CS 473 ✧ Spring 2020
Homework 1

Due Wednesday, February 5, 2020 at 9pm

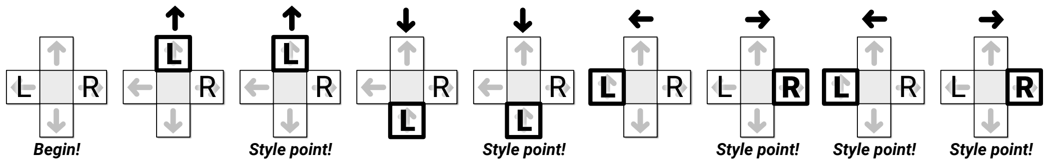
- Starting with this homework, groups of up to three students can submit joint solutions for each problem. For each numbered problem, exactly one member of each homework group should submit a solution and identify the other group members (if any).
- If you use a greedy algorithm, you ***must*** prove that it is correct, or you will get zero points ***even if your algorithm is correct***.

---

1. ***Dance Dance Revolution*** (ダンスダンスレボリューション) 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 (←, ↑, ↓, or →) 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) foot 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, all your style points are taken away and you lose the game.

   How should you move your feet to maximize your total number of style points? Assume you start the game with your left foot on ← and your right foot on →, and that you've memorized the entire sequence of arrows. For example, if the sequence is ↑↑↓↓←→←→, you can earn 5 style points by moving your feet as shown below:



   (a) Prove that for *any* sequence of n arrows, it is possible to earn at least $n/4-1$ style points.

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

2. You've been hired to store a sequence of n books on shelves in a library, using as little *vertical* space as possible. The order of the books is fixed by the cataloging system and cannot be changed; each shelf must store a contiguous interval of the given sequence of books. You can adjust the height of each shelf to match the tallest book on that shelf; in particular, you can change the height of any empty shelf to zero.

   You are given two arrays $H[1..n]$ and $W[1..n]$, where $H[i]$ and $W[i]$ are respectively the height and width of the ith book. Each shelf has the same fixed length L. Each book as width at most L, and the total width of all books on each shelf cannot exceed L. Your task is to shelve the books so that the *sum of the heights* of the shelves is as small as possible.

   (a) There is a natural greedy algorithm, which actually yields an optimal solution when all books have the same height: If $n > 0$, pack as many books as possible onto the first shelf, and then recursively shelve the remaining books.

   Show that this greedy algorithm does *not* yield an optimal solution if the books can have different heights. *[Hint: There is a small counterexample.]*

   (b) Describe and analyze an efficient algorithm to assign books to shelves to minimize the total height of the shelves.

3. (a) Any string can be decomposed into a sequence of palindromes. For example, the string BUBBASEESABANANA ("Bubba sees a banana.") can be broken into palindromes in the following ways (and 65 others):

   BUB • BASEESAB • ANANA
   B • U • BB • ASEESA • B • ANANA
   BUB • B • A • SEES • ABA • N • ANA
   B • U • BB • A • S • EE • S • A • B • A • NAN • A
   B • U • B • B • A • S • E • E • S • A • B • A • N • A • 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 should return 3.

   (b) A ***metapalindrome*** is a decomposition of a string into a sequence of palindromes, such that the sequence of palindrome lengths is itself a palindrome. For example, the string BOBSMAMASEESAUKULELE ("Bob's mama sees a ukulele") has the following metapalindromes (among others):

   BOB • S • MAM • ASEESA • UKU • L • ELE
   B • O • B • S • M • A • M • A • S • E • E • S • A • U • K • U • L • E • L • E

   The length sequences of these metapalindromes are $(3, 1, 3, 6, 3, 1, 3)$ and $(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$; notice that both of these sequences are themselves palindromes.

   Describe and analyze an efficient algorithm to find the smallest number of palindromes in any metapalindrome for a given string. For example, given the input string BOBSMAMASEESAUKULELE, your algorithm should return 7.