Tutorial 1

Divyang Mittal(17cs10012)

22-07-2019

1 Problem Statement

$A[1..m]$ and $B[1..n]$ are two 1D arrays containing m and n integers respectively, where $m \leq n$. We need to construct a sub-array $C[1..m]$ of B such that $\sum\limits_{i=1}^{m} \left|A[i] - C[i]\right|$ is minimized.

2 Recurrences

The array C is a subsequence of array B such that it's length is equal to n. So the subproblem can be defined as dp[i][j] which is the solution for $B[1..i]$ and $A[1..j]$ which gives us the subsequence $C[1..j]$ of $B[1..i]$ such that $\sum\limits_{k=1}^{j} \left|A[k] - C[k]\right|$ is minimized. From this we can easily say that our solution is dp[n][m].

There are two cases for any (i,j) pair with $i >= j$ based on the situation if we consider the element at index i from array B or not for solution C.

Therefore we get,

$$
\begin{aligned}
\text{dp[i][j]} &= 0 && \text{if j = 0} \\
&= \text{infinity} && \text{else if i<j} \\
&= \min\left(\text{dp[i-1][j-1]} + \left|A[j] - B[i]\right|,\ \text{dp[i-1][j]}\right) && \text{otherwise}
\end{aligned}
$$

3 Algorithm

Since this has overlapping subproblem property we are calculating many of the dp[i][j] repeatedly. We will calculate dp[i][j] for each pair(i,j) in a bottom-up approach and store it in a 2D array so we need not calculate it again.

Additionally an array D is used to store the path in order to build the solution.

```
//Initializing the dp 2D array
//Initialize D array to zero for all (i,j) pair
for all i in 0..n do
   for all j in i..m do
      dp[i][j] = MAX INT
   end for
   dp[i][0] = 0
end for
//Calculating dp and D
for all i in 1..n do
   for all j in 1..min(i,m) do
      if dp[i-1][j-1] + |A[j] − B[i]| < dp[i-1][j] then
         dp[i][j] = dp[i-1][j-1] + |A[j] − B[i]|
         D[i][j] = 1
      else
         dp[i][j] = dp[i-1][j]
      end if
   end for
end for
//Now building the array C
j = m
for all i in n..1 do
   if D[i][j] = 1 then
      C[j] = B[i]
      j = j - 1
   end if
end for
```

4 Demonstration

Take the following example,
A = 5, 4, 7 and B = 4, 3, 10, 5, 6

On initialization, dp array becomes

$$\begin{pmatrix} 0 & \infty & \infty & \infty \\ 0 & - & \infty & \infty \\ 0 & - & - & \infty \\ 0 & - & - & - \\ 0 & - & - & - \\ 0 & - & - & - \end{pmatrix}$$

Then after 2 iterations of i the table becomes,

$$\begin{pmatrix} 0 & \infty & \infty & \infty \\ 0 & 1 & \infty & \infty \\ 0 & 1 & 2 & \infty \\ 0 & - & - & - \\ 0 & - & - & - \\ 0 & - & - & - \end{pmatrix}$$

Finally the table becomes,

$$\begin{pmatrix} 0 & \infty & \infty & \infty \\ 0 & 1 & \infty & \infty \\ 0 & 1 & 2 & \infty \\ 0 & 1 & 2 & 5 \\ 0 & 1 & 2 & 4 \\ 0 & 1 & 2 & 3 \end{pmatrix}$$

and array D becomes,

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Hence Solution C becomes,

$$\begin{pmatrix} 4 & 3 & 6 \end{pmatrix}$$

5 Time and space complexities

Time Complexity
Size of the dp table is (m+1)*(n+1). The time taken to fill each entry is O(1) so the time taken is (m+1)*(n+1)*O(1) = O(mn). Also time taken to build solution C is O(n). Hence, Time complexity of the solution is sum of the two that is O(mn).

Space Complexity
Size of the table dp is (m+1)*(n+1), array D is (m+1)*(n+1) and array C is m. Hence, space complexity becomes sum of the three that is O(mn).