DIU Enseigner l'informatique au lycée
mercredi 8 juillet (matin)

Recherche textuelle

Exercice 1 Écrire une fonction `premiere_occurrence(m, t)` qui renvoie la position de la première occurrence de `m` dans `t`, s'il y en a une, et `None` sinon. On pourra se resservir de la fonction `occurrence` du cours. □

Exercice 2 Construire (à la main) la table de décalages de l'algorithme de Boyer-Moore pour le motif `"banane"`. □

Exercice 3 Construire (à la main) la table de décalages de l'algorithme de Boyer-Moore pour le motif `"chercher"`. □

Exercice 4 En utilisant le résultat de l'exercice précédent, dérouler manuellement l'exécution de l'algorithme de Boyer-Moore pendant le calcul de

```
recherche("chercher","chercher, rechercher et chercher encore")
```

en donnant notamment les valeurs successives de la variable `i`. Indiquer le nombre total de comparaisons de caractères. □

Programmation dynamique

Exercice 5 Expliciter (à la main) tous les appels récursifs à la fonction `rendu_monnaie` (version naïve) quand on lance `rendu_monnaie([1, 2], 3)`. Identifier les calculs redondants. □

Exercice 6 Quel est le tableau construit par la fonction `rendu_monnaie` quand on calcule `rendu_monnaie([1, 6, 10], 12)` (version programmation dynamique) ? Le calculer à la main. □

Exercice 7 Écrire une fonction `chemins(n, m)` qui calcule le nombre de chemins, sur une grille $\texttt{n} \times \texttt{m}$, qui mènent du coin supérieur gauche au coin inférieur droit, en se déplaçant uniquement le long des traits horizontaux vers la droite et le long des traits verticaux vers le bas. □

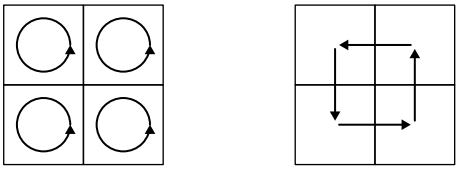
Diviser pour régner

Exercice 8 Dans cet exercice, on cherche à écrire une fonction qui effectue la rotation d'une image de 90 degrés en utilisant le principe « diviser pour régner ». Pour manipuler une image en Python, on peut utiliser par exemple la bibliothèque PIL (Python Image Library) et plus précisément son module `Image`. Avec les quatre lignes suivantes

```
from PIL import Image
im = Image.open("image.png")
largeur, hauteur = im.size
px = im.load()
```

on charge l'image contenue dans le fichier `image.png`, on obtient ses dimensions dans les variables `largeur` et `hauteur`, et la variable `px` est la matrice des pixels constituant l'image. Pour $0 \leq x <$ `largeur` et $0 \leq y <$ `hauteur`, la couleur du pixel $(\texttt{x}, \texttt{y})$ est donnée par `px[x,y]`. Une couleur est un triplet donnant les composantes rouge, vert et bleu, sous la forme d'entiers entre 0 et 255. On peut modifier la couleur d'un pixel avec une affectation de la forme `px[x,y] = c` où `c` est une couleur.

Dans cet exercice, on suppose que l'image est carrée et que sa dimension est une puissance de deux, par exemple 256×256. Notre idée consiste à découper l'image en quatre, à effectuer la rotation de 90 degrés de chacun des quatre morceaux, puis à les déplacer vers leur position finale. On peut illustrer les deux étapes ainsi :



Afin de pouvoir procéder récursivement, on va définir une fonction

```
def rotation_aux(px, x, y, t):
```

qui effectue la rotation de la portion carrée de l'image comprise entre les pixels $(\texttt{x}, \texttt{y})$ et $(\texttt{x+t}, \texttt{y+t})$. Cette fonction ne renvoie rien. Elle modifie le tableau `px` pour effectuer la rotation de cette portion de l'image, au même endroit. On suppose que `t` est une puissance de 2. Écrire le code de la fonction `rotation_aux`.

En déduire une fonction `rotation(px, taille)` qui effectue une rotation de l'image toute entière, sa dimension étant donnée par le paramètre `taille`. Une fois la rotation effectuée, on pourra sauvegarder le résultat dans un autre fichier avec la commande `im.save("rotation.png")`. □

Exercice 9 Dans cet exercice, on se propose d'appliquer le principe « diviser pour régner » pour multiplier deux entiers, avec la méthode de Karatsuba. Le principe est le suivant. Supposons deux entiers x et y ayant chacun $2n$ chiffres en base 2. On peut les écrire sous la forme

$$\begin{aligned} x &= a2^n + b \\ y &= c2^n + d \end{aligned}$$

avec $0 \leq a, b, c, d < 2^n$, c'est-à-dire avec quatre entiers a, b, c, d qui s'écrivent chacun sur n chiffres en base 2. Dès lors, on peut calculer le produit de x

et y de la façon suivante :

$$\begin{aligned} xy &= (a2^n + b)(c2^n + d) \\ &= ac2^{2n} + (ad + bc)2^n + bd \\ &= ac2^{2n} + (ac + bd - (a - b)(c - d))2^n + bd \end{aligned}$$

Cette dernière forme, d'apparence inutilement compliquée, fait apparaître seulement *trois* produits, à savoir ac, bd et $(a-b)(c-d)$. Ainsi, on a ramené la multiplication de deux entiers de $2n$ chiffres à trois multiplications d'entiers de n chiffres. Pour faire chacune de ces trois multiplications, on peut appliquer le même principe, et ainsi de suite jusqu'à obtenir de petits entiers dont la multiplication est immédiate. Au final, cela permet d'effectuer la multiplication en un temps proportionnel à $n^{1,58}$ (environ) au lieu de n^2, ce qui est un gain significatif lorsque le nombre de chiffres n est grand.

1. Écrire une fonction `taille(x)` qui renvoie le nombre de chiffres de l'entier `x` lorsqu'il est écrit en base 2.
2. Écrire une fonction `karatsuba(x, y, n)` qui calcule le produit de `x` et `y` par la méthode de Karatsuba, en supposant que `x` et `y` s'écrivent sur `n` chiffres en base 2. Indication : On peut calculer $2^{\texttt{n}}$ en Python avec l'expression `1 << n`. On peut décomposer `x` sous la forme $\texttt{a}2^{\texttt{n}} + \texttt{b}$ avec `a, b = x >> n, x % (1 << n)`.
3. En déduire une fonction `mult(x, y)` qui calcule le produit de `x` et `y`.

Remarque : Il n'est pas nécessaire d'utiliser la base 2. On pourrait tout aussi bien utiliser la base 10 par exemple. Mais les opérations liées à la base deux (multiplication, division ou reste par une puissance de deux sont faciles à réaliser pour la machine). □