Algorithmique et structures de données 2
TP2

am@up8.edu

2022

Dans ce TP :
— Révisions S1
— Outils de débuggage

1 Exercice 1 : complexités

Question 1 Complétez le tableau au dos.

Un temps de 10ns est proposé ici pour l'exécution d'un algorithme en temps constant avec une taille de données `n` valant 5.

— Afin que cela soit lisible, proposez des valeurs en choisissant l'unité la plus appropriée (ns, μs, s, h, jour, année).
Pour rappel : 1ns = 10^{-9}s, 1μs = 10^{-6}s, 1ms = 10^{-3}s, 1h = 3 600s, 1 année = 365j, etc.
— Seules les cases vides (sans « - ») sont à remplir.
— Si vous avez vu des algorithmes correspondant aux complexités du tableau, renseignez-les.

Question 2 Triez les complexité par ordre croissant en utilisant « < » tel que « 1 < 2 » signifie : la complexité de type 2 croît plus vite que la complexité de type 1 :

< < < < < < < <

	Temps	**Type**	**n = 5**	**n = 10**	**n = 50**	**n = 250**	**n = 1 000**	**n = 10 000**	**n = 1 000 000**
1	O (1)	constante	10 ns						
2	O (log(n))	logarithmique							
3	O (n log (n))	linéarithmique							
4	O (n)	linéaire							
5	O (n)	racinaire							
6	O (n !)	factorielle				-	-	-	-
7	O (2^n)	exponentielle					-	-	-
8	O (n^2)	quadratique							
9	O (n^3)	cubique							

2 Exercice 2

Question 1 compléter le programme `exercice2-1.c` pour que le print final soit correct.

Question 2 compléter le programme `exercice2-2.c` pour que le print final soit correct. Attention, vous ne pouvez pas déclarer d'autre variable que `a` et `b`.

Exercice 3 Récupérez sur moodle les codes des fonctions vus en cours et utilisez les outils de debuggage présentés pour les corriger / afficher les valeurs.

- `liste.c`, `liste.h` : créer une liste et afficher les éléments qu'elle contient (adresses et valeurs) dans `gdb`.
- testez et corrigez le code `structbug.c`.
- testez et corrigez le code `menu.c` (il faut *vider le tampon d'entrée*).

3 Exercice 3 : fuites mémoire et outil `valgrind`

- `fuite.c` : menez le diagnostic avec valgrind et corrigez le code.

4 Exercice 4 : Arbres binaires de recherche

À rendre pour le 08/02/2022 12h : Vous prendrez soin de fournir un code clair et commenté, ainsi que différents tests permettant de valider le bon fonctionnement de votre programme.

La structure d'arbre utilisée est :

```
typedef struct s_noeud_t
{
    int v;
    struct noeud* g;
    struct noeud* d;
} noeud_t;
```

Proposez une implémentation d'une fonction qui prend en paramètre un arbre binaire, et renvoie un entier non nul si celui-ci est un arbre binaire de recherche, 0 sinon. La fonction doit également déterminer les valeurs minimales et maximales de l'arbre, et les renvoyer.