**Implementation of Advanced Data Structures and Algorithms**

**Long Project 4**

***Group Members***

*Giridara Varma Elumalai*

*Srikanth Kannan*

*Praveen Erode Murugesan*

**Problem Statement:**

To implement Multi-dimensional search in a huge collection of data where every single item can have multiple attributes.

**Approach:**

* When there are multiple attributes data are usually indexed by multiple keys using efficient dictionary data structures like Balanced Trees and Hashes.
* Whenever a data is inserted it as added to the index and whenever it is deleted/updated it is removed/reassigned in the indexes as well.

**Implementation:**

* Every item is maintained as single class called Item with the Id, Price and Descriptions.
* The items are all maintained in a TreeMap with Id as the key.
* Items with the same price are indexed using TreeMap. Another indexing of items is maintained using every single description of the item as a second level of indexing.
* There is also an indexing of items with the entire set of descriptions considered as a whole key. A Hash Map is used for this indexing.
* To save space for the problem only item counts are maintained in the indexes, instead of the actual items. In real scenarios they would be a collection of items.

**Performance Analysis:**

Three dictionary data types were considered to maintain the indexes. Large number (around 1M) of insert, search and iteration were done using all three data structures to compare their performance multiple times. The data structures considered were:

* Java's ConcurrentSkipListMap
* Java's TreeMap (uses RB Tree)
* A hybrid of DoublyLinkedList & RB Tree, where the data are maintained in the list and references to the data are maintained in the RB Tree.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Structure | Running time (ms) | | | Memory used (MB) |
| Insert | Find | Iterate |
| SkipListMap | 3900 | 3100 | 30 | 68 |
| TreeMap | 1900 | 1200 | 60 | 64 |
| Hybrid TreeList | 2100 | 1200 | 30 | 92 |

Though the Hybrid TreeList performed better as good as the SkipList for iteration and as good as the TreeMap for insert & find, it consumed almost 50% more space. And the improvement in time achieved won't be significant unless we perform sequential iteration over the data for a large number of cases.

**Conclusion:**

The Project helped understand how to implement a Multi-dimensional search where items are indexed using multiple keys. Three important challenges faced were:

* To implement indexing with little code repetition. Interface and Generic class index would be a good way, but for the use case of the problem we needed some extra conditions for every indexing so separate classes were made here.
* To maintain the consistency of the indexes with little effort. For index update scenarios having a wrapper over the count helped to update the count just by a look-up and then updating in the returned object. Without the wrapper the old count had to be retrieved once and then inserted again, making 2 look-ups.
* The eternal Space-Time trade-off. Though the Hybrid TreeList was significantly faster it threw an OutOfMemory Error for the Bad inputs test case.