**Used Design Approach**

Time complexity: For building and storing data. depends on sorting

1. Min O(nlogn)
2. Max O(n^2)

For queries: O(N) for getting N objects for all type of queries.

So basic idea behind this approach is to store 3 different structure having data for customer, book, transaction. Using Hash map for getting fast access to all the data and sorting stored objects corresponding to the required queries those object would have.

Type of data structures.

1. collection of customer object (like a list of customer object)
2. collection of book object (like a list of book object)
3. collection of transaction object (like a list of transaction object)

Object Structures.

struct book{

string bookid;

float price;

string bookname;

int soldcount;

};

struct customer{

list<int> tid;

string custid;

int noOfVisits;

int purchase;

};

struct transaction{

int tid;

string custid;

string booklist;

int totalamt;

};

So for designing a database like this I want at least our object access should be very fast so I am creating 3 different hash maps.

1. map <string, struct book \*> book\_id; //1st key is book id, 2nd key is address where that book information is stored
2. map <string, struct customer \*> customer\_id; //1st key is customer id, 2nd key is address where that customer information is stored
3. map <int, struct transaction \*> transaction\_id; //1st key is transaction id, 2nd key is address where that transaction information is stored

So now I would have very fast access to the objects of any type like book, customer or transaction.

Now second thing I would try to store my data based on which my most common queries are fast.

I have to serve these queries:

1. Get top N frequent customer and number of visits they made.
   1. For this purpose, I would sort my customer object list based on his number of visits. This would directly give me top N customer.
2. Top N highest transaction and their customer id.
   1. For this purpose, I would sort my transaction object list based on total amount of transaction. This would directly give me top N transaction and their corresponding customer id.
3. Top N highest selling books, least selling books and their quantity.
   1. For this purpose, I would sort my book object list based on its sold count. This would directly give me top N most selling books, least selling books and their sold count.

After all the sorting I would rebuild hash again and will fix the pointers. This would not hurt program complexity.

For queries related to discount, I am storing total purchase amount for every customer in its object. So any time I can look into that object and can check that he is eligible for discount or not. Its time complexity would be O(1) as we have direct access to the customer object through its hash map pointer.

Program is doing extra error handling for most of the bad conditions.

Possible improvement: An in place sorting of data would make a consistent searchable database on which we can run queries at any time.

Sample input: >mediaiq.exe -t transaction.txt -p price.txt -r 3 -d 1700 -c C12397

Sample output:

C12397 2,C1238 1,C3452 1,

C12397 1602,C12397 179,C3452 178,

P8 3,P673 2,P1 1,P342 1,P42 1,P456 1,P546 1,P563 1,P675 1,

P99 0,P1 1,P342 1,P42 1,P456 1,P546 1,P563 1,P675 1,P673 2,

1