Practica 4

Algoritmos de memoria dinámica.

Fco Javier Merchán

1. Análisis

Realizar una búsqueda bibliográfica del problema a resolver, su formulación teórica y posibles alternativas de solución. Indicar si el problema es resoluble mediante la técnica de algoritmos de Programación Dinámica. Si lo es, resuelva el problema con esta técnica, describiendo el problema, qué decisiones se han tomado para abordarlo o acotarlo, y justifíquelas.

El problema que se nos presenta es la comparación de señales, es decir como de parecida son dos secuencias de números dadas. Para ello utilizaremos DTW, que viene a ser un algoritmo que calcula la coincidencia óptima entre dos secuencias, ambas no lineales en una dimensión de tiempo. También se podría calculas con algoritmos como las subsecuencia más larga no consecutiva, el cual es otro problema de la relación dada, o el coeficiente de correlación de Perarson. Este coeficiente se usa en estadística y compara el grado de similitud de dos conjunto de datos. Otras maneras de aproximación son la restricción de Sakoe-Chiba y la restricción de Itakura Parallelogram.

El algoritmo de DTW se usa a menudo en multitud de tareas, como por ejemplo para tareas de reconocimiento de voz, pulsaciones o pulsometros, entre otras.

En este caso, es posible abordar el problema con la técnica de Programación dinámica ya que representa las ideas principales de esta técnica, como son:

- Es un problema de optimización, ya que busca el máximo parecido de las dos cadenas
- Es posible expresar como una ecuación recurrente
- Para resolver un problema, este se puede dividir en etapas y se deben resolver los subproblemas de menor tamaño para resolver el problema de mayor tamaño
- Los subproblemas se solapan entre sí, este es el principal punto de inflexión con la técnica Divide y vencerás, la cual especifica que los problemas son disjuntos. En este caso, como es buscar una subsecuencia, entre dos subproblemas puede haber una subsecuencia entre estos.
- Hay un caso base, en este caso en cual las cadenas, ambas o unas sola, son de longitud 0.
- Cumple el principio de optimalidad de Bellman.

Para ver si cumple o no el principio de optimalidad de Bellman vamos a verlo con un ejemplo:

Para ello vamos a coger el caso base, el cual es óptimo trivial, ya que es un caso base. Después cogemos el caso de (1,1), el cual sigue siendo óptimo ya que coge el mínimo entre los casos bases que a su vez son óptimos. Asi vamos formando una cadena d decisiones hasta llegar a (i,j), el cual es óptimo ya que el algoritmo siempre ha tomado las decisiones optimas en cada subcaso. Por reducción a lo absurdo llegaríamos a una incoherencia ya que si cambiar uno de las decisiones del algoritmo ya no llegarías al óptimo.

2. Diseño (I)

Especificar las componentes del algoritmo, justificando su selección.

Para el algoritmo tendremos en cuenta las siguientes componentes:

- Una matriz que cuadrara los resultado de cada ecuación aplicada para el caso de DTW[i][j], rellena por valores no válidos, en este caso de -1.
- Dos secuencias de números reales ‘X’ y ‘Y’, de longitud i y j respectivamente.
- La solución DTW para los casos en los que una de las dos secuencias es de longitud 0 es de ‘0’ ya que se estaría comprando nada.
- DTW[i][j] será la resta en valor absoluto de la posición de las sucesiones X[i] y Y[j], más el mínimo de DTW[i-1][j], DTW[i][j-1] y DTW[i-1][j-1].

- La ecuación recurrente seria:

$$DTW[i][j] \begin{cases} 0 & \rightarrow \; i <= 0 \;||\; j <= 0 \\ |X[i] - Y[j]| + \min(DTW[i-1][j],\; DTW[i][j-1],\; DTW[i-1][j-1]) & \rightarrow \; i > 0 \;\&\&\; j > 0 \end{cases}$$

3. Diseño (II)

Explique cómo ha organizado la memoria para resolver el problema, y cómo se debe actualizar dicha memoria de acuerdo a la ecuación en recurrencias diseñada en el apartado anterior. Finalmente, exponga el algoritmo diseñado.

Para resolver este problema básicamente hemos cogido la ecuación recurrente y hemos ido rellenando la matriz con la solución de esta. Por lo tanto si se cambia la ecuación en recurrencias se tendría que cambiar prácticamente todo, ya que se tendría que volver a comprobar que se cumple el POB. Después si llega el caso modificar tanto el algoritmo como los casos bases.

El algoritmo quedaría tal que asi:

Procedure DTW(Vector X[0...n].,Vector Y[0....m], Matriz &sol[0...n+1][0.....m+1])
Para i=0 hasta n+1 hacer:
T[0][i]= 0; T[i][0]= 0
Fin-Para
Para i=1 hasta n+1 hacer:
Para j=1 hasta m+1 hacer:
sol[i][j] = |X[i-1]-Y[j-1]| más el min{sol[i-1][j], sol[i][j-1],sol[i-1][j-1]}
Fin-Para
Fin-Para
Devolver DTW[n][m];

Este algoritmo básicamente rellena la primera fila y columna con 0, ya que ese es el caso base, y a continuación va haciendo cada caso hasta llegar al final de ambos vectores, rellenando asi toda la tabla solución y devolviendo el ultimo valor de la tabla. El cual dice como se parecen entre si las dos secuencias de números. Como argumentos a la función, le pasaríamos las dos secuencias, asi como sus tamaños si no se puede acceder a ellos, y una matriz en la que iremos guardando todas las soluciones parciales y cálculos hechos anteriormente. Si nos fijamos la matriz tiene un tamaño igual al delos vectores, uno se identifica en sus columnas y otro por sus filas.

4. Implementación

Indique cómo se han implementado cada una de las componentes del algoritmo diseñado en el apartado anterior. Adjunte a la memoria de prácticas, en un fichero separado, el código fuente. Indique en este apartado también cómo compilar el código y cómo ejecutarlo.

La función es muy parecida al algoritmo que hemos expuesto anteriormente, pero con algunos cambias para adaptarlo al idioma de C++. Entre estos cambios es usar la librería de la STL para dar lugar a la una versión eficiente de los vectores, asi como la librería *stdlib* para la función de min y de valor absoluto. Por lo demás funciona de igual manera.

```
double DTW(vector<double> X, vector<double> Y, vector<vector<double> > &sol){
  for(int i=0; i<sol.size();i++){
    sol[i][0] = 0;
  }
  for(int i=0; i<sol[0].size();i++){
    sol[0][i] = 0;
  }
  for (int i =1 ; i < sol.size();i++){
    for(int j=1; j < sol[i].size();j++){
      sol[i][j] = abs(X[i-1]-Y[j-1]) + min(sol[i-1][j-1],min(sol[i-1][j],
                        sol[i][j-1]));

    }
  }

return sol[X.size()][Y.size()];
}
```

Para ejecutar el algoritmo basta con poner en la consola la llamada al *makefile*.
Después, pedirá la longitud de las dos cadenas y generará números aleatorios, mostrará por pantalla las dos cadenas de caracteres y después mostrará la matriz resultante con el resultado de la consulta.

5. Pruebas

Proporcione un ejemplo de ejecución del algoritmo. Indique cómo ejecutar el código proporcionado para resolver los ejemplos. Explique, a mano, cómo funciona el algoritmo resolviendo una traza del algoritmo para el caso de ejemplo dado.

Como hemos dicho anteriormente el algoritmo va rellenando línea a línea con los datos que ya tiene y el resultado lo vuelve a guardar para futuras operaciones. Por ejemplo:

- X -> 2, 5, 5, 1, 2
- Y -> 8, 5, 1, 0, 5, 1

El algoritmo rellena las primeras líneas y columna como 0, y la primera línea la rellenara de la siguiente forma:

- *DTW[1][1] = | X[1] - Y[1] | + min(DTW[0][1], DTW[1][0], DTW[0][0]).*
  - *El resultado sería |2 - 8|+ min(0, 0, 0), el resultado seria 6.*
- *DTW[1][2] = | X[1] - Y[2] | + min(DTW[0][2], DTW[1][1], DTW[0][1]).*
  - *El resultado sería |2 - 5|+ min(0, 6, 0), el resultado seria 3.*
- *DTW[1][3] = | X[1] - Y[3] | + min(DTW[0][3], DTW[1][2], DTW[0][2]).*
  - *El resultado sería |2 - 1|+ min(0, 3, 0), el resultado seria 1.*
- *DTW[1][4] = | X[1] - Y[4] | + min(DTW[0][4], DTW[1][3], DTW[0][3]).*
  - *El resultado sería |2 - 0|+ min(0, 1, 0), el resultado seria 2.*
- *DTW[1][5] = | X[1] - Y[5] | + min(DTW[0][5], DTW[1][4], DTW[0][4]).*
  - *El resultado sería |2 - 5|+ min(0, 2, 0), el resultado seria 3.*
- *DTW[1][6] = | X[1] - Y[6] | + min(DTW[0][6], DTW[1][5], DTW[0][5]).*
  - *El resultado sería |2 - 1|+ min(0, 3, 0), el resultado seria 1.*

La segunda fila la rellenara con:

- *DTW[2][1] = | X[2] - Y[1] | + min(DTW[1][1], DTW[2][0], DTW[1][0]).*
  - *El resultado sería |5 - 8|+ min(6, 0, 0), el resultado seria 3.*
- *DTW[2][2] = | X[2] - Y[2] | + min(DTW[1][2], DTW[2][1], DTW[1][1]).*
  - *El resultado sería |5 - 5|+ min(3, 3, 6), el resultado seria 3.*
- *DTW[2][3] = | X[2] - Y[3] | + min(DTW[1][3], DTW[2][2], DTW[1][2]).*
  - *El resultado sería |5 - 1|+ min(1, 3, 3), el resultado seria 5.*
- *DTW[2][4] = | X[2] - Y[4] | + min(DTW[1][4], DTW[2][3], DTW[1][3]).*
  - *El resultado sería |5 - 0|+ min(2, 5, 1), el resultado seria 6.*
- *DTW[2][5] = | X[2] - Y[5] | + min(DTW[1][5], DTW[2][4], DTW[1][4]).*
  - *El resultado sería |5 - 5|+ min(3, 6, 2), el resultado seria 2.*
- *DTW[2][6] = | X[2] - Y[6] | + min(DTW[1][6], DTW[2][5], DTW[1][5]).*
  - *El resultado sería |5 - 1|+ min(1, 2, 3), el resultado seria 5.*

La tercera fila la rellenara con:

- *DTW[3][1] = | X[3] - Y[1] | + min(DTW[2][1], DTW[3][0], DTW[2][0]).*
  - *El resultado sería |5 - 8|+ min(3, 0, 0), el resultado seria 3.*
- *DTW[3][2] = | X[3] - Y[2] | + min(DTW[2][2], DTW[3][1], DTW[2][1]).*
  - *El resultado sería |5 - 5|+ min(3, 3, 3), el resultado seria 3.*
- *DTW[3][3] = | X[3] - Y[3] | + min(DTW[2][3], DTW[3][2], DTW[2][2]).*
  - *El resultado sería |5 - 1|+ min(5, 3, 3), el resultado seria 7.*
- *DTW[3][4] = | X[3] - Y[4] | + min(DTW[2][4], DTW[3][3], DTW[2][3]).*
  - *El resultado sería |5 - 0|+ min(6, 7, 5), el resultado seria 10.*
- *DTW[3][5] = | X[3] - Y[5] | + min(DTW[2][5], DTW[3][4], DTW[2][4]).*
  - *El resultado sería |5 - 5|+ min(2, 10,),6 el resultado seria 2.*
- *DTW[3][6] = | X[3] - Y[6] | + min(DTW[2][6], DTW[3][5], DTW[2][5]).*
  - *El resultado sería |5 - 1|+ min(5, 2, 2), el resultado seria 6.*

La cuarta fila la rellenara con:

- *DTW[4][1] = | X[4] - Y[1] | + min(DTW[3][1], DTW[4][0], DTW[3][0]).*
    - *El resultado sería |1 - 8|+ min(3, 0, 0), el resultado seria 7.*
- *DTW[4][2] = | X[4] - Y[2] | + min(DTW[3][2], DTW[4][1], DTW[3][1]).*
    - *El resultado sería |1 - 5|+ min(3, 7, 3), el resultado seria 7.*
- *DTW[4][3] = | X[4] - Y[3] | + min(DTW[3][3], DTW[4][2], DTW[3][2]).*
    - *El resultado sería |1 - 1|+ min(7, 7, 3), el resultado seria 3.*
- *DTW[4][4] = | X[4] - Y[4] | + min(DTW[3][4], DTW[4][3], DTW[3][3]).*
    - *El resultado sería |1 - 0|+ min(10, 3,),7 el resultado seria 4.*
- *DTW[4][5] = | X[4] - Y[5] | + min(DTW[3][5], DTW[4][4], DTW[3][4]).*
    - *El resultado sería |1 - 5|+ min(2, 4, 1),0 el resultado seria 6.*
- *DTW[4][6] = | X[4] - Y[6] | + min(DTW[3][6], DTW[4][5], DTW[3][5]).*
    - *El resultado sería |1 - 1|+ min(6, 6, 2), el resultado seria 2.*

La última fila la rellenara con:

- *DTW[5][1] = | X[5] - Y[1] | + min(DTW[4][1], DTW[5][0], DTW[4][0]).*
    - *El resultado sería |2 - 8|+ min(7, 0, 0), el resultado seria 6.*
- *DTW[5][2] = | X[5] - Y[2] | + min(DTW[4][2], DTW[5][1], DTW[4][1]).*
    - *El resultado sería |2 - 5|+ min(7, 6, 7), el resultado seria 9.*
- *DTW[5][3] = | X[5] - Y[3] | + min(DTW[4][3], DTW[5][2], DTW[4][2]).*
    - *El resultado sería |2 - 1|+ min(3, 9, 7), el resultado seria 4.*
- *DTW[5][4] = | X[5] - Y[4] | + min(DTW[4][4], DTW[5][3], DTW[4][3]).*
    - *El resultado sería |2 - 0|+ min(4, 4, 3), el resultado seria 5.*
- *DTW[5][5] = | X[5] - Y[5] | + min(DTW[4][5], DTW[5][4], DTW[4][4]).*
    - *El resultado sería |2 - 5|+ min(6, 5, 4), el resultado seria 7.*
- *DTW[5][6] = | X[5] - Y[6] | + min(DTW[4][6], DTW[5][5], DTW[4][5]).*
    - *El resultado sería |2 - 1|+ min(2, 7, 6), el resultado seria 3.*

La matriz solución quedaría de la siguiente manera, de la cual resaltamos las opciones elegidas con el valor mínimo elegido por el algoritmo para llegar a nuestra solución, sabiendo asi como va construyendo el resultado.

0	0	0	0	0	0	0
0	6	3	1	2	3	1
0	3	3	5	6	2	5
0	3	3	8	10	2	6
0	7	7	3	4	6	2
0	6	9	4	5	7	3

Hemos resaltado en naranja la solución al algoritmo. Y en azul las opciones elegidas por el algoritmo como el mínimo.