



## Algoritmos Avanzados

### Laboratorio N°2: Método Goloso

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

10 de Junio de 2021

# Tabla de contenidos

<b>1. Introducción</b>	<b>1</b>
1.1. Contexto . . . . .	1
1.2. Motivación . . . . .	1
1.3. Teoría relevante . . . . .	1
1.4. Objetivos . . . . .	2
1.5. Estructura del informe . . . . .	2
<b>2. Método</b>	<b>4</b>
2.1. Recursos utilizados . . . . .	4
2.2. Procedimientos . . . . .	4
2.2.1. Implementación del algoritmo . . . . .	4
2.2.2. Test de la implementación . . . . .	5
2.3. Resultados y análisis . . . . .	6
<b>3. Discusión</b>	<b>9</b>
3.1. Resultados . . . . .	9
3.2. Recomendaciones . . . . .	10
<b>4. Conclusiones</b>	<b>11</b>
<b>5. Apéndice</b>	<b>12</b>
5.1. Pseudocódigo implementado . . . . .	12
5.2. Instrucciones de uso . . . . .	14
<b>Referencias</b>	<b>16</b>

# **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 goloso, el cual consiste en determinar el o los elementos más prometedores que cumplan una restricción dada dentro de un conjunto de elementos, hasta que se encuentre la solución. El como se determine estos elementos más prometedores depende netamente del contexto y la implementación del problema. Por ejemplo, para un árbol de expansión mínima que se puede

apreciar en la figura 1 se puede observar que los elementos prometedores fueron seleccionados con el criterio de elegir las aristas que presenten menos peso hasta que todos los nodos estén conectados. Los algoritmos golosos se pueden resumir como “Son algoritmos que siguen una heurística mediante la cual toman decisiones óptimas locales en cada paso, de manera muy eficiente, con la esperanza de poder encontrar un óptimo global tras una serie de pasos.” [1]

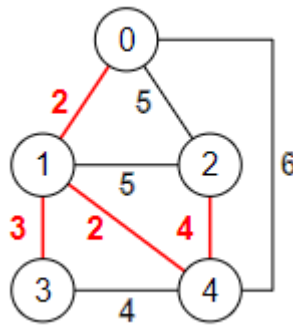


Figura 1: Ejemplo de selección de elemento prometedor

## 1.4. Objetivos

Los objetivos definidos para esta experiencia de laboratorio son:

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

## 1.5. Estructura del informe

Lo presentado hasta ahora ha sido la introducción del informe, a continuación se seguirá con la sección de Método, en la cual se detallaran los recursos utilizados y procedimientos para luego presentar los resultados con su respectivo análisis de estos. Posteriormente

se encontrara la sección de Discusión en la cual se se discutirá respecto a los resultados conseguidos al implementar el algoritmo y se darán recomendaciones respecto a lo observado. La cuarta sección corresponde a la sección de Conclusiones en la cual se listaran las conclusiones conseguidas de esta experiencia de laboratorio. La quinta sección corresponde al Apéndice, en el cual se adjuntaran anexos que ayuden a la comprensión del problema, algoritmo o de la solución implementada. Finalmente esta la sección de Referencias la cual contiene todas las fuentes que fueron citadas a lo largo del informe.

## 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 Xeon E5 2620 V2, 16GB de ram, S.O: Windows 10.

### 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 implementó la estructura: “Paquete”. La estructura “Paquete” contiene los datos referentes al id, precio, volumen y ponderación de un paquete, cabe destacar que la ponderación consiste en la razón entre el precio de un paquete y el volumen de este. La figura 2 permite observar la idea de cómo se representa un paquete de forma gráfica. Con la estructura paquete definida, se realiza la obtención del conjunto de paquetes más prometedores mediante el siguiente algoritmo: los paquetes validados son incorporados a un arreglo de paquetes, luego este arreglo es ordenado de mayor a menor según la ponderación de los paquetes, una vez ordenado se recorre el arreglo utilizando una variable auxiliar, la cual acumula el volumen de cada paquete que se va tomando según su índice hasta que este volumen acumulado supera el volumen máximo del container, al ocurrir esto se retorna el índice de tope, este índice tope representa el paquete limite hasta el cual no se supera el volumen máximo que soporta el container. Finalmente se escribe la solución escribiendo en un documento de texto plano, recorriendo el arreglo de paquetes hasta el índice de tope y por cada paquete escribir sus características.

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

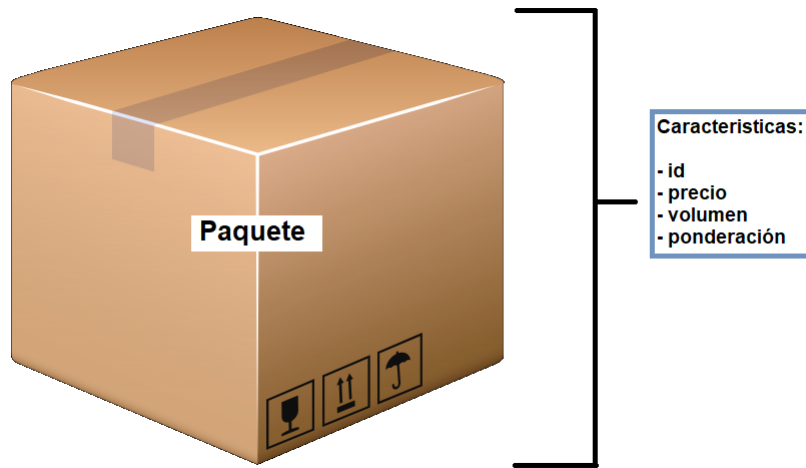


Figura 2: Paquete representado de forma gráfica.

### 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.

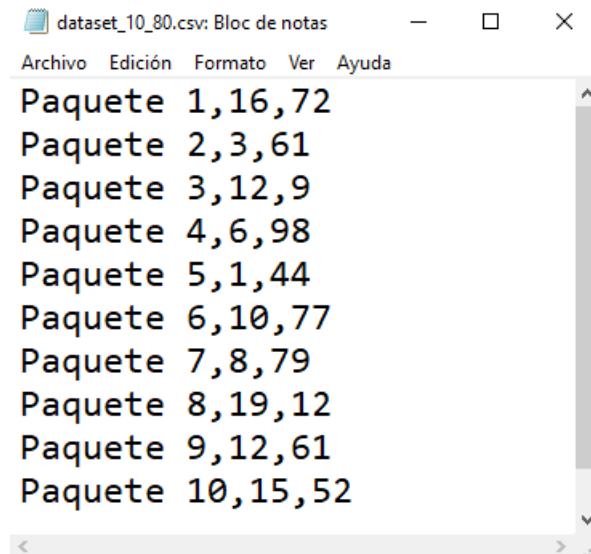


Figura 3: Ejemplo de un Dataset de 10 paquetes.

## 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 procesador **Intel core i5** corresponden las figuras 4 y 5, para el equipo con el procesador **Intel Xeon E5 2620 V2** corresponden las figuras 6 y 7. 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, 30, 40 y 50 paquetes.

Cantidad de Paquetes	Tiempo de ejecución [s]
10	0,002
20	0,002
30	0,002
40	0,002
50	0,002

Figura 4: Tabla al testear en el procesador Intel Core I5

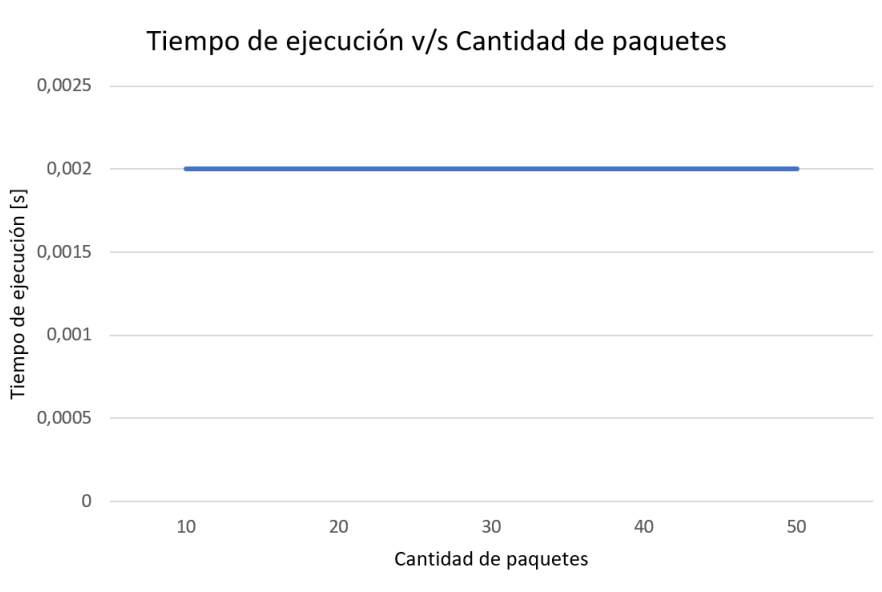


Figura 5: Gráfico de tiempo en el procesador Intel Core I5



Cantidad de Paquetes	Tiempo de ejecución [s]
10	0,001
20	0,008
30	0,003
40	0,006
50	0,005

Figura 6: Tabla al testear en el procesador Intel Xeon

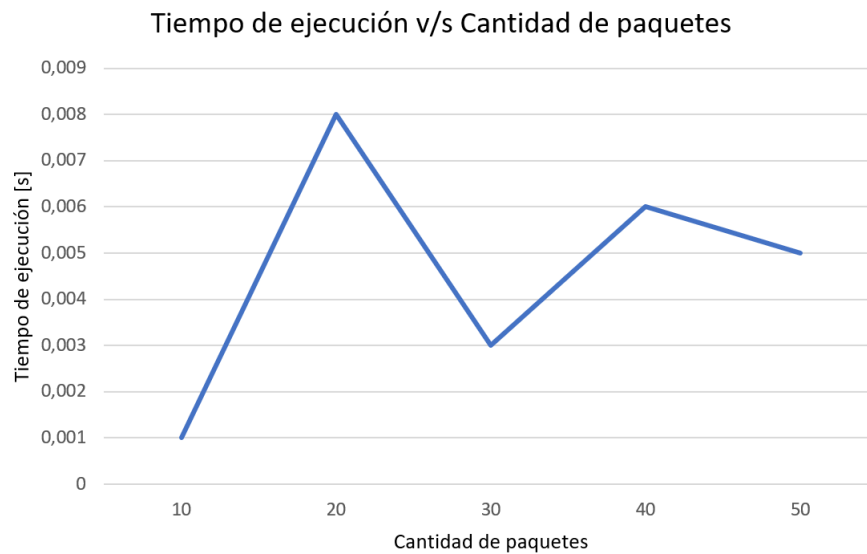


Figura 7: Gráfico de tiempo en el procesador Intel Xeon

```

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

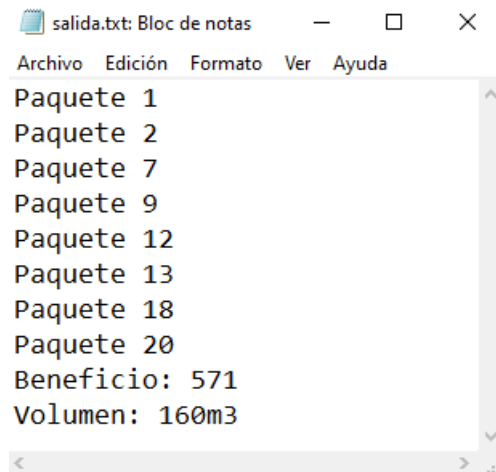


Figura 9: salida.txt para el dataset de 20 paquetes

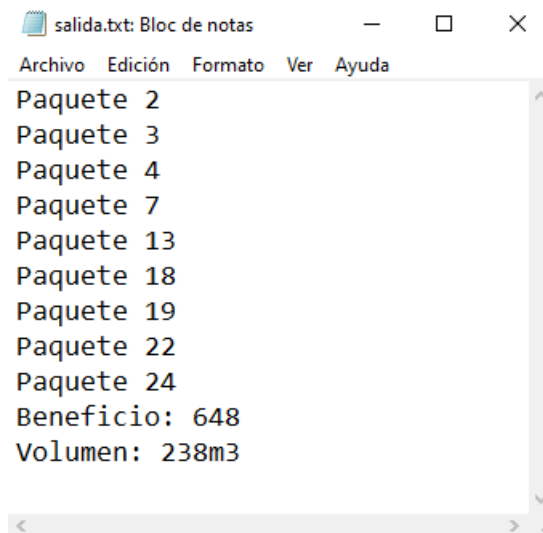


Figura 10: salida.txt para el dataset de 30 paquetes

## 3. Discusión

### 3.1. Resultados

Tomando en consideración los resultados conseguidos, es posible observar que se logró obtener una solución para todos los datasets sin mayor problema. Respecto al algoritmo goloso implementado, es posible observar que siempre entrega una solución en un tiempo acotado, sin embargo, este algoritmo no asegura el óptimo del problema, “Uno de los más versátiles y simples de entender es el denominado “algoritmo voraz” (Greedy algorithm). A pesar de que no siempre es capaz de encontrar una respuesta óptima, se lo utiliza con frecuencia dado que es muy rápido.” [2] Entonces se puede determinar que la principal ventaja de un algoritmo goloso es el que entregara una solución en un tiempo acotado y su principal desventaja es que no se puede asegurar que la solución entregada sea la mejor o la óptima. Realizando una comparación con la técnica de fuerza bruta/búsqueda exhaustiva que fue implementada en la sesión anterior de laboratorio, se puede apreciar que utilizando el método goloso se consiguió entregar una solución para todos los datasets entregados en tiempos muy acotados en comparación a fuerza bruta, por ejemplo para el tercer dataset de 30 paquetes, en la implementación de fuerza bruta tardo 23 minutos en arrojar un resultado, en cambio en esta implementación de goloso, solo tardo 2 milésimas de segundo. Este resultado refleja adecuadamente las diferencias de Orden de complejidad de ambos algoritmos, la implementación de fuerza bruta tenía una complejidad de  $O(n * 2^n)$  mientras que la implementación de goloso tiene un orden de complejidad de  $O(n * \log n)$ , en caso de tener datasets con cada vez más cantidad de paquetes, sería posible observar que el algoritmo de goloso crece de manera logarítmica. Sin embargo, el precio a pagar de goloso por esta rapidez se ve reflejado en la diferencia de las soluciones entregadas en comparación al algoritmo de fuerza bruta, mientras goloso entrega una solución aproximada, fuerza bruta siempre entregó la mejor solución para cada caso.

### **3.2. Recomendaciones**

Tomando los resultados en consideración, podría recomendarse un algoritmo de goloso para tanto grandes como pequeñas cantidades de datos, ya que se obtendrá una solución aproximada en una baja cantidad de tiempo. Sin embargo, es necesario tener en consideración que los resultados con este método pueden no ser el óptimo del problema, por lo que en caso de tener una cantidad pequeña de datos es recomendable ocupar el algoritmo de fuerza bruta para asegurar el mejor resultado.

## 4. Conclusiones

En el presente informe se efectuó el estudio del algoritmo de goloso a través del problema de la combinación y selección de paquetes. Se cumplió el objetivo de lograr comprender el algoritmo de goloso, 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 goloso, 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 soluciones aproximadas que pueden ser o no las óptimas, además fue posible encontrar solución para todos los datasets entregados para el testing. En conclusión, se recomienda el uso de esta técnica cuando se tiene un gran número de datos, debido a su rápida ejecución, en caso de que se tengan pocos datos se recomienda la búsqueda de otro algoritmo que pueda asegurar resultados óptimos.

Para finalizar, el trabajo en grupo para la realización de este laboratorio fue muy efectivo, logrando una buena comunicación y distribución del trabajo.

## 5. Apéndice

### 5.1. Pseudocódigo implementado

Pseudocódigo el programa de goloso:

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).

---

Nota: La estructura que se utilizar y que se le har referencia a lo largo del pseudocódigo es la siguiente:

```
struct Paquete{  
    int id; //id del paquete  
    int beneficio; //beneficio del paquete  
    int volumen; //volumen que ocupa el paquete  
    float ponderacion; //resultado de dividir beneficio entre volumen  
};
```

---

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

---

Se crea un arreglo de paquetes iniciales vaco

Por cada linea en el archivo:

Se lee volumen del Paquete de la linea (segundo dgito)

Si volumen Paquete < Capacidad Container:

Calcular ponderacin -> beneficio/volumen

Generar paquete segn 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 solución válida

Fin de Programa

---

8. Realizar un ordenamiento por Quicksort de la lista de paquetes de mayor a menor según su ponderación, usando la función nativa `qsort()`; disponible en la librería estándar de C `stdio.h`.

9. Debido a que la lista de paquetes se encuentra ordenada de mayor a menor según su ponderación, se calcula mediante el método goloso, el índice hasta el cual es posible incorporar paquetes a la solución final sin exceder el volumen máximo que soporta el container.

---

Se inicializa Volumen Acumulado igual a cero.

Para cada elemento del arreglo de paquetes iniciales:

Volumen Acumulado = Volumen Acumulado + Volumen elemento

Si Volumen Acumulado > Capacidad Container:

Retornar índice del ciclo -1

Retornar largoArreglo - 1 //Caso en el que todas los paquetes quepan en la solución

---

10. Se escribe un archivo de salida `.txt` que posee la información de la solución encontrada, la cual fue determinada en el paso 9 según el índice de tope de la lista de paquetes calculado, el cual no excede el volumen máximo del container. Se escribirán los paquetes de la solución encontrada, incluyendo el beneficio y el volumen que ocupa.

---

Abrir archivo con nombre `salida.txt`

Se inicializan las variables `beneficioTotal` y `volumenTotal` ambas en cero

Para cada elemento de la lista de paquetes inicial hasta indice de tope:

Leer numero de id del elemento actual

Escribir Paquete + id en archivo

beneficioTotal = beneficioTotal + beneficio elemento

volumenTotal = volumenTotal + volumen elemento

Escribir Beneficio : + beneficioTotal en archivo

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:

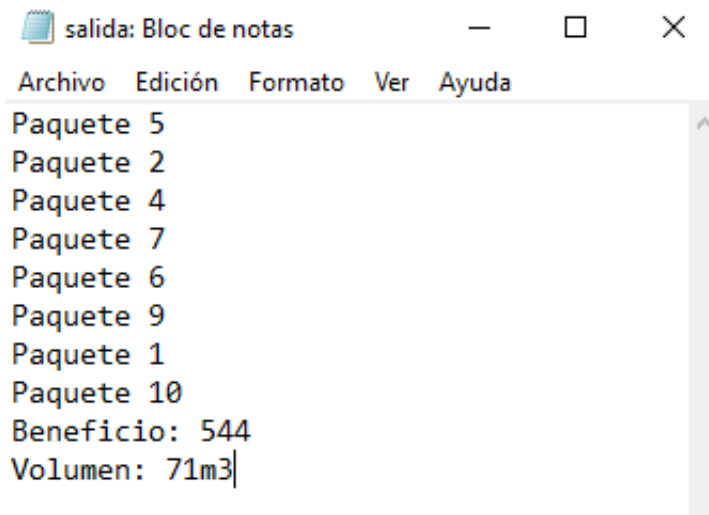
- dataset\_**P**\_**V**.csv.
- P: Cantidad de Paquetes que contiene el dataset.
- V: Volumen máximo del container.

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.



```
D:\Universidad\AA\Lab 2>l
Ingrese el nombre del archivo, con la extension incluida, por ejemplo dataset_5_20.csv:
dataset_10_80
El resultado es:
Beneficio: 544
Volumen: 71m3
Tiempo de ejecucion: 0.001000
```

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



salida: Bloc de notas

Archivo Edición Formato Ver Ayuda

Paquete 5  
Paquete 2  
Paquete 4  
Paquete 7  
Paquete 6  
Paquete 9  
Paquete 1  
Paquete 10  
Beneficio: 544  
Volumen: 71m3

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

## Referencias

- [1] U. R. J. Carlos. Algoritmos voraces. [Online]. Available: [https://www.cartagena99.com/recursos/alumnos/temarios/Algoritmos\\_voraces.pdf](https://www.cartagena99.com/recursos/alumnos/temarios/Algoritmos_voraces.pdf)
- [2] A. Palazzesi. El algoritmo voraz. [Online]. Available: <https://www.neoteo.com/el-algoritmo-voraz/>