Corrigé type

<u>Correction</u> 1. (07pts)

A.

1. Le nombre d'opérations élémentaires dans l'algorithme 1 se calcule par**………(1pt)**

$\sum_{i=0}^{n-1}\sum_{j=0}^{i-1} 1 = \sum_{i=0}^{n-1} i \ = \frac{n(n-1)}{2}$ et l'algorithme 1 est donc de complexité O(n^2).

2. **………(1pt)**

On considère tout d'abord le cas où $n = 2k+1$ est impair, pour lequel le nombre d'opérations élémentaires est $\Sigma_{i=0}^{(n-1)/2}\Sigma_{j=i}^{n-i-1}1 = \Sigma_{i=0}^{k}(2k-2i)1 = k(k+1) = (n-1)/2((n-1)/2+1)$ soit encore une complexité quadratique. Le cas pair se traite de la même manière.

B. La règle de la somme doit etre appliquée directement. **………(0.5pt)**

$$0(\max(f(n), g(n)) = \begin{cases} n^4 & \text{si } n \text{ est pair,} \\ n^3 & \text{si } n \text{ est impair.} \end{cases}$$

C.

1. **………(1pt)**

$\lim\limits_{n\to\infty}\left(\frac{f(x)}{g(x)}\right) = \lim\limits_{n\to\infty}\left(\frac{7x^2}{x^3}\right) = \lim\limits_{n\to\infty}\left(\frac{7}{x}\right) = 0$**, donc $7x^2 \in O(x^3)$ mais $x^3 \notin O(7x^2)$**

2. **………(1pt)**

$\lim\limits_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = \lim\limits_{n\to\infty}\left(\frac{\log2\ (n)}{n}\right) = 0,$

donc, $\log_2 (n) \in O(n)$ mais $n \notin O(\log (n))$

NB. La même démarche fonctionne pour un logarithme de n'importe quelle base.

3. **………(1pt)**

$\lim\limits_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = \lim\limits_{n\to\infty}\left(\frac{2^n}{3^n}\right) = \lim\limits_{n\to\infty}\left(\frac{2}{3}\right)^n = 0,$

donc, $2^n \in O(3^n)$ mais $3^n \notin O(2^n)$

D.

1.

La fonction f est une somme. On peut traduire le théorème de la somme ainsi : c'est la fonction dont le grand O domine les autres qui l'emporte. Dans le cas de f , quelle fonction domine l'autre ? Allons voir la liste du théorème 2 : c'est l'exponentielle 5^x qui domine la puissance x^2 , au sens où $x^2 \in O(5^x)$. Ainsi, $x^2 + 5^x \in O(5^x)$

La fonction cherchée est donc **$g(x) = 5^x$. ………(0.5pt)**

2.

$4x^2 + 3x + 7 \in O(x^2)$, $6 \cdot 3^x \in O(3^x)$, $5 \log(x) \in O(\log(x))$ donc $4x^2 + 3x + 7 + 6 \cdot 3^x + 5 \log(x) \in O(3^x)$, La fonction cherchée est donc **$g(x) = 3^x$** . **………(0.5pt)**

3. $f(n) = (14n + 3) \log(n) + 3n^2$ **donc $g(n) = n^2$. ………(0.5pt)**

<u>Correction</u> 2. (06pts)

1. On a deux versions:

<u>cas1.</u> on ne compte que les appels internes (pas l'appel principal);

alors, A(n)=2+A(n-1)+A(n-2) si n>=2 et A(n)=0 si n< 2.

<u>cas2.</u> on compte aussi l'appel principal, alors

B(n)=1+B(n-1)+B(n-2) si n>=2, **………(1pt)**

B(n)=1 si n<2. **………(1pt)**

2. prenons par exemple la seconde expression

il est immédiat que B(n)>B(n-1) pour tout n, Donc B(n)≥2*B(n-2), **………(1pt)**

on a donc : B(n)≥2*B(n-2)+1

Théorème 4 d'équation de récurrence (cours), $\begin{cases} B(1)=b & n<2 \\ B(n)=aB(n-1)+b & n>=2 \end{cases}$

a>=1, b>0 donc B(n) $\in\theta(a^{n-1})$
B(n) $\in\theta(2^{n-1})$ ……… (**1pt**)

3. ………(2pts)

```
int T(int n) {
  int Tab=new int [n+1];
  Tab[0]=1;
  Tab[1]=1;
  for(int i=2; i ≤n;i++)
     Tab[i]=3* Tab[i-1]- Tab[i-2];
  return Tab[n];
}
```

<u>Correction</u> 3. (07pts)

1. ………(4pts)

État *s*	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
succ(s)	{(s_1,3),(s_2,2), (s_3,4) }	{(s_2,1), (s_5,4)}	{(s_4,1)}	{(s_4,5)}	{(s_6,1)}	{(s_2,1), (s_6,1)}	{}	{}
h(*s*)	3	3	2	7	1	4	0	0
goal(*s*)	Faux	Faux	Faux	Faux	Faux	Faux	Vrai	Vrai

Itér.	Liste *open* (état, *f*, *g*), …	Liste *closed* (état, *f*, *g*), …
0	(s_0, 3, 0)	
1	(s_2, 4, 2),(s_1, 6, 3), (s_3, 11, 4)	(s_0, 3, 0)
2	(s_4, 4, 3),(s_1, 6, 3), (s_5, 7 , 3)	(s_0, 3, 0), (s_2, 4, 2)
3	(s_6, 4, 4), (s_3, 15, 8)	(s_0, 3, 0), (s_2, 4, 2), (s_4, 4, 3)
4	Solution trouvée en tête de *open* : Chemin : < s_0 , s_2 , s_4 , s_6 >	(s_0, 3, 0), (s_2, 4, 2), (s_4, 4, 3), (s_6, 4, 4)

2.
- Non. **…………..(1pt)**

Pour les états s3 et s5 , la fonction h surestime le coût pour se rendre à un état but. Le coût restant minimal sont : h*(s3)=6 et h*(s5)=1.

- Les conditions h(s5)>h*(s5) et h(s3)>h*(s3) sont suffisantes pour conclure que la fonction h n'est pas admissible. **…………..(1pt)**

3. **…………..(1pt)**

Dans ce cas, l'état s7 sera l'unique état acceptant le but. Comme aucune action ne mène à cet état, aucune solution ne pourra être trouvée (à l'exception du cas spécial où l'état initial serait s7). A* devra explorer tous les états accessibles à partir de l'état initial avant de conclure d'aucune solution n'existe. Ainsi, h devient admissible, car elle ne surestime jamais le coût restant.