

CC3001

Algoritmos y Estructuras de Datos

Diseño y Análisis de Algoritmos: Casos de Estudio

Prof. Benjamin Bustos

Casos de estudio

- Estudiaremos tres problemas
 - Subsecuencia de suma máxima
 - Subsecuencia común más larga
 - Multiplicación de matrices

2

Subsecuencia de suma máxima

- Subsecuencia de suma máxima
 - Dados enteros $A_1, \dots, A_n$ (posiblemente negativos), encontrar el máximo valor de

$$\sum_{k=i}^j A_k$$

- Si todos los números son negativos, la subsecuencia de suma máxima es 0

3

Subsecuencia de suma máxima

- Ejemplo:
 - Secuencia: -2, 11, -4, 13, -5, -2
 - Respuesta: 20
- Veremos cuatro soluciones distintas para este problema
- Primera solución (fuerza bruta):
 - Calcular la suma de todas las subsecuencias
 - Quedarse con la suma mayor

4

Subsecuencia de suma máxima

- Solución 1: Fuerza bruta

```
int maxSum = 0;
for( i=0; i<a.length; i++)
{
    for( j=i; j<a.length; j++)
    {
        int thisSum = 0;
        for( k=i; k<=j; k++)
            thisSum += a[k];
        if (thisSum > maxSum)
            maxSum = thisSum;
    }
}
```

5

Subsecuencia de suma máxima

- Tiempo: $O(n^3)$

$$\sum_{i=0}^{n-1} \sum_{j=i}^{n-1} \sum_{k=i}^j 1 = \frac{n^3 + 3n^2 + 2n}{6}$$

6

Subsecuencia de suma máxima

■ Segunda solución (mejora fuerza bruta)

- Notar que

$$\sum_{k=i}^j A_k = A_j + \sum_{k=i}^{j-1} A_k$$

- Por lo tanto, el tercer ciclo **for** se puede eliminar

7

Subsecuencia de suma máxima

■ Solución 2: Mejora a fuerza bruta

```
int maxSum = 0;
for( i=0; i<a.length; i++)
{
    int thisSum = 0;
    for (j=i; j<=a.length; j++)
    {
        thisSum += a[j];
        if (thisSum > maxSum)
            maxSum = thisSum;
    }
}
```

8

Subsecuencia de suma máxima

■ Tiempo: $O(n^2)$

■ Solución 3: Usando “dividir para reinar”

- Idea: dividir el problema en dos subproblemas del mismo tamaño
- Resolver recursivamente
- Mezclar las soluciones
- Obtener solución final

9

Subsecuencia de suma máxima

■ Dividiendo el problema

- Subsecuencia de suma máxima puede estar en tres partes:
 - Primera mitad
 - Segunda mitad
 - Cruza por el medio ambas mitades

10

Subsecuencia de suma máxima

■ Dividiendo el problema

- Ejemplo:

Primera mitad	Segunda mitad
4 -3 5 -2	-1 2 6 -2

11

Subsecuencia de suma máxima

■ Dividiendo el problema

- Ejemplo:

Primera mitad	Segunda mitad
4 -3 5 -2	-1 2 6 -2

- Suma máxima primera mitad: 6

12

Subsecuencia de suma máxima

■ Dividiendo el problema

□ Ejemplo:

Primera mitad	Segunda mitad
4 -3 5 -2	-1 2 6 -2

- Suma máxima segunda mitad: 8

13

Subsecuencia de suma máxima

■ Dividiendo el problema

□ Ejemplo:

Primera mitad	Segunda mitad
4 -3 5 -2	-1 2 6 -2

- Suma máxima incluyendo último primera mitad: 4
- Idem primer elemento segunda mitad: 7
- Total: 11 (mayor que máximo en ambas mitades)

14

Subsecuencia de suma máxima

■ Algoritmo:

- Dividir secuencia en dos (izquierda, derecha)
- Resolver recursivamente las mitades
 - Caso base: secuencia de largo 1
- Calcular suma máxima centro (borde izquierdo + borde derecho)
- Retornar max{izquierda, derecha, centro}

15

Subsecuencia de suma máxima

■ Complejidad del algoritmo:

- Dos llamadas recursivas de tamaño $n/2$
- Suma máxima centro: $O(n)$
- Ecuación de recurrencia:

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

16

Subsecuencia de suma máxima

■ Tiempo: $O(n \log(n))$

■ Solución 4: Algoritmo eficiente

□ Observaciones:

- No es necesario conocer donde esta la mejor subsecuencia
- La mejor subsecuencia no puede comenzar en un número negativo
 - Cualquier subsecuencia negativa no puede ser prefijo de la subsecuencia óptima

17

Subsecuencia de suma máxima

■ Solución 4: Algoritmo eficiente

□ Inducción (reforzada)

- Se conoce la mejor subsecuencia entre 1 y j
- Se conoce la mejor subsecuencia que termina en j

□ Algoritmo

- Se almacenan ambos valores (inicialmente 0)
- Se incrementa j en 1
- Se actualiza mejor subsecuencia si es necesario
- Si subsecuencia que termina en j es < 0 se puede descartar, volver su valor a 0

18

Subsecuencia de suma máxima

■ Seudocódigo

```
int maxSum = 0, thisSum = 0;
for( j=0; j<a.length; j++)
{
    thisSum += a[j];
    if (thisSum > maxSum)
        maxSum = thisSum;
    else if (thisSum < 0)
        thisSum = 0;
}
```

19

Subsecuencia de suma máxima

■ Tiempo de la solución eficiente: $O(n)$

20

Multiplicación de matrices

■ Problema numérico fundamental

■ A, B matrices de $N \times N$

■ Se desea calcular $C = A * B$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} & a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} & a_{21} \cdot b_{12} + a_{22} \cdot b_{22} \end{bmatrix}$$

21

Multiplicación de matrices

■ Algoritmo simple:

```
// A, B: matrices de N x N
int[][] C=new int[N][N];

for( int i=0; i<n; i++) // Inicializacion
    for (int j=0; j<n; j++)
        C[i][j]=0;

for( int i=0; i<n; i++)
    for (int j=0; j<n; j++)
        for (int k=0; k<n; k++)
            C[i][j]+=A[i][k]*B[k][j];
```

22

Multiplicación de matrices

■ Tiempo algoritmo simple: $O(N^3)$

■ Por largo tiempo se supuso cota $\Omega(N^3)$

■ En los 60', Strassen mostró como romper la barrera $\Omega(N^3)$

■ Idea del algoritmo de Strassen:

- Dividir cada matriz en cuatro cuadrantes

23

Multiplicación de matrices

■ Descomposición de $AB=C$ en cuatro cuadrantes

$$\begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{bmatrix} \begin{bmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{bmatrix} = \begin{bmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{bmatrix}$$

$$\begin{aligned} C_{1,1} &= A_{1,1}B_{1,1} + A_{1,2}B_{2,1} \\ C_{1,2} &= A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ C_{2,1} &= A_{2,1}B_{1,1} + A_{2,2}B_{2,1} \\ C_{2,2} &= A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{aligned}$$

24

Multiplicación de matrices

- Se realizan 8 multiplicaciones de matrices de $N/2 \times N/2$

$$T(n) = 8T\left(\frac{N}{2}\right) + O(N^2)$$

25

Multiplicación de matrices

- Tiempo: $O(N^3)$
- Mejora: disminuir número de subproblemas
- Estrategia de Strassen:

$$\begin{aligned}M_1 &= (A_{1,2} - A_{2,2}) \cdot (B_{2,1} + B_{2,2}) \\M_2 &= (A_{1,1} + A_{2,2}) \cdot (B_{1,1} + B_{2,2}) \\M_3 &= (A_{1,1} - A_{2,1}) \cdot (B_{1,1} + B_{1,2}) \\M_4 &= (A_{1,1} + A_{1,2}) \cdot B_{2,2} \\M_5 &= A_{1,1} \cdot (B_{1,2} - B_{2,2}) \\M_6 &= A_{2,2} \cdot (B_{2,1} - B_{1,1}) \\M_7 &= (A_{2,1} + A_{2,2}) \cdot B_{1,1}\end{aligned}$$

26

Multiplicación de matrices

- Respuesta final:

$$\begin{aligned}C_{1,1} &= M_1 + M_2 - M_4 + M_6 \\C_{1,2} &= M_4 + M_5 \\C_{2,1} &= M_6 + M_7 \\C_{2,2} &= M_2 - M_3 + M_5 - M_7\end{aligned}$$

27

Multiplicación de matrices

- Complejidad ahora satisface la recurrencia

$$\begin{aligned}T(N) &= 7T\left(\frac{N}{2}\right) + O(N^2) \\T(N) &= O(N^{\log_2(7)}) = O(N^{2,81})\end{aligned}$$

28

Multiplicación de matrices

- Detalles a considerar:
 - N no es potencia de 2 (detalle menor)
 - En la práctica, algoritmo de Strassen funciona mejor que el algoritmo simple cuando N es "grande"
 - Numéricamente inestable
- Sin embargo, representa un resultado interesante desde el punto de vista teórico

29

Subsecuencia común más larga

- Problema: comparar dos secuencias de ADN
 - ADN: secuencia de moléculas llamadas bases
 - Se puede representar como un string (A, C, G, T)
- Cómo determinar si dos secuencias son similares
 - Una es *substring* de la otra
 - Costo de transformar una en otra (distancia edición)
 - Encontrar una tercera que se parezca a ambas

30

Subsecuencia común más larga

■ Definiciones

- Subsecuencia: la secuencia con cero o más elementos dejados fuera
- Formalmente:

$$X = \langle x_1, \dots, x_m \rangle, Z = \langle z_1, \dots, z_k \rangle$$

Z es subsecuencia de X si existe secuencia de índices creciente de X tal que

$$\langle i_1, \dots, i_k \rangle, \forall j = 1, \dots, k \quad x_{i_j} = z_j$$

31

Subsecuencia común más larga

■ Definiciones

- Z es subsecuencia común de X e Y si es subsecuencia de X y de Y
- Ejemplos:

$$X = \langle A, B, C, B, D, A, B \rangle, Y = \langle B, D, C, A, B, A \rangle$$

$$Z = \langle B, C, A \rangle, Z' = \langle B, C, B, A \rangle$$

- Problema: encontrar subsecuencia común más larga (LCS) de X e Y

32

Subsecuencia común más larga

■ Solución por fuerza bruta:

- Enumerar todas las subsecuencias de X
- Chequear cada una si es también subsecuencia de Y
- Guardar la subsecuencia común más larga

- X tiene 2^m subsecuencias

- Tiempo: $O(2^m)$

33

Subsecuencia común más larga

- Idea: intentar dividir el problema

- Definición: i -ésimo prefijo de X

$$X = \langle x_1, \dots, x_m \rangle$$

$$X_i = \langle x_1, \dots, x_i \rangle, i = 1, \dots, m$$

- Subproblemas de LCS: prefijos de X e Y

34

Subsecuencia común más larga

■ Teorema: Subestructura óptima de una LCS

- $X(m)$ e $Y(n)$ secuencias, $Z(k)$ una LCS de X e Y

1. $x_m = y_n \Rightarrow z_k = x_m = y_n \wedge Z_{k-1}$ es LCS de X_{m-1} e Y_{n-1}
2. $x_m \neq y_n \Rightarrow z_k \neq x_m$ implica que Z es LCS de X_{m-1} e Y
3. $x_m \neq y_n \Rightarrow z_k \neq y_n$ implica que Z es LCS de X e Y_{n-1}

35

Subsecuencia común más larga

- Teorema implica revisar uno o dos subproblemas
- La solución del subproblema es parte de la solución final (óptima)
- Nota: Encontrar LCS de casos (2) y (3) del Teorema implica calcular LCS de X_{m-1} e Y_{n-1}
 - Muchos subproblemas comparten otros subproblemas
 - Total subproblemas distintos: $m \cdot n$

36

Subsecuencia común más larga

- Solución: Programación dinámica

- Definición: Matriz C de $m \times n$

$$c[i, j] = \begin{cases} 0 & \text{si } i = 0 \text{ o } j = 0 \\ c[i-1, j-1] + 1 & \text{si } i, j > 0 \text{ y } x_i = y_j \\ \max\{c[i, j-1], c[i-1, j]\} & \text{si } i, j > 0 \text{ y } x_i \neq y_j \end{cases}$$

- Algoritmo: llenar tabla en forma *bottom-up*

37

Subsecuencia común más larga

- Implementación:

```
m=X.length-1; n=Y.length-1; // indices 1 a m,n
for(i=1; i<=m; i++) c[i,0]=0;
for(j=0; j<=n; j++) c[0,j]=0;

for(i=1; i<=m; i++)
  for(j=1; j<=n; j++)
    if (X[i]==Y[j]){
      c[i,j]=c[i-1,j-1]+1; b[i,j]="\n";
    }
    else if (c[i-1,j]>c[i,j-1]){
      c[i,j]=c[i-1,j]; b[i,j]="\n";
    }
    else{
      c[i,j]=c[i,j-1]; b[i,j]="-"}

return {c,b};
```

38

Subsecuencia común más larga

- Ejemplo:

- Para imprimir LCS

```
void LCS(b,X,i,j){
  if (i==0 || j==0)
    return;
  if (b[i,j]=="\n"){
    LCS(b,X,i-1,j-1);
    print(X[i]);
  }
  else if (b[i,j]=="-")
    LCS(b,X,i-1,j);
  else \ " "
    LCS(b,X,i,j-1);
}
```

	j	0	1	2	3	4	5	6
i	y _j	B	D	C	A	B	A	
0	x _i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

39