3) Linear Hashing

Perform linear hashing for the following given parameters:

Using the following sequence of hash functions:

$$H_i(K) = K(2 \cdot 2^i) \text{ with } i \in \{0, 1, 2, \ldots, n\}$$

The hash table should be initialized with 2 buckets. Each bucket has a capacity of 3 entries. If more than $\beta > \frac{2}{3}$ of the table is occupied, controlled splitting should be performed.

Insert the following values in the given order:

$$27, 13, 28, 3, 21, 8, 27, 16, 36$$

Write down what happens during each insert. Also visualize your buckets after every split.

Solution:

Number of buckets to be considered ,**N = 2**

Number of records per bucket to be considered, **b=3**

Given **hash function** to be considered

$$H_i(K) = K(2 \cdot 2^i) \text{ with } i \in \{0, 1, 2, \ldots, n\}$$

Given threshold = $\beta_s = 2/3 = 0.66$

Number of values inserted into buckets, **x = 8**

As we are using Linear hashing, we would have a pointer **p**, pointing to the first bucket initially and follows the round robin fashion to split the buckets. After applying the hash function, the output of the hash function would be the bucket numbers the values would be placed into.

Values: 27, 13, 28, 3, 21, 8, 27, 16, 36

(i) Inserting 27 in the bucket and i=0

$$H_0(27) = 27 mod (2 \cdot 2^0)$$

$$H_0(27) = 27 mod (2 \cdot 1) = 27 mod 2 = 1$$

P BUCKET 0	BUCKET 1
	27

(ii) Inserting 13 in the bucket

$$H_0(13) = 13 mod (2 \cdot 1) = 13 mod 2 = 1$$

P BUCKET 0	BUCKET 1
	27 13

(iii) Inserting 28 in the bucket

$$H_0(28) = 28 mod (2 \cdot 1) = 28 mod 2 = 0$$

P BUCKET 0	BUCKET 1
28	27 13

(iv) Inserting 3 in the bucket

$$H_0(3) = 3 mod (2 \cdot 1) = 3 mod 2 = 1$$

P BUCKET 0	BUCKET 1
28	27 13 3

(v) Inserting 21 in the bucket

$$H_0(21) = 21 mod (2 \cdot 1) = 21 mod 2 = 1$$

P BUCKET 0	BUCKET 1
28	27 13 3
	21

The bucket 1 is full and hence the value of 21 goes to the overflow bucket. We have to calculate the value of β

$$\beta = x/(b \times M)$$

$$\beta = 5/(2 \times 3) = 5/6 = 0.83$$

But $\beta_s = 0.66, \beta > \beta_s$, **hence the controlled splitting must be performed**. The pointer points to bucket 0, and hence we **split the bucket 0 into bucket 0 and bucket 2.** Bucket 0 and Bucket 2 follow H_1 and bucket 1 follows H_0

BUCKET 0	**P** BUCKET 1	BUCKET 2
28	27 13 3	
h1	**h0**	**h1**
	21	

Apply H_1 function to the values that are already present in **BUCKET 0**

$$H_1(28) = 28 mod (2 \cdot 2.2^1) = 28 mod 4 = 0$$

After H_1 function, the value of 28 remains in bucket 0.

(vi) Inserting 8 in the bucket

$$H_0(8) = 8 mod (2 \cdot 1) = 8 mod 2 = 0$$

p is the pointer pointing to bucket 1, and therefore the value of **p=1**

$$H_0(8) \geq p = False$$

$$H_0(8) < p = 0 < 1 = True$$

Hence use H_1 function to place the value of 8 in bucket

$$H_1(8) = 8 mod (2 \cdot 2.2^1) = 8 mod 4 = 0$$

BUCKET 0	**P** BUCKET 1	BUCKET 2
28 8	27 13 3	
h1	**h0**	**h1**
	21	

(vii) Inserting 27 in the bucket

For the upcoming values, we don't know if it belongs to H_0 or H_1 and hence we apply $H_0 first$.

$$H_0(27) = 27 mod (2 \cdot 1) = 27 mod 2 = 1$$

The bucket 1 is full and hence the value of 27 goes to the overflow bucket.

BUCKET 0	**P** BUCKET 1	BUCKET 2
28 8	27 13 3	
h1	**h0**	**h1**
	21 **27**	

We have to calculate the value of β

$$\beta = x/(b \times M)$$

$$\beta = 7/(3 \times 3) = 7/9 = 0.77$$

But $\beta_s = 0.66, \beta > \beta_s$, **hence the controlled splitting must be performed**. The pointer points to bucket 1, and hence we **split the bucket 1 into bucket 1 and bucket 3.**Now that all the buckets are split doubly, the pointer moves back to bucket 0 as per round robin fashion and **all the buckets follow H_1 function** and **H_0 function is eliminated.**

H_1 function is applied to all the values that are present previously in the table and the values get reshuffled due to this.

$$H_1(27) = 27 mod (2 \cdot 2.2^1) = 27 mod 4 = 3$$

$$H_1(13) = 13 mod (2 \cdot 2.2^1) = 13 mod 4 = 1$$

$$H_1(28) = 28 mod (2 \cdot 2.2^1) = 28 mod 4 = 0$$

$$H_1(3) = 3 mod (2 \cdot 2.2^1) = 3 mod 4 = 3$$

$$H_1(21) = 21 mod (2 \cdot 2.2^1) = 21 mod 4 = 1$$

$$H_1(8) = 8 mod (2 \cdot 2.2^1) = 8 mod 4 = 0$$

$$H_1(27) = 27 mod (2 \cdot 2.2^1) = 27 mod 4 = 3$$

P BUCKET 0	BUCKET 1	BUCKET 2	BUCKET 3
28 4	13 21		27 3
h1	**h1**	**h1**	**h1**

(viii) Inserting 16 in the bucket

$$H_1(16) = 16 mod (2 \cdot 2.2^1) = 16 mod 4 = 0$$

P BUCKET 0	BUCKET 1	BUCKET 2	BUCKET 3
28 4 16	13 21		27 3
h1	**h1**	**h1**	**h1**

(ix) Inserting 36 in the bucket

$$H_1(36) = 36 mod (2 \cdot 2.2^1) = 36 mod 4 = 0$$

The bucket 0 is full and hence the value of 27 goes to the overflow bucket.

P BUCKET 0	BUCKET 1	BUCKET 2	BUCKET 3
28 4 16	13 21		27 3
h1	**h1**	**h1**	**h1**
36			

We have to calculate the value of β

$$\beta = x/(b \times M)$$

$$\beta = 9/(4 \times 3) = 9/12 = 0.75$$

But $\beta_s = 0.66, \beta > \beta_s$, **hence the controlled splitting must be performed**. The pointer points to bucket 0, and hence we **split the bucket 0 into bucket 0 and bucket 4.**Bucket 0 and Bucket 4 follow H_2 function and bucket1, bucket 2 and bucket 3 follows H_1 function.

Apply H_2 for the existing elements of the bucket 0

BUCKET 0	**P** BUCKET 1	BUCKET 2	BUCKET 3	BUCKET 4
16	13 21		27 3	28 4
h2	**h1**	**h1**	**h1**	**h2**
36				

Applying H_2 function for 36 we get

$$H_2(36) = 36 mod (2 \cdot 2.2^2) = 36 mod 8 = 4$$

Final output of values in the buckets are as follows

BUCKET 0	**P** BUCKET 1	BUCKET 2	BUCKET 3	BUCKET 4
16	13 21		27 3	28 4 36
h2	**h1**	**h1**	**h1**	**h2**

4) Top-k Algorithms

Apply the FA and TA algorithm for $k = 2$, using addition as aggregation function, on the following three index lists. Write down all index list accesses, as well as the current top-k documents after each step. How many sequential and how many random accesses were executed?

L_1
d_2 0.9
d_3 0.8
d_1 0.5
d_6 0.4
d_5 0.3
d_8 0.2
d_7 0.1

L_2
d_1 0.8
d_2 0.7
d_3 0.5
d_6 0.4
d_8 0.3
d_4 0.3
d_7 0.1

L_3
d_3 0.9
d_4 0.8
d_1 0.6
d_2 0.4
d_5 0.3
d_7 0.2
d_8 0.2

Solution:

(i) Fagin's Algorithm

For k=2, we do sequential reads until we find all top 2 values as k=2.

Aggregation function: **sum**

L_1
d_2 0.9
d_3 0.8
d_1 0.5

L_2
d_1 0.8
d_2 0.7
d_3 0.5

L_3
d_3 0.9
d_4 0.8
d_1 0.6

Calculating the values of the documents

For the pages that are not in the sequential scan, we consider as 0 by default.

$$d1 = 0.5 + 0.8 + 0.6 = 1.9$$

$$d2 = 0.9 + 0.7 + 0 = 1.6$$

$$d3 = 0.8 + 0.5 + 0.9 = 2.2$$

Top values of k=2 so far are **d3,d1** We have performed 3 sequential reads per row in the index list, hence we have performed **9 sequential reads** on the whole so far.

There are possibilities that the partially scanned documents might have the better values and henc we perform random access on the partially scanned documents.

$$d2 = 0.9 + 0.7 + 0.4 = 2$$

$$d1 = 0 + 0.3 + 0.8 = 1.1$$

d_2 takes 2 random scans, one for L1 and other for L2. d4 takes one random scan, summing up to 3 random scans. As, d_2 value is greater than d_1, we consider d_2 to be the top element along with d_3

Hence, **d3,d2** are the top 2 documents for k=2.
Sequential accesses executed : 9
Random accesses executed : 3

(ii) Threshold Algorithm: Read sequentially from each index list and perform random reads for every document found in the sequence list. We compare the threshold of the aggregated values of the documents with the scan line score. When the value of **aggregated values > scan line score**, then we terminate the algorithm.

L_1
d_2 0.9

L_2
d_1 0.8

L_3
d_3 0.9

SEQUENTIAL ACCESS	RANDOM ACCESS
L1: d_2 **0.9**	**L2:** d_2 0.7, **L3:** d_2 0.4
L2: d_1 **0.8**	**L1:** d_1 0.5, **L3:** d_1 0.6
L3: d_3 **0.9**	**L2:** d_3 0.8, **L3:** d_3 0.5

Aggregation sum of the documents:

$$d2 = 0.9 + 0.7 + 0.4 = 2$$

$$d1 = 0.8 + 0.5 + 0.6 = 1.9$$

$$d3 = 0.9 + 0.8 + 0.5 = 2.2$$

Scan line score = L1 + L2 + L3 = 0.9 + 0.8 + 0.9 = 2.6
Current top-k: {d3,d2}
The aggregated values of **d3 and d2 < scan line score** - continue algorithm
Sequential access for L1: d3 0.8

L_1
d_2 0.9
d_3 0.8

L_2
d_1 0.8

L_3
d_3 0.9

Scan line score = 0.8 + 0.8 + 0.9 = 2.5
The aggregated values of **d3 and d2 < scan line score** - continue algorithm
Sequential access for L2: d2 0.7

L_1
d_2 0.9
d_3 0.8

L_2
d_1 0.8
d_2 0.7

L_3
d_3 0.9

Scan line score = 0.8 + 0.7 + 0.9 = 2.4

The aggregated values of **d3 and d2 < scan line score** - continue algorithm

Sequential access for L3: d4 0.8

Random access for L1 d4:0 and L2 d4:0.3

Aggregation of d4 $d4 = 0 + 0.8 + 0.3 = 1.1$

L_1
d_2 0.9
d_3 0.8

L_2
d_1 0.8
d_2 0.7

L_3
d_3 0.9
d_4 0.8

Scan line score = 0.8 + 0.7 + 0.8 = 2.3

The aggregated values of **d3 and d2 < scan line score** - continue algorithm

Sequential access for L1: d1 0.5

L_1
d_2 0.9
d_3 0.8
d_1 0.5

L_2
d_1 0.8
d_2 0.7

L_3
d_3 0.9
d_4 0.8

Scan line score = 0.5 + 0.7 + 0.8 = 2

The aggregated values of **d3 and d2 < scan line score** - continue algorithm

Sequential access for L2: d1 0.5

L_1
d_2 0.9
d_3 0.8
d_1 0.5

L_2
d_1 0.8
d_2 0.7
d_3 0.5

L_3
d_3 0.9
d_4 0.8

Scan line score = 0.5 + 0.5 + 0.8 = 1.8

The aggregated values of **d3 and d2 > scan line score - Terminate algorithm**

Sequential Accesses: 8 Random Accesses : 8