Algorithms II(H)
Assessed Exercise 2017-18
Dynamic Programming with Memoization

Apoorva Tamaskar

November 17, 2017

Problem 1

Include a status report which, for any non-working implementation, should state clearly what happens on compilation (in the case of compile-time errors) or on execution (in the case of run-time errors). If there are no compile-time or run-time errors, all that is required is a single sentence indicating this.

Solution 1

No compilation errors.
Run time errors:- Segmentation fault. Whenever the strings are read form a file, the code terminates by giving segmentation fault.

Problem 2

Try running the iterative algorithm and the recursive algorithm with memoization to compute an LCS for the strings in files strings10000a.txt and strings10000b.txt. What general behaviour do you notice and why is this the case?

Solution 2

Due to the segmentation fault occurring when the strings were read from an file, I ran the algorithms over randomly generated strings of same lengths over alphabets of size 2.
Iterative algorithm ran in the shortest amount of time.
Recursion without memoization did not terminate within acceptable time period(¿1 min).
Recursion with memoization terminated, but took time larger than Iterative method.
Recursion without memoization is slowest due to the exponential amounts of calculations, where are the other 2 have polynomial complexity.
Recursion with memoization is slower than iterative version due to the extra number of comparisons steps involved on each level of the algorithm. In general it is possible that the recursive calls are expensive. Tail Call Optimization is a possible solution to make it better(Not implemented).

Problem 3

Try generating strings of increasing lengths (assume that both strings have the same length) over an alphabet of size 4, and executing all three types of algorithm to find an LCS. What trends do you notice, and why? When answering this question, you should refer to (i) the relative speed of the algorithms and (ii) the largest instances solvable within, say, a minute.

Solution

asd
Iterative version could solve for 2 string of length 65000,10000 over alphabet of size 4 in time 12 sec.
Recursion without Memoization could find the solution to strings of length 23,24. Increasing lengths above this, the program did not terminate in less than a min. **Recursion with Memoization** could find a solution for 2 strings of length 60000 and 10000 over alphabet of size 4 within 13 sec. Increasing the size above this gives segmentation fault.

Problem 4

Try generating strings of length 5000 (again assume that both strings have the same length) over an alphabet that increases from 2 up to 10, executing the iterative algorithm and the recursive algorithm with memoization to find an LCS. What do you notice regarding how the time taken in each case varies with the growth in the alphabet size? Why do you think this is?

Solution

As the alphabet sized is increased, it can be observed that the running time increases.
While analyzing the algorithm, we need to keep in mind that we had assumed that each operation of comparison takes equal amount of time, however this assumption is not valid. This constant amount of time does depend on the alphabet size.
As alphabet size increases, the constant goes up, which in turn increases the running time of the algorithm.