Foundation of Algorithms prog Assignment

Austin Jin

November 22, 2022

The following is a problem to be completed by the individual (i.e., it is not collaborative) and then implemented. Please follow the requirements provided under Syllabus Course Information under the link Programming Assign- ment Requirements.
For small input instances, n 30, you may output your results to the command window. Larger input instances must be written to a file. In either case, you may need to produce an output file for trace runs or asymptotic performance analysis.
1. Programming (non-collaborative): You are consulting for a group of people (who would prefer not to be mentioned here by name) whose job consists of monitoring and analyzing electronic signals coming from ships in the Atlantic ocean. They want a fast algorithm for a basic primitive that arises frequently: "untangling" a superposition of two known signals.
Specifically, they are picturing a situation in which each of two ships is emitting a short sequence of 0s and 1s over and over, and they want to make sure that the signal they are hearing is simply an interweaving of these two emissions, with nothing extra added in. The short sequence emitted by each ship is known to the ship and to you, the receiver.
Given a string x consisting of 0s and 1s, we write xk to denote k copies of x concatenated together. We say that string x is a repetition of x if it is a prefix of xk for some number k. So, x = 10110110110 is a repetition of x = 101.
We say that a string s is an interweaving of x and y if its symbols can be partitioned into two (not necessarily contiguous) subsequence s and s so that s is a repetition of x and s is a repetition of y. Each symbol in s must belong to exactly one of s of s.
For example,if x = 101 and y = 0,then s = 100010101 is an interweaving of x

and y since characters 1,2,5,7,8, and 9 form 101101 – a repetition of x – and the remaining characters 3,4,6 form 000 – a repetition of y. In terms of our application, x and y are the repeating sequences from the two ships, and s is the signal we are receiving. We want to make sure s "unravels" into simple repetitions of x and y.

You want a "fast algorithm for ...'untangling' a superposition of two known signals", x and y. You receive some set of symbols s. You do not know if the signal you receive has its first symbol in x, y, or a symbol that is extra and added in. You need to use your knowledge of x and y and the received signal s to determine if s consists only of valid interwoven symbols from x and y. That may require your solution to discard some symbols at the beginning or end of s to get to at least a full match of x and a full match of y. The received signal could be too short to do this. If you can't determine that the signal is an interweaving of x and y, either because s is not long enough to do so or because s contains symbols not in x or y then you cannot decide that s is an interweaving of x and y.

(a) [50 points] Give an efficient algorithm that takes strings s, x, and y and decides if s is an interweaving of x and y. Derive the computational complexity of your algorithm.

```
function DFS(s, i, j, A, B, C)
    x = A.length
    y = B.length
    z = C.length
    if Reach the end of the string C, s == z then
     return True
    if A[i] == C[s], B[j] == C[s], s < d then ▷ Both A[i] and B[j] satisfy the C[s] requirement
        x = dfs(s+1, i+1, j, A, B, C)
        y = dfs(s+1, i, j+1, A, B, C)
        dp[s][i][j] = Choose a better path x or y
     return dp[s][i][j]
    if A[i] == C[s], s < d then
        x = dfs(s+1, i+1, j, A, B, C)
        dp[s][i][j] = x return dp[s][i][j]
    if B[j] == C[s], s < d then
        y = dfs(s+1, i, j+1, A, B, C)
        dp[s][i][j] = y
     return dp[s][i][j]
    dp[s][i][j] = 0
     return dp[s][i][j]
function ISINTERLEAVED(A, B, C)
    m = A.length
    n = B.length
    s = C.length
    if m + n > s then return False
    Create a 3D array dp with each length equals to s
     return dfs(0, 0, 0, A, B, C)
```

	q0	1	0	1		A = 101
q0	T	T	T	F		B = 100
0	F	T	T	T		C = 10010100
0	F	T	F	T		
1	F	T	T	T		
0	F	F	F	T		
0	F	F	F	T		

Depth first searching and dynamic programming have been applied to solve this problem. At the first time we used 2D matrix to determine whether can we find pattern A and B in string C which only contained one A and one B, the matrix would present like picture above. The point begin with q0 state and read every character from string C. For each character, the program would compare it to character of pattern A and B at certain position A[i] or B[j]. If A[i] equals to the C[s], the program would mark True for that cell and record it as a vertex. So, for the true statement, we can get at least a path from top left to the bottom right. Similarly we can also apply this strategy to our complex pattern matching. In our algorithm, we build a 3D matrix rather than 2D to test whether we can find a path from top to the bottom of the string C only using A and B patterns, since string C might have repetitive sequence of pattern A or B and we also need to extend axis x and y with repeating pattern A and B to satisfy string S. This algorithm is fast and memorized vertices and edges of the matching, so this algorithm used extra space to save time complexity. The space complexity would be $O(s^3)$. This problem is NP-complete Problem. And the time complexity would be $O(s * m * n)$.

(b) [50 points] Implement your algorithm above and test its run time to verify your complexity analysis. Re- member that CPU time is not a valid measure for testing run time. You must use something such as the number of comparisons.

```
101010101 is interleaved of 10 and 1
time complexity: 17
O(|A|*|B|*|C|) = 2 * 1 * 9 = 18
0000000 is not interleaved of 101 and 0
time complexity: 9
O(|A|*|B|*|C|) = 3 * 1 * 7 = 21
100000100 is not interleaved of 101 and 00100
time complexity: 6
O(|A|*|B|*|C|) = 3 * 5 * 9 = 135
10001001 is interleaved of 101 and 00100
time complexity: 12
O(|A|*|B|*|C|) = 3 * 5 * 8 = 120
100011001 is interleaved of 101 and 00100
time complexity: 15
O(|A|*|B|*|C|) = 3 * 5 * 9 = 135
1000111001 is not interleaved of 101 and 00100
time complexity: 16
O(|A|*|B|*|C|) = 3 * 5 * 10 = 150
```

From the result we can see that the time complexity would be always smaller than $O(s * m * n)$ when character number of pattern A and B is larger than 1.