2002 Sun Microsystems and TopCoder Collegiate Challenge – Problem Statement

PolePosition PROBLEM STATEMENT:

An auto race is occurring, and the drivers want to figure out what sort of starting advantage they need to win or tie the race.

The racetrack is a rectangle. It has lanes that split the track into concentric rectangles. If the innermost lane is of length L and width W, then the lane immediately outside of that lane will have length L + 2 and width W + 2.
Example:

```
       width W + 2
    ----------------
   |     width W    |
l  |    --------    | The area defined by X's is the area inside the track that
is not used.
e  |   | lane 0 | l | The small lane delimited by the area inside the track and
the smallest dashed
n  | l |  XXXX  | a | rectangle containing the inside of the track (lane 0) is of
length L and width W.
g  | e |  XXXX  | n | The width of the lane is the distance horizontally at
either the top or bottom
t  | n |  XXXX  | e | between the two adjacent corners of the lane.  The length
of the lane is the
h  | g |  XXXX  |   | distance vertically on either the left or right side
between the two adjacent
   | t |  XXXX  | 1 | corners of the lane. The measure is taken only between two
corners that are
L  | h |  XXXX  |   | vertically or horizontally aligned. The next lane (lane 1)
is of length L + 2 and
+  |   |  XXXX  |   | width W + 2.  If there were a third lane, it would be of
length
2  | L | lane 0 |   | (L + 2) + 2 = L + 4 and of width (W + 2) + 2 = W + 4.
   |   ---------    | The width of lane N is given by W + 2 * N, and the length
by L + 2 * N.
   |    lane 1      | The total distance required to complete the inner lap is
    ----------------  2 * (L - 2) + 2 * (W - 2) + 4.
```

If the race is 10 laps, and L = 5, W = 6, then the driver in lane 0 only has to travel a distance of 180 units, while the driver in lane 1 has to travel 260 units. The driver in lane 2 has to travel 340 units, etc.

If a driver is in lane N of the track, then there must be a driver in every lane from lane 0 to lane N-1, that is, empty lanes must be at the outside of the track. Every driver races in every race and there are enough lanes on the track to accommodate all drivers. Exactly one driver can be in a each lane.

Given an int[] of speeds for each driver (units of time the driver needs to complete one unit of track), the number of laps to be raced, and the length and width of the inner lane, return an int[] that contains the number of the highest numbered lane (0 based) that the corresponding driver could start in and win or tie the race. The driver does not have to win every time he is in this position, merely win or tie based on one possible driver line up.
If the driver could never win or tie, put a -1 into the corresponding element of the (int[]) return value.

For the purpose of this problem, a driver wins (or ties for first place) if that driver arrives at the finish line before every other driver or at the same time as one or more other drivers and before the remaining drivers. The driver does not complete driving through a unit of distance on the track until all the time he needs to drive through it has passed. For example, if a driver's speed is 4, and the driver has just completed 10 units of track, the driver will be considered to instantaneously complete driving through the next unit of track 4 units of time later.

DEFINITION:

Class Name: PolePosition
Method Name: worstStart
Parameters: int[], int, int, int
Returns: int[]

Method signature: int[] worstStart(int[] driverSpeeds, int trackLength, int trackWidth, int lapsInRace)
(be sure your method is public)

TopCoder will ensure the validity of the inputs. Inputs are considered valid if and only if all of the following criteria are met:
 *driverSpeeds contains between 2 and 50 elements, inclusive.
 *elements of driverSpeeds are between 1 and 100, inclusive.
 *No two elements of driverSpeeds are equal.
 *trackLength is between 2 and 100, inclusive.
 *trackWidth is between 2 and 100, inclusive.
 *lapsInRace is between 1 and 100, inclusive.

WORKED EXAMPLES:

1.
driverSpeeds = { 3, 5 }, trackLength = 5, trackWidth = 5, lapsInRace = 2.
If driver 0 is in lane 0, and driver 1 is in lane 1, driver 0 completes the race after (laps * lane length * speed) 2 * 16 * 3 = 96 units of time. Driver 1 completes the race after 2 * 24 * 5 = 240 units of time and driver 0 wins. If driver 0 is in lane 1 and driver 1 is in lane 0 then driver 1 will complete the race in 2 * 16 * 5 = 160 units of time and driver 0 will complete the race in 2 * 24 * 3 = 144 units of time. Hence, driver 1 can never win or tie no matter what lane he starts in and driver 0 wins or ties no matter what lane he starts in.
Return { 1, -1 }.

2.
driverSpeeds = { 9, 10 }, trackLength = 2, trackWidth = 2, lapsInRace = 10.
If driver 0 is in lane 0, and driver 1 is in lane 1, then we can see that driver 0 easily wins, since he is faster than driver 1 and has less distance to cover.
If driver 0 is in lane 1, and driver 1 is in lane 0, then driver 0 crosses the finish line after 10 * 12 * 9 = 1080 units of time and driver 1 crosses the finish line after 10 * 4 * 10 = 400 units of time. Return { 0, 0 }, since neither driver wins unless that driver is in the pole position.

3.
driverSpeeds = { 12, 20, 60 }

trackLength = 4, trackWidth = 4, numLaps = 1.

Lane 0 is of distance 12, lane 1 is of distance 20, and lane 2 is of distance 28.
The following table shows the finish times of the three drivers.
The elements of the grid are in the format "driver number/time to finish":

Driver in Lane 0	Driver in Lane 1	Driver in Lane 2	winning driver(s)
0/144	1/400	2/1680	0
0/144	2/1200	1/560	0
1/240	0/240	2/1680	0,1
1/240	2/1200	0/336	1
2/720	0/240	1/560	0
2/720	1/400	0/336	0

Driver 0 can win or tie when in lanes 0, 1, or 2, driver 1 can win or tie when in lane 0, and driver 2 can never win. Note specifically, that driver 0 only wins in lane 2 once (with driver 2 in lane 0) and loses the other time (with driver 1 in lane 0). However, since he only needs to win in one combination of lanes, his worst starting position is still 2.
Return { 2, 0, -1 }.

TEST CASES:
{ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 }, 10, 10, 6 returns {9, 2, 0, -1, -1, -1, -1, -1, -1, -1}.
{ 1, 2, 7, 9, 10, 45 }, 1, 1, 1 returns {5, 2, 0, 0, 0, -1}.