Université Mohamed Khider Biskra
Faculté des sciences exactes et des sciences de la nature et de la vie
Département d'informatique

Unité : CO(M1)
Date : 04/06/2021
Durée : 1h

Examen

Exercice 1. (07pts)

A. Calculer les complexités des codes suivants

1.
```
for ( int i = 0 ; i < n ; i ++ ) {
    for( int j = 0 ; j < i ;j ++ ) {
        x = x + 1 ; }  }
```

2.
```
for ( int i = 0 ; i < n ; i ++ ) {
    for( int j = i ; j < n-i ;j ++ ) {
        x = x + 1 ; }  }
```

B. Soient deux modules de complexité respectives O(f(n)) et O(g(n)) où

$$f(n) = \begin{cases} n^4 & \text{si } n \text{ est pair} \\ n^2 & \text{si } n \text{ est impair.} \end{cases} \qquad g(n) = \begin{cases} n^2 & \text{si } n \text{ est pair,} \\ n^3 & \text{si } n \text{ est impair.} \end{cases}$$

Déterminer la complexité de ces deux modules s'ils sont exécutés séquentiellement.

C. Est-ce que f(x)∈ O(g(x))? Est-ce que g(x)∈ O(f(x))?

1. Soit $f(x)= 7x^2$ et $g(x)=x^3$.
2. soit $f(n) = \log_2 (n)$ et $g(n)=n$.
3. Soit $f(n) = 2^n$ et $g(n) = 3^n$.

D. Trouvez une fonction g(x), la plus simple possible, telle que f (x) ∈ O(g(x)).

1. $f(x) = x^2 + 5^x$
2. $f(x) = 4x^2 + 3x+ 7+ 6* 3^x + 5 \log(x)$
3. $f(n) = (14n+ 3) \log(n)+ 3n^2$

Exercice 2. (06pts)

Soit la suite T(n) définie par T(1)=T(0)=1 et T(n)=3*T(n-1)-T(n-2) pour n>1. Pour calculer T(n) pour l'entrée n, on propose l'algorithme naif récursif :

```
function T(n: in NATURAL) return NATURAL
begin
  if n<2 then return 1;
  else return 3*T(n-1)-T(n-2);
 endif
end T;
```

Soit A(n) le nombre d'appels récursifs à T effectués lors de l'exécution de la function T(n).

1. Exprimez A(n) en fonction de A(n-1) et A(n-2).
2. Qu'en déduire sur la complexité de l'algorithme naif?
3. Proposez un algorithme en O(n) en supposant qu'on est dans le cadre du cout uniforme, en ne prenant pas en compte la taille des opérandes.

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

Le tableau suivant défini les fonctions successeurs succ, heuristique h et but goal. La fonction successeur succ(s) retourne un ensemble de paires {(s1 , c1), …, (sn , cn)} tel que s est un état donné, si est un état successeur, et ci est le coût pour passer de l'état s à si . La fonction h(s) retourne est estimation de la distance entre un état s et un état satisfaisant au but.

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

1. Donnez une trace d'exécution de l'algorithme A* en utilisant les fonctions définies précédemment et en considérant l'état initial s. Pour chaque état dans les listes open et closed, donnez ses valeurs f et g.

Itér.	Liste *open* (état, *f*, *g*), …	Liste *closed* (état, *f*, *g*), …
0	(s_0, 3, 0)	

2. La fonction heuristique h est-elle admissible? Justifiez.

3. Si la fonction but(goal) était modifiée tel que B(s6)=Faux, que se passera-t-il? Donnez le maximum d'observations que vous pouvez?

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	**FAUX**	Vrai

Bon courage