IFT 2125 Hiver 2018 **Pierre McKenzie**
Introduction à l'algorithmique

Devoir 1

Remise : sous forme *papier*, au plus tard le VE 26 janvier à l'heure du *début* de la démo

1. Un algorithme A prend $\frac{1}{10}2^{\sqrt{n}}$ secondes et un algorithme B prend $100n^3$ secondes pour résoudre un exemplaire de taille n d'un problème.

   (a) Quelles tailles maximales d'exemplaires A et B peuvent-ils résoudre en 30 jours ? *What are the maximal instance sizes that A and B can solve in 30 days ?*

   (b) Quelles seraient ces tailles sur un ordinateur un million de fois plus rapide ? *How about on a computer one million times faster ?*

   (c) B est-il infiniment souvent plus rapide que A ? *Is B infinitely often faster than A ?*

2. Soient $f(n) = n^{2\log n}$, $g(n) = (n\log n)^3$, $h(n) = 2^{(\log n)^3}$ et $t(n) = f(n)+g(n)+h(n)$. Placez $O(f)$, $O(g)$, $O(h)$ et $O(t)$ en ordre d'inclusion et justifiez vos réponses, en prenant soin de justifier également les inégalités. *Order the big-O's by inclusion, and justify your answers, not forgetting about the inequalities.*

3. Avec f et g de la question précédente, complétez le tableau ci-dessous à l'aide de $\subset, =, \supset$, ou inscrivez $\neq$ lorsqu'aucune des trois relations ne s'applique. Justifiez *une seule* des entrées du tableau, celle (ou une de celles) que vous considérez la moins facile. *Fill the table, and justify any one of its entries that gave you the most difficulty (pick randomly if you found all of them trivial).*

	$O(f)$	$O(g)$	$O(f(n)\times g(n))$	$\Omega(f)$	$\Omega(g)$	$\Theta(f)$
$O(f)$	$=$					
$O(g)$	$O(g)\,?\,?\,O(f)$	$=$				
$O(f(n)\times g(n))$			$=$			
$\Omega(f)$				$=$		
$\Omega(g)$					$=$	
$\Theta(f)$						$=$

4. Rappelons que S_m est l'ensemble des permutations de $\{1,2,\ldots,m\}$. Considérez le problème PERMUTA2

   DONNÉE: $k \in \mathbb{N}$ et $p, p_1, p_2 \in S_m$.
   DÉCIDER: s'il existe $\ell \leq k$ et $i_1, i_2, \ldots, i_\ell \in \{1,2\}$ tels que $p = p_{i_1} * p_{i_2} * \cdots * p_{i_\ell}$.

   (a) Donnez une méthode `perm2(k,p,p1,p2)` en Python, commentée, résolvant PERMUTA2 :

```
>>> p = tuple([3,1,2,5,6,4])
>>> p1 = tuple([2,3,4,5,6,1])
>>> p2 = tuple([2,1,3,4,5,6])
>>> k = 5
>>> perm2(k,p,p1,p2)
False
>>> k=15
>>> perm2(k,p,p1,p2)
True
>>> p = tuple([3,1,2,4,6,4])
>>> perm2(k,p,p1,p2)
Erreur: une donnée n'est pas une permutation
```

   Indice. Une méthode "bête" vaudra tous ses points. Vous pouvez supposer que `p,p1,p2` sont des tuples d'une même longueur m, le nombre de points permutés.

(b) Le fichier `exemplaires.py` fournit 7 exemplaires de PERM2. Importez `exemplaires` dans votre méthode. Complétez ensuite au mieux, *avec* ou *sans* l'aide de votre méthode (*sur papier, à la main*) le tableau ci-dessous avec TRUE, FALSE, ERREUR ou SAIS PAS.

Exemplaire	Valeur réponse sur cet exemplaire
`exemplaire 1` (ci-dessus)	TRUE
`exemplaire 2` (ci-dessus)	FALSE
`exemplaire 3` (ci-dessus)	ERREUR
`exemplaire 4`	
`exemplaire 5`	
`exemplaire 6`	
`exemplaire 7`	

(c) Imprimez votre méthode et joignez-la à votre devoir.

(d) Déposez votre méthode sur Studium pour que Stéphanie puisse la tester.

English translation The input to the PERMUTA2 problem consists of $k \in \mathbb{N}$ and $p, p_1, p_2 \in S_m$, where S_m is the set of permutations of $\{1, 2, \ldots, m\}$. The problem is to determine whether there exist $\ell \leq k$ and $i_1, i_2, \ldots, i_\ell$ such that $p = p_{i_1} * p_{i_2} * \cdots * p_{i_\ell}$.

You are asked to produce a Python method, properly commented, that solves PERMUTA2. Any method, including brute force, is fine. You may assume that `p,p1,p2` are tuples of the same length m, the number of points permuted.

Then you are asked to fill in the above table *by hand*, as best you can, *with* or *without* the help of your Python program, for the 7 instances provided by the file `exemplaires.py` on Studium. (Import `exemplaires` into your file.)

Please hand in a printed copy of your method together with your assignment on paper, and also upload your method on Studium for Stéphanie to be able to test it.