Algorithmique et Structures de données 1

LIV 2022-2023

Travaux Dirigés 3

Site du cours : `https://defelice.up8.site/algo-struct.html`
Les exercices marqués de (@) sont à faire dans un second temps.

Exercice 1. *Fibonacci*

La suite $(F_n)_{n\in\mathbb{N}}$ de Fibonacci est définie ainsi : $F_0 = 0$, $F_1 = 1$ et $F_{n+2} = F_n + F_{n+1}$.

1. Écrire une fonction `int fiboR(int n)` qui renvoie le `n`-ième terme de la suite. Le corps de la fonction ne doit pas contenir de boucle (`for` ou `while`).
2. Écrire une fonction `int fiboB(int n)` qui renvoie le `n`-ieme terme de la suite. Le corps de la fonction ne doit pas contenir d'appel à une fonction.

Exercice 2. *Tableau*

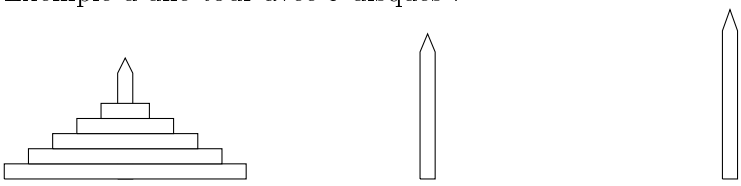
Écrire deux fonctions (récursives) `int sommeF(int taille,int* T)` et `int sommeD(int taille,int* T)`. Chacune doit faire la somme des éléments du tableau `T` mais sans utiliser de boucle.

— `sommeF` doit faire la somme en <<commençant>> par la fin du tableau.
— `sommeD` doit faire la somme en << commençant >> par le début.

Exercice 3. *La tour d' Halboï*

La légende de la tour de Hanoï raconte que dans un temple, 64 disques d'or tous troués en leur milieu, sont empilés sur trois aiguilles de diamant. À la naissance du monde, les disques étaient tous enfilés sur la plus petite des trois aiguilles, le disque au plus grand diamètre reposant sur le sol et chaque autre disque reposait sur un disque ayant un diamètre plus grand. Les prêtres de Brahmâ déplacent continuellement les disques selon les règles suivantes : Un seul disque est déplacé à la fois. Un disque déplacé ne peut être qu'enfilé sur une des trois aiguilles et posé : soit sur le sol, soit sur un disque plus grand. La légende dit aussi que l'instant où tous les disques seront empilés sur la plus grande aiguille sera aussi celui de la fin du monde.

Exemple d'une tour avec 5 disques :



1. Quels sont les mouvements à faire pour déplacer une pile de 3 disques d'une aguille à une autre ? Et une pile de 4 disques ?
2. Comment peut-on déplacer une pile de n disques d'une aiguille A à une aiguille C ?
3. Écrire une fonction
   `void hanoi(int n,char aiguilleDepart,char aiguilleIntermediaire,char aiguilleArrive)` en C qui affiche la liste des déplacements à faire pour une tour de n disques. Les aiguilles seront appelés `'A' 'B' 'C'`.

   Exemple :

```
l'exécution de
hanoi(3,'A,'B','C');
doit afficher  :

A->C
A->B
C->B
```

```
A->C
B->A
B->C
A->C
fin
```

Pour afficher vous pouvez utiliser `printf("%c->%c",aiguille1,aiguille2)`. Indice : la fonction tient sur moins de 10 lignes de code.

4. En supposant que les moines déplacent un disque par seconde, estimez la durée du monde. Devez-vous vous inquiéter ? (Conseil : essayer d'abord sur de petites valeurs de n puis en déduire une régle générale et prouvez-la).

Exercice 4. *Le meilleur des cas*

On utilise l'algorithme de tri bulle suivant :

```
void triBulle(int taille,int* T) {
   int echangeFait=1;
   int i;
   while(echangeFait) {
      echangeFait=0;
      for(i=0;i<taille-1;i++)
         if(T[i+1] < T[i]) { // ici comparaison
            echange(T+i,T+i+1);
            echangeFait=1;
         }
   }
}
```

On voudrait étudier la complexité en temps en étudiant le nombre de comparaisons effectuées lors d'une exécution de l'algorithme.

1. D'un point de vue pratique qu'est ce qui est le plus interressant : avoir un nombre de comparaisons elevées ou au contraire bas ?
2. Donner un exemple de tableau de taille 5 où le nombre de comparaisons est minimum. Combien de comparaisons utilise-t-on ?
3. Donner un exemple de tableau de taille 5 où le nombre de compraisons est maximum. Combien de comparaison utilise-t-on ?
4. Décrivez un tableau de taille n ayant un nombre de comparaisons le plus elevé (le pire des cas).
5. Quel est la place mémoire occupée en fonction de n la taille du tableau d'entrée ?

Exercice 5. *Le compte est bon !*

Soit E une liste de nombres entiers et s un nombre. Peut-on trouver une sous-liste de E dont la somme fait s ? Par exemple si $E = \{1, 5, 7, 9, 14\}$ et $s = 13$ la réponse est oui car $13 = 5 + 7 + 1$.

1. On cherche à écrire une fonction en langage C qui résout ce problème. Elle doit renvoyer 1 si Oui et 0 sinon. `int compte(int E[],int tailleDeE,int s)`
2. Quelle est la complexité dans le pire des cas en temps et en espace de l'algorithme utilisé (la réponse ici dépend de la réponse précédente) ?