Name: Michael Rogers
ID: 105667404

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 4 **Spring 2019, CU-Boulder**

The ecology department is planning a large-scale fish migration study that involves electronically tagging and releasing millions of fish across North America, waiting six months, then trapping the fish and recording the where they found each fish, according to its tracking number. In previous smaller-scale experiments, the field scientists used a hand-held device that had a sensor for reading the electronic sensor and a small onboard hard drive that used a predefined table for storing the tag ID, timestamp, and current GPS. In this table, every possible tag had a preset row which allowed for very fast (constant-time) insertions and lookups. While the team would like to re-use this hardware, they do not think that there is enough hard drive space to account for a table with millions of rows. Help them figure out another solution that provides fast insertions and lookups without requiring large memory allocations. HINT: an individual scientist will only tag a few thousand fish at a time.

Name: Michael Rogers
ID: 105667404

CSCI 3104, Algorithms **Profs. Grochow & Layer**
Explain-It-Back 4 **Spring 2019, CU-Boulder**

Dear Ecology department collegues,

This sounds like a very interesting study that you guys are preparing for, and I understand the issue that you are facing. While the current data storage system you are using works quickly to insert and search for elements in the table, it will require far too much space to store the entire dataset in one hard drive. Luckily, I believe we can fix this problem while still maintaining something similar in speed to what you are used to. Right now, weather you know it or not, you are using a strategy called hashing to store your data. This is an extremely effective method when trying to increase the efficiency of a program. Hashing is a method of storing data in which you run data through a function called a hash function that will identify where to place the data in the table, then we can go back to the hash function to find a certain datapoint very easily later. Think of the hash function as an instruction guide to finding and inserting data points in a table. This is a very effective method for sorting through this data, however we cannot store every fish in one table because there will be far too many rows of data for one hard drive to handle. The best way to handle this is we need to split up the data between multiple hard drives. If we can make it so that the data collected by each individual scientist is stored on its own hard drive, then we can continue using the hashing method that you are used to while also continuing to use the same hardware. By continuing to use hashing as our method for storing and searching for data in each table, we can maintain the speed that you are used to. Another reason to switch to this strategy is that instead of using a predefined table, we can use multiple tables that are built as we collect the data. This means that instead of wasting space on rows that might not be used, we can use only the number of rows that we need to store all the data. I think that you will find that this strategy of dividing up the data between multiple hard drives will maintain the speed you are looking for while also fixing the problem with hard drive space. If you need any help implementing this method feel free to contact me.

Good Luck,
A Computer Scientist you know