Longest Increasing Subsequence (LIS), Longest Common Substring, Longest Common Subsequence (LCS)

Pedro O. Pérez M., PhD.

Análisis y diseño de algoritmos avanzados
Tecnológico de Monterrey

pperezm@tec.mx

08-2022

Contenido

¿Qué es una subsecuencia?

Considera una cadena A = [a, b, c, d]. Una subsecuencia(también llamada subcadena) se obtiene eliminando 0 o más símbolos (no necesariamente consecutivos) de A.
Por ejemplo:

- Los ejemplos [a, b, c, d] , [a], [b], [c], [d], [a, d], [a, c], [b, d] son subsecuencias de A.
- Pero [d, b], [d, a], [d, a, c] no son subsecuencias de A.

Subsecuencia creciente más larga

Dado un arreglo de n símbolos comparables, dar la subsecuencia creciente más larga. Por ejemplo, A = [-7, 10, 9, 2, 3, 8, 8, 1], la subsecuencia creciente más larga es [-7, 2, 3, 8].

Procedure 1 LIS

Input: $A : Array$

```
Aux : Array
Aux[1] ← 1
for i ← 2 to n do
    count ← 0
    for j ← 1 to (i − 1) do
        if A[j] < A[i] then
            count ← MAX(count, Aux[j])
        end if
    end for
    Aux[i] ← count + 1
end for
```

```
count ← 0
for i ← 1 to n do
    count ← MAX(count, Aux[i])
end for
return count
```

Testing the CATCHER [1]

Un contratista militar del Departamento de Defensa acaba de completar una serie de pruebas preliminares para un nuevo misil defensivo llamado FREEDOM, que es capaz de interceptar múltiples misiles ofensivos entrantes. Se supone que el FREEDOM es un notable misil defensivo. Puede moverse hacia adelante, lateralmente y hacia abajo a velocidades muy rápidas, y puede interceptar un misil ofensivo sin sufrir daños. Pero tiene un defecto importante. Aunque puede dispararse para alcanzar cualquier elevación inicial, no tiene poder para moverse más alto que el último misil que ha interceptado.
Las pruebas que completó el contratista fueron simulaciones por computadora de campo de batalla y condiciones de ataque hostil. Como solo eran preliminares, las simulaciones solo probaron la capacidad de movimiento vertical del FREEDOM. En cada simulación, el FREEDOM fue disparado contra una secuencia de misiles ofensivos que llegaban a intervalos de tiempo fijos. La única información disponible para el FREEDOM para cada misil entrante era su altura en el punto donde podía ser interceptada y donde aparecía en la secuencia de misiles. Cada misil entrante para una ejecución de prueba se representa en la secuencia solo una vez.

[1] `https://onlinejudge.org/external/2/231.pdf`

El resultado de cada prueba se informa como la secuencia de misiles entrantes y el número total de esos misiles que son interceptados por el FREEDOM en esa prueba.
La Oficina de Contabilidad General quiere asegurarse de que los resultados de la prueba de simulación presentados por el contratista militar sean alcanzables, dadas las limitaciones del FREEDOM. Escribir una solución que tome datos de entrada que representen el patrón de misiles entrantes para una prueba y genere la cantidad máxima de misiles que el FREEDOM puede interceptar para esa prueba. Para cualquier misil entrante en la prueba, el FREEDOM puede interceptarlo si y solo si cumple una de estas dos condiciones:
1. El misil entrante es el primer misil que se intercepta en esta prueba.
-o-
2. El misil fue disparado después del último misil que fue interceptado y no es más alto que el último misil que fue interceptado.

Entrada

Los datos de entrada para cualquier prueba consisten en una secuencia de uno o más enteros no negativos, todos los cuales son menores o iguales a 32,767, que representan las alturas de los misiles entrantes (el patrón de prueba). El último número en cada secuencia es -1, lo que significa el final de los datos para esa prueba.

Salida

La salida consta del número máximo de misiles entrantes que el FREEDOM posiblemente podría interceptar para la prueba. Nota: El número de misiles para cualquier prueba dada no está limitado. Si la solución se basa en un algoritmo ineficiente, es posible que no se ejecute en el tiempo asignado.

Ejemplo de entrada

```
389
207
155
300
299
170
158
65
-1
```

Ejemplo de salida

```
6
```

Subsecuencia común más larga

Dado dos cadenas, A y B, determinar la subsecuencia común más larga de ambas cadenas. Por ejemplo, si A = [c, d, c, c, f, g, e] y B = [e, c, c, e, g, f, e], ¿cuál la subsecuencia común más larga?

Ahora veamos como se plantea el problema: considere dos secuencias A = [a_1, a_2, ..., a_m] y B = [b_1, b_2, ..., b_n] . Para resolver el problema nos fijaremos en los últimos dos símbolos: a_m y b_n . Como podemos ver hay dos posibilidades:

- Caso 1: $a_m = b_n$. En este caso, la subsecuencia común más larga debe contener a_m . Simplemente basta encontrar la subsecuencia común más larga de [a_1, a_2, ..., a_{m-1}] y [b_1, b_2, ..., b_{n-1}].
- Caso 2: $a_m \neq b_n$. En este caso, puede hacerse corresponder [a_1, a_2, ..., a_m] con [b_1, b_2, ..., b_{n-1}] y también [a_1, a_2, ..., a_{m-1}] con [b_1, b_2, ..., b_n], y nos quedamos con el mayor de los dos resultados.

Procedure 2 LCS

Input: $A, B : Array$
  $M : Matrix[0..A.length][0..B.length]$
  $m \leftarrow A.length$
  $n \leftarrow B.length$
  $INIT(M, 0)$
  for $i \leftarrow 1$ to m **do**
    for $j \leftarrow 1$ to n **do**
      if $A[i] = B[j]$ **then**
        $M[i][j] \leftarrow 1 + M[i-1][j-1]$
      else
        $M[i][j] \leftarrow MAX(M[i-1][j], M[i][j-1])$
      end if
    end for
  end for
  return $M[m][n]$

History Grading [2]

Muchos problemas en Ciencias de la Computación implican maximizar alguna medida de acuerdo con restricciones. Considere un examen de historia en el que se pide a los estudiantes que pongan varios eventos históricos en orden cronológico. Los estudiantes que ordenen todos los eventos correctamente recibirán crédito completo, pero ¿cómo se debe otorgar crédito parcial a los estudiantes que clasifiquen incorrectamente uno o más de los eventos históricos? Algunas posibilidades de crédito parcial incluyen:
1. 1 punto por cada evento cuyo rango coincida con su rango correcto.
2. 1 punto por cada evento en la secuencia más larga (no necesariamente contigua) de eventos que estén en el orden correcto entre sí.
Por ejemplo, si cuatro eventos están ordenados correctamente 1 2 3 4, entonces el orden 1 3 2 4 recibiría una puntuación de 2 usando el primer método (los eventos 1 y 4 están clasificados correctamente) y una puntuación de 3 usando el segundo método (evento las secuencias 1 2 4 y 1 3 4 están ambas en el orden correcto entre sí). En este problema, se le pide que escriba un programa para calificar tales preguntas utilizando el segundo método.

[2] https://onlinejudge.org/external/1/111.pdf

Dado el orden cronológico correcto de n eventos 1, 2, ..., n como c_1, c_2,. . . c_n donde $1 \geq c_i \geq$ n denota la clasificación del evento i en el orden cronológico correcto y una secuencia de respuestas de los estudiantes r_1, r_2, ..., r_n donde $1 \geq$ ri $\geq$ n denota el rango cronológico dado por el estudiante al evento i; determinar la longitud de la secuencia más larga (no necesariamente contigua) de eventos en las respuestas de los estudiantes que están en el orden cronológico correcto entre sí.

Entrada

El archivo de entrada contiene uno o más casos de prueba, cada uno de ellos como se describe a continuación.

La primera línea de la entrada constará de un entero n que indica el número de eventos con 2 $\geq$ n $\geq$ 20. La segunda línea contendrá n números enteros, indicando el orden cronológico correcto de n eventos. Las líneas restantes constarán cada una de n números enteros y cada línea representará el orden cronológico de los n eventos por parte del estudiante. Todas las líneas contendrán n números en el rango [1...n], con cada número apareciendo exactamente una vez por línea, y con cada número separado de otros números en la misma línea por uno o más espacios.

Salida

Para cada caso de prueba, la salida debe seguir la descripción a continuación Para cada clasificación de eventos de los estudiantes, su programa debe imprimir la puntuación de esa clasificación. Debe haber una línea de salida para cada clasificación de estudiantes.

Advertencia: lea atentamente la descripción y considere la diferencia entre .ºrdenarz çlasificar".

Ejemplo de entrada

```
4
4 2 3 1
1 3 2 4
3 2 1 4
2 3 4 1
10
3 1 2 4 9 5 10 6 8 7
1 2 3 4 5 6 7 8 9 10
4 7 2 3 10 6 9 1 5 8
3 1 2 4 9 5 10 6 8 7
2 10 1 3 8 4 9 5 7 6
```

Ejemplo de salida

```
1
2
3
6
5
10
9
```

Subcadena común más larga

El problema de la subcadena común más larga (Longest Common Substring) consiste en encontrar aquella subcadena continua que puede coincidir en todos las cadenas de entrada no necesariamente del mismo tamaño. Este problema puede arrojar múltiples soluciones. Por ejemplo, teniendo los strings: “AABCAB” y “CABCBABACC”, el tamaño de la subcadena más larga es 3: “ABC”, “CAB” y “ABA”.

Ahora veamos como se plantea el problema: considere dos secuencias A = [a_1, a_2, ..., a_m] y B = [b_1, b_2, ..., b_n] . Para resolver el problema nos fijaremos en los últimos dos símbolos: a_m y b_n . Como podemos ver hay dos posibilidades:

- Caso 1: $a_m = b_n$. En este caso, la subcadena común más larga debe contener a_m . Primero debemos encontrar la subcadena común más larga de [a_1, a_2, ..., a_{m-1}] y [b_1, b_2, ..., b_{n-1}] + 1, y luego comparar ese resultado con el máximo previo.
- Caso 2: $a_m \neq b_n$. En este caso, la longitud más larga será cero.

Procedure 3 LCS

Input: $A, B : Array$
  $M : Matrix[0..A.length][0..B.length]$
  $m \leftarrow A.length$
  $n \leftarrow B.length$
  for $i \leftarrow 1$ to m **do**
    if $A[i] = B[0]$ **then**
      $M[i][0] \leftarrow 1$, $maximum \leftarrow 1$
    end if
  end for
  for $i \leftarrow 1$ to n **do**
    if $A[0] = B[i]$ **then**
      $M[i][0] \leftarrow 1$, $maximum \leftarrow 1$
    end if
  end for

```
for i ← 1 to m do
    for j ← 1 to n do
        if A[i] = B[j] then
            M[i][j] ← 1 + M[i − 1][j − 1]
            maximum ← MAX(M[i][j], max)
        else
            M[i][j] ← 0
        end if
    end for
end for
return maximum
```