



UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO
UNIDAD ACADÉMICA PROFESIONAL TIANGUISTENCO

PROYECTO:

Problema de la suma de subconjuntos

ALUMNO:

Rubi Esmeralda Rosales Chavero

CARRERA:

Ingeniería en software

UNIDAD DE APRENDIZAJE:

Programación Paralela

PROFESOR:

Ing. en Sw. Gustavo Gómez Vergara

Fecha de entrega: 18/12/2020

ÍNDICE

Problemática	3
Objetivos	3
Delimitación del problema	4
Metodología	4
Comunicación	
Requerimientos funcionales	5
Requerimientos no funcionales	5
Planeación	
Diagrama de gantt	6
Modelado	
Análisis y diseño	7
Construcción	
Código	10
Puebas	13
Despliegue	
Programa paralelo	15
Programa estructurado	15
Conclusiones	17
Referencias	17

PROBLEMÁTICA

El problema de la suma de subconjuntos es un problema importante en la teoría de la complejidad y en la criptografía. El problema es este: dado un conjunto de enteros, ¿existe algún subconjunto cuya suma sea exactamente cero?

Por ejemplo, dado el conjunto $\{-7, -3, -2, 5, 8\}$, la respuesta es SI, porque el subconjunto $\{-3, -2, 5\}$ suma cero. Este problema es NP-completo (Wikipedia, 2020).

OBJETIVOS

GENERAL	Realizar dos programas en lenguaje java que generen un conjunto de n elementos, obtengan la suma de todos sus subconjuntos e indiquen cuales son aquellos que su suma sea 0 y el tiempo total de su ejecución
ESPECÍFICOS	El usuario dará los parámetros para la generación y llenado del conjunto.
	El primer programa deberá ser elaborado bajo el paradigma de programación estructurada.
	El segundo programa deberá ser elaborado bajo el paradigma de programación paralela.

DELIMITACIÓN DEL PROBLEMA

El rendimiento que tengan ambos programas, es decir los tiempos de ejecución, están directamente ligados con el hardware del ordenador en el que se ejecuten en el ámbito de procesamiento y memoria RAM.

Las características del equipo que se ocupa en el desarrollo de este proyecto son las siguientes: procesador AMD Radeon R5 y memoria RAM de 8 GB.

METODOLOGÍA

El modelo en cascada, a veces llamado ciclo de vida clásico, sugiere un enfoque sistemático y secuencial para el desarrollo del software, que comienza con la especificación de los requerimientos por parte del cliente y avanza a través de planeación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado (Pressman, 2020).

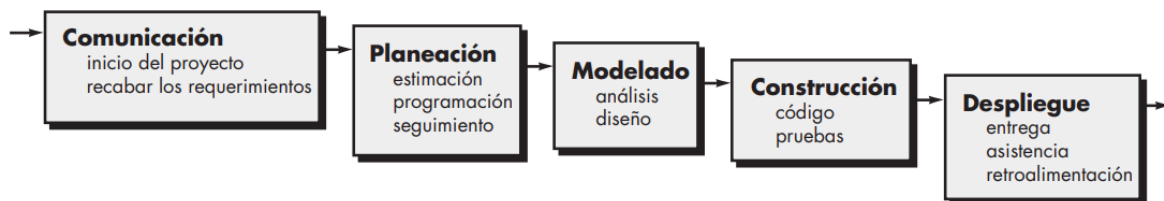


Imagen 1 Etapas de la metodología de cascada

La metodología de cascada es el paradigma más antiguo de la ingeniería de software. Sin embargo, sirve como un modelo de proceso útil para este proyecto ya que los requerimientos son fijos y el trabajo avanzará en forma lineal hacia el final.

COMUNICACIÓN

REQUERIMIENTOS FUNCIONALES

1. El usuario dará 3 parámetros para la generación y llenado del conjunto, los cuales son: n, entero mayor a dos, min, entero negativo y max, entero positivo.
2. Indicar al usuario la cantidad total de subconjuntos antes de iniciar con el proceso de sumar subconjuntos.
3. La generación de subconjuntos se dividirá según su longitud, la cual inicia con una variable i igual a n y a disminuyendo en 1 hasta tener un valor igual a 2.
4. En el programa paralelo:
 - a. El proceso se hará paralelo en el punto 3 de esta lista, ya que habrá i cantidad de hilos que realicen la generación de subconjuntos (dependiendo de su longitud).
 - b. Para mostrar las sumas de subconjuntos de usara un JTextArea distinto dependiendo de la longitud de los subconjuntos.
 - c. Para mostrar los resultados de subconjuntos que sumen 0 habrá otro JTextArea. Esto se realizará de forma paralela con el punto 4b.
5. En el programa estructurado:
 - a. La generación de subconjuntos será de forma lineal en el punto 3 de esta lista. Se hará uso de un ciclo for para encontrar todos los subconjuntos.
 - b. Se mostrará al usuario cada subconjunto encontrado seguido de su suma.
 - c. Al finalizar el punto 5b de esta lista se mostrarán los subconjuntos que sumen 0.

REQUERIMIENTOS NO FUNCIONALES

6. El tiempo de ejecución del programa paralelo deberá ser menor al del programa estructurado.
7. El programa estructurado pedirá los datos al usuario por medio de un cuadro de dialogo y mostrará los resultados en consola.
8. El programa paralelo hará uso de una interfaz grafica en la que el usuario escriba los datos para el conjunto y también pueda visualizar los resultados.

PLANEACIÓN

DIAGRAMA DE GANTT

Actividad	Fecha Inicio	Fecha Fin	Semana 1					Semana 2					Semana 3				
			30	01	02	03	04	07	08	09	10	11	14	15	16	17	18
Problemática y objetivos	30/11/2020	30/11/2020															
Delimitación y metodología	30/11/2020	30/11/2020															
Etapa 1 Comunicación	01/12/2020	02/12/2020															
Etapa 2 Planeación	02/12/2020	02/12/2020															
Etapa 3 Modelado	03/12/2020	07/12/2020															
Etapa 4 Construcción	08/12/2020	14/12/2020															
Etapa 5 Despliegue	15/12/2020	16/12/2020															
Revisión de los programas	16/12/2020	16/12/2020															
Revisión de documentación	17/12/2020	17/12/2020															
Entrega final	18/12/2020	18/12/2020															

MODELADO

ANÁLISIS Y DISEÑO

Para realizar el análisis se retomarán los requerimientos (expuestos en la primera etapa de esta metodología) que necesiten una explicación más amplia para su total comprensión. De ser necesario se mostrará un cuadro con las diferencias entre el programa paralelo y el estructurado, en caso de no mostrarse deberá entenderse que es igual para ambos programas.

En cuanto al diseño, se mencionará la manera en que se solicitan y muestran los datos, pero esto se profundizará en la última etapa de la metodología (despliegue). Se omite en este apartado la creación de diagramas ya que se trata de una única función para ambos programas, la cual es introducir los datos para generar el conjunto y observar los resultados.

Requerimiento 1: *El usuario dará 3 parámetros para la generación y llenado del conjunto, los cuales son: n, entero mayor a dos, min, entero negativo y max, entero positivo.*

El conjunto será un array de tipo entero de longitud n. Será llenado usando la función `random()` de la librería `Math` en java usando la siguiente “formula” que guarda en `valorEntero` un valor entre `min` y `max` (ambos incluidos); `min` debe ser menor que `max` y se requiere el cast a `int` porque la función `random()` retorna un `double`.

```
int valorEntero = (int) (Math.random()*(max-min+1)+min);
```

Programa estructurado	Programa paralelo
Uso de cuadros de dialogo de la clase <code>JOptionPane</code> para que el usuario ingrese los valores. Inmediatamente después continuará el proceso.	Uso de <code>(jTextField)</code> , donde el usuario escribirá los valores, y <code>(jButton)</code> para continuar con el proceso.

Requerimiento 2: Indicar al usuario la cantidad total de subconjuntos antes de iniciar con el proceso de sumar subconjuntos.

Para saber cuántos subconjuntos generará el programa debemos saber dos cosas: la primera es que no se tomarán en cuenta las permutaciones, solo las combinaciones. Veamos un ejemplo para que quede más claro: supongamos que tenemos el conjunto $\{-7, -3, -2, 5, 8\}$, del cual podemos obtener los subconjuntos $\{-3, -2, 5\}$, $\{5, -3, -2\}$ ó $\{-2, -3, 5\}$ que son permutaciones del subconjunto con esos 3 elementos $\{5, -2, -3\}$; como la suma al final es la misma, únicamente se considerará una combinación y no sus diversas permutaciones.

La segunda cosa que debemos saber es la fórmula para obtener las combinaciones

$$\text{Número de combinaciones} = \frac{n!}{(n - k)! k!}$$

Donde n es el número total de objetos y k es el número de objetos que tomamos para la combinación.

Traduciendo esto al programa n es equivalente a la longitud del conjunto y k tomará el valor de r en un ciclo for que inicia con r igual a n y va disminuyendo en 1 hasta ser igual a 2. Con cada iteración se va acumulando el resultado y de esa manera sabremos el total de subconjuntos que habrán de generarse.

$$\text{Total de subconjuntos} = \sum_{r=2}^{r=n} \left(\frac{n!}{(n - r)! r!} \right)_r$$

Requerimiento 3: La generación de subconjuntos se dividirá según su longitud, la cual inicia con una variable r igual a n y va disminuyendo en 1 hasta tener un valor igual a 2.

Si prestamos atención, este requerimiento nos habla de un ciclo for igual al del requerimiento anterior.

Usaremos esta estructura de control para llamar al método que genere todos los subconjuntos de determinada longitud, como se muestra a continuación.


```
for (r=n; r>=2; r--)  
{  
    subconjuntos(conjunto, r);  
}
```

El método “subconjuntos” es de tipo void, recibe el conjunto y el número de elementos que deben tener los subconjuntos generados. Podemos entender este método como algo parecido a la función de calcular combinaciones en una calculadora nCr , n es el conjunto y r es el numero de elementos de los subconjuntos.

Nota: el método “subconjuntos” no existe en realidad en el programa, es usado solo como ejemplo para la explicación de este requerimiento.

Requerimiento 4a: *El proceso se hará paralelo en el punto 3 de esta lista, ya que habrá r cantidad de hilos que realicen la generación de subconjuntos (dependiendo de su longitud).*

Esto consiste en que dentro del for anterior se iniciara un hilo por cada iteración para que ejecute el método “subconjuntos”. Cada hilo generará los subconjuntos de una longitud distinta, que va de 2 a n .

Requerimiento 4c: Para mostrar los resultados de subconjuntos que sumen 0 habrá otro JTextArea. Esto se realizará de forma paralela con el punto 4b.

Quiere decir que, conforme se vayan encontrando los subconjuntos que sumen 0, se irán mostrando en el JTextArea de resultados. Sin esperar a que termine de generar y sumar todos los subconjuntos como en el programa estructurado

CONSTRUCCIÓN

CÓDIGO

El proyecto en java “SumarSubconjuntos” contiene los dos programas, tanto el estructurado como el paralelo, cada uno en un paquete diferente:

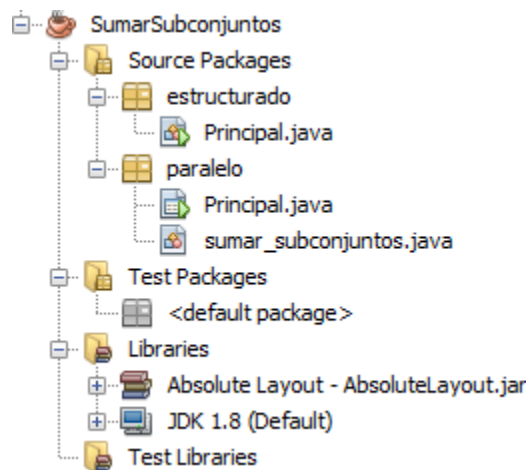


Imagen 2 Proyecto en Java

El algoritmo principal es el de sumar subconjuntos. Se trata de un método recursivo, a continuación se muestra el código original, y posteriormente las modificaciones que se aplicaron en él para adaptarlo a cada programa.

```
static ArrayList convis = new ArrayList<>();

static void combinations(int[] arr, int len, int startPosition, int[] result)
{
    if (len == 0)
    {
        convis.add(result);
        return;
    }
    for (int i = startPosition; i <= arr.length - len; i++)
    {
        result[result.length - len] = arr[i];
        combinations(arr, len - 1, i + 1, result);
    }
}
```

En el programa estructurado se trata de un método recursivo que genera el subconjunto, le asigna un numero con el contador “cont”, al obtenerlo realiza su suma y la imprime en consola. En caso de que la suma sea 0 almacena el contador, que corresponde con el numero del subconjunto, en un ArrayList “respuesta”.

```
for (int r = n; r >= 2; r--)
{
    suma_subconjuntos(conjunto, r, 0, new int[r]);
}
```

Imagen 3 Llamada al método desde el main

```
private static void suma_subconjuntos(int[] arr, int r, int posicion_inicio, int[] resultado)
{
    if (r == 0)
    {
        int suma = 0;

        for (int j = 0; j < resultado.length; j++)
        {
            suma += resultado[j];
        }

        if (suma == 0)
        {
            respuesta.add(cont);
        }

        System.out.println("Subconjunto " + cont + ": " + Arrays.toString(resultado) + " = " + suma);
        cont++;
        return;
    }
    for (int i = posicion_inicio; i <= arr.length - r; i++)
    {
        resultado[resultado.length - r] = arr[i];
        suma_subconjuntos(arr, r - 1, i + 1, resultado);
    }
}
```

Imagen 4 Método suma subconjuntos del programa estructurado

En el programa en paralelo el algoritmo es prácticamente el mismo, cambia en donde se muestra el resultado, ya que, este cuenta con una interfaz gráfica. Se manda un JTextArea a cada hilo para que en él muestre los subconjuntos de r cantidad de elementos.

```
for (int r = n; r >= 2; r--)
{
    javax.swing.JTextArea ta = new JTextArea("Subconjuntos con " + r + " elementos\n\n");
    ta.setEditable(false);
    ta.setBackground(new Color(240, 240, 240));
    Panel_Subconjuntos.add(ta);
    Panel_Subconjuntos.updateUI();
    Thread t = new sumar_subconjuntos(TextArea_Resultados, ta, conjunto, r);
    t.start();
}
```

Imagen 5 Llamada al método desde el JFrame

```
public class sumar_subconjuntos extends Thread
{
    javax.swing.JTextArea rs = null;
    javax.swing.JTextArea sc = null;
    int c[] = null;
    int r = -1;

    public sumar_subconjuntos(javax.swing.JTextArea resultados,
        javax.swing.JTextArea subconjuntos, int conjunto[], int r)
    {
        this.rs = resultados;
        this.sc = subconjuntos;
        this.c = conjunto;
        this.r = r;
    }

    @Override
    public void run()
    {
        suma_subconjuntos(c, r, 0, new int[r]);
        if (r==Principal.limite)
        {
            long fin_tiempo = System.nanoTime();
            Principal.TextField_Tiempo.setText(String.valueOf(Principal.
                tiempo(Principal.inicio_tiempo, fin_tiempo)));
            Principal.Button_Reset.setEnabled(true);
        }
    }
}
```

Imagen 6 Manejo de hilo y variables

```

private void suma_subconjuntos(int[] arr, int r, int posicion_inicio,
    int[] resultado)
{
    if (r == 0)
    {
        int suma = 0;
        for (int j = 0; j < resultado.length; j++)
        {
            suma += resultado[j];
        }
        if (suma == 0)
        {
            rs.append(Arrays.toString(resultado) + "\n");
        }
        sc.append(Arrays.toString(resultado) + " = " + suma + "\n");
        return;
    }
    for (int i = posicion_inicio; i <= arr.length - r; i++)
    {
        resultado[resultado.length - r] = arr[i];
        suma_subconjuntos(arr, r - 1, i + 1, resultado);
    }
}

```

Imagen 7 Método suma subconjuntos del programa paralelo

PUEBAS

Antes de hacer pruebas considero que se tienen que validar las entradas. Esto se hace mediante la estructura if en ambos programas; no se mostrará ese código en este apartado ya que se enfoca mas en los tiempos de ejecución de los programas.

Ambos cuentan el tiempo total de ejecución. En el programa estructurado, se captura el tiempo desde que termina de capturar los datos correctos ingresados por el usuario y hasta que termina de mostrar los conjuntos cuya suma fue 0.

En el caso de programa paralelo, el tiempo se captura desde que el usuario pulsa el botón y hasta que termina el calculo de subconjuntos de la longitud que de el mayor numero de combinaciones. Por ejemplo, en un conjunto de 6 elementos, el mayor numero de subconjuntos es en los que tienen 3 elementos.

Con el fin de comparar esos tiempos entre el programa estructurado y paralelo, enseguida se muestra una tabla con los resultados de tiempo de 5 ejecuciones distintas para cada tiempo y para cada programa con un rango de generación de números aleatorios entre -50 y 50.

Longitud conjunto	5	10	15
Numero subconjuntos	26	1013	32752

Programa	Tiempo en segundos		
Programación estructurada	0.0279814	0.5603922	11.0301866
	0.0245532	0.2921464	12.2208642
	0.0254544	0.5218381	11.9066893
	0.0159424	0.3171847	13.4198265
	0.0159499	0.4324868	13.9228956
Promedio	0.02197626	0.42480964	12.50009244

Programa	Tiempo en segundos		
Programación paralela	0.0742355	0.0739493	0.1861823
	0.1103944	0.1802547	0.3487786
	0.0351008	0.065048	0.1870507
	0.0195368	0.0346191	0.1717888
	0.0510815	0.041946	0.1361057
Promedio	0.0580698	0.07916342	0.20598122

DESPLIEGUE

Finalmente se mostrará cómo se ingresan los datos y como se muestran los resultados en ambos programas.

PROGRAMA PARALELO

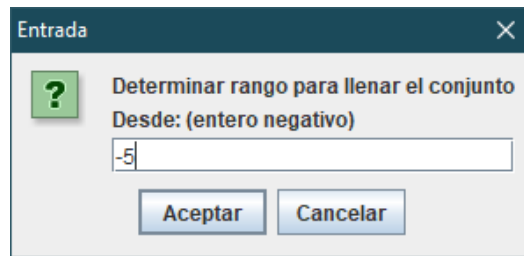
The screenshot shows a window titled "Sumar subconjuntos". At the top, there are input fields for "n" (value 5), "min" (value -5), and "max" (value 5), along with "iniciar" and "reset" buttons. Below this is a "Conjunto" input field containing "[0, 2, 4, 0, -4]" and a "No. subconjuntos" field showing "26". The main area is divided into two sections: "Resultados" on the left and "Subconjuntos" on the right. The "Resultados" section lists the elements of the set: [0, 4, 0, -4], [0, 4, -4], [4, 0, -4], [0, 0], and [4, -4]. The "Subconjuntos" section is divided into four columns based on the number of elements: "Subconjuntos con 5 elementos", "Subconjuntos con 4 elementos", "Subconjuntos con 3 elementos", and "Subconjuntos con 2 elementos". Each column lists subconjuntos and their corresponding sum. For example, under "Subconjuntos con 5 elementos", it shows "[0, 2, 4, 0, -4] = 2". At the bottom, there is a "Tiempo total de ejecución" field showing "0.028628699" and a "segundos" label.

Imagen 8 Interfaz del programa paralelo

PROGRAMA ESTRUCTURADO

The screenshot shows a dialog box titled "Entrada". It contains a green question mark icon and the text "Escriba la longitud del conjunto (entero mayor a 2)". Below this is an input field with the value "5". At the bottom, there are two buttons: "Aceptar" and "Cancelar".

Imagen 9 Solicitar valor de n

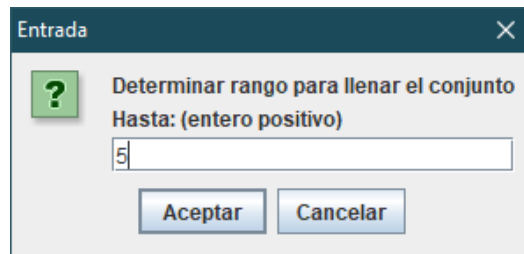


Entrada

? Determinar rango para llenar el conjunto
Desde: (entero negativo)

Aceptar Cancelar

Imagen 10 Solicitar valor de min

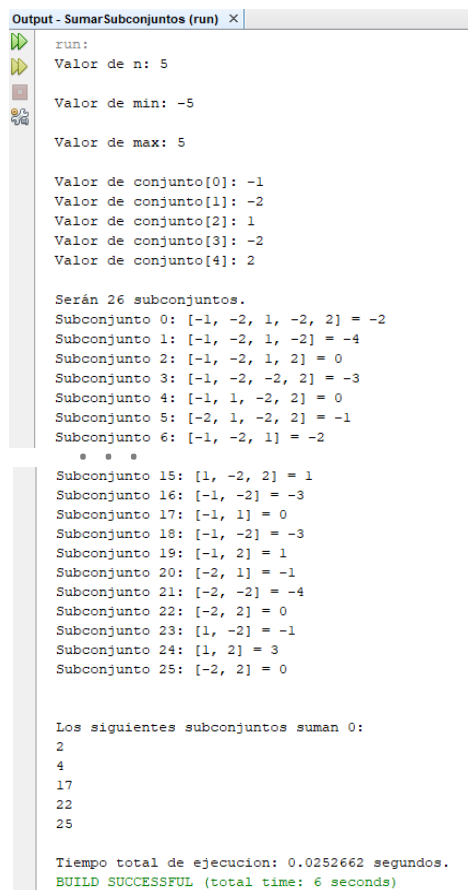


Entrada

? Determinar rango para llenar el conjunto
Hasta: (entero positivo)

Aceptar Cancelar

Imagen 11 Solicitar valor de max



```

Output - SumarSubconjuntos (run) X
run:
Valor de n: 5
Valor de min: -5
Valor de max: 5

Valor de conjunto[0]: -1
Valor de conjunto[1]: -2
Valor de conjunto[2]: 1
Valor de conjunto[3]: -2
Valor de conjunto[4]: 2

Serán 26 subconjuntos.
Subconjunto 0: [-1, -2, 1, -2, 2] = -2
Subconjunto 1: [-1, -2, 1, 2] = -4
Subconjunto 2: [-1, -2, 1, 2] = 0
Subconjunto 3: [-1, -2, -2, 2] = -3
Subconjunto 4: [-1, 1, -2, 2] = 0
Subconjunto 5: [-2, 1, -2, 2] = -1
Subconjunto 6: [-1, -2, 1] = -2
. . .
Subconjunto 15: [1, -2, 2] = 1
Subconjunto 16: [-1, -2] = -3
Subconjunto 17: [-1, 1] = 0
Subconjunto 18: [-1, -2] = -3
Subconjunto 19: [-1, 2] = 1
Subconjunto 20: [-2, 1] = -1
Subconjunto 21: [-2, -2] = -4
Subconjunto 22: [-2, 2] = 0
Subconjunto 23: [1, -2] = -1
Subconjunto 24: [1, 2] = 3
Subconjunto 25: [-2, 2] = 0

Los siguientes subconjuntos suman 0:
2
4
17
22
25

Tiempo total de ejecucion: 0.0252662 segundos.
BUILD SUCCESSFUL (total time: 6 seconds)

```

Imagen 12 Resultado en consola del programa estructurado (fragmento)

CONCLUSIONES

Este proyecto me ayudo a comprender mejor en que consiste el paradigma de la programación paralela; quede satisfecha con el programa realizado y su funcionamiento, sin embargo, me gustaría probar con otros métodos para la generación de subconjuntos que sean mas eficientes y evaluar si puede disminuir el tiempo de ejecución.

Fue un poco complicado al principio tratar de pensar la manera en que podría implementar la programación paralela. Tras esta experiencia creo que resulta mejor primero programar de forma estructurada, así es más fácil ver que partes pueden realizarse simultáneamente.

REFERENCIAS

Pressman. (2020). IngenieríaDe Software(7.aed.). MCGRAW HILL EDDUCATION.
Wikipedia. (2020, 5 mayo). *Problema de la suma de subconjuntos*. Wikipedia, la enciclopedia libre.
https://es.wikipedia.org/wiki/Problema_de_la_suma_de_subconjuntos