Tema 15: Preprocesamiento. Algoritmos Numéricos

Evaluación de Polinomios

Multiplicación de Matrices

Resolución de Ecuaciones Lineales

Eficiencia de Algoritmos Numericos

- Los programas numericos suelen hacer cálculos muy concretos un gran numero de veces
- Pequeñas, casi insignificantes, mejoras pueden producir importantes ahorros de tiempo debido a la gran cantidad de veces que se hace un cierto calculo

Preprocesamiento

- Sea I el conjunto de los casos de un problema, y supongamos que cada caso $i \in I$ consiste en dos componentes $j \in J$ y $k \in K$ (es decir $I \subseteq JxK$)
- Un algoritmo de preprocesamiento para este problema es un algoritmo A que acepta como input algun elemento $j \in J$ y produce como output otro algoritmo B_j
- Ese algoritmo B_j debe ser tal tal que si $k \in K$ y $(j,k) \in I$, entonces la aplicación de B_j en k da la solución del caso (j,k) del problema original.

Ejemplo

- Sea J un conjunto de gramaticas para una familia de lenguajes de programacion (C, Fortran, Cobol, Pascal …) y K un conjunto de programas
- El problema general es saber si un programa dado es sintacticamente corecto en alguno de los lenguajes dados
- Aqui I es el conjunto de casos del tipo: ¿Es válido el programa k en el lenguaje que define la gramatica $j \in J$?

Solución del ejemplo

- Un posible algoritmo de preprocesamiento para este ejemplo es un generador de compiladores:
- Aplicado a la gramatica $j \in J$ genera un compilador B_j para el lenguaje en cuestión
- Por tanto para saber si $k \in K$ es un programa en el lenguaje j, simplemente aplicamos el compilador B_j a K

Preprocesamiento

- Sea:
  - A(j) = tiempo para producir B_j dado j
  - $b_j(k)$ = tiempo para aplicar B_j a k
  - T(j,k) = tiempo para resolver (j,k) directamente
- Generalmente $b_j(k) \leq t(j,k) \leq a(j) + b_j(k)$
- No interesa el preprocesamiento si

$$b_j(k) > t(j,k)$$

Utilidad del Preprocesamiento

- Suele ser util en dos situaciones:
  - Emergencias: Necesitamos ser capaces de resolver cualquier caso muy rapidamente
  - Hay que resolver una serie de casos para un mismo valor de j: (j, k_1), (j, k_2), …, (j, k_n). El tiempo consumido en resolver todos los casos si trabajamos sin preprocesamiento es

$$t_1 = \Sigma_{i=1..n} (j, k_i)$$

  - y

$$t_2 = a(j) + \Sigma_{i=1..n} b_j(k_i)$$

  - Si trabajamos con preprocesamiento
  - Cuando n es suficientemente grande, t_2 suele ser mayor que t_1
- Lo estudiaremos asociado a problemas de tipo numérico

Elementos de Análisis Numerico

- Contaremos las adiciones y multiplicaciones
- Como normalmente las adiciones son mucho mas rápidas que las multiplicaciones, la reducción de las multiplicaciones a costa del aumento de las adiciones, puede producir mejoras
- Nuestros analisis se basaran en la potencia mayor de un polinomio o en el tamaño de las matrices con las que estemos trabajando

Cálculo de Polinomios

- Usaremos la forma general de un polinomio

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

- en la que los valores de los coeficientes se suponen conocidos y constantes
- El valor de x será el input y el output será el valor del polinomio usando ese valor de x

Algoritmo de Evaluación Estandar

```
result = a[0] + a[1]*x
xPower = x
for i = 2 to n do
    xPower = xPower * x
    result = result + a[i]*xPower
end for
return result
```

Análisis

- Antes del lazo, hay
  - Una multiplicación
  - Una adición
- El lazo for se hace N-1 veces
  - Hay dos multiplicaciones en el lazo
  - Hay una adición en el lazo
- Hay un total de
  - 2N-1 multiplicaciones
  - N adiciones

El Método de Horner

- Se basa en la factorización de un polinomio
- Nuestra ecuación general puede factorizarse como

$$p(x) = (\{[(a_n x + a_{n-1})*x + a_{n-2}]*x + \ldots + a_2\}*x + a_1)*x + a_0$$

- Por ejemplo, la ecuación

$$p(x) \;=\; x^3 - 5x^2 + 7x - 4$$

se factorizaría como

$$p(x) \;=\; [(x-5)*x+7]*x-4$$

El Algoritmo de Horner

```
result = a[n]
for i = n - 1 down to 0 do
    result = result * x
    result = result + a[i]
end for
return result
```

Análisis

- El lazo for se hace N veces
  - Hay una multiplicación en el lazo
  - Hay una adición en el lazo
- Hay un total de
  - N multiplicaciones
  - N adiciones
- Nos ahorramos N-1 multiplicaciones sobre el algoritmo estandar

Preprocesamiento de Coeficientes 1

- Usa la factorización de un polinomio, considerada a partir de polinomios de grado mitad del original
- Por ejemplo, cuando el algoritmo estandar tuviera que hacer 255 multiplicaciones para calcular x^{256}, nosotros podriamos considerar el cuadrado de x y el del resultado, con lo que ahorrariamos mucho tiempo para obtener el mismo resultado

Preprocesamiento de Coeficientes 2

- Suponemos polinomios mónicos (a_n=1), con la mayor potencia siendo de valor uno menos que cierta potencia de 2.
- Si nuestro polinomio tiene como mayor potencia 2^k-1, lo podemos factorizar como:

$$p(x) = \left(x^j + b\right) * q(x) + r(x)$$

donde $j = 2^{k-1}$

Preprocesamiento de Coeficientes 3

- Si elegimos b de modo que sea a_{j-1} - 1,

$$p(x) = \left(x^j + b\right) * q(x) + r(x)$$

entonces q(x) y r(x) tambien seran mónicos, con lo que el proceso podrá aplicarse recursivamente sobre ellos tambien

Preprocesamiento de Coeficientes Ejemplo 1

- Si consideramos

$$p(x) = x^3 - 5x^2 + 7x - 4$$

como la mayor potencia es $3 = 2^2-1$, entonces j sería $2^1 = 2$, y b valdría $a_1 - 1 = 6$

- Así, nuestro factor es $x^2 + 6$, y dividimos p(x) por este polinomio para encontrar q(x) y r(x)

Preprocesamiento de Coeficientes
Ejemplo 2

- La división es:

$$
\begin{array}{r|r}
 & x-5 \\
\cline{2-2}
x^2+6 & x^3-5x^2+7x-\ \ 4 \\
 & x^3-5x^2+6x-30 \\
\cline{2-2}
 & x+26
\end{array}
$$

que da

$$p(x) \;=\; \left(x^2+6\right)*(x-5)+(x+26)$$

Análisis 1

- Analizamos el preprocesamiento de coeficientes desarrollando una ecuación de recurrencia para el numero de multiplicaciones y adiciones
- En nuestra factorización, partimos el polinomio en otros dos mas pequeños, y haciendo una multiplicación mas y dos sumas adicionales

Análisis 2

- Sea M(k) el numero de multiplicaciones requeridas para evaluar el polinomio de grado $N = 2^k - 1$.
- Sea $A(k) = M(k) - k + 1$ el numero de multiplicaciones requeridas si no contamos las usadas en el cálculo de $x^2, x^4, \ldots, x^{(n+1)/2}$.
- Se obtiene la siguiente ecuación recurrente,

$$
\begin{array}{ll}
A(0) = 0 & \text{si } k = 1 \\
A(k) = 2A(k-1) + 1 & \text{si } k \geq 2
\end{array}
$$

Análisis 3

- Resolviendo esta ecuacion obtenemos

$$A(k) = 2^{k-1} - 1, \text{ cuando } k \geq 1,$$

- y asi

$$M(k) = 2^{k-1} + k - 2$$

- En otras palabras, $(N-3)/2 + \log(N+1)$ multiplicaciones son suficientes para evaluar un polinomio de grado $N = 2^k - 1$.

Comparación de los Algoritmos para Polinomios 1

- En el ejemplo que hemos visto:
  - **Algoritmo Estandar:**
    - 5 multiplicaciones y 3 adiciones
  - **Método de Horner**
    - 3 multiplicaciones y 3 adiciones
  - **Preprocesamiento de Coeficientes**
    - 2 multiplicaciones y 4 adiciones

Comparación de los Algoritmos para Polinomios 2

- En general, para un polinomio de grado N:
  - **Algoritmo Estandar:**
    - 2N-1 multiplicaciones y N adiciones
  - **Método de Horner**
    - N multiplicaciones y N adiciones
  - **Preprocesamiento de coeficientes**
    - N/2 + lg N multiplicaciones y (3N-1)/2 adiciones