



Algoritmos Avanzados

Laboratorio N°1: Fuerza Bruta

Integrantes: Christian Méndez
Israel Arias
Sección: 0-A-1
Profesor(a): Aileen Esparza

9 de Mayo de 2021

Tabla de contenidos

1. Introducción	1
1.1. Contexto	1
1.2. Motivación	1
1.3. Teoría relevante	1
1.4. Objetivos	2
2. Método	3
2.1. Recursos utilizados	3
2.2. Procedimientos	3
2.2.1. Implementación del algoritmo	3
2.2.2. Test de la implementación	5
2.3. Resultados y análisis	5
3. Discusión	9
3.1. Resultados	9
3.2. Recomendaciones	10
4. Conclusiones	11
5. Apéndice	12
5.1. Pseudocódigo implementado	12
5.2. Instrucciones de uso	14
Referencias	16

1. Introducción

1.1. Contexto

En el presente laboratorio, se plantea una situación en la cual, con el fin de abrir un nuevo negocio, tenemos un presupuesto el cual será invertido en productos, también se tiene la información del precio de venta y volumen de cada uno de los productos ofrecidos, por lo que se busca determinar la combinación de productos más lucrativa dado el volumen disponible del container en que serán importados.

1.2. Motivación

Existen muchos algoritmos los cuales son capaces de solucionar problemas tanto industriales como cotidianos, estos algoritmos presentan una alta utilidad en muchos tipos de problemas en los cuales la mente humana no resulta eficiente debido a la alta cantidad de datos que necesitan ser analizados, por ejemplo para el problema planteado es posible el determinar la mejor combinación de productos, lo que pareciera ser una tarea sencilla, sin embargo al darse cuenta la gran cantidad de combinaciones posibles de distintos productos esta tarea parece ser interminable si es ejecutada por una persona, por lo cual resulta más eficiente el modelar computacionalmente el problema, lo que permite resolverlo en tiempos más acotados. La resolución del modelado del problema se efectúa a través de un algoritmo, por lo que el aprendizaje de distintos tipos de algoritmos para la solución de diversas problemáticas se vuelve vital en una sociedad donde los problemas que involucran grandes cantidades de datos se vuelven más comunes.

1.3. Teoría relevante

El algoritmo que se usará para resolver el problema planteado es el algoritmo de búsqueda exhaustiva o también conocido como algoritmo de fuerza bruta, el cual consiste en generar todas las combinaciones posibles para un cierto problema, para luego determinar la mejor combinación de acuerdo a la condición requerida. Por ejemplo, en un candado como el que se puede observar en la figura 1, el objetivo es abrir el candado, para ello el algoritmo

de fuerza bruta creara todas las combinaciones de 4 números existentes, hasta encontrar la única combinación que abre el candado, lo que es equivalente a una persona variando uno en uno los números del candado, probando combinaciones hasta que este eventualmente se abra. “Los algoritmos de fuerza bruta buscan hallar la solución a un problema generando cada uno de los posibles candidatos para la misma, y verificando si efectivamente cada uno de éstos cumple las restricciones o condiciones para ser la solución buscada.” [1]



Figura 1: Ejemplo Fuerza Bruta

1.4. Objetivos

Los objetivos definidos para esta experiencia de laboratorio son:

- Comprensión del algoritmo de Fuerza Bruta.
- Creación de la solución para el problema planteado usando el algoritmo de fuerza bruta.
- Implementación de la solución en un programa creado en el lenguaje de programación C.
- Trabajar en pareja de manera efectiva.

2. Método

2.1. Recursos utilizados

Los equipos utilizados para el desarrollo y posteriores pruebas de la solución implementada en el lenguaje de programación C son:

- Intel Core i5 6600, 16GB de ram, S.O: Windows 10.
- Intel Celeron n4020, 4GB de ram, S.O: MX linux (Debian).

2.2. Procedimientos

2.2.1. Implementación del algoritmo

Para el problema descrito en la introducción, se proporciona un dataset en un archivo .csv el cual contiene toda la información de los paquetes que se ofrecen a la venta, con la información del beneficio monetario que deja cada paquete y el volumen en m^3 que ocupa. Para modelar el problema se implementaron dos estructuras principales: “Paquete” y “Combinación de Paquetes”. La estructura “Paquete” contiene los datos referentes al id, precio y volumen de un paquete, en cambio la estructura “Combinación de Paquetes” representa un conjunto de paquetes conformado por: un arreglo con los ids de los paquetes que conforman la combinación, peso total del conjunto, beneficio total del conjunto, largo del arreglo con los ids y última posición del último paquete agregado a la combinación. La figura 2 permite observar la idea de cómo se relacionan los paquetes y las combinaciones de una forma gráfica. Con las estructuras principales definidas, se realizan todas las combinaciones posibles mediante el siguiente algoritmo: los paquetes validados son transformados a una combinación de paquetes y son incorporados a una lista de combinaciones, luego cada una de las combinaciones de la lista trata de crear una nueva combinación agregando un paquete siempre y cuando no supere el volumen máximo permitido, al final de este proceso se tiene la creación de absolutamente todas las combinaciones válidas para el problema propuesto. A medida que se crean las combinaciones, a la par se verifica si esta nueva combinación tiene más beneficio que la actual combinación que otorga más beneficio, en caso de tener más beneficio, se reemplaza la antigua combinación de mayor beneficio por la nueva combinación de

mayor beneficio en la variable auxiliar que la almacena. Cuando ya no quedan combinaciones por crear, se retorna la combinación que más beneficio otorga, siendo esta la combinación que soluciona el problema.

El algoritmo implementado se encuentra detallado en la sección apéndice del presente informe.

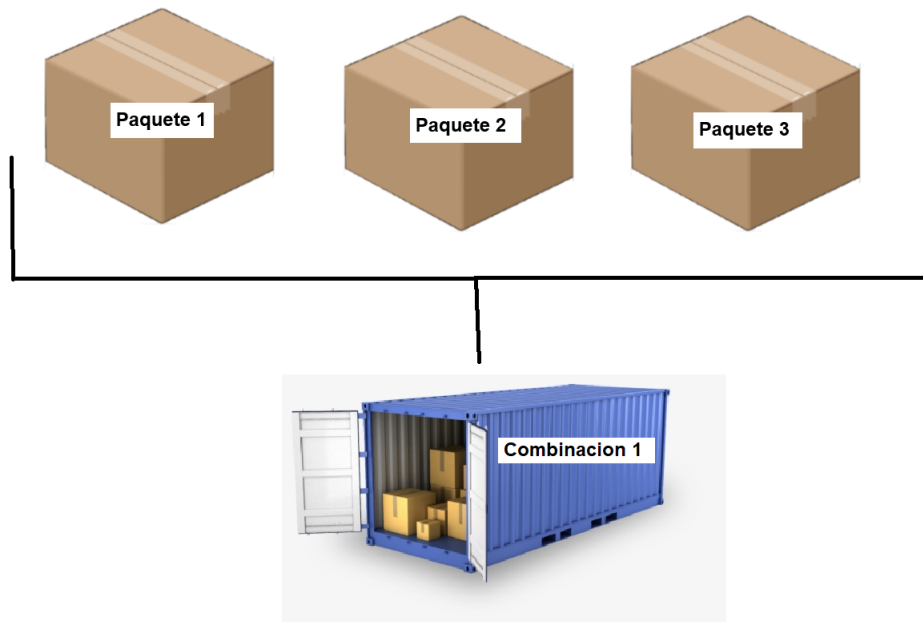


Figura 2: Relación entre Paquete y Combinación

2.2.2. Test de la implementación

Para probar la efectividad del programa se dispone de 5 datasets con distinta cantidad de paquetes, en cada uno de los datasets se indica la información relevante de los paquetes, señalando el id, volumen y beneficio de cada uno de los paquetes que posee el dataset. Para cada dataset se registra el tiempo de ejecución que toma el programa implementado en resolver el problema para los datos entregados.

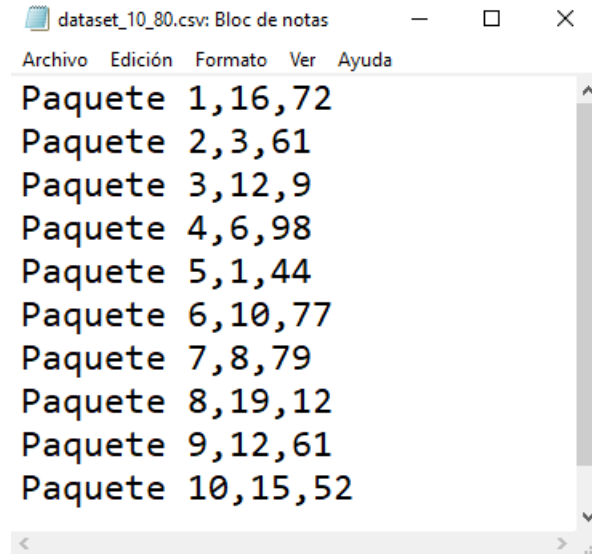


Figura 3: Ejemplo de un Dataset de 10 paquetes.

2.3. Resultados y análisis

Los resultados son presentados en ambos equipos mediante una tabla y gráfico con sus respectivos tiempos de ejecución en los que se resolvió el problema, para el equipo con sistema operativo Windows corresponden las figuras 4 y 5, para el equipo con sistema operativo Linux corresponden las figuras 6 y 7. Dadas las condiciones del problema el número de combinaciones posibles es $\sum_{i=0}^n \binom{n}{i} = 2^n$, sin embargo, al no considerarse una combinación con 0 elementos, la cantidad máxima de combinaciones es $2^n - 1$.

Además en esta sección se adjuntan los archivos de salida.txt generados por el programa que contienen la mejor combinación para los datasets de 10, 20 y 30 paquetes.

Cantidad de Paquetes	Tiempo de ejecución [s]
10	0,038
20	0,402
30	1.388,608
40	NULL
50	NULL

Figura 4: Tabla al testear en Windows

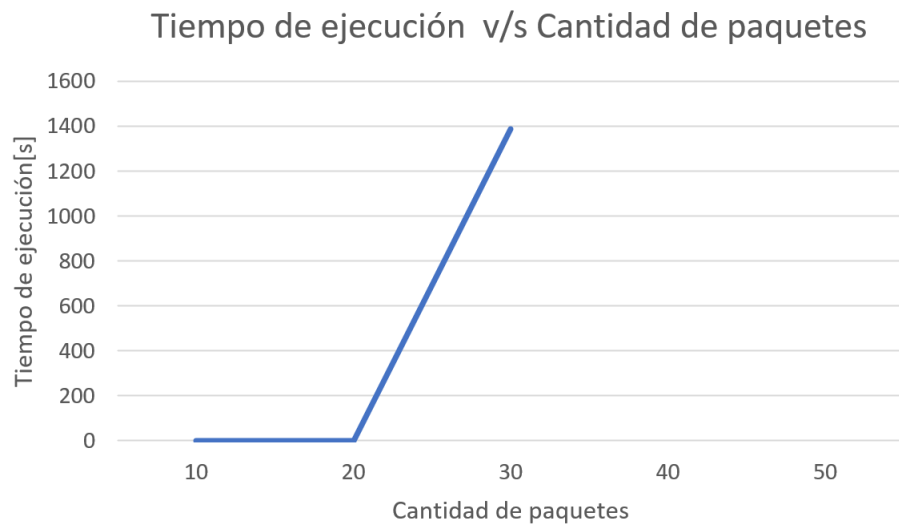


Figura 5: Gráfico de tiempo en Windows

Cantidad de Paquetes	Tiempo de ejecución [s]
10	0,001
20	0,105
30	7,730
40	NULL
50	NULL

Figura 6: Tabla al testear en Linux

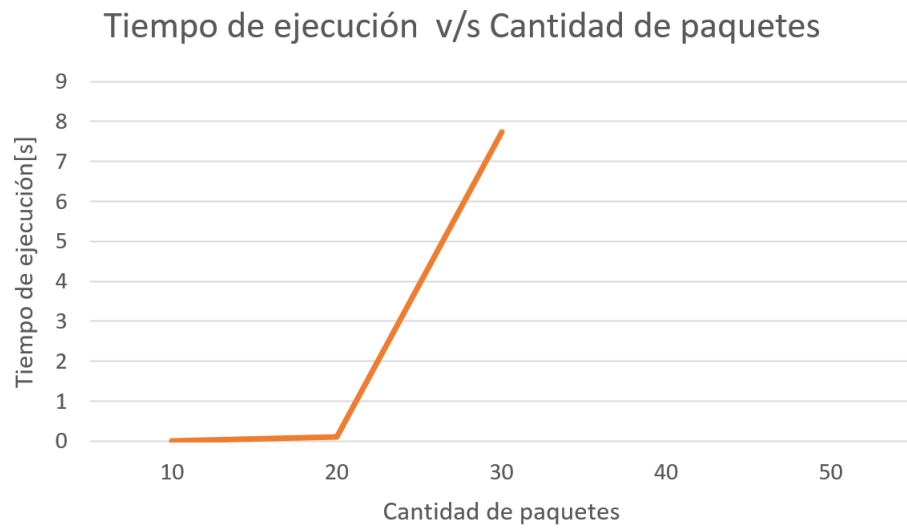


Figura 7: Gráfico de tiempo en Linux

```
salida.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
Paquete 1
Paquete 2
Paquete 4
Paquete 5
Paquete 6
Paquete 7
Paquete 9
Paquete 10
Beneficio: 544
Volumen: 71m3
```

Figura 8: salida.txt para el dataset de 10 paquetes

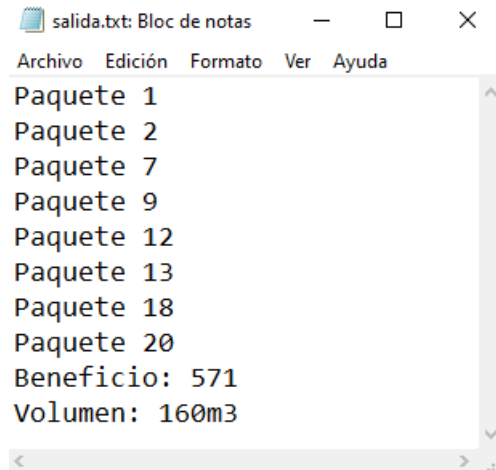


Figura 9: salida.txt para el dataset de 20 paquetes

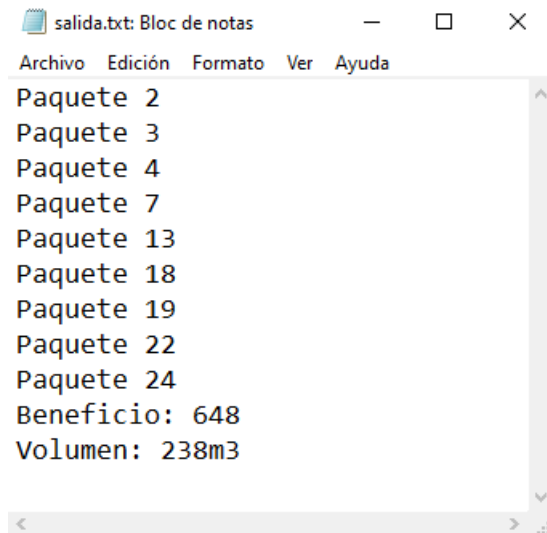


Figura 10: salida.txt para el dataset de 30 paquetes

3. Discusión

3.1. Resultados

Tomando en consideración los resultados conseguidos, es posible observar que se logró cumplir el objetivo de conseguir la mejor combinación cuando la cantidad de paquetes era de 10, 20 o 30, sin embargo, para una cantidad mayor a 30 paquetes, la solución implementada ya no es capaz de procesar la enorme cantidad de combinaciones que se crean y se congela el equipo hasta eventualmente cerrar a la fuerza la ejecución del programa, es debido a esto que en las tablas y gráficos de resultados solo se presenta información hasta 30 paquetes. Cabe destacar que, para el dataset con 30 paquetes, en el equipo con el sistema operativo Linux, solo tardó 7,7 segundos en llegar a la solución, mientras que para el mismo dataset y el mismo programa en el equipo con sistema operativo Windows 10 se tardó 23 minutos en encontrar la solución, a pesar de que el equipo con Windows 10 es de superior prestaciones de hardware que el equipo con Linux. Respecto al algoritmo de fuerza bruta implementado, es posible el observar que siempre entrega una solución, sin embargo, sus tiempos de ejecución y uso de recursos crecen de una manera exponencial mientras más cantidad de paquetes deba de analizar “En caso de que exista alguna solución al problema, un algoritmo de fuerza bruta siempre la hallará. Estos algoritmos son sencillos de implementar, pero muy ineficientes en lo que refiere a tiempo de cómputo o ejecución (algoritmos de orden exponencial en general), que perfectamente pueden dejar “colgado” el proceso en ejecución, por más veloces que sean los procesadores que se tenga.” [2] Entonces se puede determinar que la principal ventaja de un algoritmo de fuerza bruta es el que siempre encontrara una solución en caso de que esta exista y su principal desventaja es la ineficiencia que presentan en tiempos de ejecución y uso de recursos, lo que para grandes cantidades de datos puede causar el termino repentino de la ejecución causando el no lograr obtener la solución, como ocurrió para los datasets mayores a 30 paquetes.

3.2. Recomendaciones

Tomando los resultados en consideración, podría recomendarse un algoritmo de fuerza bruta para cantidades pequeñas de datos, ya que siempre se obtendrá una solución y se podría garantizar que la solución se obtendrá en una baja cantidad de tiempo. Para grandes cantidades de datos directamente resulta imposible la utilización del algoritmo de fuerza bruta, por lo que se recomendaría el uso de otro algoritmo.

4. Conclusiones

En el presente informe se efectuó el estudio del algoritmo de Fuerza Bruta a través del problema de la combinación de paquetes. Se cumplió el objetivo de lograr comprender el algoritmo de fuerza bruta, siendo capaces de destacar sus fortalezas y debilidades, además se cumplió el objetivo de idear una solución para el problema planteado utilizando el algoritmo de fuerza bruta, el cual luego fue implementado bajo la creación de un programa en el lenguaje de programación C. Las pruebas efectuadas en el programa arrojaron resultados satisfactorios logrando encontrar la solución, sin embargo, la cantidad de datos manejable se ve bastante comprometida al usar fuerza bruta, por lo cual, no fue posible determinar la solución para los archivos de 40 y 50 paquetes. En conclusión, se recomienda el uso de esta técnica cuando se tiene un bajo número de datos, en caso contrario se recomienda la búsqueda de otro algoritmo.

Para finalizar, el trabajo en grupo para la realización de este laboratorio fue muy efectivo, logrando una buena comunicación y distribución del trabajo, aparte permitió un primer acercamiento al desarrollo de una solución de manera conjunta.

5. Apéndice

5.1. Pseudocódigo implementado

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.

Por cada linea en el archivo:

Se lee peso de la linea (segundo digito)

Si peso < Capacidad Container:

Generar paquete segun estructura

Guardar paquete en arreglo inicial

Cerrar archivo

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

Si largo arreglo inicial == 0:

Mostrar por pantalla No hay solucion valida

Fin de Programa

-
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.

Se crea un arreglo de combinaciones vacio

Para cada elemento del arreglo inicial:

Crear Combinacion segun estructura usando el elemento

Guardar Combinacion en arreglo de combinaciones

Para cada elemento del arreglo de combinaciones:

Se lee el ultimo indice almacenado en la estructura de la combinacion

Para ultimo indice + 1 hasta largo de arreglo inicial - 1:

Efectuar suma entre volumen total de la combinacion y volumen del paquete

Si Suma Total < Capacidad Container:

Crear nueva combinacion tomando la combinacion antigua y agregandole el nuevo paquete

Guardar nueva combinacion en arreglo de combinaciones.

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

Se toma el primer elemento del arreglo de combinaciones como el IndiceMayorBeneficio

Se guarda el valor del beneficio del primer elemento del arreglo de combinaciones en la variable MontoBeneficioMayor

Para cada elemento del arreglo de combinaciones:

Si beneficio de la combinacion > MontoBeneficioMayor:

MontoBeneficioMayor = valor beneficio de la combinacion

IndiceMayorBeneficio = indice de la combinacion actual

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.

Se lee el valor de IndiceMayorBeneficio

Se consigue la combinacion con mayor beneficio a traves de su indice dentro del arreglo de combinaciones

Abrir archivo con nombre salida.txt

Para cada elemento dentro del arreglo de paquetes de la combinacion con mayor beneficio:

Leer numero de id del elemento actual

Escribir Paquete + id en archivo

Leer numero de beneficioTotal dentro de la combinacion

Escribir Beneficio : + beneficioTotal en archivo

Leer numero de volumenTotal dentro de la combinacion

Escribir Volumen : + volumenTotal en archivo

Cerrar archivo

Fin de Programa

5.2. Instrucciones de uso

El programa implementado busca una solución para cualquier dataset entregado por el usuario en tiempo de ejecución, para dar un dataset de forma correcta al programa, este debe encontrarse en la misma carpeta del ejecutable, además, el nombre del archivo debe contener el siguiente formato:

- dataset_**P**_**V**.csv.
- P: Cantidad de Paquetes que contiene el dataset.
- V: Volumen máximo del container.

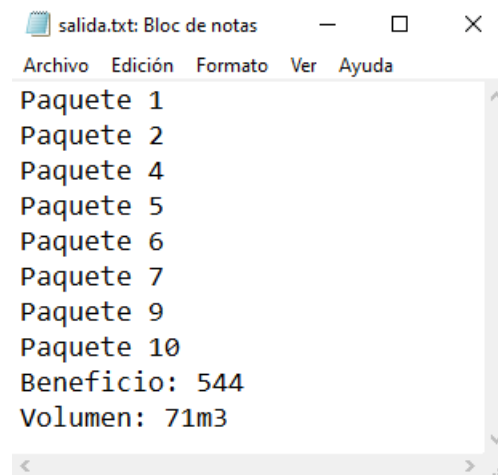
El programa finaliza cuando se indica por pantalla la mejor combinación de paquetes, dando a conocer la cantidad de tiempo en segundos que demora generar las combinaciones, el volumen

de la solución y beneficio de la solución. Además, se genera un archivo de salida “salida.txt” en el directorio del programa, el cual contiene los paquetes que conforman la mejor combinación, beneficio y volumen totales.

```
Ingrese el nombre del archivo, con la extension incluida, por ejemplo dataset_5_20.csv:
dataset_10_80.csv

Mejor combinacion final
Combinacion Beneficio:544, volumen:71, UPos:9, Largo: 8
Combinaciones: 1 ,2 ,4 ,5 ,6 ,7 ,9 ,10 ,
Tiempo de ejecucion: 0.001281
```

Figura 11: Ejemplo de ingreso de un dataset y termino de ejecución del programa.



```
salida.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
Paquete 1
Paquete 2
Paquete 4
Paquete 5
Paquete 6
Paquete 7
Paquete 9
Paquete 10
Beneficio: 544
Volumen: 71m3
```

Figura 12: Ejemplo de archivo de salida al termino de la ejecución del programa.

Referencias

- [1] S. Mawa. Algoritmos de fuerza bruta y backtracking. [Online]. Available: <https://sebamawa.wordpress.com/2017/07/09/fuerza-bruta-y-backtracking/>
- [2] ——. Algoritmos de fuerza bruta y backtracking. [Online]. Available: <https://sebamawa.wordpress.com/2017/07/09/fuerza-bruta-y-backtracking/>