Algoritmo Greedy aplicado a la resolución de un problema de biblioteca

NICOLÁS E. OLIVARES GONZÁLEZ[1,*]

[1] *Ingeniería Civil en Informática, Departamento de Ingeniería Informática, Universidad de Santiago de Chile*
[*] *Correo electrónico del autor: nicolas.olivares.g@usach.cl*

Compiled November 9, 2016

En el marco de la asignatura de Algoritmos Avanzados, impartida por el Departamento de Ingeniería Informática de la Universidad de Santiago, este artículo presenta el análisis de la solución para un problema presentado por el enunciado del laboratorio número 3 de la asignatura. Se aborda el uso de las técnicas de resolución de problemas en base a algoritmos.

OCIS codes:

http://dx.doi.org/10.1364/optica.XX.XXXXXX

1. INTRODUCCIÓN

Este artículo aborda los conceptos estudiados en cursos anteriores de algoritmos, se basa en el enfoque de resolución de problemas con el uso de algoritmo *Greedy*, *Voraz* o como es conocido generalmente *Goloso*.
Para el desarrollo de este laboratorio se utiliza el lenguaje de programación C y para la compilación del programa se trabaja en el ambiente de Linux.

2. DESPRICIÓN DEL PROBLEMA

En la biblioteca de Maipú, la bibliotecaria tiene la tarea de organizar libros en cajones para trasladar el material a una nueva dependencia que el alcalde de la comuna ha dispuesto. El problema de la bibliotecaria consiste en la manera de organizar las cajas para que se transporte la mayor cantidad de libros

3. MARCO TEÓRICO

Al momento de solucionar un problema, se plantean diversas maneras o metodos de resolución en base a las características del problema.
Para este problema se utiliza el algoritmo Greedy,

> "Un algoritmo Greedy elige, en cada paso, una solución local óptima. En general, son bastante sencillos de programar. ¿Desafortunadamente?, no siempre conducen al óptimo"[1]

4. DESCRIPCIÓN DE LA SOLUCIÓN

A. Entrada

La entrada porporcionada consta con las dimensiones *LARGO_MAX*, *ALTO_MAX*, *ANCHO_MAX*, que corresponden a las dimensiones del cajon que la bibliotecaria debe utilizar para introducir los libros.
Además se entrega el listado de libros que deben introducirse en el cajón.

B. Supuestos

Trabajando bajo el supuesto en que los libros que se introducen en un cajón son de alguna maner no rígidos, o bien líquidos y no se sobrelapan unos con otros.

C. IDEA

- Partiendo con el conjunto de libros disponibles y las dimensiones de la caja
- Se ordena el conjunto de acuerdo al volumen que ocupa, de menor a mayor. Sería el volumen el parámetro de maximización, es decir se busca meter la mayor cantidad de libros dentro del cajón para su transporte.
- Con dos conjuntos vacios, uno para los libros que quedan dentro de la caja y otro para los que se quedan afuera.
- Se recorre el conjunto ordenado verificando que sucede si se introdujera un elemento más al cajón,¿Supera la capacidad de volumen máximo?

- Se introduce el libro en los conjuntos antes mencionados, si cumple con la condición.
- De esta manera, y verificando todos los elementos del conjunto ordenado, el algoritmo debe entregar los conjuntos de libros en la caja y fuera de la caja.

Algorithm 1. Greedy

```
1:  procedure GREEDY(LIBROS, LargoMax, AltoMax, AnchoMax)
2:      LIBROS = ordenar(LIBROS)
3:      Caja = ∅
4:      Fuera = ∅
5:      Vmax = LargoMax * AltoMax * AnchoMax
6:      Vactual = 0
7:      for each  libro ∈ LIBROS do
8:          if Vactual + libro.volumen > Vmax then
9:              agregar(Caja, libro)
10:             Vactual = Vactual + libro.volumen
11:         else
12:             agregar(Fuera, libro)
13:     Vfinal = Vactual
14: return Caja, Fuera, Vfinal
```

5. ANÁLISIS DE LOS RESULTADOS

En esta sección se realizará un análisis del algoritmo implementado en base a la serie de preguntas habituales que se realizan para comprobar la validez de un algoritmo.

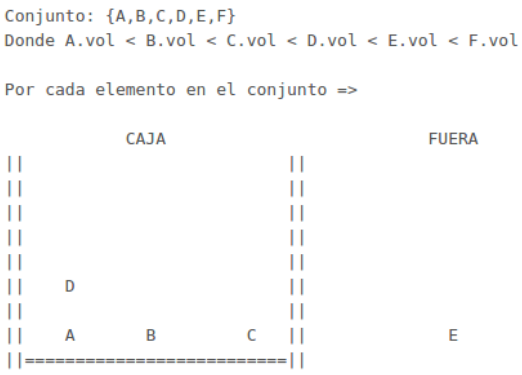
A. Análisis: Preguntas para el Algoritmo 1

1. ¿El algoritmo para en algún momento?
   Si, el algoritmo para al recorrer los libros en su totalidad.
2. Y si para, ¿lo hace con la solución?
   Este algoritmo entrega solo una solución local del problema.
3. ¿Es eficaz?¿Soluciona el problema?
   No es eficaz, porque no encuentra la solución óptima del problema en cualquier instancia.
4. ¿Es eficiente?
   Su orden de complejidad queda dado por la entrada n libros, de manera $O(n) + O(ordenamiento)$. Se requiere de n operaciones para recorrer el conjunto ordenado, pero el ordenar el conjunto, dependiendo del algoritmo de ordenamiento utilizado puede tomar distintos valores. En el caso de la implementación se usó algoritmo de burbuja, entonces $O(n^2)$. Se considera constante el agregar a los conjuntos Caja o Fuera.
5. ¿Se puede mejorar?
   Si, se puede mejorar cambiando el algoritmo usado para el ordenamiento.
6. ¿Existe otro método para solucionar el problema?
   Se puede aplicar algoritmo de optimización con la estrategia de Programación Dinámica.

6. TRAZA DE LA SOLUCIÓN

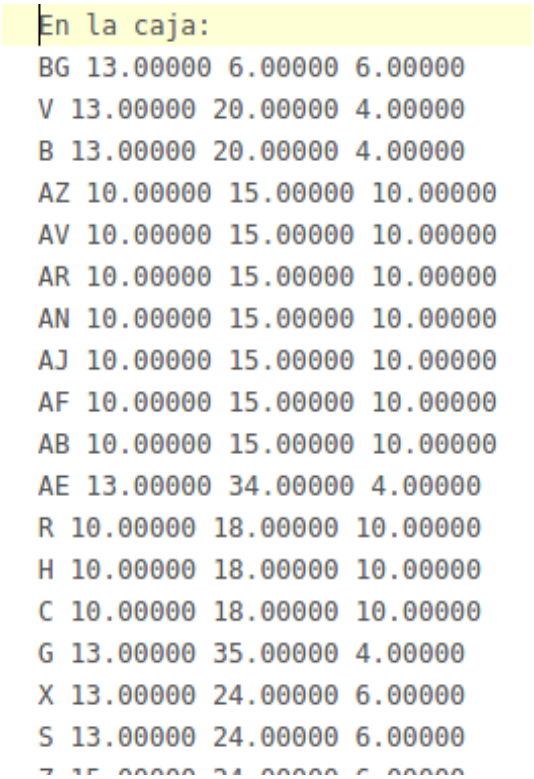
En la Figura 1 se observa una representación del algoritmo en donde se tiene el conjunto ordenado por volumen, y para cada elemento del conjunto se introduce dentro de la caja, desde los que tienen menor volumen a los con mayor volumen.
El libro E queda fuera de la caja porque si se agrega a la caja el volumen se sobrepasará.

Fig. 1. Explicación de la traza a grosso modo



En el programa generado, se guía el proceso, y en el archivo de salida se detalla la distribución de los libros.

Fig. 2. Demostración de algoritmo 1



Para el Algoritmo 1 se pueden revisar los resultados en el archivo de salida.

Fig. 3. Demostración de algoritmo 1

```
Fuera de la caja:
N 459.00000 513.00000 6.00000
BK 652.00000 688.00000 6.00000
BE 724.00000 634.00000 6.00000
BD 1263.00000 634.00000 6.00000
BF 1333.00000 634.00000 6.00000
AU 1322.00000 2440.00000 4.00000
BL 5523.00000 63444.00000 6.00000
Espacio libre dentro de la caja:
368266 centímetros cúbicos
```

7. CONCLUSIONES

Tras la realización del laboratorio y análisis de resultados, se concluye los siguiente:

- El algoritmo enstrega una solución a la bibliotecaria más no la óptima.
- Se puede mejorar el algoritmo aplicando otros algoritmos de ordenamiento, lo que reduciría la cantidad de operaciones.
- Existen métodos más eficientes para la obtención de una solución óptima. Se puede utilizar Porgramación Dinámica.
- Este problema es similar al de la mochila, basados en al optmización, un problema con restricciones de máximos y que se puede descomponer recursivamente y se operan los beneficios de los subconjuntos.

REFERENCES

1. D. Zanarini, "Algoritmos greedy," (2009).