**TP 1 Algorithme de recherche de sous-chaine**

On souhaite chercher les occurrences d’une clé dans une chaine de caractères en entrée.

Par exemple dans la chaine : « Les chaussettes de l’archiduchesse sont-elles sèches ou archi-sèches » on chercher la sous chaine « sse ».

On obtient alors « Les chaussettes de l’archiduchesse sont-elles sèches ou archi-sèches ».

On souhaite obtenir la position dans la chaine de la clé.

**Question 1**

Proposer à l’oral un algorithme naïf pour résoudre le problème.

**Question 2**

Copier un texte sur internet dans un fichier texte.

Ecrire une fonction *lire\_fichier(name) -> str :* qui prend en argument le chemin vers un fichier et retourne son contenu sous forme d’une seule chaine de caractère

**Question 3**

Ecrire une fonction *recherche\_naive(texte, cle) -> list[int] :* qui prend le texte et la sous chaine à chercher et retourne la liste des positions où la sous chaine est présente.

La liste est vide si le motif n’est pas présent.

Exemple :

recherche\_naive(« coucou », « cle ») # []

recherche\_naive(« abcdebgbc », « bc ») # [1, 7]

On peut parcourir le texte et tester à chaque emplacement si la clé est présente.

Exemple :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | abcdebgbc | ab | bc | non |
| 1 | abcdebgbc | bc | bc | oui |
| 2 | abcdebgbc | cd | bc | non |
| 3 | abcdebgbc | de | bc | non |
| 4 | abcdebgbc | eb | bc | non |
| 5 | abcdebgbc | bg | bc | non |
| 6 | abcdebgbc | gb | bc | non |
| 7 | abcdebgbc | bc | bc | oui |

On retrouve le résultat [1, 7]

On appelle ce fonctionnement une « fenêtre glissante », il faut en général 2 boucles pour l’implémenter, une pour parcourir la liste et une pour parcourir la fenêtre.

**Question 4 Algorithme de Boyer-Moore « simplifié » (****Horspool)**

Pour l’exercice réimplémentez la fonction précédente *recherche\_naive* en *echerche\_naive\_inversee* où la clé est parcourue en sens inverse. Par exemple au lieu de regarder si le caractère *i* est b et le caractère *i+1* est c, regarder si le caractère **i** est c et le caractère *i-1* est b.

**Question 5**

Dans le cas où la comparaison avec un caractère de la clé échoue, essayez de trouver les comparaisons suivantes que vous pouvez sauter.

Potentiellement faites le sur un exemple tout simple avec la comparaison de 2 mots comme « tartine » qu’on cherche dans le mot « medicament »(évidement on ne va pas trouver le mot mais pour voir les comparaison qu’il est possible de sauter)

**Question 6**

Ecrire une fonction calcule\_table\_saut(cle) → dict[str, int] qui indique pour chaque lettre de l’alphabet le nombre de comparaisons avec la clé que l’on peut sauter

**Question 7**

Utiliser cette table dans votre recherche (vous pouvez le faire dans une nouvelle fonction *Horspool).*