# Esiee-Paris - cours d'algorithmique - TP 3 - lundi 30 mars ou mercredi 1e avril 2015

**R. Natowicz, I. Alamé, A. Çela, D. Courivaud**

**1. Calcul de la p-ième valeur d'un tableau.** Il s'agit du premier exercice du 3e TD. Donner les résultats des appels `pieme(T,0,10,p)` pour $T = [2, 1, 3, 1, 5, 6, 2, 4, 2, 4]$ et toute valeur $p$, $p \in [1 : 11]$.

---

La valeur $v_p$ est la $p$-ième valeur du tableau $T$ si celui-ci contient $p-1$ valeurs inférieures ou égales à $v_p$. Pour calculer la $p$-ième valeur on peut trier le tableau et retourner $T[p-1]$. Cet algorithme est en $\Theta(n \log n)$. Nous proposons de développer un algorithme stochastique dont la complexité est presque toujours en $\Theta(n)$. On note $pieme(T, i, j, p)$ la $p$-ième valeur du sous-tableau $T[i : j]$. L'algorithme repose sur une définition récurrente :

1. si $p = 1$ et $j - i = 1$ : $pieme(T, i, j, p) = T[i]$ ; (cas de base, soluble sans récurrence)
2. sinon (cas généraux de la récurrence) supposons que le sous-tableau $T[i : j]$ vérifie $T[i : k] \leq T[k] < T[k+1 : j]$.
   (a) Que pouvez-vous dire de la $p$-ième valeur lorsqu'elle est dans la partie gauche ( $p \leq k - i$ ) ?
   (b) Que pouvez-vous dire de la $p$-ième valeur lorsqu'elle est dans la partie droite ( $k - i < p$ ) ?
3. écrire la fonction `int pieme(int[]T, int i, int j, int p)`.

---

**2. Appariement d'un motif et d'un texte.** Il s'agit du second exercice du 3e TD. Donner des exemples d'exécution. Vous aurez besoin de quelques méthodes de la classe `String` : `charAt(i)` qui retourne le caractère d'indice $i-1$ ; `length()` qui retourne la longueur de la chaîne ; `isEmpty()` qui retourne la vérité de la proposition "la chaîne est vide".
Remarque : votre programme doit être construit pas à pas sur les différents cas (cas de base et cas de récurrence.)

---

Un motif est une suite de caractères pouvant contenir les caractères spéciaux % et $\star$ en nombres quelconques. Soit $M[0 : m]$ un motif et soit $T[0 : n]$ un texte. Les caractères spéciaux qui apparaissent éventuellement dans le texte n'y ont aucun statut particulier.
Dans un appariement du motif au texte (*matching*), chaque caractère % du motif est apparié avec un (et un seul) caractère quelconque du texte ; chaque caractère $\star$ est apparié avec une suite de caractères du texte, suite de longueur quelconque, éventuellement nulle ; et chaque caractère non spécial est apparié avec le même caractère dans le texte.
Exemples: le motif « bon$\star$, comm$\star$t a%%ez-$\star$ ? » s'apparie au texte « bonjour, comment allez-vous ? » ;
le motif « bon$\star$, bienv$\star$ d%%s l'uni$\star$ d'algo$\star$miq$\star\star$ » s'apparie au texte « bonjour, bienvenue dans l'unité d'algorithmique ».

*Cas de base (soluble sans récurrence) :*

1. le motif vide ne s'apparie qu'avec le texte vide ;

2. le texte vide n'est apparié qu'avec les motifs ne contenant que des caractères $\star$ (éventuellement aucun) ;

3. si le motif et le texte sont non vides, si le motif ne commence pas par un caractère spécial, et si les premiers caractères respectifs du motif et du texte sont différents, le motif ne s'apparie pas au texte.

*Cas généraux :* dans tous les cas ci-dessous, le motif $M[p : m]$ et le texte $T[q : n]$ sont non vides. À quelle condition le motif s'apparie-t-il au texte ?

1. Si le motif ne commence pas par un caractère spécial et si $M[p] = T[q]$ ;

2. si le motif commence par le caractère spécial % ;

3. si le motif commence par le caractère spécial $\star$.

Écrire la fonction `boolean match(String M,String T,int p,int q,int m,int n)` dont l'appel

```
match(M,T,0,0,m,n)
```

retourne la vérité de la proposition « le motif $M$ s'apparie au texte $T$ » (`m=M.length()` et `n=T.length()` sont les longueurs du motif et du texte.)

---

**3. Fibonacci en temps exponentiel.** Écrire le programme naïf de calcul de la fonction de Fibonacci (programme construit pas à pas sur la définition de cette fonction, $fibo(0) = 0$, $fibo(1) = 1$, $fibo(n \geq 2) = fibo(n-2) + fibo(n-1)$) ; tracer la courbe du temps d'exécution en fonction de $n$, pour $n$ allant de 0 à 50.
La fonction `long System.currentTimeMillis()` vous permettra de mesurer les temps de calcul d'une fonction en mémorisant la date avant et après l'appel (le temps de calcul est la différence entre les deux dates.)

**4. Existence d'un sous-sac de somme x.** On se donne un sac $S[0:n]$ de $n$ entiers positifs ou nuls et un entier $x$. On veut connaître la valeur de vérité de la proposition $P(x)$ : "Il existe un sous-sac de $S$ dont la somme est $x$." Exemples avec $S = [2, 0, 2, 4, 6]$ : il existe un sous-sac de somme $x = 4$ ; il n'existe pas de sous-sac de somme $x = 15$ $(14 = 1 + (2 + 0 + 2 + 4 + 6))$.

Écrire la fonction `boolean existeSsdsx(int[] S,int k,int n,int x)` (il existe un sous-sac de $S[k : n]$ de somme $x$) dont l'appel principal `existeSsdsx(S,0,S.length,x)` retourne la valeur de vérité de la proposition $P(x)$. Cette fonction est construite sur la décomposition récursive suivante :

1. cas de base : si $x = 0$, il existe un sous-sac de $S[k : n]$ de somme $x$ (en effet : le sac vide est un sous-sac de $S[k : n]$, sa somme est nulle) ;
2. cas de base : si $x \neq 0$ et si $k = n$, il n'existe pas de sous-sac de $S[k : n]$ de somme $x$ (en effet : le sac vide est le seul sous-sac de $S[n : n]$ ; sa somme est nulle, donc différente de $x$) ;
3. cas général : $x > 0$ et $k < n$ : ... Donner ce cas général.

Donner les exemples d'exécution pour le sac $S$ ci-dessus et toute valeur $x$, $x \in [0 : 16]$.

Tracer la courbe du temps de calcul de l'appel principal `boolean existeSsdsx(S,0,n,x)` pour $S[0 : n] = [0 : n]$, $n \in [0 : 31]$, $x = \Sigma_{i \in [0:n]}\, S[i] \;=\; 1 + \frac{(n-1) \times n}{2}$.

Remarque : dans le cours Algorithmique-2 nous pourrons résoudre ce problème de façon très efficace. De plus, s'il existe un sous-ensemble de somme $x$, nous calculerons l'un d'eux.

---

Votre rapport de TP est à déposer avant mercredi 15 avril dans mon (RN) casier, secrétariat du département Informatique, bureau 5251.