UNIVERSIDAD DE COSTA RICA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS DE LA
COMPUTACIÓN E INFORMÁTICA

CI-0124– Computabilidad
Prof. Juan Jose Vargas Morales

Tarea de programación 1

Elaborado por:

José Ignacio Víquez Rojas B78451

Descripción del problema a resolver

Se tiene un conjunto de 10000 números positivos, los cuales se quieren dividir en 3 subconjuntos disjuntos A, B y C, tales que todo número de la lista pertenece a un solo subconjunto y la suma de números en cada subconjunto sea lo más parecida posible a la de los otros dos subconjuntos. Si hay números repetidos, es claro que algunos pueden estar en un subconjunto y otros en otro.

Si la suma de los valores en cada subconjunto es s1, s2 y s3, queremos minimizar la suma de cuadrados de sus diferencias:

f(s1, s2, s3) = (s1 - s2)^2 + (s1 - s3)^2 + (s2 - s3)^2.

Para evitar trabajar con números muy grandes, dividiremos la función f entre una constante grande, como 10^12. Es decir, trabajaremos con la función

f(s1, s2, s3) = ((s1 - s2)^2 + (s1 - s3)^2 + (s2 - s3)^2) / 10^12.

Observe que si los 3 subconjuntos sumaran lo mismo, f(s1, s2, s3) = 0.

Técnica Empleada para resolver el problema

Para resolver este problema se tomó la idea de una reflexión llamada “las piedras y el frasco” que trata de encontrar la mejor manera de “acomodar” piedras de distintos tamaños en un frasco de cristal. Se basa en poner las piedras más grandes en el frasco primero, luego las medianas y finalmente las más pequeñas.

Explicado de donde se sacó la idea ahora pasemos a explicar como se implementó. Se puede explicar en varios pasos:

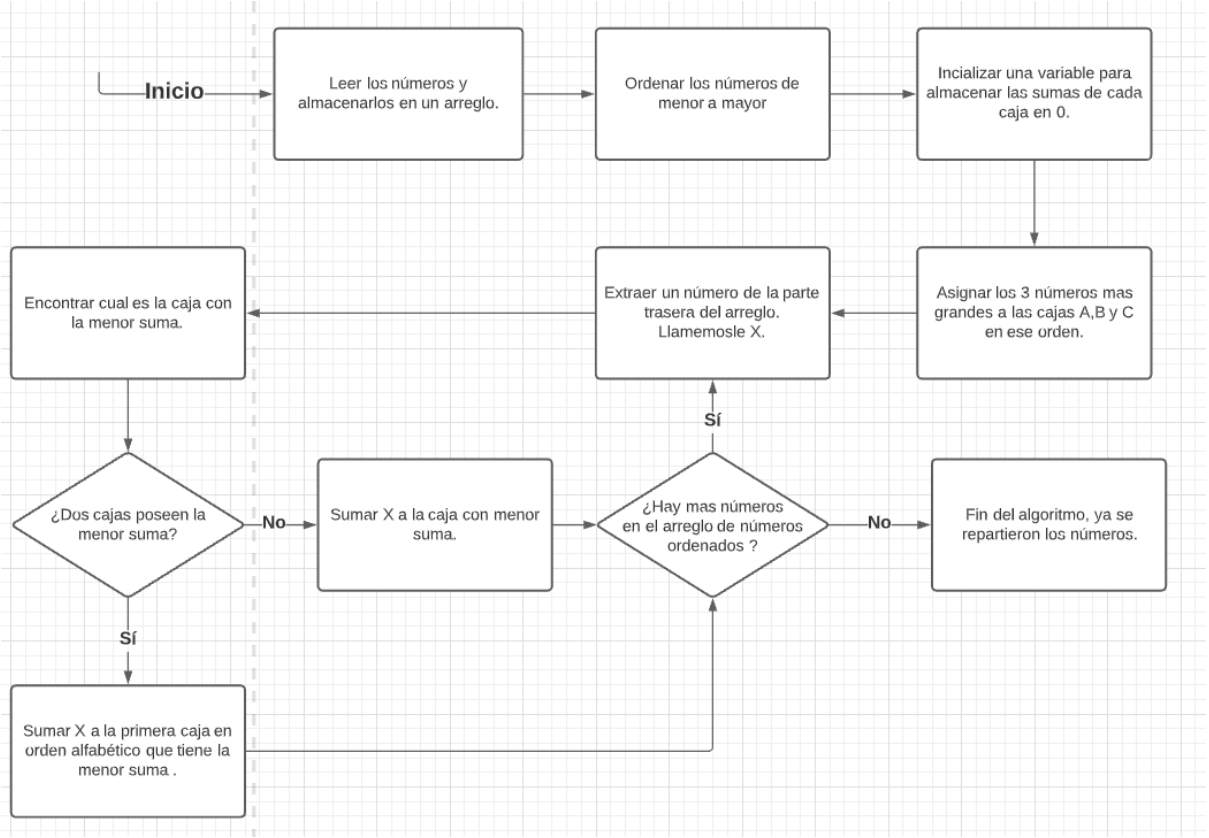
1. Se leyeron los datos del archivo de nombre “numbers.txt” y se almacenaron en un *array* de tipo *long* con el nombre “números”.
2. Una vez almacenados los números en el *array* se ordenaron de menor a mayor, o sea en las primeras posiciones del array estaban los números más bajos y en las últimas los más grandes.
3. Con los números ordenados se procedió a repartirlos en las diferentes cajas, para contextos de esa tarea se trabajó con las cajas con nombres “cajaA”, “cajaB” y “cajaC”. Para cada caja se guardó el valor de la suma que llevaban en cada momento, se denotan como “sumaA”, “sumaB” y “sumaC”. Ahora pasando a cómo se repartieron los números se hizo de la siguiente manera:
    3.1. Al inicio del programa se asignaron los tres primeros números más altos a las cajas en el orden A-B-C. Esto se realiza para inicializar las sumas de cada caja.

3.2. Se lee un nuevo número del *array* de números ordenados de atrás a adelante, o sea un número de los más grandes. En este punto se revisan las sumas totales de cada caja y se busca al menor.

3.3. Cuando encontramos la caja que tiene el valor de la suma menor le sumamos el dígito que fue extraído en el punto 3.2. En caso de que existan dos cajas con el mismo valor de suma y este sea el de la suma más baja se lo sumamos a la primera caja en orden alfabético.

3.4. Repetimos los pasos 3.2 y 3.3 hasta que se acaben los números del arreglo de números ordenados.

Veamos el siguiente diagrama de flujo que explica el método:



Duración para hallar la respuesta

La duración del programa fue de 0.2398 segundos aproximadamente.

Resultados encontrados

Por la forma de la solución propuesta solo se realizó solo una evaluación, ya que el ejecutar el programa más de una vez con este método resultaría en los mismos resultados.

Diferencia obtenida sin dividir entre 10^12: 96

Diferencia obtenida al dividir : 0,000000000096

Los resultados obtenidos se muestran además en el documento llamado “salida.txt” que se crea al ejecutar el programa. En este documento se muestra el desglose de qué números quedaron en que caja.

Línea de comando usada para compilar el programa

Compilar con : gcc Tarea1.c -o Tarea1 -lm