

DJILLALI LIABES UNIVERSITY OF SIDI BEL ABBES  
FACULTY OF EXACT SCIENCES  
DEPARTMENT OF COMPUTER SCIENCES



*Module : Algorithmique et Complexité*  
1ST YEAR OF MASTER'S DEGREE IN  
NETWORKS, INFORMATION SYSTEMS & SECURITY (RSSI)  
2021/2022

---

## **Le problème de la sous somme avec une approche récursive**

---

*Author:*  
HADJAZI M.Hisham  
AMUER Wassim Malik  
*Group: 01 / RSSI*

*Supervisor:*  
Dr. MME. BELKHODJA  
ZENAIIDI Lamia

*A paper submitted in fulfilment of the requirements for the  
TP-04*

December 2, 2021

# Contents

<b>1</b>	<b>Solutions of TP-04</b>	<b>1</b>
1.1	Q1/ Ecrire le code java correspondant à la fonction Somme. . . . .	2
1.2	Q2/ Améliorer le code pour que la fonction puisse afficher les sous-ensembles trouvés. . . . .	2
1.3	Q3/Pour tester le code, prévoir : . . . . .	3
1.3.1	une lecture de la taille n du tableau E et de la valeur de la somme S à partir du clavier . . . . .	3
1.3.2	une définition du tableau E au niveau du code avec un remplissage manuel . . . . .	3
1.3.3	une génération automatique et aléatoire du tableau E . . . . .	3
1.3.4	un affichage de la somme, des tableaux concernés, des résultats obtenus, du nombre d'appels récursifs exécutés. . . . .	4
1.4	Q4/Tester le code pour différentes valeurs de n =5, 10, 20, 40,.... . . . .	4
1.4.1	Using True or False Somme class . . . . .	4
1.4.2	Using Modified Algoethm to capture result with Somme2 class . . . . .	6
1.5	Q5/ Faire des captures d'écran des exécutions : . . . . .	8
	1- pour un tableau et une somme de votre choix ; . . . . .	8
	2- pour n=5, n=10 avec des tableaux générés aléatoirement et une somme lue à partir du clavier. . . . .	9
1.6	Conclusion . . . . .	10
1.6.1	First function Somme . . . . .	10
1.6.2	Second function Somme2 . . . . .	10
<b>A</b>	<b>Appendix A</b>	<b>11</b>
A.1	Java Code for Somme.java . . . . .	11
A.2	Java Code for Somme2.java . . . . .	13

## Chapter 1

# Solutions of TP-04

### Notes regarding this solution :

This solution and the executions of the code in it was done in the following machine :

- *Machine*: Lenovo Ideapad S210
- *CPU*: Intel Celeron 1037U 1800 MHz
- *RAM*: 8GB DDR3l
- *OS* : Linux Mint 20.2 Cinnamon Kernel v.5.4.0-88
- *IDE* : Eclipse IDE for Java Developers Version: 2019-12 (4.14.0)
- *Java version*: 11.0.11

### Définition du problème :

Etant donné un ensemble E de n entiers non négatifs, et une valeur S, il s'agit de déterminer s'il existe un sous ensemble de E dont la somme des éléments est égale à S.

### Exemple :

E= 3, 6, 2, 7, 9 ; S=9 ; les sous-ensembles 3,6, 2,7 et 9 réalisent la solution. Si S=4 aucun sous ensemble ne réalise la solution. Pour résoudre ce problème, on s'intéresse à une approche de type « **diviser pour régner** ». On peut alors définir une fonction récursive qui retourne vrai s'il existe un sous ensemble de E dont la somme des éléments est égale à S et retourne faux dans le cas contraire.

```
boolean Somme(int E[], int n, int S)
{ if (S == 0) return true;
  if (n == 0 && S > 0) return false;
  if (E[n-1] > S) return Somme(E, n-1, S);
  return Somme(E, n-1, S) || Somme(E, n-1, S - E[n-1]); }
```

FIGURE 1.1: sous somme Algorithm

## 1.1 Q1/ Ecrire le code java correspondant à la fonction Somme.

```
1 private static boolean Sum_Exist(int[] values, int targetSum,
2   int n) {
3     if (targetSum == 0) {
4       return true;
5     }
6     if (n == 0 && targetSum > 0) {
7       return false;
8     }
9     if (values[n - 1] > targetSum) {
10      return Sum_Exist(values, targetSum, n - 1);
11    }
12    return Sum_Exist(values, targetSum - values[n - 1], n - 1)
        || Sum_Exist(values, targetSum, n - 1);
13 }
```

## 1.2 Q2/ Améliorer le code pour que la fonction puisse afficher les sous-ensembles trouvés.

```
13 static void Sum_Exist(int values[], int n, Vector<Integer>
14   subarray, int target, int target2) {
15   if (target == 0) {
16     System.out.println("The Sum of subset " + Arrays.
17       toString(subarray.toArray()) + " = " + target2);
18     return;
19   },
20   if (n == 0)
21     return;
22   Sum_Exist(values, n - 1, subarray, target, target2);
23   Vector<Integer> subarray2 = new Vector<Integer>(subarray);
24   subarray2.add(values[n - 1]);
25   Sum_Exist(values, n - 1, subarray2, target - values[n - 1],
26     target2);
27 }
```

### 1.3 Q3/Pour tester le code, prévoir :

1. une lecture de la taille  $n$  du tableau  $E$  et de la valeur de la somme  $S$  à partir du clavier,
2. une définition du tableau  $E$  au niveau du code avec un remplissage manuel,
3. une génération automatique et aléatoire du tableau  $E$ ,
4. un affichage de la somme, des tableaux concernés, des résultats obtenus, du nombre d'appels récurrents exécutés.

#### 1.3.1 une lecture de la taille $n$ du tableau $E$ et de la valeur de la somme $S$ à partir du clavier

```
25 Scanner keyboard = new Scanner(System.in);
26 System.out.println("Enter the size of the Array n =\n");
27 int n = keyboard.nextInt();

28 System.out.println("The target you are looking for =");
29 int target = keyboard.nextInt();
30 int target2 = target;
```

#### 1.3.2 une définition du tableau $E$ au niveau du code avec un remplissage manuel

```
31 System.out.println("\nManual Filling Activated");
32 for (int i = 0; i < n; i++) {
33     System.out.println("Value " + (i + 1) + " =");
34     values[i] = keyboard.nextInt();
35 }
```

#### 1.3.3 une génération automatique et aléatoire du tableau $E$

```
36 System.out.println("\nAutomatic filling Activated");
37 Random random = new Random();
38 for (int i = 0; i < n; i++) {
39     values[i] = random.nextInt(15);
40 }
```

### 1.3.4 un affichage de la somme, des tableaux concernés, des résultats obtenus, du nombre d'appels récursifs exécutés.

```

41         if (target == 0)
42             System.out.println("The Sum of subset " + Arrays.
                toString(subarray.toArray()) + " = " + target2);

43         System.out.println("The Array is : " + Arrays.toString(
                values) + "\n");
44         System.out.println("The Target Sum is : " + n + "\n");
45         System.out.println("The founded sums (if any) are : \n");
46         Sum_Exist(values, n, v, target, target2);
47         System.out.println("\nThe number of recursive calls are : "
                + cpt + " Times\n");

48         static long cpt = 0;
49         static void Sum_Exist(int values[], int n, Vector<Integer>
                subarray, int target, int target2) {
50             cpt++;
51             .
52             .
53             .

```

## 1.4 Q4/Tester le code pour différentes valeurs de n =5, 10, 20, 40,....

### 1.4.1 Using True or False Somme class

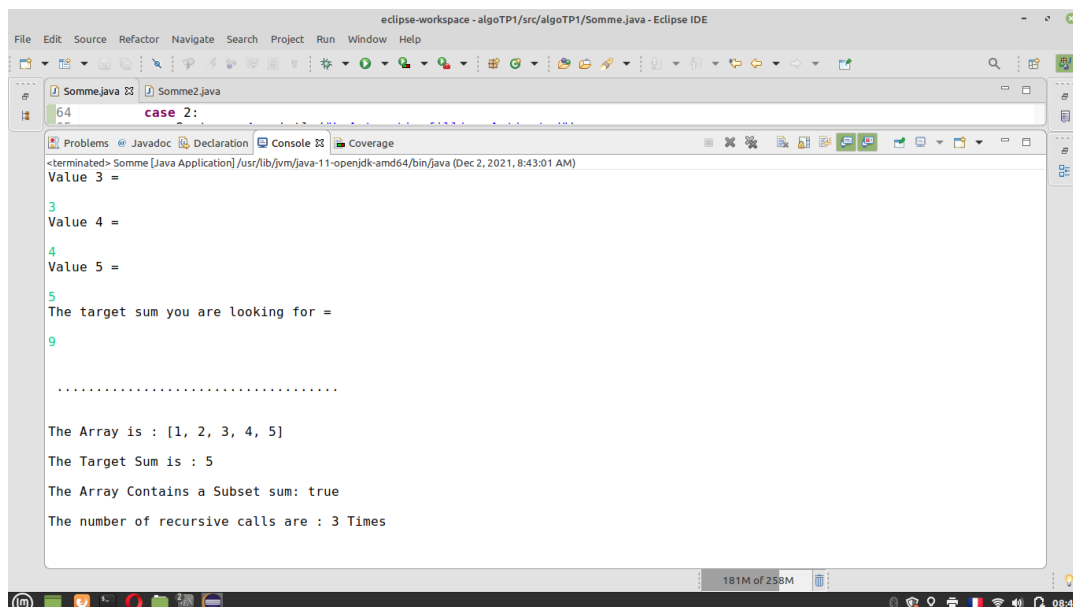
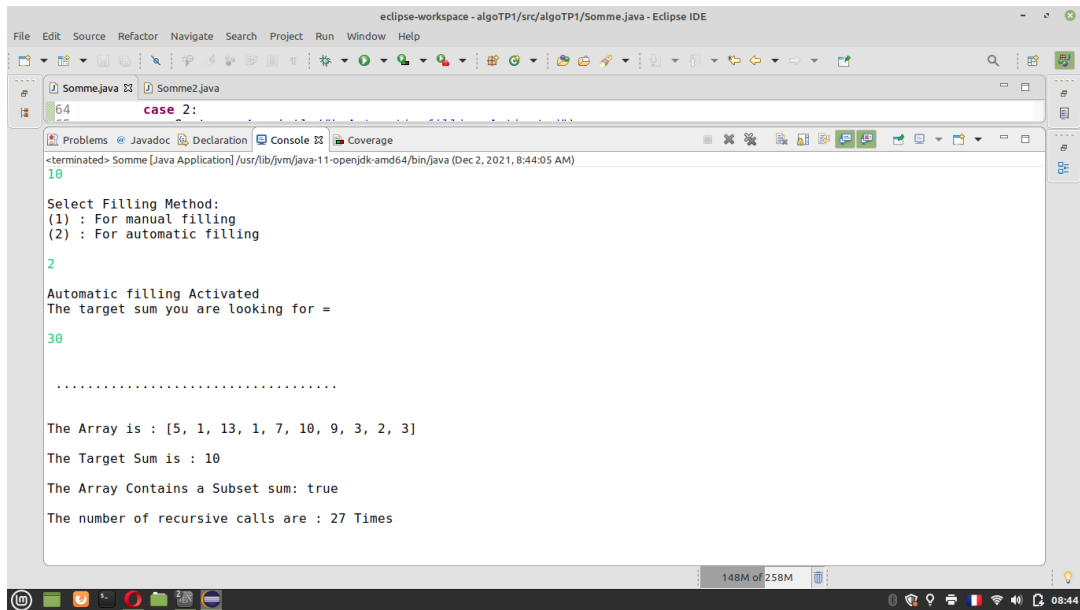
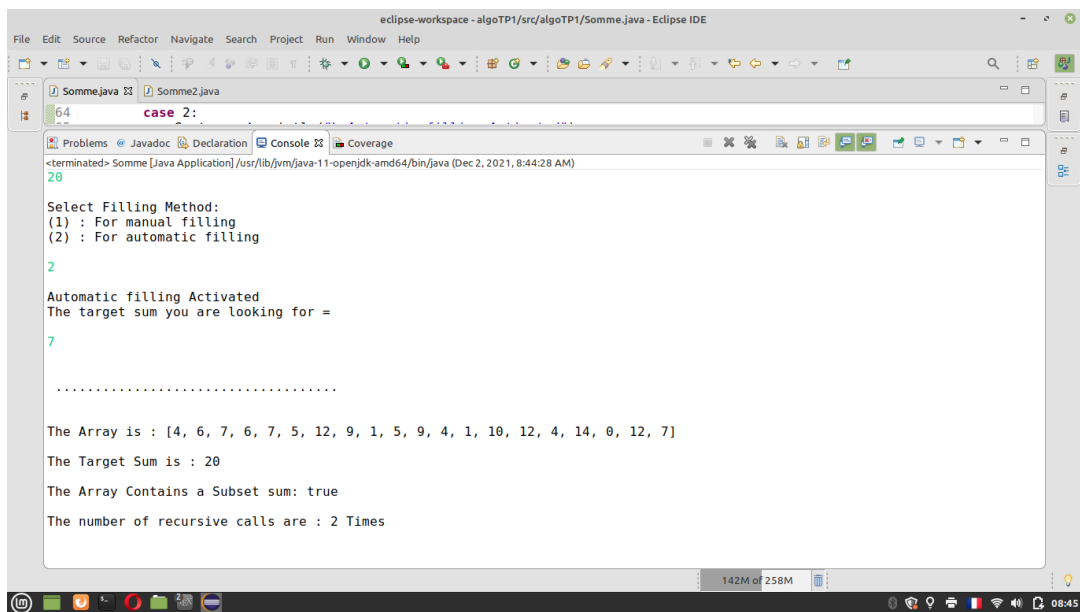


FIGURE 1.2: n=5



```
eclipse-workspace - algoTP1/src/algotP1/Somme.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Somme.java 64 case 2:
Problems Javadoc Declaration Console Coverage
<terminated> Somme [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 2, 2021, 8:44:05 AM)
10
Select Filling Method:
(1) : For manual filling
(2) : For automatic filling
2
Automatic filling Activated
The target sum you are looking for =
30
.....
The Array is : [5, 1, 13, 1, 7, 10, 9, 3, 2, 3]
The Target Sum is : 10
The Array Contains a Subset sum: true
The number of recursive calls are : 27 Times
148M of 258M
08:44
```

FIGURE 1.3:  $n=10$ 

```
eclipse-workspace - algoTP1/src/algotP1/Somme.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Somme.java 64 case 2:
Problems Javadoc Declaration Console Coverage
<terminated> Somme [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 2, 2021, 8:44:28 AM)
20
Select Filling Method:
(1) : For manual filling
(2) : For automatic filling
2
Automatic filling Activated
The target sum you are looking for =
7
.....
The Array is : [4, 6, 7, 6, 7, 5, 12, 9, 1, 5, 9, 4, 1, 10, 12, 4, 14, 0, 12, 7]
The Target Sum is : 20
The Array Contains a Subset sum: true
The number of recursive calls are : 2 Times
142M of 258M
08:45
```

FIGURE 1.4:  $n=20$

```

eclipse-workspace - algoTP1/src/algotP1/Somme.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Somme.java 64 case 2:
Problems Javadoc Declaration Console Coverage
Somme [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 2, 2021, 8:45:20 AM)
Enter the size of the Array n =
40

Select Filling Method:
(1) : For manual filling
(2) : For automatic filling
2

Automatic filling Activated
The target sum you are looking for =
10000

.....

The Array is : [11, 0, 3, 8, 4, 11, 11, 12, 12, 14, 10, 8, 1, 5, 14, 14, 6, 11, 9, 0, 13, 11, 9, 8, 4, 3, 5, 10, 0, 13, 11, 14]
The Target Sum is : 40

131M of 258M
08:45

```

FIGURE 1.5:  $n=40$ 

## 1.4.2 Using Modified Algorithm to capture result with Somme2 class

```

eclipse-workspace - algoTP1/src/algotP1/Somme2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Somme.java Somme2.java
break
Problems Javadoc Declaration Console Coverage
<terminated> Somme2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 1, 2021, 3:56:43 PM)
Select Filling Method:
(1) : For manual filling
(2) : For automatic filling
2

Automatic filling Activated
The target you are looking for =
20

.....

The Array is : [9, 2, 11, 11, 6]
The Target Sum is : 5

The founded sums (if any) are :
The Sum of subset [11, 9] = 20
The Sum of subset [11, 9] = 20

The number of recursive calls are : 63 Times

119M of 258M
15:56

```

FIGURE 1.6:  $n=5$



```

eclipse-workspace - algoTP1/src/algoTP1/Somme2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Somme.java Somme2.java
break

Problems Javadoc Declaration Console Coverage

<terminated> Somme2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 1, 2021, 3:57:05 PM)
(1) : For manual filling
(2) : For automatic filling

2

Automatic filling Activated
The target you are looking for =
15

.....

The Array is : [6, 5, 8, 12, 5, 11, 2, 13, 11, 14]

The Target Sum is : 10

The founded sums (if any) are :

The Sum of subset [2, 8, 5] = 15
The Sum of subset [2, 5, 8] = 15
The Sum of subset [13, 2] = 15

The number of recursive calls are : 1913 Times

212M of 258M
15:57

```

FIGURE 1.7: n=10

```

eclipse-workspace - algoTP1/src/algoTP1/Somme2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Somme.java Somme2.java
break

Problems Javadoc Declaration Console Coverage

<terminated> Somme2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 1, 2021, 3:57:39 PM)
The Sum of subset [2, 2, 1, 4, 1] = 10
The Sum of subset [2, 2, 1, 4, 0, 1] = 10
The Sum of subset [2, 5, 3] = 10
The Sum of subset [2, 5, 2, 1] = 10
The Sum of subset [2, 5, 2, 0, 1] = 10
The Sum of subset [2, 5, 2, 1] = 10
The Sum of subset [9, 1] = 10
The Sum of subset [9, 0, 1] = 10
The Sum of subset [9, 1] = 10
The Sum of subset [8, 1, 1] = 10
The Sum of subset [8, 1, 0, 1] = 10
The Sum of subset [8, 2] = 10
The Sum of subset [8, 2] = 10
The Sum of subset [6, 3, 1] = 10
The Sum of subset [6, 3, 0, 1] = 10
The Sum of subset [6, 4] = 10
The Sum of subset [6, 1, 3] = 10
The Sum of subset [6, 2, 1, 1] = 10
The Sum of subset [6, 2, 1, 0, 1] = 10
The Sum of subset [6, 2, 1, 1] = 10
The Sum of subset [6, 2, 1, 0, 1] = 10
The Sum of subset [6, 2, 2] = 10

The number of recursive calls are : 1974773 Times

122M of 258M
15:57

```

FIGURE 1.8: n=20

```

eclipse-workspace - algoTP1/src/algoTP1/Somme2.java - Eclipse IDE
Somme2.java
60 case 1:
61     System.out.println("\nManual Filling Activated");
62     for (int i = 0; i < n; i++) {
        ...
    }

Somme2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 1, 2021, 5:13:19 PM)
Automatic filling Activated
The target you are looking for =
3
.....

The Array is : [7, 5, 9, 14, 3, 2, 11, 6, 12, 5, 12, 1, 8, 1, 13, 12, 7, 1, 10, 4, 12, 4, 0, 1, 14, 6, 13, 7, 12, 8, 5, 7, 12, 5, 2, 11, 11, 8, 4, 1]
The Target Sum is : 40
The founded sums (if any) are :
The Sum of subset [3] = 3
The Sum of subset [1, 2] = 3
The Sum of subset [1, 2] = 3
The Sum of subset [1, 2] = 3
The Sum of subset [1, 1, 1] = 3
The Sum of subset [0, 3] = 3
The Sum of subset [0, 1, 2] = 3
The Sum of subset [0, 1, 2] = 3
The Sum of subset [0, 1, 2] = 3
The Sum of subset [0, 1, 1, 1] = 3
The Sum of subset [1, 1, 1] = 3
The Sum of subset [1, 1, 1] = 3
The Sum of subset [1, 1, 1] = 3
The Sum of subset [1, 0, 2] = 3
The Sum of subset [1, 0, 1, 1] = 3
The Sum of subset [1, 0, 1, 1] = 3
The Sum of subset [1, 0, 1, 1] = 3
The Sum of subset [2, 1] = 3
The Sum of subset [2, 1] = 3
The Sum of subset [2, 1] = 3
The Sum of subset [2, 0, 1] = 3
The Sum of subset [2, 0, 1] = 3
The Sum of subset [2, 0, 1] = 3
The Sum of subset [2, 1] = 3

```

FIGURE 1.9: n=40

## 1.5 Q5/ Faire des captures d'écran des exécutions :

1- pour un tableau et une somme de votre choix ;

Choosing sum = 10 and manual filling of array of size 6

```

eclipse-workspace - algoTP1/src/algoTP1/Somme2.java - Eclipse IDE
Somme2.java
50     System.out.println("Value " + (i + 1) + " = ");
51     values[i] = keyboard.nextInt();
    ...
}

Somme2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 1, 2021, 4:05:43 PM)
terminated> Somme2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 1, 2021, 4:05:43 PM)
Value 3 =
4
Value 4 =
5
Value 5 =
6
The target you are looking for =
10
.....

The Array is : [2, 3, 4, 5, 6]
The Target Sum is : 5
The founded sums (if any) are :
The Sum of subset [5, 3, 2] = 10
The Sum of subset [6, 4] = 10
The number of recursive calls are : 57 Times

```

FIGURE 1.10: Question 5 part 1

2- pour  $n=5$ ,  $n=10$  avec des tableaux générés aléatoirement et une somme lue à partir du clavier.

```

eclipse-workspace - algoTP1/src/algotP1/Somme2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Somme2.java
50 System.out.println("Value " + (i + 1) + " = ");
51 values[i] = keyboard.nextInt();

<terminated> Somme2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 1, 2021, 4:11:19 PM)
Select Filling Method:
(1) : For manual filling
(2) : For automatic filling
2
Automatic filling Activated
The target you are looking for =
10
.....

The Array is : [7, 10, 14, 0, 8]
The Target Sum is : 5
The founded sums (if any) are :
The Sum of subset [10] = 10
The Sum of subset [0, 10] = 10
The number of recursive calls are : 59 Times
219M of 258M
16:11

```

FIGURE 1.11: Question 5 part 2,1

```

eclipse-workspace - algoTP1/src/algotP1/Somme2.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Somme2.java
50 System.out.println("Value " + (i + 1) + " = ");
51 values[i] = keyboard.nextInt();

<terminated> Somme2 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (Dec 1, 2021, 4:12:16 PM)
Automatic filling Activated
The target you are looking for =
17
.....

The Array is : [0, 2, 1, 7, 12, 6, 13, 5, 12, 10]
The Target Sum is : 10
The founded sums (if any) are :
The Sum of subset [5, 12] = 17
The Sum of subset [12, 5] = 17
The Sum of subset [10, 7] = 17
The Sum of subset [10, 6, 1] = 17
The Sum of subset [10, 5, 2] = 17
The number of recursive calls are : 1741 Times
171M of 258M
16:12

```

FIGURE 1.12: Question 5 part 2,2

## 1.6 Conclusion

### 1.6.1 First function Somme

This is a bit tricky as its complexity vary depending on the problem, we know that if we give it a sum that doesn't exist it will have to pass through all the possibilities which makes its **worst case**  $T(n) = \mathcal{O}(2^n)$ . however for it is **impossible** to calculate **best case** and **average case** as different sums will yield different complexities.

### 1.6.2 Second function Somme2

This function unlike **Somme** has to verify all the possibilities in order to print out all possible solutions, which makes it an **average case** of  $T(n) = \theta(2^n)$  and a **best case** of  $T(n) = \Omega(2^n)$  and a **worst case** of  $T(n) = \mathcal{O}(2^n)$ . here we find that **divide and conquer** is not helping in reducing the time complexity of our recursive calls.

## Appendix A

# Appendix A

### A.1 Java Code for Somme.java

```
54 //TP4 Algorithmique et Complexite 2021-2022
55
56 //Nom:HADJAZI
57 //Prenom: Mohammed Hisham
58 //Specialite:  RSSI      Groupe: 01
59
60 //Nom:Ameur
61 //Prenom: Wassim Malik
62 //Specialite:  RSSI      Groupe: 01
63
64
65
66 import java.util.Arrays;
67 import java.util.Random;
68 import java.util.Scanner;
69
70 public class Somme {
71     // counter initialization
72     static long cpt = 0;
73
74     // Returns true if there exists a subsequence with the given
75     // sum
76     private static boolean Sum_Exist(int[] values, int targetSum,
77     int n) {
78         // increase counter by cpt = cpt + 1
79         cpt++;
80         // return true if subset is found
81         if (targetSum == 0) {
82             return true;
83         }
84         // base case: no items left, or sum becomes negative
85         if (n == 0 && targetSum > 0) {
86             return false;
87         }
88         // last element greater than the sum remove it and continue
89         // We cant include this item in the subset. We dont have a
90         // choice.
91         if (values[n - 1] > targetSum) {
92             return Sum_Exist(values, targetSum, n - 1);
93         }
94         // include last or exclude last.
```

```

92     // We have a choice to either include this item in the
93     // subset or dont.
94     // Whichever choice leads us to the target sum will be
95     // chosen.
96     return Sum_Exist(values, targetSum - values[n - 1], n - 1)
97     || Sum_Exist(values, targetSum, n - 1);
98 }
99
100 public static void main(String[] args) {
101     Scanner keyboard = new Scanner(System.in);
102
103     System.out.println("Enter the size of the Array n =");
104     int n = keyboard.nextInt();
105     int[] values = new int[n];
106
107     System.out.println("\nSelect Filling Method:\n(1) : For
108     manual filling\n(2) : For automatic filling\n");
109     int a = keyboard.nextInt();
110
111     switch (a) {
112     case 1:
113         System.out.println("\nManual Filling Activated");
114         for (int i = 0; i < n; i++) {
115             System.out.println("Value " + (i + 1) + " =\n");
116             values[i] = keyboard.nextInt();
117         }
118         break;
119     case 2:
120         System.out.println("\nAutomatic filling Activated");
121         Random random = new Random();
122         for (int i = 0; i < n; i++) {
123             values[i] = random.nextInt(15);
124         }
125         break;
126     }
127
128     System.out.println("The target sum you are looking for =\n"
129     );
130     int target = keyboard.nextInt();
131     keyboard.close();
132
133     System.out.println("\n\n
134     ..... \n\n");
135     System.out.println("The Array is : " + Arrays.toString(
136     values) + "\n");
137     System.out.println("The Target Sum is : " + n + "\n");
138     System.out.println("The Array Contains a Subset sum: " +
139     Sum_Exist(values, target, n));
140     System.out.println("\nThe number of recursive calls are : "
141     + cpt + " Times\n");
142 }

```

## A.2 Java Code for Somme2.java

```
135 //TP4 Algorithmique et Complexite 2021-2022
136
137 //Nom:HADJAZI
138 //Prenom: Mohammed Hisham
139 //Specialite:  RSSI      Groupe: 01
140
141 //Nom:Ameur
142 //Prenom: Wassim Malik
143 //Specialite:  RSSI      Groupe: 01
144
145
146
147 import java.util.Arrays;
148 import java.util.Random;
149 import java.util.Scanner;
150 import java.util.Vector;
151
152 public class Somme2 {
153     static long cpt = 0;
154
155     // The vector subarray stores current subset.
156     static void Sum_Exist(int values[], int n, Vector<Integer>
157         subarray, int target, int target2) {
158         cpt++;
159         // If remaining target is 0, then print all
160         // values of current subset.
161         if (target == 0) {
162             System.out.println("The Sum of subset " + Arrays.
163                 toString(subarray.toArray()) + " = " + target2);
164             return;
165         }
166
167         // If no remaining values,
168         if (n == 0)
169             return;
170
171         // We consider two cases for every element.
172         // a) We do not include last element.
173         // b) We include last element in current subset.
174         Sum_Exist(values, n - 1, subarray, target, target2);
175         Vector<Integer> subarray2 = new Vector<Integer>(subarray);
176         subarray2.add(values[n - 1]);
177         Sum_Exist(values, n - 1, subarray2, target - values[n - 1],
178             target2);
179     }
180
181     public static void main(String[] args) {
182
183         Scanner keyboard = new Scanner(System.in);
184
185         System.out.println("Enter the size of the Array n =\n");
186         int n = keyboard.nextInt();
187         int[] values = new int[n];
188         Vector<Integer> v = new Vector<Integer>();
```

```
187     System.out.println("Select Filling Method:\n(1) : For
      manual filling\n(2) : For automatic filling\n");
188     int a = keyboard.nextInt();
189
190     switch (a) {
191     case 1:
192         System.out.println("\nManual Filling Activated");
193         for (int i = 0; i < n; i++) {
194             System.out.println("Value " + (i + 1) + " =");
195             values[i] = keyboard.nextInt();
196         }
197         break;
198     case 2:
199         System.out.println("\nAutomatic filling Activated");
200         Random random = new Random();
201         for (int i = 0; i < n; i++) {
202             values[i] = random.nextInt(15);
203         }
204         break;
205     }
206
207     System.out.println("The target you are looking for =");
208     int target = keyboard.nextInt();
209     int target2 = target;
210     keyboard.close();
211
212     System.out.println("\n\n
      ..... \n\n");
213     System.out.println("The Array is : " + Arrays.toString(
      values) + "\n");
214     System.out.println("The Target Sum is : " + n + "\n");
215     System.out.println("The founded sums (if any) are : \n");
216     Sum_Exist(values, n, v, target, target2);
217     System.out.println("\nThe number of recursive calls are : "
      + cpt + " Times\n");
218 }
219 }
```



# Bibliography

- [1] *Recursive program to print all subsets with given sum*. July 2021. URL: [https://www.geeksforgeeks.org/recursive-program-to-print-all-subsets-with-given-sum/#\\_=\\_](https://www.geeksforgeeks.org/recursive-program-to-print-all-subsets-with-given-sum/#_=_).