TP3 Noté

Récupérez sur *Moodle* les fichiers `progListeSC.h`, `progListeSC.cpp`, `fichierTP3.h`, `fichierTP3.cpp`, `mainTP3.cpp` et `Makefile`.

À la fin de la séance, vous déposerez votre fichier fichierTP3.cpp.

Vous répondrez aux questions de complexité et temps de calcul directement dans le fichier.

Veillez à ce qu'il n'y ait pas d'erreur de compilation ! Sinon vous aurez 0 !

On appelle *Liste Intervalle* une liste triée d'entiers consécutifs. Par exemple les 4 séquences suivantes sont des Listes Intervalle : $(1, 2, 3)$, $(4, 5, 6, 7, 8, 9)$, (7), $()$.
Ouvrez le fichier `fichierTP3.cpp`.

Question 1 Complétez la fonction `estListeIntervalle` qui vérifie si une Liste est une Liste Intervalle.
Compilez (`make`). Exécutez la première partie du `main`, avec l'option 1.

Question 2 La suite donne un algorithme construisant une Liste Intervalle :

Algorithme : créerListeIntervalle(**d** l :entier , **d** p :entier) : ListeSC
Données : l est un entier positif, p est un entier
Résultat : Renvoie la Liste Intervalle de longueur l dont le premier élément est l'entier p

Par exemple `créerListeIntervalle(3,4)` doit renvoyer la liste dont la séquence d'éléments est $(4, 5, 6)$.
La fonction `creerListeIntervalle1` donne un premier algorithme pour cette opération. Cet algorithme utilise l'opération `insererFinLSC`. Quelle est la complexité de `creerListeIntervalle1`, exprimée en fonction de la longueur de la Liste Intervalle à construire ?
Proposez dans la fonction `creerListeIntervalle2` un second algorithme pour cette opération de complexité inférieure à celle de `creerListeIntervalle1`. Quelle est sa complexité ?
Proposez dans la fonction `creerListeIntervalle3` un troisième algorithme qui doit être récursif. Quelle est sa complexité ?
Compilez et testez avec l'option 2.

Question 3 L'exécution avec l'option 3 permet de comparer les temps d'exécution des 3 fonctions. Faites plusieurs essais en augmentant la longueur de la liste intervalle à générer.
Quels sont les temps de calcul de ces 3 fonctions pour construire une liste de taille 50000 ?

Question 4 Soit l'opération `transfListeIntervalle` qui transforme une liste triée sans répétition en liste intervalle.

Algorithme : transfListeIntervalle(**dr** L : ListeSC)
Données : L est une liste d'entiers **non vide, triée et sans répétition**
Résultat : Modifie la liste L en y insérant des éléments de sorte qu'elle soit une Liste Intervalle

Par exemple si le paramètre a pour séquence d'éléments (2, 5, 6, 8), la fonction modifie cette liste de sorte qu'elle devienne (2, 3, 4, 5, 6, 7, 8).
Complétez la fonction `transfListeIntervalle` pour qu'elle réalise cette opération. Votre algorithme doit modifier la liste paramètre et non en créer une nouvelle ; les cellules existantes doivent donc apparaître dans la liste résultat. Quelle est la complexité de votre algorithme ?
Compilez et testez avec l'option 4.

Question 5 Écrivez une fonction qui calcule l'intersection de 2 listes intervalles.

Algorithme : intersectionListesIntervalles(**d** $L1$: ListeSC, **d** $L2$: ListeSC) : ListeSC
Données : $L1$ et $L2$ sont 2 listes intervalles
Résultat : Renvoie une nouvelle liste intervalle, intersection des listes intervalles $L1$ et $L2$, qui ne partage aucune cellule avec les listes $L1$ et $L2$.

Par exemple l'intersection des listes intervalles (2, 3, 4, 5, 6) et (5, 6, 7, 8, 9) est la liste intervalle (5, 6).
Complétez la fonction `intersectionListesIntervalles`. Le résultat doit être une nouvelle liste. Il serait de bon ton que la complexité de cet algorithme soit linéaire dans la somme des tailles des listes $L1$ et $L2$.
Compilez et testez avec l'option 5.

Question 6 Écrivez une fonction qui étant donnée une liste en extrait la plus longue sous-liste qui est une liste d'intervalle.

Algorithme : plusLgSsLiInterv(**dr** L : ListeSC)
Données : L une liste
Résultat : Le résultat est la plus longue sous–liste intervalle de L ; aucun élément n'est recopié, il faut modifier L en supprimant les éléments n'appartenant pas à la plus longue sous–liste intervalle ; en cas d'égalité le résultat est l'une des plus longues sous–listes intervalle.

Par exemple si L est la liste (2 3 6 8 9 4 5 6 7 9 10), après `plusLgSsLiInterv(L)`, L est la liste (4 5 6 7).
Quelle est la complexité de votre algorithme ?