This is an individual project. This project is very challenging and carries a negligible weight (3%) in the total grade, so you may consider skipping it unless you want an A-level grade ($A^-$, A, $A^+$). Make your algorithms as efficient as possible, and write rigorous analyses. You may write the analyses as comments or attach a separate text file (only a plain text file is accepted). The submission format is similar to assignments (please prepare the files yourself).

**Question 1 (30 points).** In Lab 10, we have seen how to merge $k$ sorted arrays into a single sorted array. In that Lab, all the input arrays have almost the same length. (You may think of the situation the power is cut off during the execution of recursive mergesort, and we resume the task after the power is back. Then we have a set of sorted arrays almost the same length.)

However, for the general case, when the input arrays can have arbitrary lengths, this algorithm is very inefficient. Consider, for example, the input

$$[1, 2, 3, ..., \frac{n}{2}], [\frac{n}{2} + 1], [\frac{n}{2} + 2], ..., [n].$$

One way to fix it is always to merge two shortest arrays. For example,

$$\begin{aligned}
&[1, 2, 3, \ldots, 8], [9], [10], [11], [12], [13], [14], [15], [16].\\
&[1, 2, 3, \ldots, 8], [9, 10], [11], [12], [13], [14], [15], [16].\\
&[1, 2, 3, \ldots, 8], [9, 10], [11], [12], [13, 16], [14], [15].\\
&\qquad\vdots\\
&[1, 2, 3, \ldots, 8], [9, 10], [11, 12], [13, 16], [14, 15].\\
&\qquad\vdots\\
&[1, 2, 3, \ldots, 8], [9, 10, 13, 16], [11, 12, 14, 15].\\
&[1, 2, 3, \ldots, 8], [9, 10, 11, \ldots, 16].\\
&[1, 2, 3, \ldots, 16].
\end{aligned}$$

Implement this algorithm. *Hint: The key to this approach is to find two shortest arrays efficiently, which has to be done in constant time. You need a data structure that is easy to update to keep track of the lengths of the remaining arrays.*

**Question 2 (70 points).** We can store a symmetric binary matrix $B$ in a compact way. We use a two-dimensional array a, where the array a[i] stores the indices of one's in the ith row of $B$. (Yes, the index starts from 0.) For example, the matrices

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

can be represented as $[[3, 1], [0, 2], [3, 1], [2, 0], []]$ and $[[1], [0, 2], [3, 1], [2]]$, respectively.

Given such a compact representation, the task is to find a pair of identical rows. Return $(i, j)$ if there are distinct $i$ and $j$ such that rows $i$ and $j$ are the same, or $(0, 0)$ otherwise. For example, you should return either $(0, 2)$ or $(1, 3)$ for the first input, and you need to return $(0, 0)$ for the second.

*Hint: If you solve the first question correctly, the second should not be that difficult. Otherwise, it's an impossible task. In the input, the indices for each row are not sorted. You need to start by sorting them without using any sorting algorithms.*