# Práctica 3. “Algorimos voraces y programación dinámica”

Profesor:Miriam Pescaor Rojas

Departamento de Ciencias e Ingeniería de la Computación,

Análisis de Algoritmos, ESCOM-IPN
Alumnos: Cazares Cruz Jeremy Sajid
Alvares Aguilar Jesús
Grupo: 3CM7

October 19, 2021

## 1 Introducción

Los algoritmos voraces suelen ser muy eficientes sin embargo existen problemas donde no encuentran una solución óptima. Una alternativa es el uso de técnicas de programación dinámica

## 2 Marco teórico

### 2.1 Algoritmos voraces

Un algoritmo voraz es una estrategia de búsqueda por la cual se sigue una heurística consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima.
Los algoritmos voraces suelen ser bastante simples. Se emplean sobre todo para resolver problemas de optimización, como, por ejemplo, encontrar la secuencia óptima para procesar un conjunto de tareas por un computador, hallar el camino mínimo de un grafo, etc.

### 2.2 Algoritmos dinamicos

La Programación Dinámica nos permite resolver un problema hallando soluciones sucesivas a sub-problemas de menor tamaño y ligándolas como solución optima del problema. Consiste en solucionar el presente suponiendo que en cada etapa futura siempre se tomaran las decisiones correctas.

# 3 Problemas a resolver

## 3.1 Problema de devolver el cambio

El problema consiste en desarrollar un algoritmo para pagar una cierta cantidad de dinero a un cliente, empelando el menor número posible de monedas.

**Algoritmo 1:** Algoritmo voráz de devolver el cambio

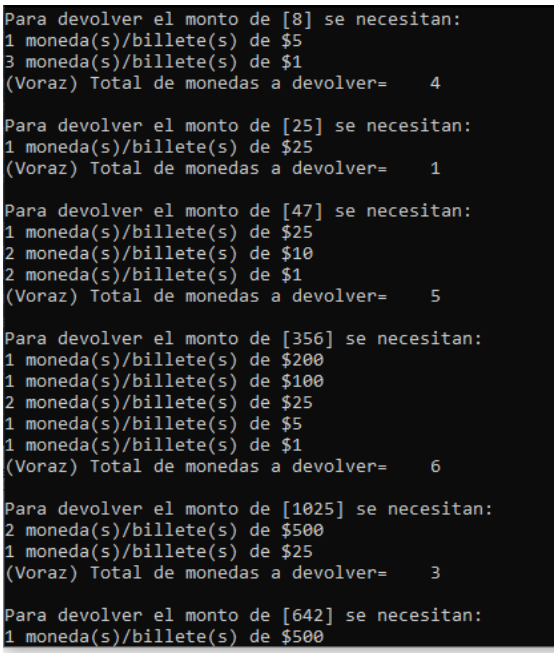
```
   Data: D: denomicaciones de las monedas y/o billetes
1  begin
2      DevolverCambio(n)
3      S ← ∅, S es el conjunto que obtendrá la solución;
4      s ← 0, la suma de los elementos en S;
5      while s ≠ n do
6          x ← el mayor elemento de D tal que x + s ≤ n.
7          if no existe el elemento then
8              devolver no "NO ENCUENTRO SOLUCIÓN";
9          else
10             S∪ moneda de valor x;
11             s ← s + x;
12     Devolver S;
```

| Denominaciones | Devolver | | | | | |
|---|---|---|---|---|---|---|
| | 8 | 25 | 47 | 356 | 1025 | 642 |
| D1 = {500, 200,100, 25, 10, 5, 1}<br>D2 = {8,6, 4, 1} | | | | | | |

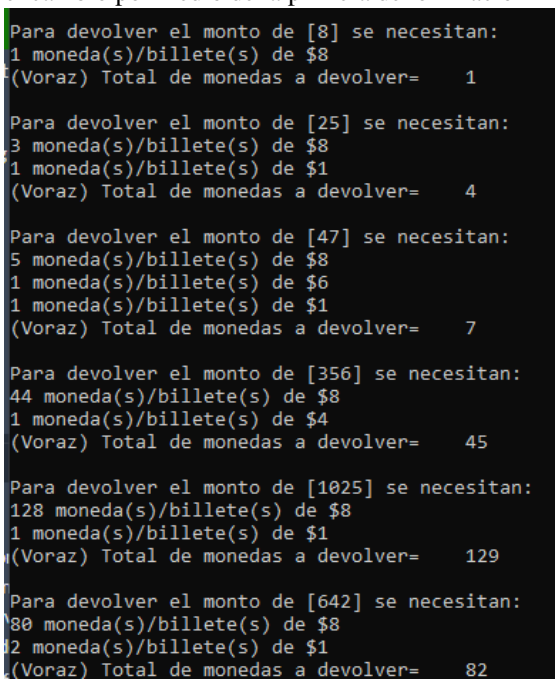
Para la solución mediante algoritmo voraz se tomó en cuenta desde el elemento mayor al menor, siendo que este número no fuera mayor al valor que devolver, de tal manera que se realizaba una constante resta a N que sería el dinero para devolver así hasta llegar al resultado, sin embargo, en algunos casos no es el óptimo con un ejemplo como que existan una denominación d={1,3,4} donde para obtener el número 6 iniciando con la denominación de mayor rango el cuál sería 4 quedaría como {4,1,1} donde la solución óptima al problema sería {3,3} teniendo que dar una menor cantidad de monedas

La solución óptima se alcanza mediante el algoritmo dinámico el cuál es mediante la utilización de una tabla $C[1, \ldots n], [0, \ldots, N]$ ($C[i, j]$) donde, como ya se mencionó con anterioridad, i representa la denominación de las monedas y j la cantidad por pagar, siendo así capaz de no usar una misma denominación en la mayoría de los casos

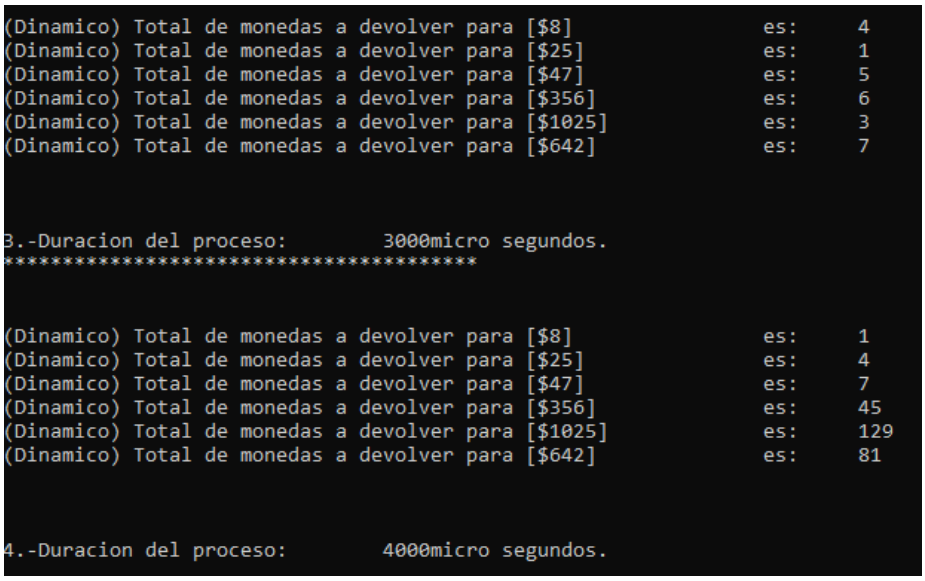
Para la solución en código se utilizó tres archivos. texto dos para las denominaciones y uno para los montos a pagar de tal manera que se pudiera automatizar el proceso y arrojar los resultados sin necesidad de ingresar datos y denominaciones, tal como se requería en la práctica, a continuación, se verán algunos resultados




En esta imagen se puede ver el total de monedas correspondientes y las usadas para dar el cambio por medio de la primera denominación




En esta segunda imagen se observa el cambio de las monedas mediante la segunda denominación



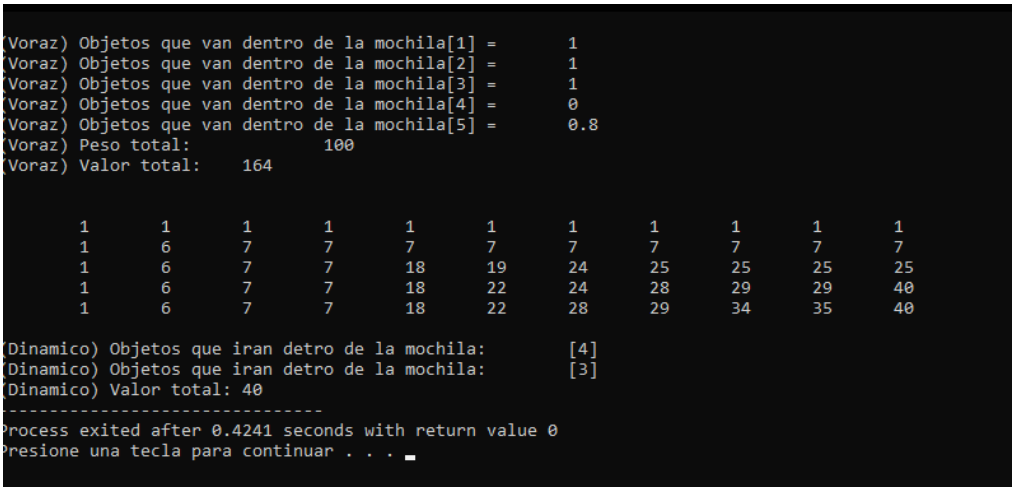

En esta tercera imagen además de observarse los tiempos se observa el total de monedas que regresa el algoritmo dinámico
Como en el caso de la cantidad de monedas para devolver la cantidad de 642 el algoritmo voraz determino que se necesitaban 82 monedas, mientras que el algoritmo dinámico un total de 81 monedas, siendo un cambio pequeño pero importante

## 3.2 El problema de la mochila fraccionaria

Nos dan $n$ objetos y una mochila. Para $i = 1, 2, \ldots, n$, el objeto $i$ tiene un peso positivo $w_i$ y valor positivo $v_i$. La mochila puede llevar un peso que no sobrepase $W$. Nuestro objetivo es llenar la mochila de tal manera que se maximice el valor de los objetos transportados, respetando la limitación de capacidad de la mochila. En una versión del problema se puede suponer que se puede tomar un trozo de los objetos, de manera que podamos decidir llevar solo una fracción $x$ de estos (la versión del problema que no permite fraccionar objetos es más compleja). En este caso el objeto $i$ contribuye en $x, w$, al peso total de la mochila, y en, $x, v$ al valor de la carga.

La mochila fraccionaria es más fácil de resolver de forma voraz por la razón de que los objetos pueden fraccionarse como su nombre indica de tal manera que las n combinaciones posibles siempre iniciaran con el objeto de mayor peso y el ultimo objeto será una fracción de este, así como no es necesario llegar al caso óptimo debido a que la mochila al poder tomar objetos fragmentados siempre llegará al límite del peso y valor sin importar por donde empiece el algoritmo a considerar
Por otra parte la mochila entera, el cual fue resuelto por medio de algoritmo dinámico, debe ser capaz de encontrar la solución dada en el texto la cual a su vez la solución óptima, los objetos: $\{3,4\}$ en cuanto peso donde el límite es 11 unidades, siendo así que los objetos tienen un peso de $\{5,6\}$ dando como resultado el límite del peso o bien por medio de los valores de los objetos el cual el límite es 40 y los objetos correspondientes son los de valor $\{18, 22\}$ llegando al límite de valor 40, el cual define como

la solución óptima al problema, estos valores son la solución a llegar debido que solo se toman dos objetos, siendo así lo más óptimo, se llena por completo la mochila sin necesidad de algún otro objeto o fracción del mismo y no sobrepasa los límites de valor y peso del planteamiento del problema
Al igual que en el problema de devolver el cambio los algoritmos dinámicos nos dan la solución óptima del problema, así como la solución a la mochila se realizó de la misma forma mediante el uso de tablas en la cual nos ayuda a llegar a la solución teniendo en cuenta los pesos y valores de los objetos, así como su resultado en caso de tomar el objeto como se puede observar en la imagen siguiente:




# 4 Resultados obtenidos

Mediante la aplicación de los algoritmos voraces y dinámicos se pudo observar la diferencia entre ambos tipos de algoritmos de tal manera que la programación dinámica ofrece la solución más optima al problema dado, como ya se pudo observar en el ejemplo dado de devolver el cambio el algoritmo voraz si da una solución al problema, pero esta no siempre es la óptima

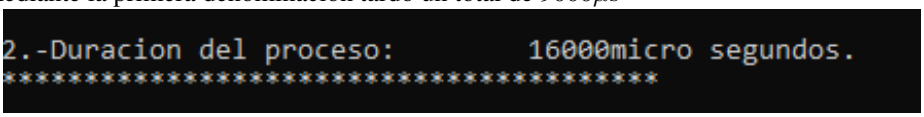
Esto también se observó mediante la solución de la mochila entera y fraccionaria los cuales reafirman que el algoritmo dinámico da resultados óptimos y los voraces no siempre llegan a tener esta cualidad. En cuanto tiempos mediante la ejecución se vieron tiempos bastantes parecidos de tal manera que en algunos casos solo llega a variar por unos microsegundos

## 4.1 Tiempos

En los algoritmos voraces y dinámicos para la solución de devolver el cambio se nota que los tiempos entre los tipos de algoritmos varía de una forma un poco significante de tal manera que se logra ver que en el caso de los algoritmos dinámicos con ambas

denominaciones llegan a ser más rápidos que los algoritmos voraces, esto se puede apreciar de mejor manera en las siguientes imagenes:


Para este primer caso se tiene que el proceso de devolver los 6 valores por método voraz y mediante la primera denominación tardo un total de 9000$\mu$s

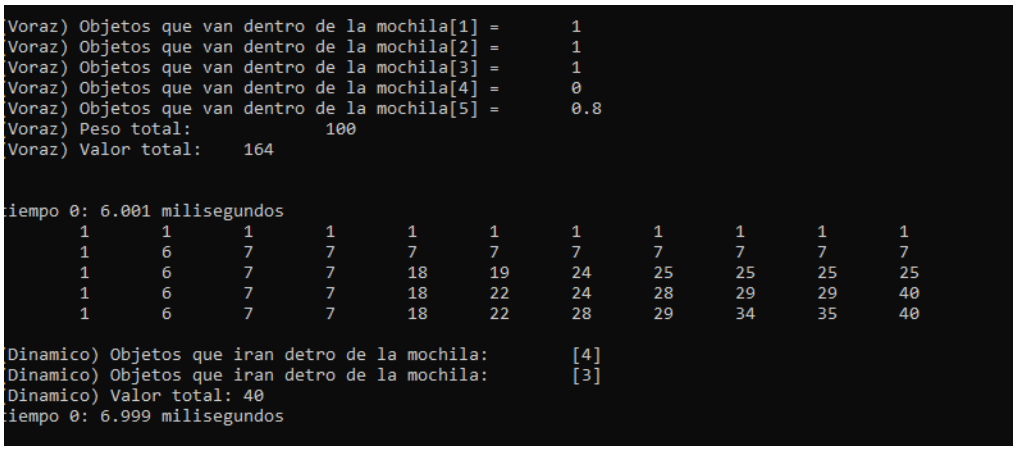



En el segundo caso donde se inició de igual manera por método voraz pero con la segunda denominación se obtuvo un tiempo de 16000$\mu$s


Finalmente, para su forma dinámica se tiene que en la denominación 1 un total de 3000$\mu$s, siendo 6000$\mu$s más rápido que el método voraz ya antes visto

Y en el caso de la segunda denominación tardando un total de 4000$\mu$s siendo 4 veces más rápido por la forma dinámica en algunos casos usando menos monedas

En el caso del problema de la mochila los tiempos si llegan a variar entre dinámico y voraz de tal manera que el dinámico llega a tardar caso en su mayoría 1 milisegundo como se puede a continuación:




# 5 Conclusiones

Los algoritmos dinamicos dan la solución óptima a diferentes problemas planteados asi como tienden a tener una mayor rapidez de respuesta ante estos problemas, de tal manera que es mejor la implementacion del un algoritmo dinamico a uno voraz para llegar a la respuesta más certera