Série 2 : Complexité des algorithmes, Structure de données avancées et NP-Compéltude

Exercice1 :

Déterminer un algorithme qui teste si un tableau de taille n est un ”tableau de permutation” (i.e. tous les éléments sont distincts et compris entre 1 et n).

1. Donner un premier algorithme naïf qui soit quadratique.
2. Donner un second algorithme linéaire utilisant un tableau auxiliaire.
3. Donner un troisième algorithme linéaire sans utiliser un tableau auxiliaire.

Exercice 2 : Produit matriciel

On considère deux matrices carrées (d'entiers) d'ordre *n*, *A* et *B*. Le produit de *A* par *B* est une matrice carrée *C* définie par :

$$C_{i,j} = \sum_{k=1}^{n} A_{i,k} * B_{k,j}$$

1. Donner un algorithme calculant le produit de deux matrices représentées sous forme d'un tableau à deux dimensions. Calculer la complexité de cet algorithme.
   Doit-on préciser dans quels cas (pire cas, meilleur des cas, cas moyen) cette complexité est obtenue ?
2. Modifier l'algorithme précédent lorsque la matrice *A* est de dimension (*m,n)* et la matrice *B* de dimension (*n, p).* Quelle est alors la complexité de l'algorithme ?

Exercice 3 :

Considérer le théorème suivant sur le pgcd ou le plus grand commun diviseur : pgcd(a,b) = pgcd(b, a mod b)

1. Utiliser le théorème pour écrire un algorithme de calcul itératif du pgcd de deux entiers a et b.
2. Calculer la complexité en temps de l'algorithme
3. Ecrire la version récursive de l'algorithme et calculer sa complexité
4. Ecrire un algorithme pour calculer le ppcm de deux nombres a et b et calculer sa complexité

Exercice 4 :

Considérer un tableau constitué des n premiers nombres entiers (1, 2, …, n). Une méthode de détermination des nombres premiers inférieurs au sens large à n, consiste à considérer le nombre 2 qui est premier puis à éliminer tous les nombres multiples de 2 car ils ne sont pas premiers, ensuite itérer ce processus en continuant avec le nombre suivant non éliminé c'est-à-dire 3 jusqu'à traiter tous les entiers du tableau.

1. Illustrer chaque itération du procédé de calcul des nombres premiers décrit ci-dessus sur les 10 premiers nombres entiers

2. Ecrire un algorithme de calcul et d'impression des nombres premiers inférieurs au sens large à n. Il est recommandé par souci de simplification de l'algorithme de mettre le nombre à 0 s'il est premier et à 1 sinon, une fois qu'il est traité.
3. Calculer la complexité de l'algorithme.

Exercice 5: Complexité en fonction de deux paramètres

Déterminer la complexité des algorithmes suivants (par rapport au nombre d'itérations effectuées), où *m* et *n* sont deux entiers positifs.

Algorithme A

```
i ← 1 ; j ← 1
tant que (i≤m) et (j≤n) faire
    i←i+1
   j←j+1
fait
```

Algorithme B

```
i ← 1 ; j ← 1
tant que (i≤m) ou (j≤n) faire
     i←i+1
    j←j+1
fait
```

Algorithme C

```
i ← 1 ; j ← 1
tant que (j≤n) faire
 si i≤m
 alors
    i←i+1
sinon
   j←j+1
fin si
fait
```

Algorithme D

```
i ← 1 ; j ← 1
tant que (j≤n) faire
 si i≤m
 alors
    i←i+1
sinon
    j←j+1 ;   i ← 1
fin si
fait
```

Exercice 6 : Recherche séquentielle vs Recherche dichotomique

On considère un tableau *A* de *n* éléments, que l'on suppose trié en ordre croissant. On cherche à construire un algorithme permettant de déterminer la position d'une valeur *clé*. (On suppose que *clé* existe dans le tableau on recherchera la première occurrence de cette valeur).

I. Considérant un parcours séquentiel
   1. Donner un algorithme itératif qui résout ce problème. Indiquer un invariant de boucle pour cet algorithme.
   2. A quoi correspond le pire des cas ? En déduire la complexité en *O* de l'algorithme.
   3. A quoi correspond le meilleur des cas ? En déduire la complexité en *O* de l'algorithme.
   4. Ecrire cet algorithme sous forme récursive.

Comment faut-il modifier ces versions (itérative et récursive) de l'algorithme si l'on n'est pas sûr que *clé* appartienne au tableau ?

I. Reprendre les questions précédentes en considérant un parcours dichotomique

Exercice 7 :

1) Ecrire un algorithme pour inverser l'ordre d'une liste. Calculer sa complexité
2) Ecrire un algorithme pour créer une liste doublement chainée à partir d'une liste quelconque. Calculer sa complexité

Exercice 8 :

Considérer deux listes de nombres entiers triées par ordre croissant. On souhaiterait fusionner ces deux listes pour obtenir une troisième liste triée.

Ecrire un algorithme de fusion de deux listes triées

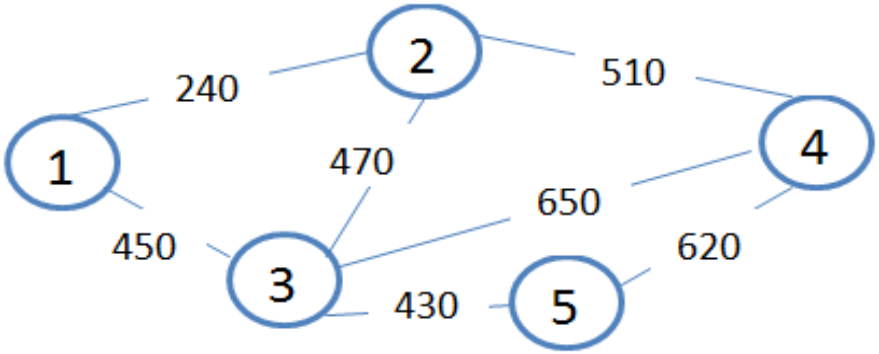
a. En utilisant la structure de tableau
b. En utilisant la structure dynamique de listes

Quelle est la complexité de votre algorithme.

Exercice 9:

Soit une variante du problème de voyageur de commerce simplifié défini comme suit :

- Soient en entrée un ensemble de ville relié entre elle par des routes de distance connue
- Un commerçant a besoin de définir un chemin de langueur <=K passant par toutes les villes. (dans l'exemple K=2000).



-

On s'intéresse à la résolution de cette variante du problème en utilisant l'algorithme de Djikstra permettant de déterminer le chemin de plus court entre deux nœuds dans un graphe

1. Proposer une modélisation adéquate du problème
2. proposer un algorithme permettant de résoudre cette variante du problème.

Calculer la complexité de l'algorithme proposé. Que peut-on conclure ?

En rajoutant à cette variante du problème une contrainte d'unicité du passage sur chaque ville on obtient la variante classique du problème du voyageur de commence où le commerçant a besoin de passer par chaque ville une et une seule fois.

1. Proposer un algorithme permettant de résoudre cette variante du problème
2. Calculer la complexité du problème. Que peut-on conclure ?
3. Proposer un algorithme de validation pour cette variante du problème

Que peut-on conclure ?