Challenges Week 3

RESTAURANTS - 25 points

Bob is a big fan of restaurants. In a few days time, Bob will be travelling to a city that is renowned for its many high quality restaurants. His plan is to visit a different restaurant every day. Since the restaurants are already quite expensive, Bob wants to save on his travelling costs.
In order to achieve this he needs to determine the best place to stay by minimising the average (Manhattan) distance to each restaurant. Once he has picked a place to stay, he will stay there for the entire trip.
Bob has drawn out a grid of size $n \times n$ with r restaurants. He can stay at any of the points on the grid. Bob wanted to write a program that can help him with this, but a curious incident caused his computer to end up in the neighbour's lawnmower. Therefore, you will have to work it out for him.

Figure 1: The neighbour mowing his lawn. Please note the desktop in the lawnmower.

Input: The first line contains n and r, the size of the city grid and the number of restaurants. Next is n lines each containing n binary digits. If there is a restaurant at a given square, then this is marked with a 1. If not, the square is marked with a 0.

Output: The coordinates of the best spot for Bob to stay. The top left square of the city grid is (0,0). If there is more than one best spot, output the one with the lowest numbered coordinates.

Constraints: $1 \leq n \leq 3000$
$1 \leq r \leq n^2$

Example

Input	Output
6 6 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	2 1

The sum of the Manhattan distances of each restaurant to (2,1) is 16. This is also true for (3,1), (2,2) and (3,2), but (2,1) has the lowest numbered coordinates. Additionally, if there are 2 possible locations (a, b) and (b, a) you should pick the location with the lowest x-coordinate.

NOSTER - 60 points

Noster (not to be confused with Nestor) has a list of updates they might implement in the future. Within this list, the updates appear in order that they can be implemented.

For each update, they also have an estimated happiness score of their users that would be the result of implementing said update. The primary goal of the Noster developers is to push out as many updates as possible. However, they only want to implement updates that improve (or maintain) the happiness score of their users compared to the happiness of the previous update (contrary to popular belief). Sometimes there are multiple possible combinations that satisfy these two conditions. In this case, the Noster developers prioritise updates with lower happiness values. So if they have the choice between `3, 4, 5, 7` and `3, 4, 6, 7`, they will pick the first one.

Figure 2: An average Noster board meeting.

Noster wants to know the total gain in happiness score they would get if they implement as many updates as they can, while making the increase in happiness between updates as small as possible. This gain in happiness score is calculated as the difference between the happiness of the first update and the happiness of the last update. Note that the happiness score can start out negative.

Input: The first line contains the number of updates that Noster can implement. Next is a list separated by spaces consisting of integers. Each of these integers represent the resulting happiness score after implementing the update at said index.

Output: The number of updates u that Noster will implement and the difference d between the happiness of the first update and the happiness of the last update in the Noster update list, given all the constraints above.

Constraints:
$0 < n < 10^8$ where n is the number of updates.
$-10^8 < h < 10^8$ where h is the happiness score

Example

Input	Output
`6` `3 2 2 2 2 4`	`5 2`

Explanation example 1

The Noster developers could start with the update that gives 3 happiness, but if they do that the only happiness increase they could get is to immediately release the update with 4 happiness, resulting in only 2 updates. Instead, they do not release update 3, and release all of the updates with happiness 2. Consequently, the updates they will release are `2, 2, 2, 2, 4` which are a total of 5 updates and have a difference in happiness of $4 - 2 = 2$. Therefore the output should be `5 2`.

Input	Output
`11` `4 3 5 4 8 12 6 27 33 29 5`	`6 26`

Explanation example 2

The Noster developers aim to release as many updates as possible, while keeping the happiness scores for each update as low as possible. This means that they could start out with an update with happiness of 4, followed by 5 or 4: `{4, 5, ...}` or `{4, 4, ...}`. However, they prefer to keep the happiness as low as possible, and choose to start their release plan with 3 instead of 4. This is possible because the roadmap that starts with 3 contains as many updates as the one that starts with 4: `{3, 5, 8, ...}` or `{3, 4, 8, ...}`.

Again, since they want to keep happiness as low as possible, instead of choosing 5 as their second update, they will choose 4. Consequently the start of the updates causes a happiness of `{3, 4, 8}`. They include the happiness updates `8, 12` instead of the update `6`, since `8, 12` are more updates than just `6`.

After `12` they can release the update `27`, and after that either `29` or `33`. They choose to release the update `29` because this only has an increase of 2 happiness, while the other option has an increase of 6. In conclusion, the update cycle started with `3` and ended with `29`, making the difference in happiness $29 - 3 = 26$. They have released a total of 6 updates, making the output of the program `6 26`.