Documento - Problema de la suma de los subconjuntos

índice

1. Análisis 2

1.1. Problemática 2

1.2. Requerimientos funcionales 2

1.3. Requerimientos no funcionales 2

1.4. Tipos de datos 3

1.5. Casos de uso 3

1.6. Metodología 3

1.7. Cronograma de actividades 4

2. Diseño 4

2.1. Tipo de programación 6

2.2. Compilador del lenguaje de programación 6

2.3. Estructuras de control 6

3. Implementación 7

3.1. Variables ocupadas 7

3.2. Clases ocupadas. 7

3.2.1. Clases de programa lineal 8

3.2.2. Clases del programa paralelo 8

4. Pruebas 9

4.1. Prueba 1: Conjunto de 5 elementos 9

4.2. Prueba 2: Conjunto de 12 elementos 10

4.3. Prueba 3: Conjunto de 20 elementos 11

5. Mantenimiento 12

1. Análisis

1.1. Problemática

El problema de la suma de subconjuntos es importante dentro de la teoría de la complejidad y en la criptografía. El problema consiste en encontrar todos los subconjuntos de un conjunto cuya suma sea igual a 0, es decir, dado un subconjunto Wn encontrar todos los subconjuntos que sumen 0.

$$W = \{-8, -3, -2, 5, 8\}$$

La respuesta a este subconjunto sería:

$$W' = \{-3, -2, 5\}, \{-8, 8\}$$

El programa que se desarrollará deberá brindar principalmente un tiempo de resolución del problema en un tiempo aceptable, puesto que la suma de subconjunto pertenece a los problemas NP-completos, esto quiere decir que resolverlo implica un coste exponencial de tiempo y dependiendo de la computadora también recursos.

1.2. Requerimientos funcionales

- **RF1** Mostrar todos los subconjuntos del conjunto.
- **RF2** Mostrar todos los subconjuntos cuya suma sea igual a la brindada por el usuario.
- **RF3** Mostrar todos los subconjuntos que no sumen la suma deseada
- **RF4** Mostrar todos los subconjuntos del conjunto generado.
- **RF4** Generar un conjunto de tamaño *n* que el usuario brinde.
- **RF5** Capturar la suma deseada que se calculará.

1.3. Requerimientos no funcionales

- **RNF1** Tiempo de ejecución aceptable.
- **RNF2** Mostrar el tiempo de resolución de los procesos que muestren subconjuntos.
- **RNF3** Mostrar el tiempo de ejecución de la resolución de los 3.
- **RNF4** Mostrar el numero de resultados de cada proceso que muestren subconjuntos.

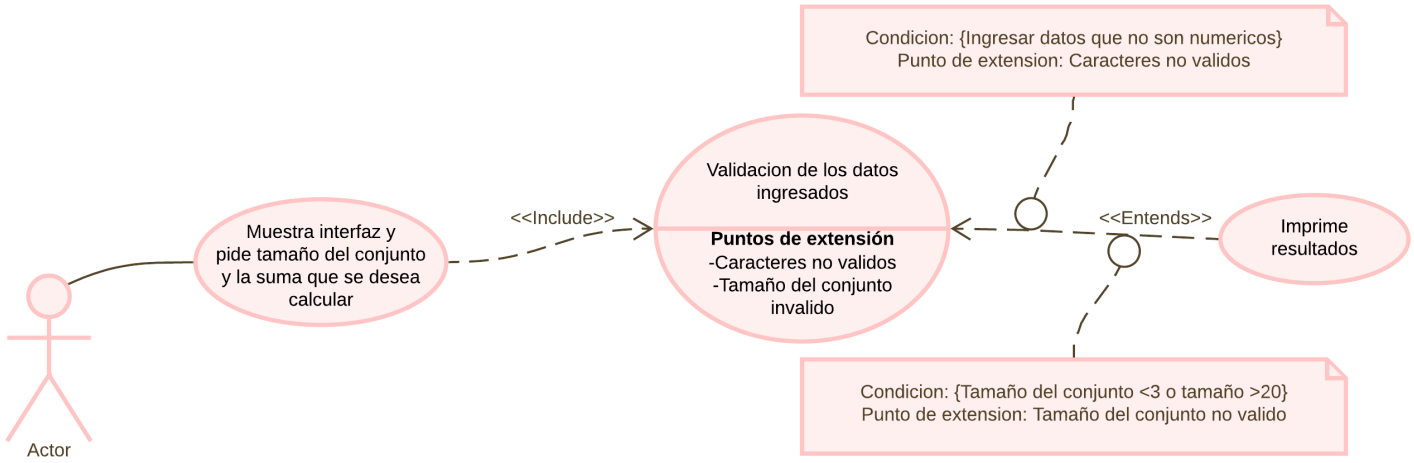
- **RNF5** Validación de campos de texto.

1.4. Tipos de datos

Los tipos de datos que ocupará el programa son enteros, arreglo de enteros, Arreglos de listas de objetos enteros y listas de listas que almacenen objetos de tipo entero. Los cuales en el lenguaje Java se conocen como:

- Entero: int.
- Arreglo de enteros: int []
- Listas de listas de objetos enteros: List<List<Integer>>
- Arreglo de listas de objetos enteros: ArrayList <Integer>

1.5. Casos de uso



1.6. Metodología

Cascada

El modelo de desarrollo en cascada sigue una serie de etapas de forma sucesiva, la etapa siguiente empieza cuando termina la etapa anterior. Esta metodología no permite regresar a la fase anterior por lo que se debe tener cuidado al definir los requerimientos.

El desarrollo en cascada es un procedimiento lineal que se caracteriza por dividir los procesos de desarrollo en sucesivas fases de proyecto. Al contrario que en los modelos iterativos, cada una de estas fases se ejecuta tan solo una vez y los resultados de cada una de las fases sirven como hipótesis de partida para la siguiente.

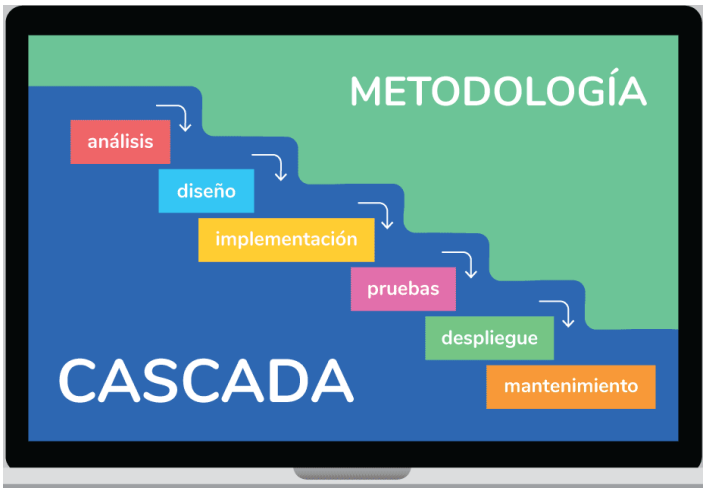


Ilustración 1 Fases de la metodología en cascada

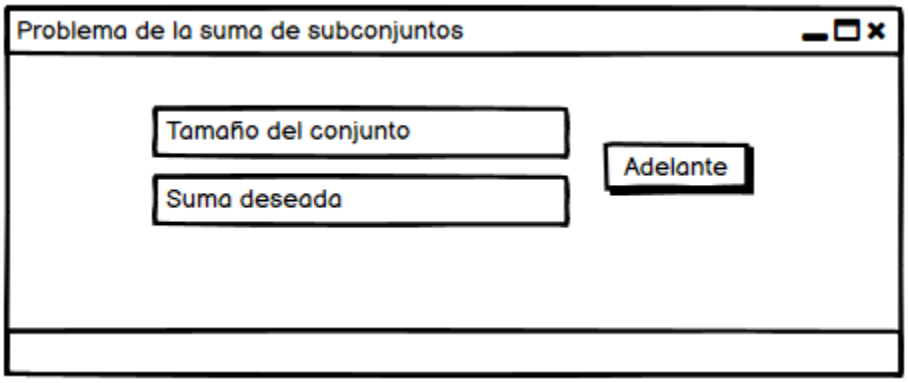
1.7. Cronograma de actividades

Nombre de la actividad: "Problema de la suma de subconjuntos"	Inicio	Final	Noviembre																				
			Semana 1							Semana 2							Semana 3						
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Realizacion de la documentacion (Analisis y diseño)	03/11/2020	04/11/2020																					
Crear algoritmo que solucione el problema linealmente	04/11/2020	06/11/2020																					
Crear interfaz del programa	06/11/2020	07/11/2020																					
Validar campos del programa	09/11/2020	10/11/2020																					
Implementación de hilos que impriman los resultados	11/11/2020	12/11/2020																					
Pruebas programa y optimizacion de código	13/11/2020	17/11/2020																					

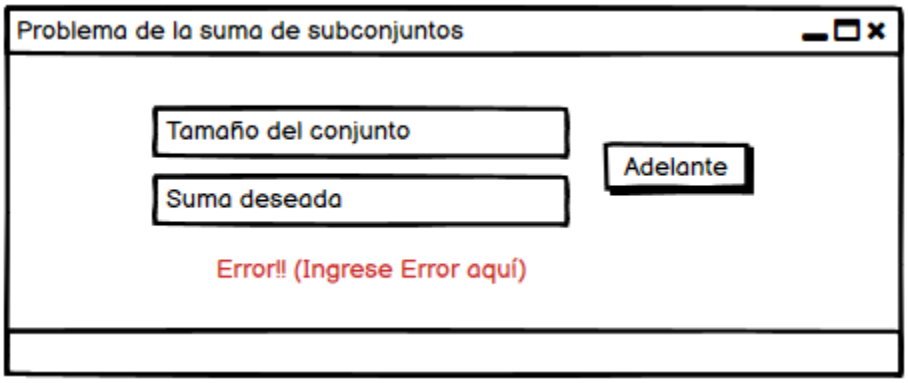
Terminologia de colores	
#	No se labora
	Realizacion de actividades
	Fecha de entrega

2. Diseño

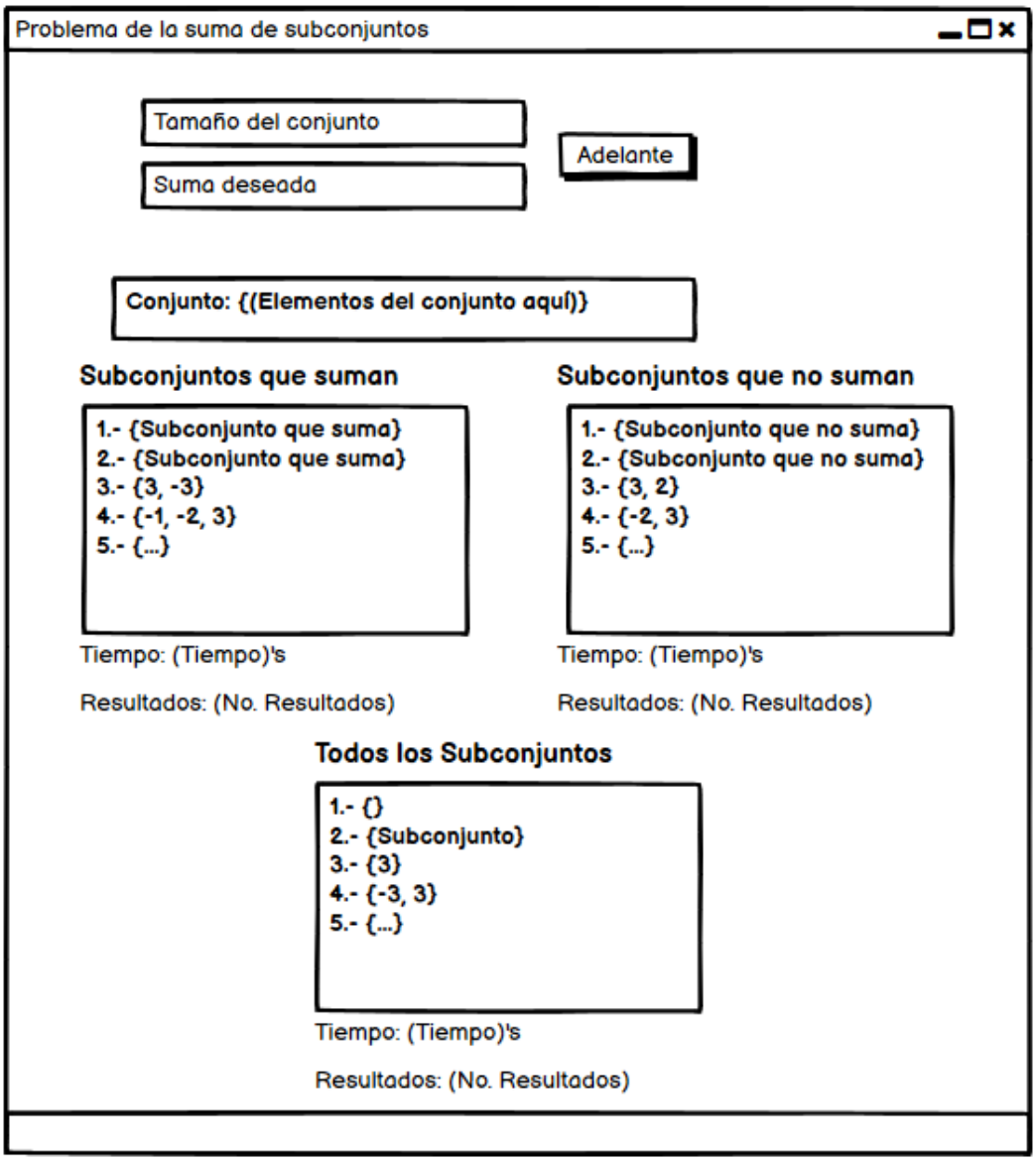
La interfaz se dividirá em 2 partes, la primera pedirá los datos al usuario cuando es la primera vez que se realiza un cálculo de subconjuntos.



En la interfaz deberá de marcar los errores que se cometan, para ello se imprimirán de como en el siguiente prototipo.



Una vez validados los campos empieza el cálculo de los subconjuntos.



2.1. Tipo de programación

Programación Paralela (Concurrente): Esta se ocupará para calcular de manera simultánea los subconjuntos que suman, que no suman y todos los subconjuntos del conjunto.

Programación Orientada a Objetos (POO): Los métodos que componen al programa se administran mediante objetos, de este modo se mantiene un orden y el código es más legible.

2.2. Compilador del lenguaje de programación

NetBeans IDE es un entorno de desarrollo integrado (IDE), modular, de base estándar (normalizado), escrito en el lenguaje de programación Java. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación, en este caso, se utilizará para poder crear la interfaz del programa y programar en Java.

2.3. Estructuras de control

Las estructuras de control que se utilizarán del lenguaje Java en este programa son:

Condición if

if (*Condición***)**

{

}

Condición if - else

if (*Condición***)**

{

} else

{

}

Bucle for

for (int **;** *condicion* **;** *incremento***)**

{

}

Bucle for each o Bucle for extendido

for(*objeto* **:** *arreglo de objetos***)**

{

}

3. Implementación

Para calcular todos los subconjuntos se optó por un método de combinaciones para utilizar únicamente ciclos for y condicionales. El código calcula el numero de resultados encontrados que cumplen la suma deseada y muestra también el tiempo que tardó en ejecutarse cada operación.

Las operaciones que el programa realiza son

1. Calcular y mostrar los subconjuntos que suman la suma ingresada por el usuario.
2. Calcular y mostrar los subconjuntos que no suman la suma ingresada por el usuario.
3. Calcular y mostrar todos los subconjuntos del conjunto.
4. Llenar el conjunto de manera automática.

3.1. Variables ocupadas

Int [] conjunto; Contiene a todos los elementos del conjunto.

Int sumD; Almacena la suma que desea encontrar el usuario en los subconjuntos

List<List<Integer>> subsets = new ArrayList<>(); Esta variable almacenará a todos los subconjuntos incluyendo el vacío.

String ns = ""; Guarda todos los subconjunto que no suman

String s = “”; Esta variable guarda todos los subconjuntos que suman

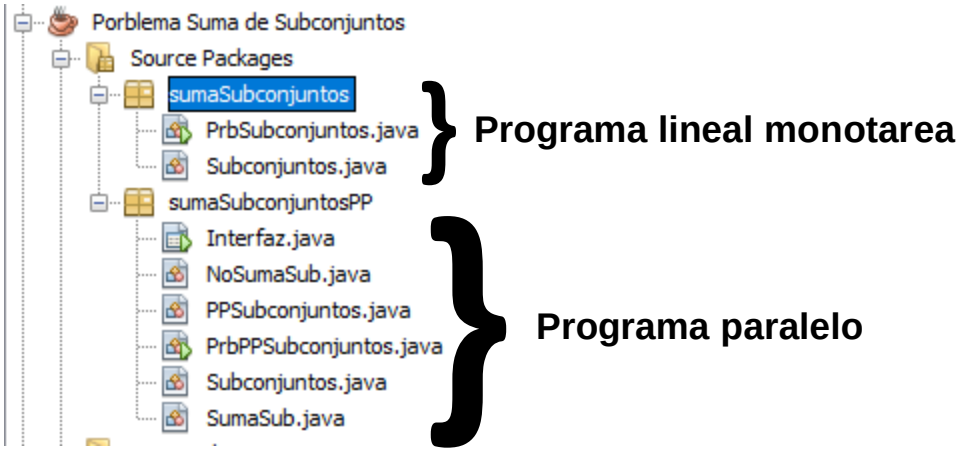
int sum = 0; Cuenta a todos subconjuntos que suman

int nSum = 1; Cuenta a todos los subconjuntos que no cumplen la suma, inicia en 0 debido a que se toma en cuenta el subconjunto vacío.

Thread t = new Thread(); Hilo que ejecutará las instrucciones de las operaciones del cálculo de subconjuntos.

3.2. Clases ocupadas.

El proyecto de Java se puede abrir con la IDE NetBeans, en ella encontrará dos paquetes uno de ellos contiene el programa sin hilos estructurado *“sumaSubconjuntos”*, este programa imprime por consola los resultados mientras que en el otro paquete *“sumaSubconjuntosPP”* encontrará el programa con una interfaz gráfica y ejecución con hilos.



Para ejecutar el programa sin hilos ejecute la clase *"PrbSubconjuntos"* del paquete *"sumaSubconjuntos"* sin embargo, si quiere ejecutar el programa paralelo ejecute la clase *"Interfaz"* del paquete *"sumaSubconjuntosPP"*.

3.2.1. Clases de programa lineal

PrbSumaSubconjuntos

En este podremos seleccionar el tamaño del conjuntos y ejecutar el programa. Para cambiar el tamaño del conjunto

Subconjuntos

Esta clase contiene todos los métodos para llenar el conjunto, así como para calcular los subconjuntos.

3.2.2. Clases del programa paralelo

Las clases importantes del programa paralelo son las siguientes.

Interfaz: Esta clase permite la ejecución del programa paralelo y contiene todo el código de las interfaces que muestra el programa, también recolecta las información como el tamaño del conjunto y la suma deseada.

NoSumaSub: Esta clase realiza el cálculo de todos los subconjuntos que nos suman la suma deseada.

SumaSub: Esta clase se encarga únicamente de calcular los subconjuntos que suman la suma deseada.

Subconjuntos: Esta clase se encarga de imprimir a todos los subconjuntos, incluyendo el vacío.

PPSubconjuntos: Contiene los métodos para llenar el conjunto y también contiene el método para calcular los tiempo de ejecución de las operaciones.

4. Pruebas

Las pruebas principalmente se realizaron para confirmar que el programa paralelo resuelve en mejor tiempo la suma de subconjuntos que el programa lineal. Para la obtención de resultados se han promediado los tiempos en segundos de tres intentos por cada prueba con la suma deseada de cero, al final de cada apartado se muestran los promedios tanto del programa paralelo como el lineal y las diferencias de tiempos.

4.1. Prueba 1: Conjunto de 5 elementos

Programa lineal

Primer intento

```
Tiempos: Segundos
    Calculo subconjuntos: 6.699E-4
```

Segundo intento

```
Tiempos: Segundos
    Calculo subconjuntos: 6.798E-4
```

Tercer intento

```
Tiempos: Segundos
    Calculo subconjuntos: 6.967E-4
```

Programa Paralelo

Primer intento

```
Tiempo: 0.0027265's Terminado
```

Segundo intento

```
Tiempo: 6.015E-4's Terminado
```

Tercer intento

```
Tiempo: 7.75E-4's Terminado
```

Conclusiones

El programa paralelo y el programa lineal muestran resultados muy parecidos mismos resultado.

Promedio de tiempo del programa lineal: 0.000463’s

Promedio de tiempo del programa paralelo: 0.001255’s

Diferencia de tiempo: 0.0007928’s a favor del programa lineal

4.2. Prueba 2: Conjunto de 12 elementos

Programa lineal

Primer intento

```
Tiempos: Segundos
    Calculo subconjuntos: 0.7794419
```

Segundo intento

```
Tiempos: Segundos
    Calculo subconjuntos: 0.6864417
```

Tercer intento

```
Tiempos: Segundos
    Calculo subconjuntos: 0.6157084
```

Programa Paralelo

Primer intento

```
Tiempo: 0.0358726's Terminado
```

Segundo intento

```
Tiempo: 0.0291509's Terminado
```

Tercer intento

```
Tiempo: 0.0120735's Terminado
```

Conclusiones

Ahora el programa paralelo muestra una ventaja de tiempo considerable, esto muestra que la implementación de los hilos ha funcionado.

Promedio de tiempo del programa lineal: 0.693864’s

Promedio de tiempo del programa paralelo: 0.00208370’s

Diferencia de tiempo: 0.6917803’s a favor del programa paralelo.

4.3. Prueba 3: Conjunto de 20 elementos

Esta prueba se limitó a 1 intento en el programa lineal debido a que pasados 5 minutos el programa aun no resolvía el problema.

Programa lineal

Primer intento

Indeterminado, intento cancelado después de haber transcurrido 10 minutos

Programa Paralelo

Primer intento

```
Tiempo: 3.3572007's Terminado
```

Segundo intento

```
Tiempo: 2.8384978's Terminado
```

Tercer intento

```
Tiempo: 1.5759971's Terminado
```

Conclusiones

El programa lineal ni siquiera pudo terminar la suma de los subconjuntos.

Promedio de tiempo del programa lineal: Indeterminado

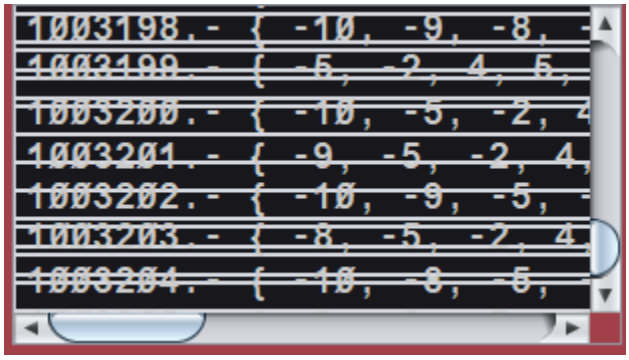
Promedio de tiempo del programa paralelo: 2.59050652’s

Diferencia de tiempo: Tiempo a favor del programa paralelo.

5. Mantenimiento

El programa puede mejorar en muchos aspectos, como por ejemplo realizar la suma de subconjuntos de varios conjuntos al mismo tiempo, que el botón de *“Adelante no se”* bloquee y abra otra ventana que resuelva otro subconjuntos.

Otra mejora es la del bug de las líneas que aparecen en las áreas de texto donde se imprimen los resultados. Este bug no fue tratado debido a que no altera el resultado del programa y los números del subconjuntos son legibles.



Aumentar el rango de calculo del programa, ya que a partir de los 50 -100 elementos del conjunto el programa se cierra.