Séance de renforcement

CPUMons

25 octobre 2022

Avant de commencer... Ce document comporte plusieurs exercices dont la difficulté est renseignée par un nombre et est *globalement* croissante. Il est évidemment fortement recommandé de réfléchir au préalable sur papier avant de se lancer dans la partie implémentation.
Répartition de la difficulté :

- Niveau 1 : problème facile, ne nécessite pas de code complexe pour être mis en œuvre ;
- Niveau 2 : problème moyen, demande une réflexion plus poussée sur la présentation du code ;
- Niveau 3 : problème délicat, demande une réflexion poussée sur le problème en soi.
- Niveau 4 : problème difficile.

1 Exercice 1 (Niveau 2)

Le professeur Molét [1] a reçu un télégraphe urgent venant de Laponie pour un problème sur le logiciel de détection d'enfant sage et doit donc partir de toute urgence.Il vous confie la lourde tâche, à vous son assistant, de préparer la salle pour son prochain examen. Il vous informe que pour faciliter la correction, les étudiants [2] sont triés suivant une liste et vous donne aussi le local où aura lieu le test.
Faisant de la programmation, vous visualiser directement la classe comme une matrice de m rangée contenant n sièges et décidez donc d'écrire un programme qui transforme la liste en une matrice représentant la classe.

Votre tâche, implémentez une fonction **changeInMatrix** qui prend en paramètre une liste *tab* (par exemple [1,2,3,4,5,6]), le nombre de lignes n et le nombre de colonnes m et qui retourne la matrice associée à ces valeurs. (Ici : Si n = 2 et m = 3 alors on a [[1,2,3], [4,5,6]])

Ps : N'oubliez d'implémenter correctement la visualisation de la matrice afin de vous facilitez la vie lors de la distribution de copies.

Pss : On considère que tout les étudiants pourront entrer dans le local et que si il reste des places libres alors le programme retournera un zéro à l'endroit vide.

1. aucune ressemblance avec un de vos enseignants
2. représenté par un matricule

2 Exercice 2 (Niveau 2.5)

Pour votre projet de fin de BAC1, vous avez besoin de dessiner plusieurs images en 3D, ne voyant pas très bien comment faire vous allez voir un infographiste qui vous explique que pour dessiner une figure en 3D il faut dessiner chaque face dans un ordre précis. En effet, il sera beaucoup plus compliqué de dessiner la face arrière d'un objet si la face avant bloque le chemin. Pour être sûr d'avoir compris, vous allez vous entraîner à dessiner des boîtes se superposant.

Tâche : Vous devez réaliser un code représentant l'image attendue en 2D. L'image a nbLignes lignes et nbColonnes colonnes. Les lignes sont numérotées de 0 à nbLignes - 1 et les colonnes de 0 à nbColonnes - 1. La couleur de chaque rectangle est définie par un caractère. Par défaut, chaque pixel est de la couleur '.' .

Entrée :
La première ligne de l'entrée contient deux entiers : nbLignes et nbColonnes séparés par un espace.
La seconde ligne contient un unique entier : nbRectangles qui est le nombre de rectangles qui vont être ajoutés à l'image.
Les nbRectangles lignes suivantes contiennent chacune quatre entiers iLig1, iCol1, iLig2 et iCol2 décrivant les coins respectivement en haut à gauche et en bas à droite du rectangle considéré, ainsi qu'un caractère couleur indiquant la couleur du rectangle.
Les rectangles doivent être dessinés dans l'ordre dans lequel ils sont donnés en entrée.

Sortie :
Votre programme doit afficher nbLignes lignes de nbColonnes caractères chacune décrivant l'image obtenue.

Exemple :
9 19
4
1 3 7 5 o
5 2 6 16 -
1 12 7 14 u
2 1 2 16 s
Donnera :

```
...................
...ooo......uuu....
.sssssssssssssss..
...ooo......uuu....
...ooo......uuu....
..----------uuu--..
..----------uuu--..
...ooo......uuu....
...................
```

3 Exercice 3 (Niveau 3)

Suite à une guerre entre deux pays, vous êtes mobilisé pour aider le pays à vaincre l'ennemi ! Votre rôle : Intercepter les communications adverses et décoder leurs messages.
La structure de leur message est présenté comme ceci : composé uniquement d'une suite de chiffre compris entre 0 et 25. Chaque chiffre représente une lettre de l'alphabet. Comme l'ennemi parle dans une langue où la fréquence de e est très importante, vous pouvez utiliser cette information pour décoder leurs messages. En plus de ça, les chiffres sont décalé de manière à ce qu'on ne trouve pas la lettre tout de suite.

Voici un exemple de message : 18 16 13 3 17 18 10 3 16 2 3 16 7 20 25 18 7 20 3 17 21 6 25 18 3 10 17 3.

Exercice : Créer une fonction qui demande en entrée une chaine de caractère composé de chiffre. La fonction doit retourner le message décodé. Pour rendre les choses plus faciles, chaque nombre est distancé d'un espace. Hypothèse en plus, le message contient un nombre de e tels qu'il l'est le plus fréquent.

Petit conseil : Faites plusieurs fonctions pour ne pas vous perdre dans le code.

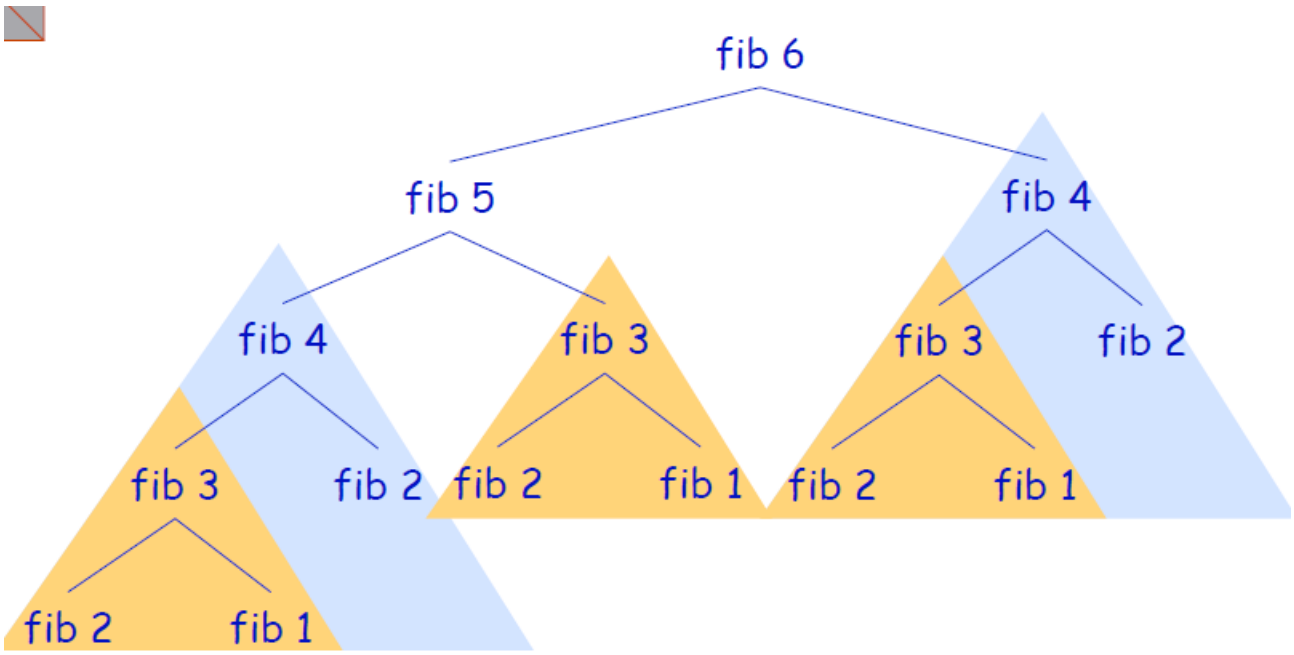
4 Exercice 4 (Niveau 1.5)

Comme vous avez pu voir en cours, les algorithmes récursifs sont fort pratique à implémenter mais en contre partie la rapidité n'est pas forcement au rendez-vous.

1) Pour commencer, codez un algorithme **RÉCURSIF** qui prends en paramètre un naturel n et qui vous renvoies le n-ieme nombre de la suite de fibonacci.
La suite de fibonacci en commençant par 0 et 1, chaque élément suivant est la somme des deux derniers éléments de la suite.
Exemple : fibo(35) donnera 9227465 (prends environ 2.5 secondes)

(Niveau 3)

2) Vous remarquez sûrement que pour des valeurs "assez grandes", l'algorithme récursif est assez lent à calculer.
Observons pour fibo(6) :



Vous voyez qu'il faut calculer plusieurs fois de façon séparée les même fibo(), vous pensez qu'il serait préférable de stocker cette valeur pour ne pas la recalculer plusieurs fois.
Vous tentez donc d'implémenter un algorithme **ITÉRATIF** de fibonacci prenant les même paramètre que précédemment qui utilise ce que vous venez de remarquer.

5 Concours

Pour ceux qui veulent s'entraîner pour participer aux concours, voici quelques liens pour vous permettre de travailler sur ce qui vous intéresse à votre propre rythme :

1. FranceIOI : nécessite de débloquer les premiers niveaux, mais recouvre une très grande variété d'algorithmes et de problèmes différents rangés par thématique ;
2. Isograd : sur ce site, vous retrouverez de nombreux anciens concours dont, en particulier, les BattleDev précédentes qui constituent un *excellent* point de départ dans le monde des concours ;
3. Google Code Jam : les énoncés des années précédentes y sont disponibles. Le niveau requis est, bien évidemment, progressif et les problèmes sont en général assez intéressants ;
4. Kattis : site reprenant de *très* nombreux problèmes. Il vous est conseillé de tester un ou deux problèmes "triviaux" afin de bien vérifier si vous n'avez pas de problèmes avec les entrées et sorties ; puis, de passer aux faciles et, rapidement, aux moyens (les difficiles portant très bien leur nom).