HW 6: Solved Problem

Instructors: Hassanieh, Miller

CS/ECE 374 B: Algorithms & Models of Computation, Spring 2020 Version: **1.0**

1 A *shuffle* of two strings X and Y is formed by interspersing the characters into a new string, keeping the characters of X and Y in the same order. For example, the string BANANAANANAS is a shuffle of the strings BANANA and ANANAS in several different ways:

$\texttt{\underline{BANANA}ANANAS}$ $\texttt{\underline{BAN}ANA\underline{ANA}NAS}$ $\texttt{\underline{B}AN\underline{AN}A\underline{A}NA\underline{NA}S}$

Similarly, PRODGYRNAMAMMIINCG and DYPRONGARMAMMICING are shuffles of DYNAMIC and PROGRAMMING:

$\texttt{PRO\underline{D}G\underline{Y}R\underline{NAM}AMMI\underline{I}N\underline{C}G}$ $\texttt{\underline{DY}PRO\underline{N}G\underline{A}R\underline{M}AMM\underline{IC}ING}$

Given three strings $A[1\,..\,m]$, $B[1\,..\,n]$, and $C[1\,..\,m+n]$, describe and analyze an algorithm to determine whether C is a shuffle of A and B.

Solution:

We define a boolean function $\mathit{Shuf}(i,j)$, which is TRUE if and only if the prefix $C[1\,..\,i+j]$ is a shuffle of the prefixes $A[1\,..\,i]$ and $B[1\,..\,j]$. This function satisfies the following recurrence:

$$\mathit{Shuf}(i,j) = \begin{cases} \textsc{True} & \text{if } i=j=0 \\ \mathit{Shuf}(0,j-1) \land (B[j]=C[j]) & \text{if } i=0 \text{ and } j>0 \\ \mathit{Shuf}(i-1,0) \land (A[i]=C[i]) & \text{if } i>0 \text{ and } j=0 \\ \big(\mathit{Shuf}(i-1,j) \land (A[i]=C[i+j])\big) & \\ \qquad \lor \big(\mathit{Shuf}(i,j-1) \land (B[j]=C[i+j])\big) & \text{if } i>0 \text{ and } j>0 \end{cases}$$

We need to compute $\mathit{Shuf}(m,n)$.

We can memoize all function values into a two-dimensional array $\mathit{Shuf}[0\,..\,m][0\,..\,n]$. Each array entry $\mathit{Shuf}[i,j]$ depends only on the entries immediately below and immediately to the right: $\mathit{Shuf}[i-1,j]$ and $\mathit{Shuf}[i,j-1]$. Thus, we can fill the array in standard row-major order. The original recurrence gives us the following pseudocode which runs in ***$O(mn)$ time***:

```
Shuffle?($A[1..m]$, $B[1..n]$, $C[1..m+n]$):
$Shuf[0,0] \gets$ TRUE
for $j \gets 1$ to $n$
   $Shuf[0,j] \gets Shuf[0,j-1] \land (B[j] = C[j])$
for $i \gets 1$ to $n$
   $Shuf[i,0] \gets Shuf[i-1,0] \land (A[i] = B[i])$
   for $j \gets 1$ to $n$
      $Shuf[i,j] \gets$ FALSE
      if $A[i] = C[i+j]$
         $Shuf[i,j] \gets Shuf[i,j] \lor Shuf[i-1,j]$
      if $B[i] = C[i+j]$
         $Shuf[i,j] \gets Shuf[i,j] \lor Shuf[i,j-1]$
return $Shuf[m,n]$
```

Rubric: Standard dynamic programming rubric:
For problems worth 10 poins:

- 6 points for a correct recurrence, described either using mathematical notation or as pseudocode for a recursive algorithm.
  - + 1 point for a clear **English** description of the function you are trying to evaluate. (Otherwise, we don't even know what you are *trying* to do.) **Automatic zero if the English description is missing.**
  - + 1 point for stating how to call your function to get the final answer.
  - + 1 point for base case(s). $-1/2$ for one *minor* bug, like a typo or an off-by-one error.
  - + 3 points for recursive case(s). -1 for each *minor* bug, like a typo or an off-by-one error. **No credit for the rest of the problem if the recursive case(s) are incorrect.**
- 4 points for details of the dynamic programming algorithm
  - + 1 point for describing the memoization data structure
  - + 2 points for describing a correct evaluation order; a clear picture is usually sufficient. If you use nested loops, be sure to specify the nesting order.
  - + 1 point for time analysis
- It is *not* necessary to state a space bound.
- For problems that ask for an algorithm that computes an optimal *structure*—such as a subset, partition, subsequence, or tree—an algorithm that computes only the *value* or *cost* of the optimal structure is sufficient for full credit, unless the problem says otherwise.
- Official solutions usually include pseudocode for the final iterative dynamic programming algorithm, ***but iterative psuedocode is not required for full credit***. If your solution includes iterative pseudocode, you do not need to separately describe the recurrence, memoization structure, or evaluation order. (But you still need to describe the underlying recursive function in English.)
- Official solutions will provide target time bounds. Algorithms that are faster than this target are worth more points; slower algorithms are worth fewer points, typically by 2 or 3 points (out of 10) for each factor of n. Partial credit is scaled to the new maximum score, and all points above 10 are recorded as extra credit.

  We rarely include these target time bounds in the actual questions, because when we have included them, significantly more students turned in algorithms that meet the target time bound but did not work (earning 0/10) instead of correct algorithms that are slower than the target time bound (earning 8/10).