Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Ghardaïa
Faculté des Sciences et de la Technologie
Département des Mathématiques et d'Informatique

Concours d'accès à la formation de troisième cycle en vue de l'obtention du diplôme de doctorat au titre de l'année universitaire 2021-2022

Filière : Informatique

Spécialité : Systèmes Intelligents et Apprentissage Automatique

Module : Conception et Analyse des Algorithmes **Durée : 01 heure 30'** **Coefficient : 1**

Exercice 1 : Usage de l'espace mémoire et ordre du grandeur (4,50 points)

Supposons que la bibliothèque Java java.util.LinkedList est implémentée en utilisant une liste linéaire doublement chainée, en maintenant une référence sur le premier et le dernier nœud dans la liste ainsi que sa taille.

```
public class LinkedList<Item> {
    private Node first; // Le premier noeud de la LLC
    private Node last; // Le dernier  noeud de la LLC
    private int N; // Le nombre d'éléments dans la LLC
    private class Node {
        private Item item; // L'élément de type Item n'est pas un type primitif.
        private Node next, prev; // L'élément  suivant et précédent
    }
    ...
}
```

En utilisant un modèle de coût mémoire 64-bits :

1. Calculer le nombre d'octets utilisé par un objet «**Node**», sachant qu'un objet interne requiert un extra entête de 8 octets pour faire référence à l'instance qui le contient.
2. Calculer le nombre d'octets utilisé par un objet «**LinkedList**» pour sauvegarder N éléments (Ne tenez pas en compte l'espace mémoire pour les éléments eux même mais prenez en compte l'espace pour les références mémoire envers eux).
3. Quel est l'ordre de grandeur du temps d'exécution dans le **pire des cas** de chaque opération ci-dessous :

Fonction	Commentaire
addFirst(item)	Ajouter un élément au début de la liste.
get(i)	Renvoyer l'élément à la position i dans la liste.
set(i, item)	Remplacer l'élément de la position i par l'élément « item ».
removeLast()	Supprimer et renvoyer l'élément de la dernière position de la liste.
contains(item)	Est ce que l'élément "item" existe dans la liste ?

Exercice 2 : Méthodes de tri (04.50 points)

Considérons un tableau qui contient deux copies successives des entiers 1 à n dans l'ordre croissant. Par exemple, voici un tableau quand $n = 8$:

1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8

Notez que la longueur du tableau est $2n$.

1. Combien de comparaisons (en fonction de n) le tri par **sélection** fait-il pour trier ce type de tableau ? Utilisez la notation tilde.
2. Combien de comparaisons (en fonction de n) le tri par **insertion** fait-il pour trier ce type de tableau ? Utilisez la notation tilde.
3. Combien de comparaisons (en fonction de n) le tri par **fusion** fait-il pour trier ce type de tableau ? Supposez que n est une puissance de 2. Utilisez la notation tilde.

Exercice 3 : Problème du voyageur de commerce (06.50 points)

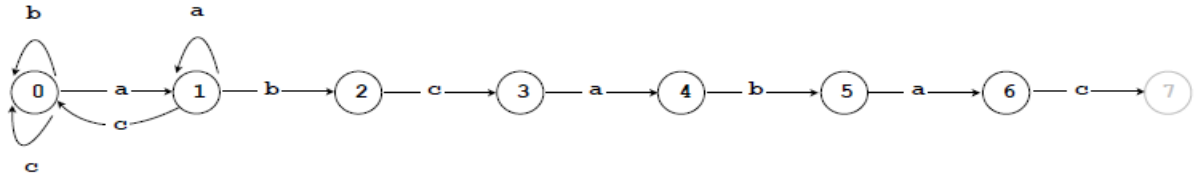
1. Définir le problème du voyageur de commerce (Traveling Salesperson Problem, TSP).
2. Donner un pseudo-algorithme de la solution brute du problème du voyageur de commerce.
3. Calculer la complexité temporelle de la solution brute.
4. Donner le principe des algorithmes gloutons en général et celui pour le problème du voyageur de commerce.
5. Présenter l'algorithme glouton de base pour le problème du voyageur de commerce.
6. Donner la complexité de l'algorithme glouton de base pour le problème du voyageur de commerce.
7. Expliquer la notion de **NP-hard** et **NP-complete** par le bias du problème du voyageur de commerce

Exercice 4 : Algorithme KMP (04.50 points)

1. Créez un automate fini déterministe (DFA) de Knuth-Morris-Pratt pour le mot $\boldsymbol{abcabac}$ sur l'alphabet $\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$ en complétant la table suivante. L'état **0** est l'état initial et l'état **7** est l'état final.

	0	1	2	3	4	5	6
a	1	1		4		6	
b	0	2			5		
c	0	0	3				7

Vous pouvez utiliser la représentation graphique partiellement complète suivante du DFA.



2. Quel est le but de l'algorithme de Knuth-Morris-Pratt ?
3. Quelle est la complexité temporelle de l'algorithme KMP pour atteindre son but ?

Bon courage

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Ghardaïa
Faculté des Sciences et de la Technologie
Département des Mathématiques et d'Informatique

Concours d'accès à la formation de troisième cycle en vue de l'obtention du diplôme de doctorat au titre de l'année universitaire 2021-2022

Filière : Informatique

Spécialité : Systèmes Intelligents et Apprentissage Automatique

Module : Conception et Analyse des Algorithmes **Durée : 01 heure 30'** **Coefficient : 1**

Exercice 1 : Usage de l'espace mémoire et ordre du grandeur (4,50 points)

1. Le nombre d'octets utilisé par un objet «**Node**» **(0.25*6 = 1.50 pts)**

   L'objet «Node» utilise 48 octets tel que illustré dans la figure suivante :

	Node
16	Entête objet
8	Entête objet interne
8	Réf : item
8	Réf : next
8	Réf : prev
0	Padding

48 octets

2. Le nombre d'octets utilisé par un objet «**LinkedList**» **(0.50 + 0.25*5 = 1.75 pts)**

Un objet «LinkedList» utilise 40 octets. Dans le cas où l'objet comporte N éléments, l'espace mémoire utilisé sera 40 + 48 N octets.

	LinkedList
16	Entête objet
8	Réf : first
8	Réf : last
4	N
4	Padding

40 octets

3. **(0.25*5 = 1.25 pts)**

Fonction	Ordre de grandeur
addFirst(item)	1
get(i)	N
set(i, item)	N
removeLast()	1
contains(item)	N

Exercice 2 : Méthodes de tri (04.50 points)

1. $\sim 2n^2$. **(1.50 pts)**

Le tri par sélection effectue toujours $\sim \frac{1}{2}m^2$ comparaisons pour trier un tableau de longueur m. Ici $m = 2n$.

2. $\sim \frac{1}{2}n^2$. **(1.50 pts)**

Chaque entier i dans la moitié à droite est inversé avec n – i éléments dans la moitié à gauche. Alors le nombre d'inversions est $(n-1) + (n-2) + \ldots + 1 + 0 = \sim \frac{1}{2}n^2$.

3. $\sim \mathrm{n} \log_2 n$. **(1.50 pts)**

Considérons le haut niveau de l'algorithme récursive. Le tri des sous-tableaux gauche et droite prend chacun $\frac{1}{2}\mathrm{n} \log_2 n$ comparaisons. (Rappelons qu'un tableau trié est le meilleur cas pour le tri par fusion). La fusion des deux sous-tableaux (chacun de taille n) prend $2n - 1$ comparaisons.

Exercice 3 : Problème du voyageur de commerce (06.50 points)

1. Définition TSP **(0.75 pt)**

Dans le problème de voyageur de commerce (TSP), étant donné n villes avec une ville de départ et toutes les distances entre les paires de villes, planifiez une visite commençant et se terminant à la ville de départ qui visite chaque ville exactement une fois et a une distance minimale.

2. Pseudo-algorithme de la solution brute du TSP **(1.25 pts)**

```
bestpath = null
bestdistance = +infinity
for each permutation of cities, starting and ending at hometown   (n-1)! permutations
calculate distance of current permutation                         O(n)
        if (distance < bestdistance)                              O(1)
                bestdistance = distance
                bestpath = current permutation
return bestpath
```

3. Complexité temporelle de la solution brute **(0.75 pt)**

$(n-1)!*n \Rightarrow O(n!)$

4. **(0.75 pt)**

Principe des algorithmes gloutons: Essayez d'améliorer une situation autant que possible à ce moment particulier en pensant uniquement au présent et non au futur..
Principe de l'algorithme glouton TSP : Choisir la ville la plus proche en considérant la ville courante

5. Algorithme glouton de base pour le TSP **(1.25 pts)**

```
bestpath = []
current = hometown
citiestovisit = all other cities
while (more cities to visit)
        select city closest to current and add it to the bestpath   O(n)
        remove current city from cities to visit
        current = selected city
return bestpath
```

6. Complexité de l'algorithme glouton de base pour le TSP **(0.75 pt)**

$(n-1)*n \Rightarrow O(n^2)$, Beaucoup mieux que O(n!).

7. NP-hard et NP-complete **(1.00 pt)**

NP-hard: les problèmes qui sont au moins aussi difficiles à résoudre que le problème le plus difficile dans NP
exemple: TSP optimisation : le problème de TSP avec la contrainte "distance minimale"

NP-complete: Aucun algorithme de temps polynomial connu pour trouver une solution, mais on peut vérifier la solution en temps polynomial.
exemple: TSP décision : le problème de TSP avec la contrainte " avoir une distance inférieure à L"

Exercice 4 : Algorithme KMP (04.50 points)

1. Table de transitions **(10*0.25=2.50 pts)**

	0	1	2	3	4	5	6
a	1	1	1	4	1	6	1
b	0	2	0	0	5	0	2
c	0	0	3	0	0	3	7

2. But de l'algorithme de Knuth-Morris-Pratt **(1.00 pt)**

L'algorithme de Knuth-Morris-Pratt cherche un motif de longueur M dans un texte de longueur N.

3. Complexité **(1.00 pt)**

La complexité temporelle de l'algorithme KMP est $O(N + M)$.

Bon courage