Universidad de Santiago de Chile
Departamento ingeniería informática
Ingeniería civil informática
Algoritmos Avanzados
Laboratorio

02-2018

Fuerza bruta

Catalina Andrea Morales Rojas
e-mail: catalina.morales.r@usach.cl

Brute force

RESUMEN: *Para lograr resolver problemas se cuenta con una gran variedad de algoritmos, uno de los más conocidos corresponde a fuerza bruta. A continuación se presenta el análisis e implementación del problema de encontrar la inversión que genera la mayor utilidad en base a una lista ingresada mediante el uso de esta técnica.*

PALABRAS CLAVE: fuerza bruta, complejidad algorítmica.

ABSTRACT: *To solve problems, there is a great variety of algorithms, one of the best known corresponds to brute force. Below is the analysis and implementation of the problem of finding the investment that generates the greatest utility based on a list entered through the use of this technique.*

KEYWORDS: *brute force, algorithm complexity.*

1 INTRODUCCIÓN

Fuerza bruta es un algoritmo sencillo de implementar y que entrega la solución siempre que esta exista. Es por esto que a continuación se expone la implementación realizada a un problema que debe entregar como resultado la mejor combinación en base a una serie de parámetros.

El presente documento busca analizar las ventajas y desventajas que presenta este algoritmo, junto con el desarrollo realizado en la resolución del problema.

2 DESCRIPCIÓN DEL PROBLEMA

El problema consiste en encontrar una combinación de inversiones (en base a lo ingresado en el archivo de entrada), que cuente con la mayor utilidad y que su costo total sea menor o igual al que la empresa que solicita el algoritmo está dispuesto a pagar.

3 MARCO TEÓRICO

El principio de fuerza bruta es generar todas las combinaciones en base a una serie de entradas, para luego, usando un criterio descartar aquellas que no son soluciones aceptadas.

Debido a que son generadas todas las combinaciones posibles, el algoritmo no es eficiente, ya que a mayor número de variables, mayor será el costo de ejecución.

Por otro lado, fuerza bruta siempre entrega la solución óptima en caso de existir una y es sencillo de implementar.

4 DESCRIPCIÓN DE LA SOLUCIÓN

El algoritmo descrito a continuación fue implementado utilizando fuerza bruta mediante el lenguaje de programación C.

4.1 IDEA

La idea principal es generar todas las combinaciones, y luego encontrar aquella que cumpla con que el monto total es menor al monto que tiene la empresa disponible para realizar la compra, y que además genere la máxima utilidad.
Cada combinación puede tener desde 1 a n inversiones (siendo n la cantidad de inversiones ingresadas en el archivo de entrada). El orden en que son entregadas las inversiones no es importante, por lo cual el algoritmo es optimizado y no entregará combinaciones que cuenten con distinto orden y las mismas inversiones.
Debido a que la representación de números binarios permite generar todas las posibles combinaciones, se busca obtener las combinaciones, en donde un 1 representa que la variable si se utiliza y un 0 que no. Por ejemplo, si se tiene como entrada los costos L= {100,200,400}, se generan 6 combinaciones:

0 = 000 = “”
1 = 001 = “100”
2 = 010 = “200”
.
.
.
7 = 111 = “100 200 400”
A medida que se genera la combinación, las utilidades y el costo serán sumados, para luego comparar y elegir aquella combinación que no sobrepase el costo inicial y que genere el máximo de utilidad.
Para calcular la cantidad de veces que se debe realizar este algoritmo se utiliza:
2^n, en donde n corresponde a la cantidad de inversiones.

4.2 REPRESENTACIÓN Y ESTRUCTURA

La información es manejada mediante dos arreglos estáticos, los cuales se encargan de almacenar las inversiones y utilidades que se ingresan en el archivo de entrada.
Además, a medida que se realizan todas las combinaciones, aquella que genere la mayor utilidad es almacenada en otro arreglo.

4.3 PSEUDOCÓDIGO

```
bruteForce(cantidad, capitalInicial,
costos[],utilidadEntrada[])
cantidad = 2^(cantidad + 1) - 1
utilidad = 0
costo = 0
CostoTotal = 0
utilidadMaxima = 0

Mientras la cantidad > 1
        aux = cantidad
        valor = 0

        j = 0
// Se vacían los valores de la tabla de
combinaciones para almacenar las nuevas
combinaciones.

        Mientras j < cantidad
                combinaciones[j] = 0
                j = j + 1
//Se calcula el valor binario de cada valor y en
base a esto se obtienen las combinaciones
        Mientras aux > 1
                módulo = aux%2
                Si modulo = 1 entonces
                        utilidad = utilidad +
utilidadEntrada[valor]
                        costo = costo +
costos[valor]
                        combinaciones[valor] =
costos[valor]

                valor = valor + 1
                aux = aux/2
                Si aux = 1
                        utilidad = utilidad +
utilidadEntrada[valor]
```

```
                    costo = costo +
costos[valor]
                    combinaciones[valor] =
costos[valor]
                    valor = valor + 1
//Luego de generar cada combinación se
compara con el máximo anterior y con el
costo, si la combinación es mejor, se
reemplaza
        Si costo <=capitalInicial y utilidad >
utilidadMaxima entonces
                combinación = cantidad
                utilidadMaxima = utilidad
                CostoTotal = capital

Retornar combinación con la mayor utilidad
```

4.4 TRAZA

Entrada: Costos = {100,200}
Utilidad = {20,10}
Costo inicial = 300
cantidad = 2

Cantidad = 4
Utilidad = 0
Costo = 0
Costo total = 0
utilidad Máxima = 0

Número binario: 00
Costo: 0
Utilidad: 0
Combinación = " "

Número binario: 01
Costo: 100
Utilidad: 20
Combinación = "100"

Número binario: 10
Costo: 100
Utilidad: 10
Combinación: " 200"

Número binario: 11
Costo: 300
Utilidad: 30
Combinación: " 100 200"

Retornar combinación: "100 200"
Utilidad final = 30
Costo final = 300

4.5 ORDEN DE COMPLEJIDAD

La función bruteForce funciona en base a dos ciclos anidados, en donde el primer ciclo se realiza según la cantidad de combinaciones que se pueden generar (2^n, siendo n la cantidad de inversiones ingresadas en el archivo de entrada), y el segundo con respecto a la cantidad de veces que se debe iterar para transformar cada número decimal en binario (m). Por lo tanto, la complejidad del algoritmo es de:

$$\sim O(2^n)$$

5 ANÁLISIS DE LA SOLUCIÓN

5.1 ANÁLISIS DE IMPLEMENTACIÓN

1-¿Se detiene?
El algoritmo se detiene cuando genera todas las combinaciones de inversiones.

2-¿Entrega solución?
Entrega como resultado la primera combinación de inversiones encontrada con la mayor utilidad.

3-Eficiencia

La complejidad del algoritmo es de orden ~ $O(2^n)$, la cual no es eficiente, ya que no es polinómica.

4-¿Se puede mejorar?

El algoritmo se puede mejorar si además de elegir el algoritmo con la mayor utilidad entregará aquel que posee un menor costo.

5-¿Otra idea?

Se puede realizar el mismo algoritmo mediante backtraking, de manera que solo se generan aquellas combinaciones que tienen un costo menor al costo inicial del problema.

5.2 EJECUCIÓN

Para la ejecución del programa se debe tener instalado el compilador GNU gcc.

- Para compilar el programa:

```
$ gcc lab_avanzados.c funciones.c  -o lab
```

- para ejecutar el programa:

```
$ ./lab
```

El programa solicitará ingresar el nombre del archivo que contiene la entrada, junto con su extensión:

```
$ Ingrese el nombre del archivo: entrada.txt
```

Para poder entrar al modo debug:

```
$ gcc lab_avanzados.c funciones.c -DDEBUG -o lab
```

6 CONCLUSIONES

A partir del trabajo desarrollado es posible concluir que el algoritmo de fuerza bruta (o también conocido como enumeración explícita) es un buen algoritmo cuando se deben generar combinaciones con una cantidad reducida de variables, ya que permite encontrar el óptimo de la solución (si es que existe una) y es sencillo de implementar.
Por otro lado, el tiempo de ejecución del algoritmo aumenta de manera exponencial cuando lo realizan la cantidad de variables involucradas, como en este caso, en donde la complejidad fue de $O(2^n)$, por lo cual no es una buena estrategia para cuando se necesite eficiencia en la resolución de un problema.