
Quick 1 – 4 octobre 2024 – Durée 1 h

Une feuille A4 manuscrite recto-verso autorisée. Les calculatrices et les téléphones (incluant smartphone, tablettes,... tout ce qui contient une interface réseau) sont interdits.
Les dictionnaires pour les personnes de langue étrangère sont autorisés.
Le barème est indicatif.

Vous êtes vivement encouragés à illustrer vos recherches et vos réponses par des schémas.

1 Écriture d'algorithme et complexité (13 points)

On s'intéresse dans tout ce problème à la reconnaissance d'anagrammes.

Deux phrases sont anagrammes si l'une peut être obtenue en permutant les lettres de l'autre.

Par exemple « *Le réchauffement climatique* » et « *Ce fuel qui tache le firmament* » sont des anagrammes.

Par souci de simplicité, on ne veut tenir compte ni des espaces, ni des accents, ni des majuscules. **On suppose donc que les chaînes fournies sont constituées uniquement de lettres minuscules.**

Ainsi, les deux phrases données en exemple plus haut seront en réalité fournies sous la forme des chaînes `lerechauffementclimatique` et `cefuelquitachelefirmament`.

Les algorithmes que l'on écrira vérifieront tous la même spécification :

Anagrammes(S, T)

{ **paramètres** : deux chaînes *S* et *T* (tableaux de caractères) de même longueur *n*
valeur de retour : un booléen *True* si *S* et *T* sont anagrammes l'une de l'autre, *False* sinon
effets de bord : *S* et *T* sont susceptibles d'être modifiées par l'algorithme }

(On suppose que les deux chaînes ont la même longueur *n*, sinon il est clair qu'elles ne peuvent pas être anagrammes.)

1. On propose l'algorithme naïf suivant, qui vérifie si chaque lettre de la chaîne S est présente dans la chaîne T .

```

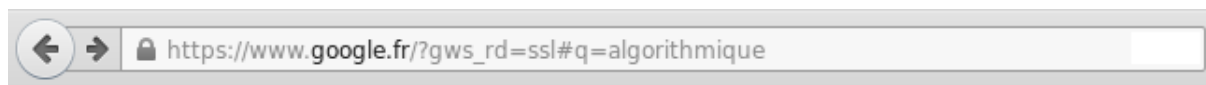
ANAGRAMMES_NAIF( $S, T$ )
  pour  $i = 0$  à  $n - 1$  faire
     $j := 0$ 
    tant que  $j < n$  et  $S[i] \neq T[j]$  faire
       $j := j + 1$ 
    si  $j = n$  alors
      renvoyer  $False$ 
  renvoyer  $True$ 

```

- (a) (1 point) Expliquer pourquoi cet algorithme est incorrect, et donner un exemple de deux chaînes de mêmes longueurs pour lesquelles il ne renvoie pas le résultat attendu.
- (b) (2 points) Écrire un algorithme qui corrige le problème que vous avez identifié, tout en conservant un fonctionnement similaire à notre algorithme naïf.
- (c) (2 points) Quelle est la complexité (au pire) de l'algorithme naïf donné dans l'énoncé ?
- (d) (1 point) Est-ce que les modifications que vous avez effectuées pour corriger l'algorithme changent sa complexité ?
2. On admet pour cette partie disposer d'un algorithme qui trie un tableau de taille n en effectuant au pire $n \log n$ comparaisons.
- (a) (2 points) Écrire un algorithme qui profite de cette possibilité pour déterminer plus efficacement si les deux chaînes fournies en argument sont anagrammes l'une de l'autre.
- (b) (2 points) Quelle est la complexité totale de cet algorithme ?
(toujours en fonction de la longueur n des chaînes à comparer)
3. On propose une dernière méthode basée sur un comptage des lettres.
On suppose pour cela qu'on travaille sur un alphabet fini A , dont on numérote arbitrairement les symboles de 0 à $|A| - 1$.
Dans notre exemple $A = \{a, b, \dots, z\}$ et on peut associer a à 0, b à 1, ... z à 25, mais on pourrait adapter ce principe pour d'autres alphabets, comme par exemple dans le code ASCII.
- (a) (2 points) Écrire un algorithme qui profite de cette possibilité pour déterminer plus efficacement si les deux chaînes fournies en argument sont anagrammes l'une de l'autre.
- (b) (1 point) Quelle est la complexité de votre algorithme ?
(en fonction de la longueur n des chaînes à comparer et de la taille $|A|$ de l'alphabet)

2 Structure de données (7 points)

À l'aide du type pile fourni page suivante, on vous demande d'implémenter un type Historique simulant les opérations permises par la barre d'un navigateur Internet :



La spécification des trois opérations à fournir est la suivante. Elles modifient toutes l'historique H par effet de bord.

(Lorsqu'une opération n'est pas possible, l'historique ne sera pas modifié.)

Visiter(A, H)

```
{ paramètres : A : une adresse web, H : un historique
  valeur de retour : aucune
  description : mémorise dans H l'adresse web courante A afin de pouvoir la retrouver
  après la visite d'autres pages
  effets de bord : H est modifié
}
```

Reculer(H)

```
{ paramètres : H : un historique
  valeur de retour : la nouvelle adresse web courante
  description : va à la page visitée juste avant la page courante dans l'historique.
  effets de bord : H est modifié
}
```

Avancer(H)

```
{ paramètres : H : un historique
  valeur de retour : la nouvelle adresse web courante
  description : va à la page visitée juste après la page courante dans l'historique
  (ce qui permet d'annuler l'effet d'une opération Reculer)
  effets de bord : H est modifié
}
```

1. (3 points) Représenter à l'aide de schémas comment vous allez utiliser la structure de pile pour mémoriser les pages web visitées et permettre de consulter cet historique.
2. (4 points) Écrire le pseudo-code correspondant à chacune des 3 opérations Visiter, Reculer et Avancer.

Pour réaliser cet historique, vous utiliserez le type abstrait **Pile** ci-dessous, où les primitives Empiler et Dépiler opèrent par effet de bord sur leurs arguments. Il n'est **pas demandé** d'implémenter les primitives du type **Pile**.

PileVide()

```
{ paramètres : aucun
  valeur de retour : une pile vide
}
```

Empiler(x, P)

```
{ paramètres : x : un élément, P : une pile
  valeur de retour : aucune
  effet de bord : après l'appel à Empiler, P contient x à son sommet, suivi des éléments déjà présents dans P
}
```

Sommet(P)

```
{ paramètres : P : une pile
  précondition : P n'est pas vide
  valeur de retour : l'élément en tête de P
  effet de bord : P n'est pas modifiée
}
```

Depiler(P)

```
{ paramètres : P : une pile
  précondition : P n'est pas vide
  valeur de retour : aucune
  effet de bord : après l'appel à Depiler, P contient les mêmes éléments qu'avant sauf son sommet
}
```

EstVide(P)

```
{ paramètres : P : une pile
  valeur de retour : un booléen
  description : détermine si P est vide
}
```