Weijia Xu

Professor Arun

Algorithms & Structured Programming CS455

Oct.04, 2023

1.Consider an array A of distinct numbers in ascending order and a given threshhold T. Design a **divide-and-conquer** algorithm that is as asymptotically efficient as you can (in running time) to compute the number of pairs (i, j), i < j, such that (A[j] - A[i]) < T.

* 1. (3 pts) Describe the algorithm in a paragraph or two. Explain the key ideas in  it and why it finds a correct solution.

Divide: Split the array ‘A’ into two roughly equal halves: left and right.

Conquer: Recursively find the number of pairs in both ‘left’ and ‘right’ that satisfy the condition ‘(A[j] -A[i]) < T.

Combine: Count the number of pairs where ‘i’ is in ‘left’ and ‘j’ is in ‘right’ that satisfy the condition. This can be efficiently done by iterating through every element in ‘left’ and finding how many elements in ‘right’ starting from the smallest element are such that their difference with ‘left[i]’ is less than ‘T’.

* 1. (3 pts) Analyze the running time of your algorithm on inputs in which the number of pairs in the solution is constant, i.e. does not grow with the number of elements n in A.

For the divide step, simply split the array into two halves, which takes constant time O(1).

For the combine step, the linear scan of ‘left’ and ‘right’ arrays ensures that the time complexity is O(n).

Given the recurrence relation:

T(n) = 2T(n/2) + O(n)

By the master theorem, this recurrence relation resolves to:  
T(n) = O(nlogn)

* 1. (3 pts) Implement it in Python and run it on A = [2, 7, 14, 22, 30, 37, 43, 50, 57, 63, 71, 78, 85, 91, 98, 105, 112, 118, 125, 133] and T = 8.

A digital clock with numbers and symbols

Description automatically generated

1. Consider the problem of finding a longest increasing subsequence in a sequence of integers. (Remember as done in class when covering the LCS problem, a subsequence of a sequence is not necessarily a contiguous slice of the latter.)
   1. (3 pts) Design a dynamic programming algorithm for this problem and explain it.

Initialize a list ‘lengths’ of the same length as the input array and fill it with ‘1’s. So the value at ‘lengths[i]’ represents the length of the LIS ending at index ‘I’. For each element in the array, we compare it with every element before it. If the current element is greater, then it can be included in the subsequence of the previous element, potentially creating a longer subsequence. And update ‘lengths[i]’ by taking the maximum between its current value and the length of the increasing subsequence that ends in the previous elements +1.

* 1. (3 pts) Implement it in Python and run it on the sequence [3, 2, 5, 1, 6, 3, 9,2] to output an increasing subsequence that is the longest.

A number and numbers on a black background

Description automatically generated

1. Consider this slightly modified definition of edit distance. The edit distance between two strings S and T is the minimum number of insertion, deletion, substitution, or **transposition** operations needed to transform S to T (or vice versa). We've added a new operation which is to transpose adjacent characters. Consider CTAC and GCTCA. Their edit distance is 2 per the modified definition as AC can be transformed to CA with a single transposition operation.
   1. (3 pts) Describe the modified dynamic programming algorithm.

To find the modified edit distance, use of a 2D table where ‘dp[i][j]’ represents the minimum edit distance between the first ‘i’ characters of string ‘s’ and the first ‘j’ characters of string ‘T’.

Initialization:

dp[i][0] = i (transforming a string of length ‘i’ to an empty string requires ‘i’ deletions)

dp[0][j] = j(transforming an empty string to a string of length ‘j’ requires ‘j’ insertions)

Recurrence:

For each ‘i’ and ‘j’, have the following possibilities:

* Insert a character into S: dp[i][j-1] +1
* Delete a character from S: dp[i-1][j] +1
* Substitute a character in S: dp[i-1][j-1] + 1 or dp[i-1][j-1]
* Transpose two adjacent characters in S: dp[i-2][j-2] + 1
  1. (3 pts) Implement this (modified) algorithm in Python and run it on (CTAC, GCTCA) to output the edit distance as well as an actual sequence of operations that corresponds to it.

A black screen with white text

Description automatically generated