CS/ECE 374 A ✧ Fall 2019
Homework 5

Due Tuesday, October 15, 2019 at 8pm

---

1. Farmers Boggis, Bunce, and Bean have set up an obstacle course for Mr. Fox. The course consists of a long row of booths, each with a number painted on the front with bright red paint. Formally, Mr. Fox is given an array $A[1..n]$, where $A[i]$ is the number painted on the front of the ith booth. Each number $A[i]$ could be positive, negative, or zero. Everyone agrees with the following rules:

    - Mr. Fox must visit all the booths in order from 1 to n.
    - At each booth, Mr. Fox must say one word: either "Ring!" or "Ding!"
    - If Mr. Fox says "Ring!" at the ith booth, he earns a reward of $A[i]$ chickens. (If $A[i] < 0$, Mr. Fox pays a penalty of $-A[i]$ chickens.)
    - If Mr. Fox says "Ding!" at the ith booth, he pays a penalty of $A[i]$ chickens. (If $A[i] < 0$, Mr. Fox earns a reward of $-A[i]$ chickens.)
    - Mr. Fox is forbidden to say the same word more than three times in a row. For example, if he says "Ring!" at booths 6, 7, and 8, then he must say "Ding!" at booth 9.
    - All accounts will be settled at the end, after Mr. Fox visits every booth and the umpire calls "Hot box!" Mr. Fox does not actually have to carry chickens (or anti-chickens) through the obstacle course.
    - Finally, if Mr. Fox violates any of the rules, or if he ends the obstacle course owing the farmers chickens, the farmers will shoot him.

    Describe and analyze an algorithm to compute the largest number of chickens that Mr. Fox can earn by running the obstacle course, given the array $A[1..n]$ of numbers as input. *[Hint: Watch out for the burning pine cone!]*

2. Recall that a *supersequence* of a string w is any string obtained from w by inserting zero or more symbols. For example, the strings `STRING`, `STIRRING`, and `MISTERFINNIGAN` are all supersequences of the string `STRING`.

    (a) Recall that a *palindrome* is a string that is equal to its reversal, like the empty string, `A`, `HANNAH`, or `AMANAPLANACATACANALPANAMA`. Describe an algorithm to compute the length of the shortest palindrome supersequence of a given string.

    (b) A *dromedrome* is an even-length string whose first half is equal to its second half, like the empty string, `AA`, `ACKACK`, or `AMANAPLANAMANAPLAN`. Describe an algorithm to compute the length of the shortest dromedrome supersequence of a given string.

    For example, given the string `SUPERSEQUENCE` as input, your algorithm for part (a) should return 21 (the length of `SUPECNRSEQUQESRNCEPUS`), and your algorithm for part (b) should return 20 (the length of `SEQUPERNCESEQUPERNCE`). The input to both algorithms is an array $A[1..n]$ representing a string.

3. Suppose you are given a DFA M with k states for Jeff's favorite regular language $L \subseteq (0+1)^*$.

   (a) Describe and analyze an algorithm that decides whether a given bit-string belongs to the language L^*.

   (b) Describe and analyze an algorithm that partitions a given bit-string into as many substrings as possible, such that L contains every substring in the partition. Your algorithm should return only the number of substrings, not their actual positions. (In light of your algorithm from part (a), you can assume that an appropriate partition exists.)

   For example, suppose L is the set of all bit-strings that start and end with 1 and whose length is *not* divisible by 3.[1] Then given the input string 101111000011010110111101, your algorithm for part (a) should return TRUE, and your algorithm for part (b) should return the integer 5, which is the length of the following partition:

   1011 • 1 • 10000110101 • 1011 • 1101

   The input to both algorithms consists of (some reasonable representation of) the DFA M and an array $A[1..n]$ of bits. Express the running time of your algorithms as functions of both k (the number of states in M) and n (the length of the input string).

   *[Hint: Do **not** try to build a DFA for L^*.]*

[1]This is not actually Jeff's favorite regular language.

Solved Problem

4. 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.

   BANANAANANAS BANANAANANAS BANANAANANAS

   Similarly, the strings PRODGYRNAMAMMIINCG and DYPRONGARMAMMICING are both shuffles of DYNAMIC and PROGRAMMING:

   PRODGYRNAMAMMIINCG DYPRONGARMAMMICING

   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:

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

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:

```
SHUFFLE?($A[1..m]$, $B[1..n]$, $C[1..m+n]$):
    $Shuf[0,0] \leftarrow$ TRUE
    for $j \leftarrow 1$ to $n$
        $Shuf[0,j] \leftarrow Shuf[0,j-1] \wedge (B[j]=C[j])$
    for $i \leftarrow 1$ to $n$
        $Shuf[i,0] \leftarrow Shuf[i-1,0] \wedge (A[i]=B[i])$
        for $j \leftarrow 1$ to $n$
            $Shuf[i,j] \leftarrow$ FALSE
            if $A[i]=C[i+j]$
                $Shuf[i,j] \leftarrow Shuf[i,j] \vee Shuf[i-1,j]$
            if $B[i]=C[i+j]$
                $Shuf[i,j] \leftarrow Shuf[i,j] \vee Shuf[i,j-1]$
    return $Shuf[m,n]$
```

The algorithm runs in ***O*(*mn*) time**. ■

Rubric: Max 10 points: Standard dynamic programming rubric. No proofs required. Max 7 points for a slower polynomial-time algorithm; scale partial credit accordingly.

Standard dynamic programming rubric. For problems worth 10 points:

- 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're *trying* to do.) **Deadly Sin: 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). −½ 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 specifically says otherwise.
- Official solutions usually include pseudocode for the final iterative dynamic programming algorithm, ***but iterative pseudocode 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 ***do*** 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 didn't work (earning 0/10) instead of correct algorithms that are slower than the target time bound (earning 8/10).