Programación Dinámica aplicada 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 27, 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 4 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 y en el actual curso de algoritmos avanzados, se basa en el enfoque de resolución de problemas con el uso de la Programación Dinámica.
El objetivo general de este laboratorio es demostrar el conocimiento del concepto de Programación Dinámica, para la resolución de problemas.
Dentro de los objetivos específicos de éste laboratorio estan:

- Plantear la idea y el algoritmo
- Desarrollar un programa que resuleve el problema.
- Analizar los resultados y la traza

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.
En este documento se puede encontrar las siguientes secciones:

- Despcripción del problema: En esta sección es posible obtener la dimensión del problema planteado.
- Marco teórico: Se rescatan definiciones teoricas y estado del arte de los conocimientos utilizados en este laboratorio.
- Descripción de la solución: Se aborda el desarrollo del laboratorio, explicando el cómo se construyó la solución.
- Análisis de resultados: En esta sección se evaluan los resultados obtenidos al ejecutar el laboratorio
- Traza de la solución: Para esta sección se muestra cómo se realiza el proceso de la solución.
- Conlusiones: En esta última instancia se rescatan las comparaciones entre los objetivos y lo alcanzado.

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.
Este problema corresponde al mismo utilizado en el laboratorio 3 de la asignatura, en cuya ocasión se utilizó el algoritmo Greedy para la obtención de una solución.

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 un algoritmo de Programación Dinámica. Según [1], lo siguiente define a al método de Programación Dinámica:

- El método de programación dinámica sirve para resolver problemas combinando las soluciones de subproblemas.
- Normalmente es utilizada para resolver problemas de optimización.
- Al construir un algoritmo usando la estrategia de programación dinamica es necesario:

1. Caracterizar la estructura de una solución óptima

2. Definir recursivamente *el valor* de una solución óptima.
3. Computar *el valor* de una solución en forma *bottom-up*
4. Construir una solución óptima a partir de la información computada.

Por otro lado:

"La Programación Dinámica no sólo tiene sentido aplicarla por razones de eficiencia, sino porque además presenta un método capaz de resolver de manera eficiente problemas cuya solución ha sido abordada por otras técnicas y ha fracasado."[2]

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 cajón 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 manera no rígidos, o bien líquidos y no se sobrelapan unos con otros.

C. IDEA

Se tiene un cajón de capacidad máxima igual al V en 1 y un conjunto S de n elementos

$$V = LARGO\_MAX \times ANCHO\_MAX \times ALTO\_MAX \quad \textbf{(1)}$$

Cada libro l_i tiene un volumen v_i

- Paso 1: Supongamos que tenemos la forma óptima de almacenar los n libros en un cajón, tal que en 2 se tiene dicha forma óptima de almacenar.

  $$S_n = \{l_1, l_2, ..., l_n\} \quad \textbf{(2)}$$

  Ahora bien, se busca un subproblema que también sea óptimo para una *subcajon* de k libros. Definimos el parámetro v como el volúmen máximo para la *subcajon* representada en 3

  $$S_k = \{l_1, l_2, ..., l_k\} \quad \textbf{(3)}$$

  Dicho lo anterior, el problema será calcular $V[k, v]$ para encontrar la solución óptima para S_k en un *subcajon*
- Paso 2: Asumiendo que $V[i, j]$ con $i = 0, 1, 2, 3, ..., k-1$ y $j = 0, 1, 2, 3, ..., w$.
  Recursivamente se tiene lo observado en 4

  $$V[k, v] = \begin{cases} V[k-1, v] & si \quad v_k > v \\ max\{V[k-1, v], V[k-1, v-v_k] + v_k\} & sino \end{cases} \quad \textbf{(4)}$$

  El significado de esta expresión, dice que, el mejor subconjunto S_k que tiene un volúmen total v:
  - El mejor subconjunto S_{k-1} que tiene un volúmen total $\leq v$, ó
  - El mejor subconjunto S_{k-1} que tiene un volúmen total $v - v_k$ más el elemento k.

  Otra forma de explicar esta expresión recursiva:
  - Caso $v_k > v$. Si intentamos ingresar el elemento k al conjunto este sobrepasaría el volumen v de ese subconjunto S_{k-1} por lo tanto el elemento k no puede ser parte del conjunto.
  - Caso $v_k \leq v$. En este caso el elemenot k si puede ingresar a S_{k-1} y se escoge el caso con mayor valor.
- Paso 3: En el algoritmo 1 se muestra cómo se rellena la tabla con los valores para el *Bot tom-up*

Algorithm 1. Obtenicón de valores óptimos

```
1:  procedure ALGORITMO(Vmax, n)
2:      for v = 0 : Vmax do
3:          V[0, v] = 0
4:      for i = 1 : n do
5:          V[i, 0] = 0
6:      for i = 1 : n do
7:          for v = 0 : Vmax do
8:              if v_i ≤ v then
9:                  if v_i + V[i − 1, v − v_i] > V[i − 1, v] then
10:                     V[i, v] = v_i + V[i − 1, v − v_i]
11:                 else
12:                     V[i, v] = V[i − 1, v]
13:             else
14:                 V[i, v] = V[i − 1, v]
        return V
```

- Paso 4: La tabla V[i,v] captura aquellos volúmenes acumulados para cada combinación de elementos con volúmen v. Ahora bien, para armar y obtener los elementos que van en el cajón se arma otro algoritmo que revisa los resultados de esta tabla.

Algorithm 2. Obtenicón de elemntos que van en el cajón

```
1:  procedure ALGORITMO(Vmax, n)
2:      i = n
3:      k = Vmax
4:      Cajon = ϕ
5:      Fuera = ϕ
6:      while i > 0  y  k > 0 do
7:          if V[i, k] ≠ V[i − 1, k] then
8:              Cajon + {i}
9:              i = i − 1
10:             k = k − w
11:         else
12:             Fuera + {i}
13:             i = i − 1
        return Cajon,Fuera
```

5. ANÁLISIS DE LOS RESULTADOS

En esta sección se realizará un análisis del algoritmo implementado en base a la serie de preguntas habit-

uales que se realizan para comprobar la validez de un algoritmo. Para este caso se tienen dos algoritmos asi que se analizaran en conjunto como parte de una solución.

A. Análisis: Preguntas para los Algoritmos 1 y 2

1. ¿Los algoritmos paran en algún momento?
   Si, ambos algoritmos paran ya que trabajan con matriz con limites establecidos en proceso contenidos.

2. Y si paran, ¿lo hacen con la solución?
   En el caso del algoritmo para obtener la tabla de valores del volumen para con lo que se le solicita que es armar la tabla de los volumenes para las combinaciones de elementos y volumenes. Mientras que el algoritmo que recopila los libros que deben ir en la caja dada la configuracion de la tabla, para con la solución óptima.

3. ¿Son eficaces?¿Solucionan el problema?
   Son eficaz considerando que soluciona de manera óptima el problema.

4. ¿Son eficientes?
   Si establecemos una complejidad conjunta de ambos algoritmos tenemos: para el algoritmo 1 una complejidad de $O(nm)$ siendo n el número de libros disponibles y m el volumen posible desde 0 al máximo volumen almacenable en la caja. Para el segundo algoritmo, este corresponde a $O(m)$ ya que m por lo general será mayor que n. Por lo tanto al ser polinómicas las complejidades, estos algoritmos son eficientes.

5. ¿Se puede mejorar?
   Este algritmo al ser diseñado con respecto al análisis previo, se podría mejorar planteando una descomposición diferente, pero en lo general dado que se debe de igual manera obtener algún dato en base al llenado de una tabla que generará complejidades de orden cuadrático.

6. ¿Existe otro método para solucionar el problema?
   Se puede aplicar para este problema algoritmos aproximativos o bien como se vio en el laboratorio 3, algoritmo Greedy, el cuál no asegura el óptimo.

6. TRAZA DE LA SOLUCIÓN

En la Figura 1 se observa una representación del algoritmo 1 donde van los datos de volumenes acumulados por subcajón.

Fig. 1. Explicación de la traza a grosso modo

```
Libros: {1,2,3,4,5,6}
Volúmen máximo = 9 cm³

n|v| 0 || 1 || 2 || 3 || 4 || 5 || 6 || 7 || 8 || 9 |
-----------------------------------------------------
|0|  0 || 0 || 0 || 0 || 0 || 0 || 0 || 0 || 0 || 0 |
-----------------------------------------------------
|1|  0 ||                                           |
---------------   Aca van el resto   ---------------
|2|  0 ||                                           |
--------------- de datos obtenidos desde -----------
|3|  0 ||                                           |
---------------    algoritmo 1      ---------------
|4|  0 ||                                           |
-----------------------------------------------------
|5|  0 ||                                           |
-----------------------------------------------------
|6|  0 ||                                           |
|===================================================
```

Fig. 2. Parte 1 de la salida del programa

```
En la caja:
EN 50 50 50
EM 10 10 8
EL 30 8 2
EK 23 5 6
EJ 12 30 5
EI 15 24 6
EH 12 6 6
EG 14 7 5
EF 11 8 9
EE 15 9 14
ED 11 10 9
EC 10 10 8
EB 30 8 2
EA 23 5 6
DZ 11 10 9
```

Fig. 3. Parte 2 de la salida del programa

```
EV 11 8 9
EU 15 9 14
ET 11 10 9
ES 10 10 8
ER 12 6 6
EQ 14 7 5
EP 11 8 9
EO 15 9 14

Espacio libre dentro de la caja:
64302 centímetros cúbicos
```

7. CONCLUSIONES

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

- El algoritmo entrega una solución óptima a la bibliotecaria.
- Se pueden aplicar otras técnicas para resolverlo pero es con la programación dinámica que se obtienen resultados eficientes.

- En la comparativa con la solución del laboratorio 3, se observa una mejora debido a la optimización del problema.
- 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. En este caso la programación dinámica elimina la recursividad, haciendo la solución más eficiente.

En base a los objetivos planteados, se tiene que:
Se demostraron los conocimientos obtenidos sobre Programación Dinámica, desarrollando una idea y algoritmo.
El programa funciona y puede ser cotejado por el lector con la versión anterior del laboratorio 3, quedan a disposición del lector estas pruebas.
El análisis estándar del algoritmo cumple con las preguntas realizadas al algoritmo correspondiente.

REFERENCES

1. J. Baier, "Taller de programación avanzada/programación dinámica," (2004).
2. R. G. y Antonio Vallecillo, "Técnicas de diseño de algoritmos," (2000).