Code Jam to I/O for Women 2018 - Code Jam to I/O for Women

Burger Optimization

Problem

In 2017, Google learned about a serious Android bug: in the burger emoji, the cheese was directly on top of the lower bun, rather than on the patty itself. Really, who makes a burger that way? Sundar, our CEO, vowed to drop everything and address this issue immediately.

To prevent this sort of situation in the future, the Code Jam team has constructed a mathematical model for understanding burgers. A burger consists of a stack of **K** ingredients between two buns, with each ingredient appearing exactly once. We are interested in the *distance-to-bun* value of each ingredient. The distance-to-bun value of an ingredient is the minimum number of other ingredients between the that ingredient and a bun:

- If **K** is even, then the distance-to-bun values for the ingredients (starting with the ingredient at the top of the stack) are: 0, 1, ..., **K**/2 - 1, **K**/2 - 1, ..., 1, 0.
- If **K** is odd, then they are: 0, 1, ..., ((**K** - 1) / 2) - 1, (**K** - 1) / 2, ((**K** - 1) / 2) - 1, ..., 1, 0.

After carrying out a lot of focus group testing (and eating a lot of burgers), we have determined that the i-th of each of our **K** ingredients has an optimal distance-to-bun value of $\mathbf{D_i}$. We think our burger emoji users will be happiest if we choose an ordering for our ingredients that minimizes the *error* value, which we define as the sum of the squared differences between each ingredient's optimal and actual distance-to-bun values.

For example, if we have five ingredients A, B, C, D, and E with optimal distance-to-bun values of 0, 2, 1, 1, and 2, and we place them between the buns in that order, then the error is $(0-0)^2 + (2-1)^2 + (1-2)^2 + (1-1)^2 + (2-0)^2 = 6$. If we instead place them in the order C, E, B, D, A, then the error is $(1-0)^2 + (2-1)^2 + (2-2)^2 + (1-1)^2 + (0-0)^2 = 2$, which turns out to be the minimum possible error for these ingredients.

Given the list of optimal distance-to-bun values for our ingredients, can you help us determine the smallest possible error?

Input

The first line of the input gives the number of test cases, **T**; **T** test cases follow. Each begins with one line containing an integer **K**: the number of ingredients in our burger. Then, there is one more line containing **K** integers $\mathbf{D_i}$, the optimal distance-to-bun values of our ingredients.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the smallest possible error, as described above.

Limits

$1 \le \mathbf{T} \le 100$.
Time limit: 20 seconds per test set.
Memory limit: 1GB.

$0 \leq \mathbf{D_i} \leq \text{floor}((\mathbf{K}-1)/2)$, for all i. (Each optimal distance-to-bun value is within the range of attainable distance-to-bun values.)

Small dataset (Test set 1 - Visible)

$1 \leq \mathbf{K} \leq 8$.

Large dataset (Test set 2 - Hidden)

$1 \leq \mathbf{K} \leq 100$.

Sample

Input	Output
3 5 0 2 1 1 2 1 0 6 2 2 2 2 2 2	Case #1: 2 Case #2: 0 Case #3: 10

Sample Case #1 is the one illustrated in the problem statement.

In Sample Case #2, there is only one ingredient in the burger; that is not much of a burger, but our model has to be able to handle this base case! There is no confusion over how to place the one ingredient, and the error is 0.

In Sample Case #3, there are six ingredients, but all of them have an optimal distance-to-bun of 2. Any way of placing them is equivalent, and the error is $2^2 + 1^2 + 0^2 + 0^2 + 1^2 + 2^2 = 10$.