Elissa Ito December 3, 2022

6.1210 Problem Set 6

Problem 1 *(Collaborators: None)*

S Let $T(i, j)$ = the consecutive letters from A in the most left letters in $C[i + j : end]$ (which is merged from $A[i : end]$ and $B[j : end]$ with all letters not being more than three consecutively from $A[i : end]$.

R if $B[j + 1] = C[i + j + 1]$ & $T(i, j + 1) \leq 3$
then $T(i, j) = 0$
else if $A[i + 1] = C[i + j + 1]$ & $2 \geq T(i + 1, j)$
then $T(i, j) = T(i + 1, j) + 1$
else if $A[i + 1]! = C[i + j + 1]$ & $B[j + 1]! = C[i + j + 1]$
then $T(i, j) = 4$

B $T(n, m) = 0$;$T(i, m + 1) = T(n + 1, i) = 4$ since adding a letter from outside allowed indices is not possible

T Entry $T(i, j)$ depends on entries with larger i or j. Let $n = |\text{A}|$ and $m = |\text{B}|$. The table T can be filled either by row or column from the back to the front from $i = n$ to -1, and $j = m$ to -1.

O Output is YES if $T(-1, -1) \leq 3$. If not, the output is NO.

T $O(nm)$ since there are n rows and m columns in the table, with each entry taking $O(1)$.

Proof of correctness:

1. OSPP holds for all cases considered by the algorithm.
2. The cases considered by the algorithm are exhaustive.

Problem 2 *(Collaborators: None)*

Let digits with " : " between them be one number
Let
cnum = $D[i] : D[i+1]... : D[i+k]$
pnum = $D[j] : D[j+1]... : D[j+k]$

- **S** Let $T(i,j,k)$ = the length of the longest increasing digital subsequence of $D[i:n]$, where pnum is the most recently added number to the subsequence, i.e. j is the first digit of the most recently added number to the subsequence, i is the first digit of the current number, and k is the number of digits in the most recently added number.
- **R** $T(i,j,k) = \max\{\ 1 + T(i+k, j+k, k)$ if $cnum > pnum$
  $1 + T(i+k+1, j+k, k+1)$ if $cnum \leq pnum$
  $T(i+k, j, k)$
  $\}$
- **B** For all k=1,2,...n, $T(n,n,k) = 0$ because the length of the longest increasing digital subsequence of 0 digits is 0.
- **T** Entry $T(i,j,k)$ depends on entries with larger i or j or k. The table T can be filled by row from the back to the front.
- **O** Output is max[T(0,0,k) k=1,2,...n].
- **T** $O(n^4)$ since there are n items per row in 3 dimensions in the table, with each entry taking at most $O(n)$ to compute, because it takes $O(k)$ to compare two k digit numbers.

Proof of correctness:

1. OSPP holds for all cases considered by the algorithm.
2. The cases considered by the algorithm are exhaustive i.e.

Problem 3 *(Collaborators: None)*

S Let $M(i,j)$ = the number of mushrooms collected in the optimal path to position (i,j), and $P(i,j)$ = the number of optimal paths from the start (1,1) to position (i,j) i.e. the number of ways princess plum can move from (a,b) to (a+1,b) or (a,b+1) from (1,1) to (i,j) for any $0 \leq a \leq i$ and $0 \leq b \leq j$, and thus each paths should be length i+j-1.

R if (i,j) is a tree, $M(i,j) = False$ and $P(i,j) = 0$.
else if (i,j) is empty, $M(i,j) = max[M(i-1,j), M(i,j-1)]$
The max function returns whichever one is not a False or if both are False, returns False.
  and if $P(i-1,j)! = 0$ then $P(i,j) = P(i-1,j)$
  and if $P(i,j-1)! = 0$ then $P(i,j) = P(i,j-1)$
  else $P(i,j) = 0$ else (i.e. if (i,j) is a mushroom), $M(i,j) = max[M(i-1,j), M(i,j-1)] + 1$
The max function behaves the same as above.
  and if $P(i-1,j)! = 0$ then $P(i,j) = P(i-1,j)$
  and if $P(i,j-1)! = 0$ then $P(i,j) = P(i,j-1)$
  else $P(i,j) = 0$ }

B $M(1,1) = 0$, $P(1,1) = 1$. $M(a,0) = M(0,b) = 0$ and $P(a,0) = P(0,b) = 0$ for any for any a,b=1,2,...n.

T Entry $M(i,j)$ and $P(i,j)$ depend on entries with smaller i or j. The tables M and P can be filled by row from left to right and top to bottom.

O Output is $M(n,n)$ for the maximum number of mushroms for a quick path and $P(n,n)$ for the number of optimal paths that could give the maximum mushrooms collected.

T $O(n^2)$ since there are n items per row and column each for each table, with each entry taking at most $O(1)$ to compute. Thus $2 * O(n * n * 1) = O(n^2)$

Proof of correctness:

1. OSPP holds for all cases considered by the algorithm
2. The cases considered by the algorithm are exhaustive