“CS 374” Fall 2014 — Homework 5

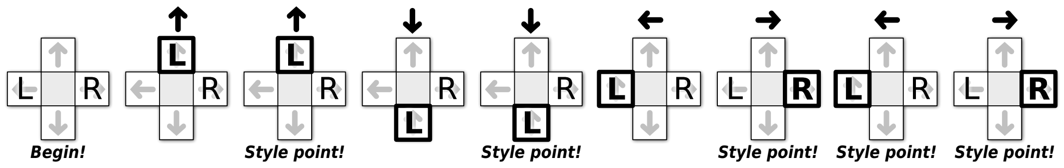
Due Tuesday, October 14, 2014 at noon

---

1. ***Dance Dance Revolution*** is a dance video game, first introduced in Japan by Konami in 1998. Players stand on a platform marked with four arrows, pointing forward, back, left, and right, arranged in a cross pattern. During play, the game plays a song and scrolls a sequence of n arrows ($\leftarrow$, $\uparrow$, $\downarrow$, or $\rightarrow$) from the bottom to the top of the screen. At the precise moment each arrow reaches the top of the screen, the player must step on the corresponding arrow on the dance platform. (The arrows are timed so that you'll step with the beat of the song.)

   You are playing a variant of this game called “Vogue Vogue Revolution”, where the goal is to play perfectly but move as little as possible. When an arrow reaches the top of the screen, if one of your feet is already on the correct arrow, you are awarded one style point for maintaining your current pose. If neither foot is on the right arrow, you must move one (and *only* one) of your feet from its current location to the correct arrow on the platform. If you ever step on the wrong arrow, or fail to step on the correct arrow, or move more than one foot at a time, or move either foot when you are already standing on the correct arrow, or insult Beyoncé, all your style points are immediately taken away and you lose.

   How should you move your feet to maximize your total number of style points? For purposes of this problem, assume you always start with you left foot on $\leftarrow$ and you right foot on $\rightarrow$, and that you've memorized the entire sequence of arrows. For example, if the sequence is $\uparrow\uparrow\downarrow\downarrow\leftarrow\rightarrow\leftarrow\rightarrow$, you can earn 5 style points by moving you feet as shown below:



   Describe and analyze an efficient algorithm to find the maximum number of style points you can earn during a given VVR routine. Your input is an array *Arrow*$[1..n]$ containing the sequence of arrows.

2. Recall that a *palindrome* is any string that is exactly the same as its reversal, like `I`, or `DEED`, or `RACECAR`, or `AMANAPLANACATACANALPANAMA`.

   Any string can be decomposed into a sequence of palindrome substrings. For example, the string `BUBBASEESABANANA` (“Bubba sees a banana.”) can be broken into palindromes in the following ways (among many others):

   `BUB` • `BASEESAB` • `ANANA`

   `B` • `U` • `BB` • `A` • `SEES` • `ABA` • `NAN` • `A`

   `B` • `U` • `BB` • `A` • `SEES` • `A` • `B` • `ANANA`

   `B` • `U` • `B` • `B` • `A` • `S` • `E` • `E` • `S` • `A` • `B` • `ANA` • `N` • `A`

   Describe and analyze an efficient algorithm to find the smallest number of palindromes that make up a given input string. For example, given the input string `BUBBASEESABANANA`, your algorithm would return the integer 3.

3. Suppose you are given a DFA $M = (\{\mathtt{0}, \mathtt{1}\}, Q, s, A, \delta)$ and a binary string $w \in \{\mathtt{0}, \mathtt{1}\}^*$.

   (a) Describe and analyze an algorithm that computes the longest subsequence of w that is accepted by M, or correctly reports that M does not accept any subsequence of w.

   *(b) **[Extra credit]** Describe and analyze an algorithm that computes the *shortest supersequence* of w that is accepted by M, or correctly reports that M does not accept any supersequence of w. (Recall that a string x is a supersequence of w if and only if w is a subsequence of x.)

   Analyze both of your algorithms in terms of the parameters $n = |w|$ and $k = |Q|$.

Rubric (for all dynamic programming problems): As usual, a score of x on the following 10-point scale corresponds to a score of $\lceil x/3 \rceil$ on the 4-point homework scale.

- 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.) **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 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.
- Official solutions usually include pseudocode for the final iterative dynamic programming algorithm, ***but this is not required for full credit***. If your solution includes iterative pseudocode, you do not need to separately describe the recurrence, data 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 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 do include them, significantly more students turn in algorithms that meet the target time bound but don't work (earning 0/10) instead of correct algorithms that are slower than the target time bound (earning 8/10).

CS 374 Fall 2014 — Homework 5 Problem 1

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

Describe and analyze an efficient algorithm to find the maximum number of style points you can earn during a given VVR routine. Your input is an array $Arrow[1..n]$ containing the sequence of arrows.

CS 374 Fall 2014 — Homework 5 Problem 2

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

Describe and analyze an efficient algorithm to find the smallest number of palindromes that make up a given input string.

CS 374 Fall 2014 — Homework 5 Problem 3

Name:	NetID:
Name:	NetID:
Name:	NetID:
Section: 1 2 3	

(a) Describe and analyze an algorithm that either computes the longest subsequence of w that is accepted by M, or correctly reports that M does not accept any subsequence of w.

*(b) **[Extra credit]** Describe and analyze an algorithm that either computes the *shortest supersequence* of w that is accepted by M, or correctly reports that M does not accept any supersequence of w.