Análisis de Algoritmos, Sem: 2018-2, 3CV1, Práctica 8, 14 de Noviembre de 2018

Práctica 8: Multiplicación de una Secuencia de Matrices

Reyes Valenzuela Alejandro

Escuela Superior de Cómputo
Instituto Politécnico Nacional, México
*areyesv*11@*gmail.com*

Resumen: En este reporte se abordará el desarrollo de un programa mediante la programación dinámica, la cuál nos va a permitir determinar el caso óptimo para la resolucióin de un problema determinado, esto con el fin de evitar la búsqueda de todas las combinaciones posibles y sólo enfocarnos en la que nos interesa (la óptima).

Palabras Clave: Programación,Dinámica, Matrices,Óptimo.

1 Introducción

Cuando debemos de solucionar problemas, en varios casos utilizamos la primera solución que encontramos, Sin importar el costo o tiempo que pueda llevar, por lo que normalmente se tienen que analizar varias opciones para determinar cuál se adapta mejor a lo que estamos buscando.

Sin embargo, puede darse el caso de que el número de opciones crece considerablemente, por lo que nos es imposible analizarlos todos desde cero; es por este motivo que la programación dinámica busca dar solución a los problemas con base a los problemas del mismo tipo resueltos previamente, esto con el fin de evitar repeticiones en operciones que nos quiten tiempo y enfocarnos en la búsqueda de la solución solicitada.

En esta práctica se abordará un ejemplo, cuyo funcionamiento se observará tanto de manera teórica como experimental.

2 Conceptos Básicos

En informática, la programación dinámica es un método para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas superpuestos y subestructuras óptimas.

Conviene resaltar que a diferencia de la programación lineal, el modelado de problemas de programación dinámica no sigue una forma estándar. Así, para cada problema será necesario especificar cada uno de los componentes que caracterizan un problema de programación dinámica.

El procedimiento general de resolución de estas situaciones se divide en el análisis recursivo de cada una de las etapas del problema, en orden inverso, es decir comenzando por la última y pasando en cada iteración a la etapa antecesora. El análisis de la primera etapa finaliza con la obtención del óptimo del problema. Para esta práctica utilizaremos dicho análisis para obtener el producto de matrices óptimo, es decir, en donde se realice el menor número de operaciones, para ello en vez de analizar todos los casos posibles en los que podemos multiplicar nuestra matriz, se analizará un algoritmo que nos devuelve de manera directa nuestro producto óptimo:

$SecuenciaMatrices(P)$
  n=p.length
  Sean $m[1,...,n][2,...,n]$ y $S[1,...,n-1][1,...,n-1]$ dos tablas
  for i = 0 to n
    $m[i][j] = 0$
  for l = 2 to n
    for i = 1 to $n - l + 1$
      $j = i + l - 1$
      $m[i][j] = +\infty$
      for k = i to $j - 1$
        $q = m[i][k] + m[k+1][j] + P_{i-1} * P_k * P_j$
        $if\ q < m[i][j]$
          $m[i][j] = q$
          $S[p][j] = k$
$return$ m y S

El algoritmo anterior genera la configuración óptima, por lo que ahora necesitamos otro para imprimir dicha configuración:

$ImprimeOptimo(S, i, j)$
$if\ i == j$
  $return\ A_i$
$else$
  Imprime (
  $ImprimeOptimo(S, i, S[i, j])$
  $ImprimeOptimo(S, S[i, j] + 1, j)$
  Imprime)

3 Experimentación y Resultados

```
[alejandro@alejandro-pc ~]$ python2 p8.py
Tamaños Matrices
[10, 5, 4, 6, 5]
Configuración óptima de producto de matrices:
( A1 ( A2 ( A3 A4 ) ) )
Minimo numero de multiplicaciones: 470
[alejandro@alejandro-pc ~]$ python2 p8.py
Tamaños Matrices
[14, 7, 8, 5, 4]
Configuración óptima de producto de matrices:
( A1 ( A2 ( A3 A4 ) ) )
Minimo numero de multiplicaciones: 776
[alejandro@alejandro-pc ~]$ python2 p8.py
Tamaños Matrices
[3, 6, 8, 2, 4]
Configuración óptima de producto de matrices:
( ( A1 ( A2 A3 ) ) A4 )
Minimo numero de multiplicaciones: 156
[alejandro@alejandro-pc ~]$
```

Ejecución del Programa

Podemos observar que nos devuelve la configuración óptima de matrices con base a un arreglo, por lo que a continuación corrobaremos éste hecho de manera teórica.

Tomaremos los casos A=$10, 5, 4, 6, 5$ y A=$3, 6, 8, 2, 4$ para comprobar que la configuración obtenida es la óptima.

Primer caso A=$10, 5, 4, 6, 5$:

$(A_1\ A_2)(A_3\ A_4)$ = Número de operaciones = 520
$A_1\ (A_2\ (A_3\ A_4))$ = Número de operaciones = 470
$A_1\ ((A_2\ A_3)\ A_4\ ))$ = Número de operaciones = 650
$((A_1\ A_2\ )\ A_3\ ))\ A_4$ Número de operaciones = 740
$(A_1\ (A_2\ A_3))\ A_4$ = Número de operaciones = 720

Podemos observar que el número menor es el 470, resultado que fue arrojado por el programa, al igual que su configuración correspondiente, por lo que el programa cumplió con su cometido en este caso.

Primer caso A=$3, 6, 8, 2, 4$:

$(A_1\ A_2)(A_3\ A_4)$ = Número de operaciones = 304
$A_1\ (A_2\ (A_3\ A_4))$ = Número de operaciones = 470
$A_1\ ((A_2\ A_3)\ A_4\ ))$ = Número de operaciones = 328
$((A_1\ A_2\ )\ A_3\ ))\ A_4$ Número de operaciones = 216
$(A_1\ (A_2\ A_3))\ A_4$ = Número de operaciones = 156

De la misma manera, el número menor fue el mismo que el arrojado por el programa, en este caso el 156, cuya configuración fue impresa por el programa, por lo que nuevamente corroboramos que la parentización propuesta fue la de menor número de operaciones.

4 Conclusiones

Reyes Valenzuela Alejandro: Con la programación dinámica pude observar que podemos analizar problemas de una mejor manera, ya que evitamos analizar todos los casos y en vez de eso reutilizamos valores que ya conocemos para poder resolver problemas de mayor complejidad de una manera más sencilla. Adicionalmente, este problema en concreto no fue difícil de programar, por lo que únicamente se procedió a calcular los otros resultados de manera teórica para comprobar los resultados.

5 Anexo

Documente que la ecuación de recurrencia P(n) = $\sum_{k=1}^{n-1} P(k)P(n - k)$ si n $\geq$ 2 y P(n = 1) = 1 es (2^n). La ecuación de recurrencia P(n) genera los números de Catalán. Mediante el método de sustitucin hacia adelante tenemos que:

P(n=2) = $\sum_{k=1}^{1} P(k)P(n - k)$ =P(1)*P(1)=1

P(n=3) = P(2)*P(1) + P(1)*P(2) = 1 * 1 + 1 * 1 = 2

P(n=4) = P(1)*P(3) + P(2)*P(2) + P(3)*P(1) = 1 * 2 + 1 * 1 + 1 * 1 = 5

P(n=5) = P(1)*P(4) + P(2)*P(3) + P(3)*P(2) + P(4)*P(1) = 1 * 5 + 1 * 2 + 2 * 1 + 5 * 1 = 14

Dichas cantidades podemos acotarlas con 2^n, por ejemplo:

P(2) = 1 c * 2^2 = 4 * c, para todo c $\leq$ 1/4

P(3) = 2 c * 2^3 = 8 * c, para todo c $\leq$ 1/4

P(4) = 5 c * 2^4 = 16 * c, para todo c $\leq$ 1/4

P(5) = 14 c * 2^5 = 32 * c, para todo c $\leq$ 1/4

Como podemos observar, los números de Catalán acotan inferiormente a la función 2^n, por lo que tenemos que P(n) ϵ Ω (2^n)

6 Bibliografía

Eumed.net (2018). [online] Available at: http://www.eumed.net/libros-gratis/2006c/216/1j.htm [Accessed 13 Nov. 2018].

Radford.edu (2018). [online] Available at: https://www.radford.edu/nokie/classes/360/dp-matrix-parens.html [Accessed 13 Nov. 2018].