Master ICONE 1ère année

TP-Complexité

Réaliser par :
Mohammed BENAOU
Mohammed RASFA

1 Algorithmique et complexité

Exercice 1

La suite de Fibonacci est une suite de nombres entiers tels que :

$$\begin{cases} F_1 = 1 \\ F_2 = 1 \\ F_n = F_n - 1 + F_n - 2 \quad pour \quad n > 2 \end{cases}$$

Pour l'algorithme récursif il faut compter le nombre d'appel pour n en entrer que l'on va noter F_n,alors la complexité de cet de cet algorithme est exponentielle $O(2^n)$.Donc pour compter le nombre d'appel récursif est très compliquer est dur $F_6 = F_5 + F_4 = F_4 + F_3 + F_3 + F_2 + ...$

Algorithm 1 Nb-Fibo

Input: un entier n
Output: un entier
begin
 a = 1 ←complexité constante O(1)
 b = 1 ←complexité constante O(1)
 for *i allant de 3 à n* **do**
  $b = a + b$
  $a = b - a$
  $complexitéLinéaireO(n)$
end
Retour $[b]$
 end

Alors l'algorithme itératif dont la complexité est de type linéaire $O(n)$ 2-on a implémente les deux algorithmes en java :

Le code source

```java
package fibonacci;
import java.util.concurrent.TimeUnit;
/**
 *
 * @author Mohammed
 */
public class Fibonacci {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        System.out.println("Test de fibonacci avec boucle for");
        long val = NB_Fibo(12);
        System.out.println("la valeur "+val);
        System.out.println("___________________________");
        System.out.println("Test de fibonacci recursif");
        int val2=Nb_Fibo_R(12);
        System.out.println("la valeur "+val2);
    }

    public static int NB_Fibo(int n) {
        int  a = 1;
        int b = 1;
        for (int k = 3; k <= n; k++) {
            long deb=System.nanoTime();
            b=a+b;
            a=b-a;
            System.out.println("le temps d execution "+k +" "+(System.nanoTime()-deb)+" ns");
        }
        return b;
    }

    public static int Nb_Fibo_R(int n){
        int x;
        if( n<= 2){
            x=1;
        }
        else {
            long deb=System.nanoTime();
            x=Nb_Fibo_R(n-1)+Nb_Fibo_R(n-2);
            System.out.println("le temps d execution : "+(System.nanoTime()-deb)+" ns");
        }
        return x;
    }
}
```

Le temps d'execution devient différent pour n = 12

2 Calculabilité et décidabilité

Exercie 2

Le problème de $n^{éme}$ nombre de Fibonacci

- Un problème calculable : Oui car l'algorithme existe pour résoudre problème.
- Un problème décidable : Non par ce que le retour n'est pas de type boolean .
- De la classe P : Oui car il existe un algorithme polynomial pour la suie de Fibonacci
- De la classe NP : Non par ce que il existe un linéaire et exponentielle
- Un problème NP-complet : Non par ce que il n'existe pas de NP pour Fibonacci
- Un problème ouvert : Non par ce on sait la solution du problème

Exercice 3

1. Soit B="Voyageur de commerce"
   le problème B se réduit en problème A s'il existe un algorithme transformant le problème B en A,ce dernier est un problème NP et le problème B est de type NP-Complet qui se réduit en A,alors A est NP-complet
2. Si on trouve un nouvel algorithme de résolution du problème A en temps polynomial alors A est dans P.
3. A est NP-complet (question 1) + A est dans P (question 2).
4. A n'est pas dans P + A dans NP donc P != NP.

Exercice 4

1. - SAT et sa variante 3-SAT
   - Le problème du cycle hamiltonien et sa variante pondérée, le problème du voyageur de commerce
   - La clique maximum (équivalent au stable maximum et à la couverture de sommets minimum)
   - Le problème ensemble dominant minimal
   - Les problèmes de coloration de graphe ;
   - Le problème du sac à dos.
2. L'exemple traité par Wikipedea pour la coloration d'un graphe est celui de l'allocation des fréquances sur les réseaux de télecom dans lequel chaque émeteur a une fréquence particulière.
   - Algorithme de Welsh et Powell
   - DSATUR

3 Machine de Türing

Exercice 5

1. La biographie d'Alan Türing

2. A) Les entrées de l'exécution d'une machine de Türing : un mot à traiter
   Sorie de l'exécution d'une machine de Türing : Oui le mot est reconnu || Non le mot n'est pas reconnu.

   B) les programmes implémentés : Ajouter 1 ; soustraire 1 ; Multiplier 2 ; Inverser 0,1 ; Doubler liste1 ; Detecter Palindromes.

Exercice 6

3.1

−le Mot 111#1111

Etat	caractère courant	Transition	Nouvel état	Resultat
0	1	1,E,R	1	E11#1111
1	1	1,1,R	1	E11#1111
1	1	1,1,R	1	E11#1111
1	#	#,#,R	1	E11#1111
1	1	1,1,R	1	E11#1111
1	1	1,1,R	1	E11#1111
1	1	1,1,R	1	E11#1111
1	1	1,1,R	1	E11#1111
1	E	E,E,L	2	E11#1111
2	1	1,E,L	2	E11#111E
3	1	1,1,L	3	E11#111E
3	1	1,1,L	3	E11#111E
3	1	1,1,L	3	E11#111E
3	1	1,1,L	3	E11#111E
3	#	#,#,L	3	E11#111E
3	1	1,1,L	3	E11#111E
3	1	1,1,L	3	E11#111E
3	1	1,1,L	3	E11#111E
3	E	E,E,R	0	E11#111E
0	1	1,E,R	1	EE1#111E
1	1	1,1,R	1	EE1#111E
1	#	#,#,R	1	EE1#111E
1	1	1,1,R	1	EE1#111E
1	1	1,1,R	1	EE1#111E
1	1	1,1,R	1	EE1#111E
1	E	E,E,L	2	EE1#111E
2	1	1,E,L	2	E11#11EE
3	1	1,1,L	3	E11#11EE
3	1	1,1,L	3	E11#11EE
3	#	#,#,L	3	EE1#11EE
3	1	1,1,L	3	EE1#11EE
0	1	1,E,R	1	EEE#11EE
1	#	#,#,R	1	EEE#11EE
1	1	1,1,R	1	EEE#11EE
1	E	E,E,L	2	EEE#11EE
2	1	1,E,L	2	EEE#1EEE

Table 1 – Application de la machine de Türing sur le mot 111#1111

le Mot 111#111

Etat	caractère courant	Transition	Nouvel état	Resultat
0	1	1,E,R	1	E11#111
1	1	1,1,R	1	E11#111
1	1	1,1,R	1	E11#111
1	#	#,#,R	1	E11#111
1	1	1,1,R	1	E11#111
1	1	1,1,R	1	E11#111
1	1	1,1,R	1	E11#111
1	E	E,E,L	2	E11#111
2	1	1,E,L	2	E11#11E
3	1	1,1,L	3	E11#11E
3	1	1,1,L	3	E11#11E
3	1	1,1,L	3	E11#11E
3	#	#,#,L	3	E11#11E
3	1	1,1,L	3	E11#11E
3	1	1,1,L	3	E11#11E
3	E	E,E,R	0	E11#11E
0	1	1,E,R	1	EE1#11E
1	1	1,1,R	1	EE1#111E
1	#	#,#,R	1	EE1#11E
1	1	1,1,R	1	EE1#11E
1	1	1,1,R	1	EE1#11E
1	E	E,E,L	2	EE1#11E
2	1	1,E,L	2	EE1#1EE
3	1	1,1,L	3	E11#1EE
3	#	#,#,L	3	EE1#11EE
3	1	1,1,L	3	EE1#1EE
0	1	1,E,R	1	EEE#1EE
1	#	#,#,R	1	EEE#1EE
1	1	1,1,R	1	EEE#1EE
1	E	E,E,L	2	EEE#EEE
2	1	1,E,L	2	EEE#EEE

TABLE 2 – Application de la machine de Türing sur le mot 111#111

3.2 les transitions de sortie

- (1) égalité => #,#,S
- (2) Non égalité => 1,1,S
- (3) égalité => E,E,S

3.3 comportement de la machine de Türing

1. la machine de Türing commence l'execution par l'état initail (0) si le caractère courant égal à 1 dans ce cas là elle le supprime et passe le curseur à droite .

2. Dans l'état 1 elle remplace le caractère courant soit par 1 soit par avec un déplacement vers la droite jusqu'à ce qu'elle trouve le vide(E) après elle supprime le caractère final et se déplace vers la gauche en passant à l'état (2).
3. pour l'état (2) si le caractère courant égal à 1 elle le remplace par le vide (E) avec un déplacement à gauche en passant à l'état 3 .
4. Dans l'etat 3 si le caractere courant est 1 ou le curseur se deplace a gauche et reste dans le meme etat.
5. Dans l'état 3 si le caractère courant est E, elle le remplace aussi par E avec un déplacement du curseur vers la droite avec une transition à l'état 0
6. En retournant vers l'état 0 si a caractère courant est 3 ou # le curseur se déplace vers la droite avec une transition vers l'état 4
7. pour l'état 4 si le caractère courant est # elle le remplace par # et elle reste dans le même état.
8. pour l'état 4 si le caractère courant est 1 || E elle le remplace par avec un déplacement vers l'état final.

3.4 Modification de la machine de Türing

voilà ci-dessus la modification pour que la machine puisse traiter n'importe quel caractère du mot d'entrée :
on rajoute les deux états E1,E2 avant l'etat initial 0

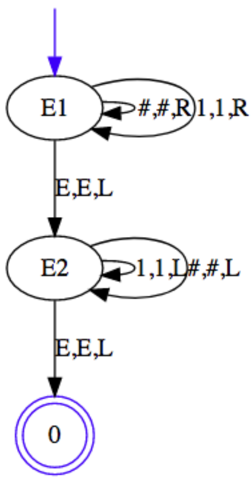

Figure 1 – Modification de la machine de Türing

1

1. http ://www.webgraphviz.com/

Exercice 7

- l'ensemble des états est Q = 0, 1, 2, 3, 4,
- l'alphabet d'entrée est = a, b,
- l'état initial est I = 0,
- le seul état final F = 4,

voilà notre machine de Türing :

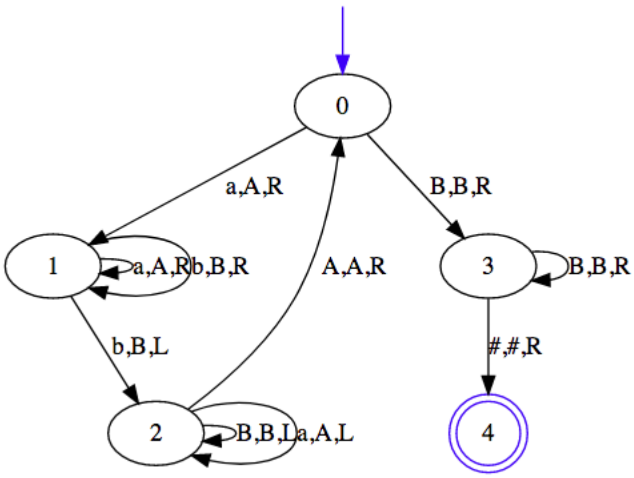

FIGURE 2 – notre la machine de Türing

[2]

2. https ://www.irif.fr/ carton/Enseignement/Complexite/MasterInfo/Cours/Exemples/turing-anbn.html