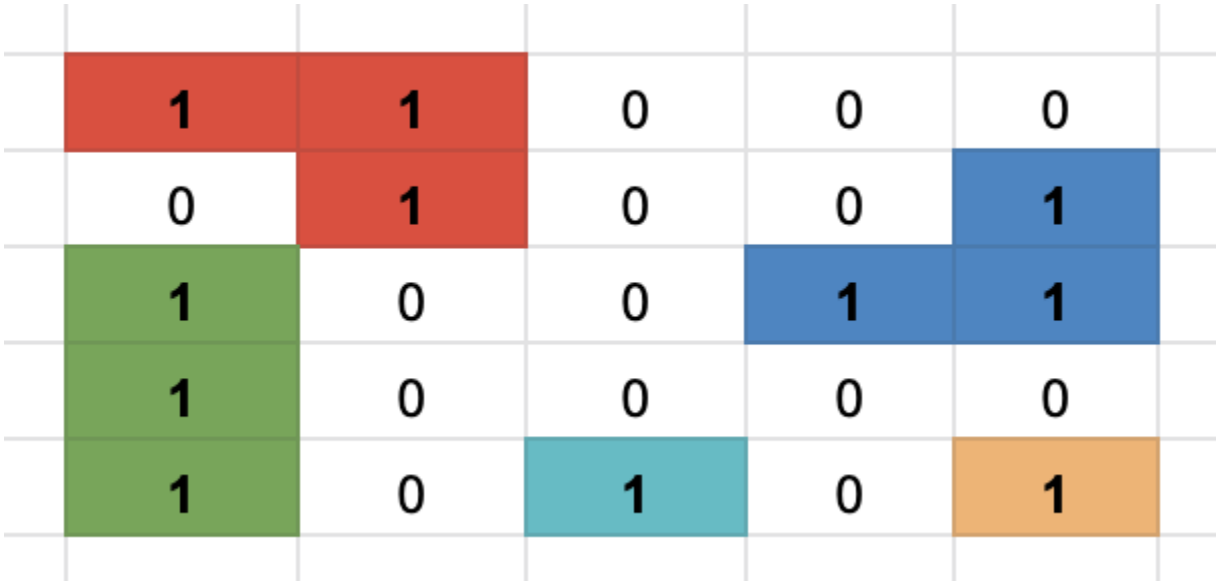
Interview Tasks



1. **Number of connected groups**

The goal of this task is to find the number of connected groups that are present on the matrix.

Two cells are considered to be connected if they both hold the value 1, and they are adjacent to each other either vertically or horizontally (they share a common border).

In the above image, we've got 5 total groups - each colored differently for visibility.

Given the following:

- Matrix dimensions N * M
- Matrix values for each cell

Write a program that will output the total number of groups present on the matrix.

Scoring mechanism:

Test cases will not be supplied. The provided solution must contain at least one input sample of your choice.

The task will be scored based on the following criteria:

- The solution is correct
- The solution is preferred to be iterative rather than recursive (Iterative solutions will be scored higher - recursive solutions are also accepted if the task cannot be solved

iteratively by the candidate. Submissions containing both types will be rewarded with bonus points.)
- The algorithmic complexity (big O) has to be calculated and explained (both for run time and space complexities)
- Code quality

The task has no other assumptions or limitations except for the above, but you may set your own as long as they are described inside the readme.

You may solve this exercise in whichever language you prefer. The input is conceptualized as a matrix, but you may use any available data structures to solve this problem.

The solution should contain the following:

- The source code
- Documentation whenever needed
- A readme (should contain a description of the solution, as well as the steps needed in order to run the code)

2. Data Store Library

For this task, you are tasked with designing and implementing a library that can be used to store and retrieve arbitrary data in multiple formats & destinations.

This library will be used by 3rd party developers, and it should expose a simple and structured API.

A record will be defined as a simple data structure where every key maps to a primitive value (fields values cannot be objects, arrays etc).

Given this definition of a record, the library must support the following operations:

- Record inserts & batch inserts
- Record query/retrieval
- Query filters (equality operations only), limit & offset
- Update and delete operations

You may use the storage format and destination of your choice, but the library must be expandable.
Support for other storage formats & destinations should not require significant code changes in order to be added (Example formats: json, xml, bytes etc. Example destinations: local drive, ftp, cloud storage etc).

Every supported storage format may be combined with any of the supported storage destinations.

The solution should support at least one format and one destination, and should also contain mock implementations of an alternative format and an alternative destination. (Example: data is stored in json format on the local drive. Mock: data stored as xml and uploaded to S3 - the mock methods do not have to be implemented)

The end user should be able to configure the library according to their needs.

The exposed API should be thread-safe, and tests must be present alongside the submitted solution.

The task will be scored based on the following criteria:

- Overall design and usability
- Simplicity of use
- Correctness
- Supported features
- Quality of code
- Quality of tests
- Documentation

This is primarily a design task - the performance of the chosen storage type does not matter.

Sample usage instructions should also be provided for the exposed interfaces.