UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA
INFORMÁTICA

Algoritmos Avanzados

Laboratorio N°1: Cálculo de complejidad

Alumnos: Israel Arias Panez.
Christian Mendez Acosta.

Sección: A-1.

Santiago -
Chile 1-2021

Pseudocódigo el programa de fuerza bruta:

1. Se Solicita el nombre del archivo por teclado.

2. Se verifica si existe el dataset según el nombre entregado, en caso de no existir vuelve al paso 1.

3. Se almacena la capacidad máxima del container por el nombre del archivo.

4. Se almacena la cantidad de paquetes por el nombre del archivo.

5. Se abre el archivo dataset (archivo .csv).

6. Lectura, validación y creación de paquetes, creación del arreglo inicial que contiene todos los paquetes leídos y validados.

7. Se verifica en caso de que no se haya guardado ningún paquete válido.

8. Se generan todas las posibles combinaciones válidas y sin repetición a partir del arreglo inicial, las cuales se almacenarán en un arreglo de combinaciones

9. Se busca la combinación que otorga el mayor beneficio, dentro del arreglo final de combinaciones.

10. Se escribe un archivo de salida .txt que posee la información de la mejor combinación encontrada, escribiendo los paquetes que la contienen, el beneficio y el volumen que ocupa.

Cálculo de complejidad:

Los pasos 1, 2, 3, 4 y 5 son constantes O(1), por lo que no serán calculados a continuación.

6.

Por cada línea en el archivo: **->O(n)** Se lee peso de la línea (segundo dígito) //C Si peso < Capacidad Container: //C Generar paquete según estructura //C Guardar paquete en arreglo inicial //C Cerrar archivo //C	O(n)

7.

Si largo arreglo inicial == 0: //C Mostrar por pantalla “No hay solución válida” //C Fin de Programa	O(1)

8.

```
Se crea un arreglo de combinaciones vacío

Para cada elemento del arreglo inicial: ->O(n)
      Crear Combinación según estructura usando el elemento //C
      Guardar Combinación en arreglo de combinaciones //C

Para cada elemento del arreglo de combinaciones: ->O(2^n)
      Se lee el último índice almacenado en la estructura de la
      combinación //C

      Para ultimo indice + 1 hasta largo de arreglo inicial - 1: ->O(n)
            Efectuar suma entre volumen total de la combinación y volumen
            del paquete //C

            Si Suma Total < Capacidad Container: //C
                  Sumar beneficio del paquete al beneficio total de la
                  combinación // C
                  Crear nueva combinación tomando la combinación antigua y
                  agregandole el nuevo paquete, fijando nuevo beneficio y
                  nuevo volumen //C
                  Guardar nueva combinación en arreglo de combinaciones.
                  //C
```

Matemáticamente la suma de todas las posibles combinaciones desde cero hasta n sin repetición está dada por la expresión:

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n$$

Sin embargo, el total de combinaciones no incluye un conjunto con cero elementos, por ende el total de combinaciones sería igual a $2^n - 1$en consecuencia el número máximo de iteraciones será de $2^n - 1$ Por lo que la complejidad del primer ciclo está dado por O(2^n), la complejidad del ciclo anidado a este, es de O(n), por lo que finalmente la complejidad de este trozo de código es de O($n \cdot 2^n$)

9.

Se toma el primer elemento del arreglo de combinaciones como el IndiceMayorBeneficio //C
 	
Se guarda el valor del beneficio del primer elemento del arreglo de combinaciones en la variable MontoBeneficioMayor //C
 	
Para cada elemento del arreglo de combinaciones: **->O(n)**
Si beneficio de la combinación > MontoBeneficioMayor: //C
MontoBeneficioMayor=valor beneficio de la combinación //C
IndiceMayorBeneficio = indice de la combinación actual//C
	O(n)

10.

Se lee el valor de IndiceMayorBeneficio //C Se consigue la combinación con mayor beneficio a través de su índice dentro del arreglo de combinaciones //C Abrir archivo con nombre salida.txt //C Para cada elemento dentro del arreglo de paquetes de la combinación con mayor beneficio: **->O(n)** Leer numero de id del elemento actual //C Escribir “Paquete” + id en archivo //C Leer numero de beneficioTotal dentro de la combinación //C Escribir “Beneficio: ” + beneficioTotal en archivo //C Leer numéro de volumenTotal dentro de la combinación //C Escribir “Volumen: “ + volumenTotal en archivo //C Cerrar archivo //C Fin de Programa	O(n)

Complejidad total

O(n) + O(1) + O($n \cdot 2^n$) + O(n) + O(n)

Por cota superior: $n \cdot 2^n \geq n \geq 1$ entonces la complejidad del algoritmo es de

O($n \cdot 2^n$)