Laboratorio N°4: Programación Dinámica

ANDRÉS MUÑOZ

Profesora: Mónica Villanueva

Ayudante: Patricio Vargas

Compiled June 5, 2017

En la siguiente investigación se desarrolla la técnica de resolución de problemas denominada *Programación Dinámica*. Con esta técnica se busca realizar la menor cantidad de pasos para entregar una solución pedida, lo que implica que el tiempo de ejecución sea menor en comparación a otras técnicas para resolver problemas. Esta técnica es aplicada a un problema en particular que consiste en indicar el efectivo mínimo que debe ser entregado por un cajero a un cliente. Además, se implementan esta técnica en lenguaje de programación C para ejemplificar y analizar sus características y comportamiento para este problema.

1. INTRODUCCIÓN

En la vida cotidiana, el ser humano intenta realizar la menor cantidad de acciones para resolver un determinado problema, como por ejemplo cuando una persona necesita realizar diferentes trámites en distintas partes de una ciudad, esta persona tratará de ir de lugar en lugar de tal forma que la distancia recorrida sea la menor para posteriormente volver a su casa.

En este informe se implementará un tipo de algoritmo, el cual será comparado con los otros vistos en las entregas anteriores. Para llevar a cabo esta comparación, se analizará el tiempo de ejecución que demora en entregar una solución en función de una entrada.

En el segundo capítulo, se da a conocer el problema planteado, explicando las diferentes restricciones. Luego en el tercer capítulo, se darán a conocer los diferentes conceptos técnicos necesarios para entender a profundidad lo planteado en este documento. En el cuarto capítulo se describe la solución que se dará al problema, aquí es donde se explica cómo se implementa un algoritmo Goloso y su tiempo de ejecución para el problema planteado. Posteriormente en el quinto capítulo se analizan los resultados obtenidos, esto se refiere a evaluar contestando unas series de preguntas el algoritmo implementado anteriormente, así como también compararlo con el algoritmo implementado en la experiencia anterior. En el siguiente capítulo se muestran tablas las cuales representan una traza de la implementación para un caso particular. Finalmente se dará a conocer lo objetivos logrados, además de interpretar lo investigado e implementado a lo largo de este proceso.

2. DESCRIPCIÓN DEL PROBLEMA

El banco regional de Concepción tiene problemas de atención al público y sus clientes se quejan constantemente de la lentitud con que avanzan las filas de las cajas.

Las quejas siempre hacen que una institución pierda prestigio y a la vez, no sea recomendado por los mismos clientes. Esto conllevaría a que muchos clientes se cambien de banco prefiriendo uno con mejor servicio de atención al público.

Al parecer el problema de la lentitud en la que avanzan las filas son los cajeros, ya que no son lo suficientemente rápidos para calcular el vuelto que deben entregar después de un pago y además no siempre lo hacen correctamente, es decir, muchas veces devuelven una cantidad errónea de dinero, lo cual provoca peleas y retrasos en la atención.

3. MARCO TEÓRICO

- **Lenguaje de programación:** Es un lenguaje formal diseñado para realizar procesos que pueden ser llevados a cabo por máquinas como las computadoras.
- **Algoritmo:** Conjunto ordenado y finito de operaciones para hallar la solución de un tipo de problemas. Un algoritmo puede ser implementado en diferentes lenguajes de programación.[1]
- **Método programación dinámica:** Es un método de resolución de problema, el cual busca entregar la solución del problema en la menor cantidad de pasos, para esto se descompone el problema, luego se expresa la solución a partir del o los problemas descompuestos, posteriormente se identifica el orden de resolución de estos problemas y finalmente se identifica la estructura de datos para llevar a cabo esta implementación del método.Para entender de mejor manera la técnica consultar apuntes [2]

4. DESCRIPCIÓN DE LA SOLUCIÓN

La solución para disminuir las quejas de los clientes, es diseñar e implementar un programa en el que se indique a los cajeros el vuelto que deben entregar y la cantidad mínima de efectivo para hacerlo.

El programa realizado, cumple con el sistema monetario con el cual rige la sucursal, el cual es el chileno. Además, el programa fue escrito en lenguaje de programación C y utilizando la técnica de resolución de problemas denominado Programación Dinámica. A continuación, se procede a explicar cómo funciona esta técnica para el caso particular de este problema.

Primero es necesario calcular el vuelto que se debe entregar al cliente, el cual se obtiene de la resta entre lo que entrega el cliente para pagar y lo que debe pagar. Luego es necesario guardar la cantidad que se tiene de cada billete o moneda. Estos dos pasos previos serán utilizados para realizar la técnica Programación Dinámica:

La idea es tener un archivo en el que se irán guardando los resultados de los vueltos ya calculados. Es por esto que primero se consulta si el resultado está en el archivo guardado, con el fin de no utilizar el algoritmo para calcular el resultado nuevamente. En otras palabras, si el resultado está guardado en el archivo se obtiene y se entrega, en caso de que no esté en el archivo, se realizarán 3 acciones. Primero se calcula el resultado, luego se guarda en el archivo y posteriormente se entrega el resultado.

A continuación, se muestra una tabla de todas las funciones implementadas para llevar a cabo la solución, en esta tabla se especifica el tiempo de ejecución y orden para cada función. Cabe recalcar que la función en la cual se implementa el concepto de Programación Dinámica tiene el nombre de *calculate*, la cual se encarga de calcular el efectivo mínimo a entregar. También mencionar que dentro de esta función subyace la función *initDeliveredCoins*, la cual reinicia el arreglo en el cual se guardarán la cantidad de cada efectivo que se debe entregar. Además, subyace la función *showChange*, que muestra en la consola la cantidad que se debe entregar de cada efectivo, entre otras funciones.

Table 1. Tiempos y orden de ejecución de funciones

Función	Tiempo	Orden
existsFile	T(n)=c	O(1)
loadValues	T(n)=c+cn	O(n)
initDeliveredCoins	T(n)=c+cn	O(n)
showChange	T(n)=c+cn	O(n)
addCash	T(n)=c+cn	O(n)
greedyDeposit	T(n)=c+cn^2	O(n^2)
greedyChange	T(n)=c+cn^2	O(n^2)
copyChange	T(n)=c	O(1)
loadChange	T(n)=c+cn	O(n)
saveChange	T(n)=c	O(1)
validateChange	T(n)=c+cn	O(n)

En la siguiente figura se muestra el código en el cual se implementa el algoritmo de Programación Dinámica:

```
// Aqui se obitiene lo que se debe entregar al cliente
// PROGRAMACION DINAMICA
if (loadChange(deliveredCoins,change))
{
    //no realiza nada por que si encuentra el vuelto
    //en el archivo, los valores se cargan inmediatamente
    //printf("Se encontro el valor en el archivo\n");
}
else
{
    greedyChange(deliveredCoins,valueCoins,change);
    saveChange(deliveredCoins,change);
    //printf("No se encontro el valor en el archivo\n");
}
```

Fig. 1. Algoritmo donde se implementa la técnica de programación dinámica

5. ANÁLISIS DE LOS RESULTADOS

Analizando el algoritmo Goloso implementado, se puede decir que siempre **termina**, esto quiere decir que no realiza bucles infinitos. También mencionar que cuando este algoritmo finaliza lo hace con la **solución**, la cual en este caso particular corresponde a la cantidad mínima de efectivo para un vuelto que se debe entregar a un cliente. Cabe recalcar que el tiempo de ejecución del algoritmo depende de la combinación entre cantidad de efectivo que se tenga que entregar, la cantidad de efectivo que se tenga en la caja, la cantidad de tipos de efectivo que se tengan y por último la cantidad de resultados guardados en el archivo. Esto quiere decir que una combinación distinta de estos datos podría dar el mismo tiempo de ejecución.

Teniendo en cuenta la tabla 1, se obtiene que el orden de la función *calculate* es de n^2, esto quiere decir que es muy eficiente. Principalmente este orden está dado por un ciclo anidado en otro, ya que es necesario verificar si se puede agregar cada una de las monedas a la solución y por cada una de éstas se va agregando una unidad hasta que no se pueda agregar más.

Es necesario mencionar que para este problema en particular se ha utilizado el método goloso para calcular los resultados, ya que garantiza el óptimo, cabe recalcar que esto es debido a que el archivo de entrada está ordenado desde el billete de mayor valor, hasta la moneda de menor valor. En el caso de que no esté ordenado el archivo de entrada, el programa no entregaría la solución correcta ni óptima. Para solucionar este problema, lo que se debe hacer, es ordenar el efectivo de mayor a menor valor, transformando la entrada al algoritmo implementado. Este arreglo al programa afectaría al tiempo de ejecución, pero no al orden de las funciones ya que de todas formas la función *calculate* sería la de mayor orden, ya que el mejor algoritmo de ordenamiento es de orden $nlog(n)$

Otro punto importante a mencionar, es que al utilizar la programación dinámica, se acelera el proceso para calcular el resultado, ya que no se vuelve a calcular si es que ya se realizó previamente, pero cabe recalcar que si el archivo en el cual se encuentran los resultado es muy grande, es mejor utilizar el método goloso por si solo, ya que el programa se demoraría más en buscar el resultado que en realizarlo nuevamente. Una otra opción para guardar los valores es utilizar la estructura de árbol binario ordenado, en el cual el tiempo de búsqueda es de orden O(nlog(n)), mucho menor a buscar en el archivo de manera secuencial con un tiempo de búsqueda de orden O(n)

6. TRAZA DE LA SOLUCIÓN

Las siguientes tablas muestran una traza del algoritmo de Programación dinámica, el cual ejemplifica de manera concreta cómo se va entregando la solución.

Para la traza se pedirá entregar el vuelto para los siguientes valores:5400, 2570, 4500, 2570, 1239, 2570, 4500, 1239, 2412, 1239, 8434

Table 2. Traza de vueltos

Tiempo	L1	L2	L3	L4	L5	L6
1	5400					
2	5400	2570				
3	5400	2570	4500			
4	5400	2570	4500			
5	5400	2570	4500	1239		
6	5400	2570	4500	1239		
7	5400	2570	4500	1239		
8	5400	2570	4500	1239		
9	5400	2570	4500	1239	2412	
10	5400	2570	4500	1239	2412	
11	5400	2570	4500	1239	2412	8434

Como se puede apreciar en la tabla 2, en la primera columna se muestra el instante de ejecución, mientras que en las siguientes columnas se muestran los valores que se encuentran en las líneas correspondientes del archivo. Los valores de color rojo son aquellos que se deben insertar al archivo, ya que no se encuentra en éste, es por esto que se debe realizar el algoritmo que calcula el vuelto de cada moneda para ese valor, para posteriormente guardarlo. Los valores de color verde, son aquellos que se encuentran en el archivo al momento de realizar una consulta sobre ese valor, de esta forma no es necesario utilizar el algoritmo para calcular las monedas mínimas para devolver ese valor, ya que se obtienen del archivo directamente.

Cabe recalcar que mientras más valores distintos se consultan más va creciendo el archivo, es decir se irá agregando una línea nueva por cada valor que se consulte y no esté en el archivo. Es por esto que el archivo puede ser demasiado grande.

7. CONCLUSIONES

Para concluir este informe se pudo observar que se cumplieron todos los diferentes objetivos. El objetivo principal fue, realizar un programa que indique la cantidad mínima de efectivo que debe entregar un cajero a un cliente, donde se utilizó la técnica de resolución de problema denominado textitProgramación Dinámica.

Posteriormente se analizó el algoritmo implementado y se llegó a la conclusión, de que es eficiente para este problema, ya que su tiempo de ejecución es de orden $O(n^2)$, a diferencia de los métodos de backtracking y fuerza bruta, los cuales generan muchas soluciones y todas las soluciones respectivamente con un tiempo de ejecución de orden mayor a n^2. Además, la solución entregada es óptima cuando los valores del efectivo están ordenados de mayor a menor, como se dijo en el capítulo de análisis de resultados, incluso se dio una forma de cómo mejorar el programa y que dicha condición no sea necesaria.

Además, se comprobó que cuando el archivo en el cual se guardan los resultados es muy grande es conveniente utilizar el método goloso por si solo, otra alternativa a la solución propuesta en el capítulo de análisis para el método de programación dinámica es utilizar es simular una memoria cache en la cual se guarden los resultados, de esta forma, la cantidad de resultados guardados tendría una cantidad máxima, y además quedaran guardados solo aquellos resultados que más se utilicen en el último periodo de tiempo.

Finalmente, se ha llevado a la práctica distintos conceptos vistos en clases, de esta forma, se ha comprendido de una forma mejor sobre éstos. Con esta cuarta entrega, es posible decir, que el óptimo de una solución dado un algoritmo implementado, depende netamente del problema. En este problema en particular al utilizar la técnica Programación Dinámica, se obtiene el óptimo, ya que se utilizó el método goloso para calcular el resultado.

REFERENCES

1. "Definición de algoritmo," http://conceptodefinicion.de/algoritmo. [Online; Último acceso 17-Abril-2017].
2. M. V. Ilufi, *Algoritmos: Teoría y aplicaciones* (2002).