Report

1. Instruction of program

You can compile the program with the Makefile and use the command “make all”. Then you will get the execution file. I have already compiled the file for MacOS system. Here is the usage example: **“./ADM-2023-A4-sort-branched-inplace 200 ../../assignment\ 03/ADM-2024-Assignment-3-data-TPCH-SF-1/l_discount-int8.csv”**. First parameter is the number of values to read from the file which is 200 in this case. Second parameter is the path of file which is the csv data from assignment 3.

Note: if the compile algorithm doesn’t work on your system, please change the gcc part of Makefile and compile the program again.

2. Implementation of sort algorithms

2.1 Branch mispreditions

I use the template of c program and just implement the sort algorithm. For easier explanation, I use the simple bubble sort, the complexity is $O(N^2)$. Then I use the compiled execution file to test the “l_discount-int8.csv” file with 200 rows.

The load time is 41 usec, the sort time is 48 usec. This algorithm may suffer from the branch misprediction. Because before update the value I use an if condition. The CPU assumes a particular result to optimize the execution. And the dataset is random, the CPU will mispredict the result.

2.2 Predicted

For predicted sort algorithm, I also use the bubble sort and the complexity is also $O(N^2)$. But this time, to avoid the branch misprediction I use bool type for condition without using the if condition. To update the sorted values I calculate the difference by multiplying diff and condition. If the condition is false the value of difference will be 0, and will not update. So that I can conduct the sort algorithm without the if condition, and also avoid the branch misprediction.

The load time is 38 usec, the sort time is 31 usec. This algorithm will not suffer from the branch misprediction, cause the CPU will not predict the result. And by using this type of algorithm, the execution efficiency is improved a lot.