3. Remplacer

Activité 2 :

1. 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.

Recherche naïve

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

Activité 2 :

1. 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.



- Prendre le temps de réfléchir,
- 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.

```

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

```

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

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

Correction

```

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

```

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

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

Sommaire

1. Importer un texte

2. Rechercher

2.1 Recherche naïve

2.2 Gestion de la casse

2.3 Évaluer l'efficacité

2.4 Algorithme de Boyer-Moore

3. Remplacer

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`.
2. Adapter la fonction `recherche_naive` pour qu'elle compte les mots sans prendre en compte la casse.

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`.
2. Adapter la fonction `recherche_naive` pour qu'elle compte les mots sans prendre en compte la casse.



- Prendre le temps de réfléchir,
- 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.

```

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

```

Correction

```

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

```

```

1 def recherche_naive(texte: str, motif: str) -> list:
2     """
3     renvoie les positions du motif dans le texte
4     """
5     res = []
6     # dernière position = taille(texte) - taille(motif)
7     for i in range(len(texte)-len(motif)+1):
8         j = 0
9         while (j < len(motif)) and \
10              (en_minuscule(motif[j]) == en_minuscule(texte[i+j])):
11             j += 1
12         # correspondance sur toute la fenêtre
13         if j == len(motif):
14             res.append(i)
15     return res

```

Utilisation de la fonction en_minuscule

Correction

```

1 def recherche_naive(texte: str, motif: str) -> list:
2     """
3     renvoie les positions du motif dans le texte
4     """
5     res = []
6     # dernière position = taille(texte) - taille(motif)
7     for i in range(len(texte)-len(motif)+1):
8         j = 0
9         while (j < len(motif)) and \
10              (en_minuscule(motif[j]) == en_minuscule(texte[i+j])):
11             j += 1
12         # correspondance sur toute la fenêtre
13         if j == len(motif):
14             res.append(i)
15     return res

```

Utilisation de la fonction en_minuscule

Sommaire

1. Importer un texte

2. Rechercher

2.1 Recherche naïve

2.2 Gestion de la casse

2.3 Évaluer l'efficacité

2.4 Algorithme de Boyer-Moore

3. Remplacer

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.

Évaluer l'efficacité

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_COMPARAISSONS` initialisée à 0.
2. Modifier la fonction `recherche_naive` pour compter le nombre de comparaisons effectuées. La variable `NB_COMPARAISSONS` sera utilisée comme variable globale.

Activité 4 :

1. Dans le programme principal, ajouter la variable `NB_COMPARAISSONS` initialisée à 0.
2. Modifier la fonction `recherche_naive` pour compter le nombre de comparaisons effectuées. La variable `NB_COMPARAISSONS` sera utilisée comme *variable globale*.



- ▶ Prendre le temps de réfléchir,
- ▶ 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.

```

1 def recherche_naive(texte: str, motif: str) -> list:
2     global NB_COMPARAISSONS
3     res = []
4     # dernière position = taille(texte) - taille(motif)
5     for i in range(len(texte)-len(motif)+1):
6         j = 0
7         # comparaison de la première lettre
8         NB_COMPARAISSONS += 1
9         while (j < len(motif)) and \
10             (en_minuscule(motif[j]) == en_minuscule(texte[i+j])):
11             j += 1
12             # comparaisons dans la fenêtre
13             NB_COMPARAISSONS += 1
14         # correspondance sur toute la fenêtre
15         if j == len(motif):
16             res.append(i)
17     return res

```

Correction

```

1 def recherche_naive(texte: str, motif: str) -> list:
2     global NB_COMPARAISSONS
3     res = []
4     # dernière position = taille(texte) - taille(motif)
5     for i in range(len(texte)-len(motif)+1):
6         j = 0
7         # comparaison de la première lettre
8         NB_COMPARAISSONS += 1
9         while (j < len(motif)) and \
10             (en_minuscule(motif[j]) == en_minuscule(texte[i+j])):
11             j += 1
12             # comparaisons dans la fenêtre
13             NB_COMPARAISSONS += 1
14         # correspondance sur toute la fenêtre
15         if j == len(motif):
16             res.append(i)
17     return res

```

```
1 NB_COMPARAISSONS = 0
2 print("nombre de caractères: ", len(livre))
3 print(recherche_naive(livre, "guerre"))
4 print("comparaisons: ", NB_COMPARAISSONS)
```

```
1 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.

Correction

```
1 NB_COMPARAISSONS = 0
2 print("nombre de caractères: ", len(livre))
3 print(recherche_naive(livre, "guerre"))
4 print("comparaisons: ", NB_COMPARAISSONS)
```

```
1 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.

Sommaire

1. Importer un texte

2. Rechercher

2.1 Recherche naïve

2.2 Gestion de la casse

2.3 Évaluer l'efficacité

2.4 Algorithme de Boyer-Moore

3. 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.



- Prendre le temps de réfléchir,
- 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.


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

Correction

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

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

Correction

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

```

1 def compare(texte: str, position: int, motif: str) -> bool:
2     # position de la dernière lettre de la fenêtre
3     en_cours = position+len(motif)-1
4     # parcours de la fenêtre à l'envers
5     for i in range(len(motif)-1, -1, -1):
6         if not ( en_minuscule(texte[en_cours]) ==
7                 en_minuscule(motif[i]) ):
8             return False
9     else:
10         en_cours = en_cours - 1
11     return True

```

Correction

```

1 def compare(texte: str, position: int, motif: str) -> bool:
2     # position de la dernière lettre de la fenêtre
3     en_cours = position+len(motif)-1
4     # parcours de la fenêtre à l'envers
5     for i in range(len(motif)-1, -1, -1):
6         if not ( en_minuscule(texte[en_cours]) ==
7                 en_minuscule(motif[i]) ):
8             return False
9     else:
10         en_cours = en_cours - 1
11     return True

```

```

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

```

Correction

```

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

```

```

1 def compare(texte: str, position: int, motif: str) -> bool:
2     global NB_COMPARAISSONS
3     # position de la dernière lettre de la fenêtre
4     en_cours = position + len(motif)-1
5     # parcours de la fenêtre à l'envers
6     for i in range(len(motif)-1, -1, -1):
7         # compare au moins la dernière lettre de la fenêtre
8         NB_COMPARAISSONS += 1
9         if not( en_minuscule(texte[en_cours]) ==
10              en_minuscule(motif[i]) ):
11             return False
12     else:
13         en_cours -= 1
14     return True

```

On peut compter les comparaisons dans la fonction compare.

Correction

```

1 def compare(texte: str, position: int, motif: str) -> bool:
2     global NB_COMPARAISSONS
3     # position de la dernière lettre de la fenêtre
4     en_cours = position + len(motif)-1
5     # parcours de la fenêtre à l'envers
6     for i in range(len(motif)-1, -1, -1):
7         # compare au moins la dernière lettre de la fenêtre
8         NB_COMPARAISSONS += 1
9         if not( en_minuscule(texte[en_cours]) ==
10              en_minuscule(motif[i]) ):
11             return False
12     else:
13         en_cours -= 1
14     return True

```

On peut compter les comparaisons dans la fonction `compare`.

1. Importer un texte
2. Rechercher
3. Remplacer

Sommaire

1. Importer un texte

2. Rechercher

3. Remplacer

Importer un texte

Rechercher

Recherche naïve

Gestion de la casse

Évaluer l'efficacité

Algorithme de Boyer-Moore

Remplacer

Activité 6 :

1. Écrire la fonction `remplacer(livre: str, motif: str, remplacement: str) → str` qui remplace le motif dans le livre par `remplacement` et renvoie le texte modifié.
NB : On pourra utiliser un slice pour récupérer un morceau du texte :

```
1 # renvoie la chaîne entre l'indice
  debut inclus et fin exclu
2 texte[debut: fin]
```

2. Remplacer le mot `guerre` par `paix`.
3. Créer alors un fichier `la-paix-des-mondes.txt` du livre modifié.

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) → str` qui remplace le motif dans le livre par `remplacement` et renvoie le texte modifié.
NB : On pourra utiliser un slice pour récupérer un morceau du texte :

```
1 # renvoie la chaîne entre l'indice
  debut inclus et fin exclu
2 texte[debut: fin]
```

2. Remplacer le mot `guerre` par `paix`.
3. Créer alors un fichier `la-paix-des-mondes.txt` du livre modifié.

Avant de regarder la correction

Avant de regarder la correction



- Prendre le temps de réfléchir,
- 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.

Importer un texte

Rechercher

Recherche naïve

Gestion de la casse

Évaluer l'efficacité

Algorithme de Boyer-Moore

Remplacer


```

1 def remplacer(livre: str, motif: str, remplacement: str) ->
  str:
2     """
3     remplace 'motif' par 'remplacement' dans 'livre'
4
5     Returns:
6         str: livre modifié
7     """
8     positions = boyer_moore(livre, motif)
9     livre_modifie = ""
10    debut = 0
11    for fin in positions:
12        livre_modifie += livre[debut: fin] + remplacement
13        # recommence à la fin du motif dans livre
14        debut = fin + len(motif)
15    return livre_modifie

```

Correction

```

1 def remplacer(livre: str, motif: str, remplacement: str) ->
  str:
2     """
3     remplace 'motif' par 'remplacement' dans 'livre'
4
5     Returns:
6         str: livre modifié
7     """
8     positions = boyer_moore(livre, motif)
9     livre_modifie = ""
10    debut = 0
11    for fin in positions:
12        livre_modifie += livre[debut: fin] + remplacement
13        # recommence à la fin du motif dans livre
14        debut = fin + len(motif)
15    return livre_modifie

```

```
1 modifie = remplacer(livre, "guerre", "paix")
2 fichier = open("la-paix-des-mondes.txt", "w",
3               encoding="utf8")
4 fichier.write(modifie)
5 fichier.close()
```

Création du livre

Correction

```
1 modifie = remplacer(livre, "guerre", "paix")
2 fichier = open("la-paix-des-mondes.txt", "w",
3               encoding="utf8")
4 fichier.write(modifie)
5 fichier.close()
```

Création du livre