Le but est de programmer en C++ quelques algorithmes sur les tableaux et d'observer en pratique les différences entre classes de complexité.

Introduction

Fichiers à récupérer Depuis l'*Espace pédagogique* récupérez les fichiers `mainTP1.cpp`, `outilsTab.cpp`, `outilsTab.h`, ainsi que les fichiers `trace1.gnu, trace2.gnu, trace3.gnu` et `Makefile`.
Le fichier `mainTP1.cpp` contient le main.
Le fichier `outilsTab.cpp` contient les définitions de fonctions sur les tableaux d'entiers (affichage, génération aléatoire, chronométrage de fonctions) ainsi que des fonctions que vous devrez compléter. Leurs spécifications sont données dans le fichier entête `outilsTab.h`.
Les fichiers `traceX.gnu` contiennent des directives `gnuplot` pour l'affichage de courbes.

Environnement de programmation Vous utiliserez votre éditeur favori. Pour la ompilation vous pouvez utiliser la commande `make` qui génère l'exécutable `tp1`.
Pour une exécution sans erreur de `tp1`, le répertoire courant doit contenir les 3 fichiers `traceX.gnu`.

Calcul du sous-tableau de somme maximum

Données : un tableau $T[1 \dots n]$ de valeurs prises dans $\mathbb{Z}$
Résultat : la somme maximale des sous-Tableaux de T

Nous avons vu en TD 4 algorithmes pour ce problème de complexité $O(n^3)$, $O(n^2)$, $O(n.log(n))$, $O(n)$ (n est la taille du tableau).
`ssTabSomMax1`, `ssTabSomMax2` et `ssTabSomMax3` sont les fonctions correspondant aux 3 premiers algorithmes.

Question 1

Complétez la fonction `ssTabSomMax4` du fichier `outilsTab.c` pour implanter l'algorithme de complexité $O(n)$.
Pour vérifier votre fonction, lancez l'exécution du programme avec l'option 1.
Le programme génère un tableau aléatoire de 1000 entiers compris entre -100 et 100, puis exécute les 4 fonctions `ssTabSomMaxX` et affiche pour chacune d'entre elles la somme maximum calculée et le temps d'exécution. Vérifiez que votre fonction `ssTabSomMax4` renvoie le même résultat que les 3 autres.

Question 2

Il s'agit d'observer les temps d'exécution de ces 4 fonctions en faisant varier la taille du tableau généré aléatoirement.
Lancez l'exécution du programme avec l'option 2.
Les temps des 4 algorithmes sont comparés pour des tableaux dont la taille varie de 100 à 1000 éléments. Sur le graphique affiché, la courbe de l'algorithme $O(n^3)$ se détache des 3 autres, à peine visibles.
Pour comparez les 3 autres algorithmes, il faut augmenter la taille des tableaux générés. Dans le deuxième graphique cette taille varie de 1000 à 20000.
Pour comparez les 2 algorithmes de complexité $O(n.log(n))$ et $O(n)$, on fait varier la taille des tableaux de 100000 à 2000000.
Quel est approximativement le temps d'exécution de l'algorithme linéaire ($O(n)$) pour calculer la somme max d'un tableau de taille 10 millions ?
Inscrivez votre réponse dans le fichier `outilsTab.cpp`.

Question 3

Toujours pour ce même problème on souhaite à présent obtenir, outre la somme maximum, les indices de début et de fin d'un sous-tableau de somme maximum.
Exemple : pour le tableau ci-dessous, il faut renvoyer la somme max 190 et les indices 2 et 10. Ces 3 valeurs sont renvoyées dans une structure `triplet` définie dans `outilsTab.h` et dont les champs entiers sont `deb, fin, somMax`.

0	*1*	**2**	*3*	*4*	*5*	*6*	*7*	*8*	*9*	**10**	*11*	*12*	*13*	*14*
31	-41	59	26	-53	58	97	-93	-23	84	35	-98	-80	-72	-85

Complétez la fonction `indSsTabSomMax` pour qu'elle renvoie **en temps linéaire** la valeur de la somme max ainsi que les indices du sous-tableau de somme max.
Vérifiez votre fonction en lançant l'exécution du programme avec l'option 3, qui génère un tableau aléatoire de taille 20 et affiche les 3 résultats.

Question 4

Ranger les pairs d'un tableau

Vous devez écrire un algorithme pour le problème :

Données : T un tableau d'entiers
Résultat : modifie le tableau T en déplaçant ses éléments de sorte que tous les nombres impairs soient placés après tous les nombres pairs. Autrement dit en fin d'algorithme, T est une permutation du tableau en donnée et pour tout couple d'indices (i, j), si $T[i]$ est pair et $T[j]$ est impair alors $i < j$.

Complétez la fonction `rangerPairs` implantant un **algorithme linéaire** pour ce problème.
Donnez de préférence un algorithme n'utilisant pas de tableau auxilliaire.
Lancez l'exécution avec l'option 4 et vérifiez si le tableau modifié est correct.