1 Cheeseville

The streets of Cheeseville form an $k \times k$ grid, and one of $k^2$ mice lives at the corner of each intersection. Note that, in this grid, the bottom left corner corresponds to $(0,0)$ (i.e. we zero-index rows and count from bottom-up instead of from the top-down).

You’re about to place a big slice of cheese at the South-West corner of Cheeseville, after which all $k^2$ mice will immediately smell it and run to the cheese as fast as they can.

There is enough cheese for all the mice, but not all cheese is the same: the mice that arrive first get to eat the best, stinkiest parts of the cheese. The problem: if two mice arrive at exactly the same time, they will fight over who gets to eat next best part of the cheese.

All mice run at exactly the same speed. All mice always take a shortest (minimum number of blocks) route, i.e. they never run North or East. But there are some small delays on each block, so paths with the same number of blocks can take slightly different time - which you hope will help as tie-breakers between the mice racing for the cheese.

Your input is a pair of matrices: an $k \times (k-1)$ matrix $W$ with delays going West, where $W_{ij}$ is the delay from street-corner $(i, j+1)$ to street-corner $(i, j)$, and a $(k-1) \times k$ matrix $S$, where $S_{ij}$ is the delay going South from $(i+1, j)$ to $(i, j)$. Design an algorithm that predicts ties, i.e. it finds a pair of mice that is expected to arrive at $(0,0)$ at exactly the same time, or notifies that such a pair doesn’t exist.

**Input:** A $k \times (k-1)$ matrix $W$ with $W_{ij} =$ the cost of going from $(i, j+1)$ to $(i, j)$ and a $(k-1) \times k$ matrix $S_{ij} =$ the cost of going from $(i+1, j)$ to $(i, j)$.

**Output:** Either a pair $(i_1, j_1), (i_2, j_2)$ of mice that will arrive at exactly the same time, or $\emptyset$ if no ties will occur.

**Example.**

**Input:**

$$W = \begin{bmatrix}
101 & 104 \\
105 & 102 & 101 & 105
\end{bmatrix}$$

$$S = \begin{bmatrix}
102 & 103 & 105 \\
103 & 105 & 105
\end{bmatrix}$$

**Output:**

$$\{(1,2), (2,1)\}$$
In the diagram above, North, East, South, and West are up, right, down, and left respectively. The mice living at (1, 2) and (2, 1) will arrive at (0, 0) at the same time (the delays they encounter are bolded). The algorithm will output [(1, 2), (2, 1)]

2 Rod Cutting

Suppose we have a rod of length $k$, where $k$ is a positive integer. We would like to cut the rod into integer-length segments such that we maximize the product of the resulting segments' lengths. Multiple cuts may be made. For example, if $k = 8$, the maximum product is 18 from cutting the rod into three pieces of length 3, 3, and 2. Write an algorithm to determine the maximum product for a rod of length $k$.

3 Encoding

Suppose we encode lowercase letters into a numeric string as follows: we encode $a$ as 1, $b$ as 2 . . . and $z$ as 26. Given a numeric string $S$ of length $n$, develop an $O(n)$ algorithm to find how many letter strings this can correspond to. For example, for the numeric string 123, the algorithm should output 3 because the letter strings that map to this numeric string are $abc, lc$, and $aw$.

4 Dice Probabilities

We wish to find the probability that rolling $k$ 6-sided fair dice will result in a sum $S$. Devise an algorithm to find this probability.