**COMP 3323 Assignment 3 – Question 1 & 2’s Analysis**

**Name: Wang Yuli**

**U.No: 3035028946**

**Question 1. Document indexing for containment query**

|  |  |
| --- | --- |
| **Doc ID** | **Terms** |
| **1** | A, C, D, E, F, G |
| **2** | A, B, C, D, H, I, J, K |
| **3** | A, B, E, F, H, I, K |
| **4** | A, B, D, H, J, K |
| **5** | A, B, D, E, H, I |
| **6** | A, C, D, E, F,H |
| **7** | B, E, F, H, J |
| **8** | E, F, G, H, I, J, K |

Numerical integer identifier: 1~11

Signature length: 8 bits

-> mapping terms to 0~7

1. **(i) Draw signature file; (ii) query strategy on containment of all A, B, C**

**Answer:**

1. Encoding of terms

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J | K |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

Encoded Database by the above encoding table

|  |  |
| --- | --- |
| **Doc ID** | **Terms** |
| **1** | 1, 3, 4, 5, 6, 7 |
| **2** | 1, 2, 3, 4, 8, 9, 10, 11 |
| **3** | 1, 2, 5, 6, 8, 10, 11 |
| **4** | 1, 2, 4, 8, 10, 11 |
| **5** | 1, 2, 4, 5, 8, 9 |
| **6** | 1, 3, 4, 5, 6, 8 |
| **7** | 2, 5, 6, 8, 10 |
| **8** | 5, 6, 7, 8, 9, 10, 11 |

By applying

|  |  |
| --- | --- |
| **Doc ID** | **Terms (of H(e))** |
| **1** | 1, 3, 4, 5, 6, 7 |
| **2** | 1, 2, 3, 4, 0, 1, 2, 3 |
| **3** | 1, 2, 5, 6, 0, 2, 3 |
| **4** | 1, 2, 4, 0, 2, 3 |
| **5** | 1, 2, 4, 5, 0, 1 |
| **6** | 1, 3, 4, 5, 6, 0 |
| **7** | 2, 5, 6, 0, 2 |
| **8** | 5, 6, 7, 0, 1, 2, 3 |

Therefore, the **signature file:**

|  |  |
| --- | --- |
| **Doc ID** | **Signatures** |
| **1** | 01011111 |
| **2** | 11111000 |
| **3** | 11110110 |
| **4** | 11111000 |
| **5** | 11101100 |
| **6** | 11011110 |
| **7** | 10100110 |
| **8** | 11110111 |

1. q =

q.sig = 11100000

Therefore, our strategy is

1. Obtain candidates: the signature file is scanned to, find Doc IDs which qualify q.sig:

*for each document in signature file:*

*if : // i.e.*

*e is marked as a candidate;*

1. Candidates obtained in above is compared to q at document level to get the final solution. This step is similar to merge-join two sorted table

Hence:

from step 1, we got candidate table:

|  |  |
| --- | --- |
| **Candidates** | **Terms (Document level)** |
| **2** | A, B, C, D, H, I, J, K |
| **3** | A, B, E, F, H, I, K |
| **4** | A, B, D, H, J, K |
| **5** | A, B, D, E, H, I |
| **8** | E, F, G, H, I, J, K |

From step 2, we do verification and got the final results:

|  |  |
| --- | --- |
| **Doc ID** | Terms (Document level) |
| **2** | A, B, C, D, H, I, J, K |

1. **(i) Draw inverted file, (ii) query strategy on containment of all A, B, C**

**Answer:**

1. Inverted File

|  |  |
| --- | --- |
| **Directory (Term)** | **Inverted List (Doc IDs)** |
| **A** | 1, 2, 3, 4, 5, 6 |
| **B** | 2, 3, 4, 5, 7 |
| **C** | 1, 2, 6 |
| **D** | 1, 2, 4, 5, 6 |
| **E** | 1, 3, 5, 6, 7, 8 |
| **F** | 1, 3, 6, 7, 8 |
| **G** | 1, 8 |
| **H** | 2, 3, 4, 5, 6, 7, 8 |
| **I** | 2, 3, 5, 8 |
| **J** | 2, 4, 7, 8 |
| **K** | 2, 3, 4, 8 |

1. q =

Strategy:

merg-join inverted lists A & B & C, resulting in list X as our final result

Hence:

From the strategy, X = {2}

1. **Discuss advantages & disadvantages of Signature File & Inverted File**

**Answer:**

Suppose we have

N documents, with maximum M terms each,

altogether we have W different terms.

L for signature length

We compare the performance of Signature File & Inverted File by different aspects as follows:

* Signature File, using signature-based indexes in slices
  + Encoding of terms:
  + Encoding database using the term encoding table:
  + Applying Hash Function:
  + Building signature file:

Evaluating Query:

* + Get candidate table, do logical AND:
  + Refinement step:

If we consider inserting new document into the database, all slices need to be updated, thus the cost is for one document

* Inverted File:
  + Building Inverted File: scan all terms for only once:

Evaluating Query: Do merge-join for predicates contained in the query

Suppose we combine the predicates by “logical AND” only, and we have P predicates: since the inverted lists are already sorted, cost is:

If we consider inserting new document into the database, all directory need to be updated, thus the cost is for inserting, and we also need to sort each directory again. Thus this procedure is very expensive

Therefore, we can conclude from the above:

1. To evaluate a containment query, Inverted File is superior than Signature Files, because of its advantage of organizing the document in the sorted list for each terms. The sorting is also performed very naturally. Computational details are as follows:

Generally, P << M. Hence, for static database:

Building file:

Evaluating Query:

Hence, Inverted File is better when perform querying in static database

1. For dynamic database with really big N, where insertion/deletion is performed very often, Signature File might be a better choice, since:

Insertion in Signature File costs

Insertion in Inverted File costs

Even when we use the most efficient sorting algorithm, i.e. insertion sort, in this case, we still need to sort all N lists, which makes the overall cost really expensive. Besides, for deletion, the Signature File costs , while Inverted File costs

Therefore, dynamic database with often insertion/deletions, Signature File will have better performance in the insertion/deletion process.

**Question 2. Analysis**

Comparing Algorithm TA & NRA, we notice that the two algorithms perform quite differently. We use different parameters to do the analysis:

TA: user locations, # of user locations, # of rounds of get\_next() call (i.e. # of get\_next() calls performed on each given node), # of get\_spd calls.

NRA: user locations, # of user locations, # of rounds of get\_next() call (i.e. # of get\_next() calls performed on each given node), # of upper bound update performed, # of lower bound update performed.

To do the performance analysis. We choose point sets {} close to & far from each other. Altogether I performed about 20 test cases. Here is 2 expressive examples below:

* **Close data set:**

**-------- TA Algorithm Performance view ---------**

**aggregation function: 1**

**user locations: 0, 1, 2,**

**# of user locations: 3**

**# of rounds (# of rows of get\_next() call): 3**

**# of get\_spd() calls: 18**

**-------- Final result --------**

**TA: meet at 1**

**-------- NRA Algorithm Performance view ---------**

**aggregation function: 1**

**user locations: 0, 1, 2,**

**# of user locations: 3**

**# of rounds (# of rows of get\_next() call): 4**

**# of update upper bounds: 18**

**# of update upper bounds: 18**

**-------- Final result --------**

**NRA: meet at 1**

* **Far data set:**

**-------- TA Algorithm Performance view ---------**

**aggregation function: 1**

**user locations: 0, 888, 200, 1000,**

**# of user locations: 4**

**# of rounds (# of rows of get\_next() call): 128**

**# of get\_spd() calls: 1536**

**-------- Final result --------**

**TA: meet at 682**

**-------- NRA Algorithm Performance view ---------**

**aggregation function: 1**

**user locations: 0, 888, 200, 1000,**

**# of user locations: 4**

**# of rounds (# of rows of get\_next() call): 378**

**# of update upper bounds: 169429**

**# of update upper bounds: 169429**

**-------- Final result --------**

**NRA: meet at 682**

From the experiments, we can know that generally, TA performs better than NRA. When the points are close to each other, there’s not much difference on *# of rounds* and other kind of costs.

However, when the points are far from each other, TA needs much fewer rounds to find the optimal solution. This is because TA use get\_spd() to get right away when encountering any node . However, NRA must follow the priority queue order to fill up the missing entries of gradually. This makes TA superior.

Besides, when the nodes are far from each other, the cost for update upper/lower bounds are also much more expensive than the get\_spd() calls. This is because, as *rounds* increases, the hash table in NRA will grow really large, making the cost for updating the bounds really expensive. Thus, TA is also better in terms of the inner loop function calls and saving memory space.