COP 3503 Homework #6: Road Race

Filenames: roadrace.java, roadracepath.java
Time Limit: 1 second (per input case)
Standard Input, Standard Output

You are in a road race where there are several lanes. Unfortunately, you can't change lanes at any time. The road is broken up into segments and you can only change lanes (if you want) at the end of each segment. However, it takes time to change lanes, so you only want to do it if the lane you are moving will immediately or eventually help you. In particular, the lanes are numbered 1 through ***k***, from left to right, and it takes $\boldsymbol{d^2 + 5}$ **seconds** to change lanes where ***d*** represents the absolute value of the difference between the lane you start in and the lane you change to.

Secondly, for each segment of the road, the length of the segment (in meters) is known. In addition, for each lane in each segment, the average speed you will travel in kilometers per hour is known. Finally, you may choose any of the lanes to start the race in initially.

When the assignment was originally posted, my intention was for it to take $0^2 + 5 = 5$ seconds to "change" lanes from lane i to lane i. However, upon reading what is written above, as a student pointed out to me, we can think of going from lane i to lane i as NOT changing lanes and incurring no penalty. Unfortunately, by the time I was made aware of this, the assignment was already due.

Thus, I created two solutions – one with my original intention (roadracepath_v1.java) and a second one which has no 5 second penalty for staying in the same lane (roadracepath_v2.path). I used these two solutions to create two different sets of outputs for both versions of the problem. (There are 8 test cases, so there are 16 outputs corresponding to version 1 and 16 outputs corresponding to version 2, to cover both part 1 and part 2.) Since I have assumed that more students assumed the first interpretation (in fact, using the second interpretation, the samples are incorrect), all of the data creates corner cases for the version 1 interpretation, which is why the version 2 outputs (particularly for path) are "boring." (This is most acutely noticeable in the path output for test case 8 for the two different versions.)

For grading purposes, the TAs will run students' code and attempt to match the output to both sets of output, giving students whichever set gives a student the higher grade.

Also, it's impossible for two optimal paths to ever "cross". (They can merge, but not cross over.) This is because of the nature of the cost function. If arbitrary costs were given for changing between each pair of lanes between each pair of segments, this wouldn't be true. What this means is that the DP can be run in either direction and the lexicographically first path can be correctly built. Had the cost function been arbitrary, then the DP has to be run backwards (as the posted solution does) to properly build the lexicographically first path.

The Problem – Part I (80% of the grade)

For a road race, given the number of segments of road, the number of lanes of road, the length of each segment (in meters), and the average traveling speed on each lane for each segment (in kilometers per hour), determine the fastest time (in seconds) that you can finish the road race.

The Problem – Part II (20% of the grade)

Using the same input as the first part, determine the lexicographically first **path** that achieves the fastest time in the road race. We denote a path in the race as a sequence of integers $p_1, p_2, \ldots, p_n$, where p_i represents the lane in which we traveled for segment i of the race. Recall that given two sequences of integers, $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$, for the former to be lexicographically first, that for the minimum value of i for which $a_i \neq b_i$, it must be the case that $a_i < b_i$.

The credit for this portion of the assignment will only be based on test cases, and a separate program must be submitted to earn any of this credit. Also, half of these test cases will have only one unique path that achieves the fastest time. Thus, if your program can generate **ANY** valid path, your total assignment grade can be up to 90%.

The Input

The first line of input contains a two space separated positive integers, $\boldsymbol{n}$ ($1 \leq \boldsymbol{n} \leq 1000$), the number of segments in the road race, and $\boldsymbol{L}$ ($2 \leq \boldsymbol{L} \leq 10$), the number of lanes on the road, respectively.

The second line of input contains $\boldsymbol{n}$ space separated integers, $\boldsymbol{d_1}, \boldsymbol{d_2}, \ldots, \boldsymbol{d_n}$, ($100 \leq \boldsymbol{d_i} \leq 100{,}000$) where $\boldsymbol{d_i}$ represents the distance of the i^{th} segment of the road race, in meters.

The following $\boldsymbol{n}$ lines contain information about the average speed of the lanes in each segment. Specifically, the i^{th} of these lines contains information about the average speed of each lane in segment i. Each of these lines will contain $\boldsymbol{L}$ space separated integers, $\boldsymbol{s_{i,j}}$, ($1 \leq i \leq \boldsymbol{n}$, $1 \leq j \leq \boldsymbol{L}$, $1 \leq \boldsymbol{s_{i,j}} \leq 150$) representing the average speed, in kilometers per hour, on lane j of segment i.

The Output – Part I

Output a decimal number, rounded to 2 decimal places, with the fastest possible time in seconds to finish the road race.

The Output – Part II

Output ***n*** integers on a single line, followed by a space (so the last integer has a space after it), on a line by itself representing which lane should be traveled on for each segment, in the order of the segments to achieve the fastest race time. If there are multiple answers, provide the lexicographically first sequence of lanes that achieves this fastest time.

Sample Input	**Sample Output Part I**	**Sample Output Part II**
2 3 1000 2000 50 50 60 60 50 50	189.00	3 1
2 3 1000 2000 50 50 50 50 50 50	221.00	1 1
5 6 1000 2000 50000 45000 5000 80 82 81 83 79 78 100 99 98 97 96 95 101 102 103 104 105 106 149 144 148 150 141 100 68 93 85 94 92 94	3113.98	4 5 6 4 4

Implementation Requirements

This assignment is testing the use of dynamic programming/memorization. It's unlikely that you will solve the problem correctly and fast enough without using this technique. Thus, there aren't any implementation restrictions beyond the problem statement and using reasonable coding style and layout. (If using tradition DP, it's okay to have the whole program in main and use some static class variables to store the common problem information.)

What To Submit

For this assignment, please submit two Java programs. Your solution to part one should be named **roadrace.java**. Your solution to part 2 should be named **roadracepath.java.**

If you only submit the first program, your maximum grade will be 80%. If you only submit the second program, even though it's harder, your maximum grade will be 20%. Thus, please make every effort to submit the first program, roadrace.java that just outputs the fastest race time. The second part is more challenging. As you can see, to be nice, I've made it worth less, but I wanted to create it as an opportunity for students. Also, don't feel bad if you can reconstruct a valid path, but have trouble reconstructing a lexicographically first valid path.