



Algorítmica

Capítulo 6. Otras Metodologías Algorítmicas.

Tema 16. Algoritmos de Precomputación

-Algoritmos numéricos

- Evaluación de polinomios, Multiplicación de matrices, Sistemas de ecuaciones lineales

Eficiencia de Algoritmos Numéricos

- Los programas numéricos suelen hacer cálculos muy concretos un gran número de veces
- Pequeñas, casi insignificantes, mejoras pueden producir importantes ahorros de tiempo debido a la gran cantidad de veces que se hace un cierto cálculo
- La eficiencia algorítmica se mide en términos de exactitud.

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 $I \subseteq J \times K$)
- Un algoritmo de preprocesamiento para este problema es un algoritmo A que acepta como input algún elemento $j \in J$ y produce como output otro algoritmo B_j
- Ese algoritmo B_j debe ser 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 gramáticas para una familia de lenguajes de programación (C, Fortran, Cobol, Pascal ...) y K un conjunto de programas
- El problema general es saber si un programa dado es sintácticamente correcto en alguno de los lenguajes dados
- Aquí I es el conjunto de casos del tipo: ¿Es válido el programa k en el lenguaje que define la gramática $j \in J$?
- Un posible algoritmo de preprocesamiento para este ejemplo es un generador de compiladores:
 - Aplicado a la gramática $j \in J$ 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 útil en dos situaciones:
 - Emergencias: Necesitamos ser capaces de resolver cualquier caso muy rápidamente
 - Hay que resolver una serie de casos para un mismo valor de j : $(j, k_1), (j, k_2), \dots, (j, k_n)$. El tiempo consumido en resolver todos los casos si trabajamos sin preprocesamiento es

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

y

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

si trabajamos con preprocesamiento

Cuando n es suficientemente grande, t_2 suele ser menor que t_1

- Lo estudiaremos asociado a problemas de tipo numérico en los que siempre intervienen unos mismos coeficientes.

Eficiencia en Algoritmos Numéricos

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

Cálculo de Polinomios

- Usaremos la forma general de un polinomio

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

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

Algoritmo de Evaluación Estándar

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

- El análisis fácil: Antes del lazo, hay una multiplicación y una adición. El lazo se hace $N-1$ veces y hay dos multiplicaciones y una adición. Por tanto hay un total de $2N-1$ multiplicaciones y N adiciones

Evaluación con el Método de Horner

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

$$p(x) = \left(\left(\left(a_n x + a_{n-1} \right) * x + a_{n-2} \right) * x + a_2 \right) * x + a_1 \big) * x + a_0$$

- Por ejemplo, el polinomio

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

se factorizaría como

Your company name

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

Evaluación con el Método de Horner

```
result = a[n]
```

```
for i = n - 1 down to 0 do
```

```
    result = result * x
```

```
    result = result + a[i]
```

```
end for
```

```
return result
```

- El análisis es sencillo: el lazo for se hace N veces y conlleva una multiplicación y una adición. Así que hay un total de N multiplicaciones y N adiciones. Por tanto nos ahorramos N-1 multiplicaciones sobre el algoritmo estándar.

Preprocesamiento de Coeficientes

- Usa la factorización de un polinomio, considerada a partir de polinomios de grado mitad del original
- Por ejemplo, cuando el algoritmo estándar tuviera que hacer 255 multiplicaciones para calcular x^{256} , nosotros podríamos considerar el cuadrado de x y el del resultado, con lo que ahorraríamos mucho tiempo para obtener el mismo resultado
- Suponemos polinomios mónicos ($a_n=1$), con la mayor potencia siendo de valor uno menos que cierta potencia de 2.
- Así, si nuestro polinomio tiene como mayor potencia 2^k-1 , lo podemos factorizar como:

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

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

Preprocesamiento de Coeficientes

- Si miramos el coeficiente del término $j - 1$ del polinomio

tomamos $b = a_{j-1} - 1$ y entonces $q(x)$ y $r(x)$ también serán mónicos, con lo que el proceso podrá aplicarse recursivamente sobre ellos también.

- Eso llevará a reducir el número de operaciones a efectuar.

Preprocesamiento de Coeficientes: Ejemplo

- Sea

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

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 $x^2 + 6$. Entonces dividimos $p(x)$ por este polinomio para encontrar $q(x)$ y $r(x)$, y se obtiene

$$q(x) = x - 5, r(x) = x + 26$$

- Y por tanto $p(x) = (x^2 + 6)(x-5) + (x + 26)$

Análisis

- Analizamos el preprocesamiento de coeficientes desarrollando una ecuación de recurrencia para el número de multiplicaciones y adiciones.
- En nuestra factorización, partimos el polinomio en otros dos mas pequeños, haciendo una multiplicación mas y dos sumas adicionales
- Sea $M(k)$ el número de multiplicaciones requeridas para evaluar el polinomio de grado $N = 2^k - 1$.
- Sea $A(k) = M(k) - k + 1$ el número de multiplicaciones requeridas si no contamos las usadas en el cálculo de $x^2, x^4, \dots, x^{(n+1)/2}$
- Se obtiene la siguiente ecuación recurrente,

$$A(0) = 0 \text{ si } k = 1$$

$$A(k) = 2A(k-1) + 1 \quad \text{si } k \geq 2$$

Análisis

- Resolviendo esta ecuación obtenemos

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

- y así

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

- 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 tres Algoritmos

- En el ejemplo que hemos visto:
 - **Algoritmo Estándar:**
 - 5 multiplicaciones y 3 adiciones
 - **Método de Horner**
 - 3 multiplicaciones y 3 adiciones
 - **Preprocesamiento de Coeficientes**
 - 2 multiplicaciones y 4 adiciones
- Para un polinomio de grado N :
 - **Algoritmo Estándar:**
 - $2N-1$ 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

Multiplicación de Matrices

- Dos matrices se pueden multiplicar si el número de columnas de la primera es igual al número de filas de la segunda
- Cada fila de la primera matriz se multiplica por cada columna de la segunda matriz
- El valor en la casilla i, j de la matriz es el resultado de la suma de los productos correspondientes a la fila i de la primera matriz por la columna j de la segunda

Multiplicación de Matrices Estándar

```
for i = 1 to a do
  for j = 1 to c do
     $R_{i,j} = G_{i,1} * H_{1,j}$ 
    for k = 2 to b
       $R_{i,j} = R_{i,j} + G_{i,k} * H_{k,j}$ 
    end for k
  end for j
end for i
```

- Por tanto, como sabemos, para multiplicar dos matrices $G(a \times b)$ y $H(b \times c)$ el algoritmo hace $a*b*c$ multiplicaciones y $a*(b-1)*c$ adiciones

Multiplicación de Matrices de Winograd

- Podemos considerar cada fila y columna como vectores:
 $V = (v_1, v_2, v_3, v_4)$ y $W = (w_1, w_2, w_3, w_4)$
 - Cada elemento del resultado será el producto de dos vectores:

$$V \bullet W = v_1 * w_1 + v_2 * w_2 + v_3 * w_3 + v_4 * w_4$$

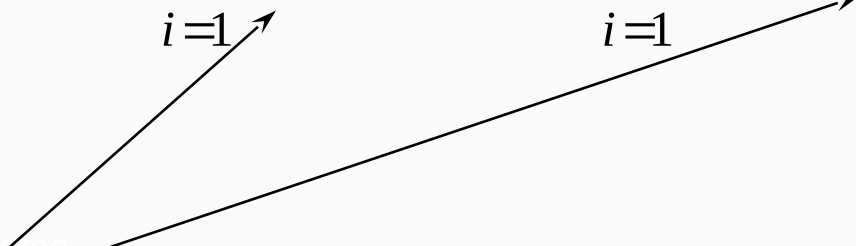
- Cada uno de esos productos se puede factorizar:

$$\begin{aligned} V \bullet W = & (v_1 + w_2) * (v_2 + w_1) + (v_3 + w_4) * (v_4 + w_3) \\ & - v_1 * v_2 - v_3 * v_4 \end{aligned}$$

- Aunque esto parece ~~mas~~ ^{mas} ~~trabajoso~~, ^{trabajoso}, las dos últimas líneas se pueden hacer solo una vez para cada fila de la primera matriz y cada columna de la segunda matriz.

Multiplicación de Matrices de Winograd

- Entonces para multiplicar una fila por una columna, tendríamos

$$V \bullet W = \sum_{i=1}^{n/2} (v_{2i-1} + w_{2i})(v_{2i} + w_{2i-1})$$
$$- \sum_{i=1}^{n/2} v_{2i-1}v_{2i} - \sum_{i=1}^{n/2} w_{2i-1}w_{2i}$$


- Donde estos dos valores se calculan una vez, pero se usan muchas veces.

Algoritmo de Winograd

Etapas de Preprocesamiento

```
d = b/2
// calcular los factores de las filas de la primera matriz (G)
for i = 1 to a do
    rowFactor[i] =  $G_{i,1} * G_{i,2}$ 
    for j = 2 to d do
        rowFactor[i] = rowFactor[i] +  $G_{i,2j-1} * G_{i,2j}$ 
    end for j
end for i
// calcular los factores de las columnas de la segunda matriz (H)
for i = 1 to c do
    columnFactor[i] =  $H_{1,i} * H_{2,i}$ 
    for j = 2 to d do
        columnFactor[i] = columnFactor[i] +  $H_{2j-1,i} * H_{2j,i}$ 
    end for j
end for i
```

Algoritmo de Winograd

Etapa de Cálculo

```
for i = 1 to a do
  for j = 1 to c do
     $R_{i,j} = \text{rowFactor}[i] - \text{columnFactor}[j]$ 
    for k = 1 to d do
       $R_{i,j} = R_{i,j} + (G_{i,2k-1} + H_{2k,j}) * (G_{i,2k} + H_{2k-1,j})$ 
    end for k
  end for j
end for i
// sumas de terminos para la dimension compartida impar
if  $(2 * (b / 2) \neq b)$  then
  for i = 1 to a do
    for j = 1 to c do
       $R_{i,j} = R_{i,j} + G_{i,b} * H_{b,j}$ 
    end for j
  end for i
end if
```

Análisis del Algoritmo de Winograd

- Etapa de Preprocesamiento:
 - El lazo “for j” se hace $d-1$ veces, realizando una multiplicación y una adición
 - El lazo “for i” se hace a (ó c) veces, llevando a cabo d multiplicaciones y $d-1$ adiciones
- Etapa de calculo para una dimensión par compartida (b) de las matrices:
 - El lazo “for k” se ejecuta d veces y hace una multiplicación y tres adiciones
 - El lazo “for j” se ejecuta c veces y hace d multiplicaciones y $3d + 1$ adiciones
 - El lazo “for i” se ejecuta a veces y hace $c*d$ multiplicaciones y $c*(3d + 1)$ adiciones

Análisis del Algoritmo de Winograd

- El algoritmo completo hace:

$a*d + c*d + a*c*d$ multiplicaciones

$a*(d-1) + c*(d-1) + a*c*(3d+1)$ adiciones

cuando b es par y $d = b/2$

Algoritmo de Strassen para la Multiplicación de Matrices

- Analizado ya con la técnica Divide y Vencerás:
- Usa siete formulas para multiplicar dos matrices 2×2
- No tiene en cuenta la conmutatividad de los productos de elementos
- Puede aplicarse recursivamente:
 - Dos matrices 4×4 se pueden multiplicar considerando cada una de ellas como una matriz 2×2 de matrices 2×2

Análisis del Algoritmo de Strassen

- Esas formulas requieren 7 multiplicaciones y 18 adiciones para multiplicar dos matrices 2×2
- El ahorro real se da cuando se aplica recursivamente, haciendo aproximadamente $N^{2.81}$ multiplicaciones y $6N^{2.81} - 6N^2$ adiciones
- Aunque no se usa en la práctica, el método de Strassen es importante porque fue el primer algoritmo que bajó de $O(N^3)$

Análisis del Algoritmo de Strassen

- Si suponemos que multiplicamos dos matrices de dimensiones idénticas $N \times N$, la comparativa de los tres algoritmos es la de esta tabla

Algoritmo	Multiplicaciones	Adiciones
Estándar	N^3	$N^3 - N^2$
Vinograd	$(N^3 + 2N^2) / 2$	$(3N^3 + 4N^2 - 4N) / 2$
Strassen	$N^{2.81}$	$6 N^{2.81} - 6N^2$

Ecuaciones Lineales

- Un sistema de ecuaciones lineales es un conjunto de N ecuaciones en N incógnitas.
- Los coeficientes (los valores a) son constantes y los resultados (los términos independientes) suelen ser los input de cada problema.
- Buscamos los valores x que satisfacen estas ecuaciones y producen los resultados indicados.
- Por ejemplo,

$$2x_1 - 4x_2 + 6x_3 = 14$$

$$6x_1 - 6x_2 + 6x_3 = 24$$

$$4x_1 + 2x_2 + 2x_3 = 18$$

Solución de Ecuaciones Lineales

- Un primer método de solución consiste en sustituir una ecuación en la otra
 - Por ejemplo, resolveríamos la primera ecuación para x_1 y entonces la sustituiríamos en el resto de las ecuaciones
- Esta sustitución reduce el número de ecuaciones y de incógnitas
- Iterando llegamos a una incógnita y una ecuación, de donde podremos ir recuperando los valores del resto
- Pero si hay muchas ecuaciones, el proceso es lento, produce errores y es difícil de implementar.

Algoritmo de Gauss-Jordan

- Este método se basa en la idea anterior
- Almacenamos las constantes en una matriz con N filas y $N+1$ columnas
- En nuestro ejemplo tendríamos:

2 -4 6 14

6 -6 6 24

4 2 2 18

Algoritmo de Gauss-Jordan

- Manipulamos algebraicamente las filas hasta obtener la matriz identidad en las primeras N columnas, con lo que los valores de las incógnitas se encontrarán en la última columna:

$$\begin{bmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & x_2 \\ 0 & 0 & 1 & x_3 \end{bmatrix}$$

Algoritmo de Gauss-Jordan

- En cada paso, tomamos una nueva fila y la dividimos por el primer elemento que no es cero
- Restamos múltiplos de esta fila a todas las demás para obtener todo ceros en esta columna, menos en esta fila
- Cuando esto lo hemos hecho N veces, cada fila tendrá un valor 1 y la última columna presentará los valores de las incógnitas

Ejemplo

- Consideremos de nuevo el ejemplo:

$$\begin{bmatrix} 2 & -4 & 6 & 14 \\ 6 & -6 & 6 & 24 \\ 4 & 2 & 2 & 18 \end{bmatrix}$$

- Comenzamos dividiendo la primera fila por 2, y entonces la restamos 6 veces de la segunda fila y cuatro de la tercera

Ejemplo

$$\begin{bmatrix} 1 & -2 & 3 & 7 \\ 0 & 6 & -12 & -18 \\ 0 & 10 & -10 & -10 \end{bmatrix}$$

- Ahora dividimos la segunda fila por 6, y entonces la sustraemos -2 veces de la primera y 10 veces de la tercera

Ejemplo

$$\begin{bmatrix} 1 & 0 & -1 & 1 \\ 0 & 1 & -2 & -3 \\ 0 & 0 & 10 & 20 \end{bmatrix}$$

- Ahora, dividimos la tercera fila por 10, y la restamos -1 veces de la primera y -2 veces de la segunda

Ejemplo

- Así logramos:

$$\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

- Y tenemos que $x_1 = 3$, $x_2 = 1$, y $x_3 = 2$, los valores que aparecen en el término independiente

Dificultades

- En la práctica, los redondeos producen resultados inexactos
- Si la matriz es singular, una fila será múltiplo exacto de alguna otra, dando lugar a un error al dividir por cero
- Hay mejores algoritmos para el problema, pero son propios de Análisis Numérico
- El cálculo de la inversa de la matriz de los coeficientes es un típico ejemplo de preprocesamiento: Regla de Cramer.

Fin de Curso

- Los objetivos del curso han sido
 - Dominar los métodos de cálculo de la eficiencia teórica de los algoritmos
 - Conocer en profundidad las técnicas de diseño de algoritmos y
 - Saber asociar a un problema el mejor algoritmo para su resolución
- Solo quedan dos aspectos que comentar:
 - El futuro a corto plazo (como será el examen), y
 - El futuro a medio plazo (como evaluaré el examen)

Últimas recomendaciones

- Pondré 6 preguntas:
 - Una será un ejercicio numérico que se puntuará sobre 10 y que, ponderado por 0.1 se sumará a la nota de prácticas (40% de prácticas).
 - Las otras 5 incluirán notaciones, greedy, DV, exploración de grafos y PD, y cada una puntuará sobre 10. La nota de teoría será la de la media de estas 5 preguntas ponderada por 0.6.
 - La nota final, será la suma de la nota de teoría (sobre 6) con las dos de prácticas (sobre 3 y sobre 1)
- Evaluación única: Las 6 preguntas del examen (cada una puntuada sobre 10)

Últimas recomendaciones

- El examen durará por lo menos 2 horas y media. El tiempo no será un problema
- No traigan dispositivos móviles o digitales. En concreto: calculadoras, teléfonos o cualquier dispositivo digital está prohibido en el examen.
- No se podrán usar apuntes o libros. Ahorrense su traslado
- Para preparar el examen,
 - No memoricen.
 - Estudien sobre libros y no sobre transparencias
 - Razonen el por qué de las cosas.
 - Relacionen métodos, conceptos
- Escriban claro, expresense bien, presenten bien el examen, repásenlo antes de entregarlo,

Y por último...

- Aprovechen las oportunidades que brindan los programas de intercambio
 - Erasmus (Europa)
 - LATAM, Canada, ...
 - Cooperación: CICODE
- Inicien contactos formativos sobre emprendimiento
- Contrato tácito
 - Formación continua. Mantengase en contacto con la Escuela y con sus profesores.