**Instructions:**

Submission should be on a shareable link with public access (preferably on Git).

**Redistribution of Keys**

**Brief:**

We have 60 million passwords protected by a set of 512 encryption keys. Ideally, we’d like the passwords to be evenly distributed across the keys, but over time we tend to see the distribution become skewed. We need a batch process to rebalance this. It will decrypt a password under one key and encrypt it under another key until the spread is even.

There is a cost in terms of I/O to count the passwords and keys, and it will change due to online transactions whilst the batch is running. We will do this count once at the start of the batch job. There is a cost for the encryption, so we only want to move an item from one key to another a maximum of once!

The task here is to determine an efficient number of moves to rebalance the load. We’d like to allow a small tolerance to ensure the batch is not constantly rebalancing the passwords across the keys every night, i.e. if a count is within 1-2% of the exact desired value, we could leave it unchanged.

We will work with smaller numbers for this example:

|  |  |  |
| --- | --- | --- |
| Key | Current number of passwords | Desired number of passwords |
| 0 | 257 | 1000 |
| 1 | 1226 | 1000 |
| 2 | 852 | 1000 |
| 3 | 3117 | 1000 |
| 4 | 0 | 1000 |
| 5 | 1006 | 1000 |
| 6 | 991 | 1000 |
| 7 | 217 | 1000 |
| 8 | 1154 | 1000 |
| 9 | 1180 | 1000 |

The example above should be possible to solve exactly.

The program output should show the initial counts, the moves to be performed, and the final counts.

If we allow a tolerance of 1%, key 5 does not need to be touched. How will this affect the algorithm?

If we add 66 passwords to each of the first 7 keys, does it still work?

Please test this with different values and tolerances.

Consider how to get the data into the program. There are several ways. Remember this is an interview, show us what you can do.

**Output**

We are looking for output along the lines of:

Total passwords = 10000

Total keys = 10

Desired passwords per key = 1000

Tolerance = 1%, acceptable passwords per key = 990-1010

Moves:

1. Move 226 from key 1 to key 0
2. Move 517 from key 3 to key 0
3. Move 148 from key 3 to key 2