Lab05-DynamicProgramming

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2021.

∗ If there is any problem, please contact TA Haolin Zhou.

∗ Name:Xin Xu   Student ID:519021910726   Email: xuxin20010203@sjtu.edu.cn

1. *Optimal Binary Search Tree.* Given a sorted sequence $K = \langle k_1, k_2, \dots, k_n \rangle$ of n distinct keys, and we wish to build a binary search tree from these keys. For each key k_i, we have a probability p_i that a search will be for k_i. Some searches may be for values not in K, and so we also have $n+1$ *dummy keys* $d_0, d_1, d_2, \dots, d_n$ representing values not in K. In particular, d_0 represents all values less than k_1, and d_n represents all values greater than k_n. For $i = 1, 2, \dots, n-1$, the dummy key d_i represents all values between k_i and k_{i+1}. For each dummy key d_i, we have a probability q_i that a search will correspond to d_i. Each key k_i is an internal node, and each dummy key d_i is a leaf. Every search is either successful (finding some key k_i) or unsuccessful (finding some dummy key d_i), and so we have $\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1$.

   (a) Prove that if an optimal binary search tree T (T has the smallest expected search cost) has a subtree T' containing keys $k_i, \dots, k_j$, then this subtree T' must be optimal as well for the subproblem with keys $k_i, \dots, k_j$ and dummy keys $d_{i-1}, \dots, d_j$.

   (b) We define $e[i, j]$ as the expected cost of searching an optimal binary search tree containing the keys $k_i, \dots, k_j$. Our goal is to compute $e[1, n]$. Write the state transition equation and pseudocode using **dynamic programming** to find the minimum expected cost of a search in a given binary tree. (**Remark**: You may use $w(i, j) = \sum_{l=i}^{j} p_l + \sum_{l=i-1}^{j} q_l$).

   (c) Implement your proposed algorithm in C/C++ and analyze the time complexity. (The framework Code-OBST.cpp is attached on the course webpage). Give the minimum search cost calculated by your algorithm. The test case is given as following:

i	0	1	2	3	4	5	6	7
p_i		0.04	0.06	0.08	0.02	0.10	0.12	0.14
q_i	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05

   (d) Please draw the structure of the optimal binary search tree in the test case, and explain the drawing process.

Solution. (a) **Proof.** We can prove it by contradiction. If an optimal binary search tree T has a subtree T' which is not optimal, then we can contruct an optimal subtree T^* and replace T' by T^*. And this replacement doesn't change the output of the research but results in a better performance. So the tree T is not optimal, and our hypothesis is wrong. Thus everysubtree of an optimal binary tree is optimal. □

(b) With the definition of $e[i, j]$ and $w(i, j)$ and the conclusion of **problem (a).**, the recursive equation of $e[i, j]$ can be defined as:
$e[i, j] = w(i, j) + min(e[i, k-1] + e[k+1, j]), i \leqslant k \leqslant j,$ and $e[i, j] = 0$ for any integer $i > j$.

Algorithm 1: Optimal Binary Search Tree

Input: a sorted sequence $K =< k_1, k_2, k_3, \dots, k_n >$, and the probability P of K that $P =< p_1, p_2, p_3, \dots, p_n >$ plus the probability of dummy keys $Q =< q_1, q_2, q_3, \dots, q_n >$.

Output: Optimal binary search tree and minimum cost.

for $i = 1$ **to** n **do**
  for $j = i$ **to** n **do**
    compute $w(i,j), w(i,j) = \sum_{l=i}^{j} p_l + \sum_{l=i-1}^{j} q_l$;

for $i = 0$ **to** $n-1$ **do**
  for $j = 1$ **to** $n-i$ **do**
    $e[j, j+i] = w(j, j+i) + min_{j \leqslant k \leqslant j+i}(e[j, k-1] + e[k+1, j+i])$;

Output $e[1, n]$;

(c) The time complexity to compute $w(i,j)$ is $O(n^3)$. The time complexity to compute $e[i][j]$ is $O(n^3)$, too. The time complexity to construct an optimal binary search tree is $O(n \log n)$ according to the conclusion of recursive tree. So, the total time complexity of OBST is $O(n^3)$. And the minimum search cost calculated by my algorithm is 2.68.

(d) Everytime the function print_tree print the root of given range and invokes itself recursively. There is a bool parameter to determine whether the print one is the left child or right child. when the range is illegal, we print dummy key d_i as the return state. The structure of the optimal binary search tree is below.

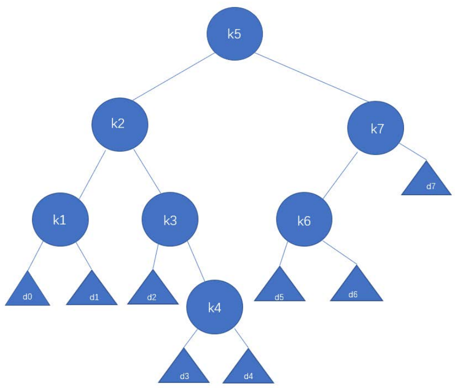

Figure 1: The Optimal Binary Search Tree

□

2. *Dynamic Time Warping Distance.* **DTW** stretches the series along the time axis in a dynamic way over different portions to enable more effective matching. Let $DTW(i,j)$ be the optimal distance between the first i and first j elements of two time series $\bar{X} = (x_1 \dots x_n)$ and $\bar{Y} = (y_1 \dots y_m)$, respectively. Note that the two time series are of lengths n and m, which may not be the same. Then, the value of $DTW(i,j)$ is defined recursively as follows:

$$DTW(i,j) = |x_i - y_j| + \min(DTW(i, j-1), DTW(i-1, j), DTW(i-1, j-1))$$

(a) Implement the proposed DTW algorithm in C/C++ and analyze the time complexity of your implementation. (The framework Code-DTW.cpp is attached on the course webpage). Two test cases have been given in the source code.

(b) The window constraint imposes a minimum level w of positional alignment between matched elements. The window constraint requires that $DTW(i, j)$ be computed only when $|i - j| \leq w$. Modify your code to add a window constraint and give the results of $w = 0$ and $w = 1$ on the two test cases.

Solution. (a) The time complexity to initiate $DTW[i][j]$ is $O(mn)$. The time to find the path is $O(m + n)$. The time to calculate the time normalized distance is $O(m + n)$ too. So, the total time complexity is $O(mn)$.

(b) The .cpp file is in the folder. There are some mistakes in it, but I really don't know how to correct it.

$\square$

Remark: You need to include your .pdf and .tex and 2 source code files in your uploaded .rar or .zip file. Screenshots of test case results are acceptable.