Algorithms and Data Structures 2020 Exam–Question 3

Nathan Sharp | s1869292 | B130263

15-12-2020

Question 3

(a). Recurrence as follows,

$lcs(x_1, \cdots, x_n, y_1, \cdots y_m) =$

Case 1: $lcs(x_1, \cdots x_{n-1}, y_1, \cdots, y_{m-1}) + 1$ if $x_m = y_m$

Case 2: $max(lcs(x_1, \cdots x_{n-1}, y_1, \cdots, y_m), lcs(x_1, \cdots x_n, y_1, \cdots, y_{m-1}))$ if otherwise

(b).

```
def lcs(x[1...n], y[1...m]):
    table = n+1 by m+1 array of zeros
    for i from 1 to n:
        for j from 1 to m:
            calcCell(i, j, table)

    return table[n,m]

def calcCell(a, b, table):
    if x[a] = y[b]:
        table[a][b] = 1 + table[a-1,b-1]
    else:
        table[i][j] = max(table[n, m-1], table[n-1, m])
```

(c). Running time is as follows,

$T(n, m) =$

Case 1: $\Theta(1)$ if $n = 0$ or $m = 0$

Case 2: $nm \cdot T_{calcCell} + T_{initTableLn2}$ if otherwise

$T_(calcCell) \in \Theta(1)$ by inspection

$T_{initTableLn2} \in \Theta(1)$ or $\Theta(nm)$ (depends on implementation and doesnt matter either way)

Hence $\underline{T(nm) \in \Theta(nm)}$

Proof of correctness of my algorithm as follows,

There are two situations,

Case 1: last elem of x's and y's are equal. hence the solution is equal to the solution with the two last elements removed. This is optimal by inspection.

Case 2: last elems of x's and y's not equal. The subsequence is calculated between either $lcs(x_1, \cdots x_{n-1}, y_1, \cdots, y_m)$ or $lcs(x_1, \cdots x_n, y_1, \cdots, y_{m-1})$. Since we want the max length we take the maximum of our options aka. equal the our recurrence.