Feuille de Travaux Pratiques n^o 1
Calculs de temps d'exécution

Le TP est à réaliser **en binôme** (un seul monôme est autorisé par groupe de TP, si celui-ci possède un nombre impair d'étudiant). Trois séances de TP encadrées sont consacrées à cette feuille de TP. Le TP sera évalué par un compte-rendu que vous devrez déposer sur madoc (voir les dates limites sur madoc en fonction de votre groupe de TP) au format PDF. Le nom du fichier devra respecter le format suivant : *TP1_nom1_nom2.pdf*
Un malus sera appliqué à chaque fois qu'une des consignes données ci-dessus n'aura pas été respectée. Notamment, en cas de retard, le malus sera le suivant : -1 point par heure de retard (toute heure entamée étant comptée comme complète).

L'objectif de ce TP est de vérifier que les résultats théoriques obtenus en TD correspondent (du moins en ordre de grandeur) aux résultats dans la pratique. Le langage de programmation utilisé pour implémenter les algorithmes et pouvoir ainsi répondre aux différentes questions est laissé au choix.

Pour chaque exercice, les instructions sont séparées en deux parties : la partie programmation qui détaille les étapes nécessaires pour implémenter et tester les algorithmes (cette partie n'est pas à rendre) ; la partie compte-rendu qui précise les éléments à indiquer dans le rapport que vous devez déposer sur madoc.

Exercice 1.1 (Algorithme `mystere`)
Dans cet exercice, nous allons traiter l'algorithme `mystere` de l'Exercice 1.2 de la feuille de TD1.

Partie programmation :

1. Implémenter un générateur de tableaux de booléens, capable de produire (soit à la volée, soit en les stockant) tous les tableaux de booléens ayant une taille n.
2. Implémenter l'algorithme `mystere`.
3. Réaliser un programme qui, pour un n donné, écrit dans un fichier les temps au mieux, au pire et en moyenne de `mystere`.
   Remarques :
   — le calcul du temps d'exécution ne doit se faire que pour l'appel à `mystere` (la génération des tableaux doit être exclue de ce calcul de temps) ;
   — `mystere` étant très rapide (entre $O(1)$ et $O(n)$), il faudra peut-être le ralentir artificiellement, par exemple en le forçant à s'arrêter pendant une durée donnée dès qu'on entre dans `mystere`.
4. Tester ce programme pour toutes les valeurs de $n \geq 1$. On s'arrêtera quand, pour un n donné, le temps cumulé d'exécution de l'algorithme `mystere` sur l'ensemble des tableaux de taille n (*sans compter la génération des tableaux*) dépasse 3 minutes. On laissera cependant les tests se terminer pour cette valeur de n, qui sera donc la dernière testée.

Partie compte-rendu :

1. Expliquer (en français) la méthode (l'algorithme) mise en place pour générer les instances de `mystere`.
2. Reporter sur un graphique (par exemple en utilisant un tableur) les temps au mieux, en moyenne et au pire de `mystere` (en ordonnée) en fonction de n (en abscisse). Vous préciserez le nombre de valeurs de n que vous avez testées tout en respectant les contraintes ci-dessus.
3. Discuter les résultats obtenus en pratique par rapport aux résultats théoriques calculés en TD.

Exercice 1.2 (Calcul de la Plus Grande Somme Consécutive)

Dans cet exercice, nous allons nous intéresser à trois algorithmes qui résolvent le problème MAX-SOMME-CONSÉCUTIVE (Exercice 1.8 de la feuille TD1) : `MaxSomme1`, `MaxSomme2` et `MaxSomme3`.

Partie programmation :
La différence ici est que les tableaux en entrée contiennent des entiers positifs et négatifs : on ne peut donc pas tous les générer pour une taille n donnée. Pour chaque valeur de n testée, on générera donc un *grand échantillon* de tableaux suivant les règles ci-dessous :

- chaque entier du tableau sera choisi au hasard entre $-\frac{n}{2}$ et $\frac{n}{2}$ (attention : pas de 0 dans le tableau, et au moins un positif **et** un négatif (cf énoncé de l'Exercice 1.8)) ;
- pour un n donné, on arrête les tests quand le temps cumulé d'exécution de l'algorithme *sans compter la génération des tableaux* dépasse 3 minutes.

Les valeurs de n à tester ici sont les suivantes : de 50 en 50 jusque 1000, de 500 en 500 jusque 10.000, etc.

Pour le reste, les étapes sont les mêmes que dans l'Exercice 1.1 ci-dessus, c'est-à-dire :

1. Génération des tableaux.
2. Implémentation des algorithmes `MaxSomme1`, `MaxSomme2` et `MaxSomme3`.
3. Tests pour chaque valeur de n jusque 100.000.

Partie compte-rendu :

1. Expliquer (en français) la méthode (l'algorithme) mise en place pour générer les instances des algorithmes testés.
2. Reporter sur un graphique (par exemple en utilisant un tableur) les temps au mieux, en moyenne et au pire de chaque algorithme (en ordonnée) en fonction de n (en abscisse). Sur un autre graphique, vous indiquerez le nombre de tableaux générés (en ordonnée) pour chaque valeur de n (en abscisse).
3. Discuter les résultats obtenus en pratique par rapport aux résultats théoriques calculés en TD.