**THEORY PART**

**Question 1**

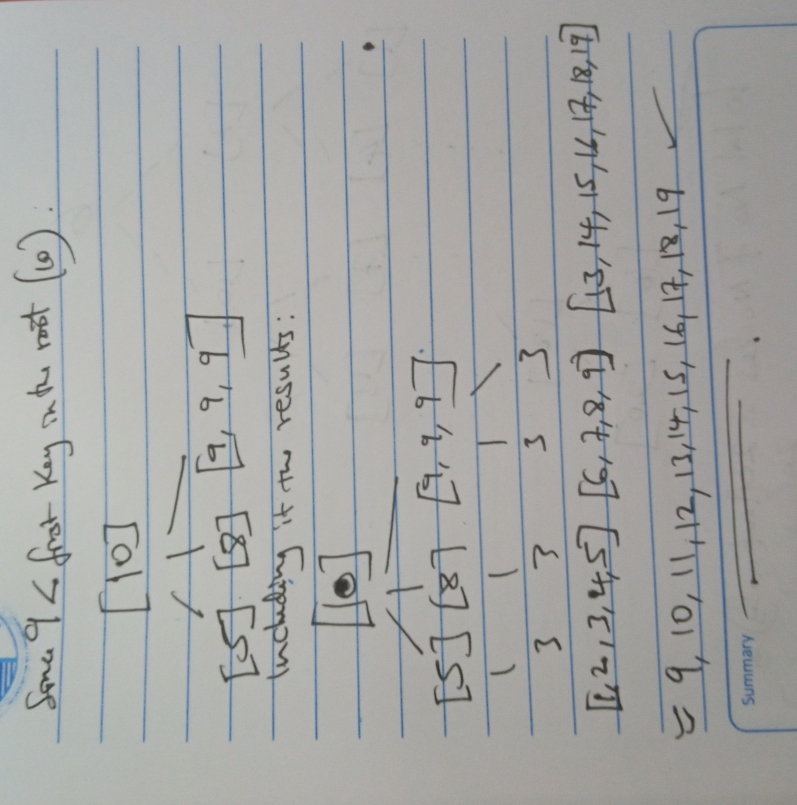
P = ceil(logM(N/B))

N = 1,000,000 pages

B = 6 buffers

P = ceil(log6(1,000,000/6)) = ceil(log(166,667)) ≈ 18

**Question 2**



**Question 3**

* h0(x) takes the rightmost 2 bits of key x as the hash value.

0, 4, 8, 12, 16, and 20 hashed to bucket 0

1, 5, 9, 13, 17, and 21 hashed to bucket 1.

* h1(x) takes the rightmost 3 bits of key x as the hash value.

0, 1, 2, 3, 8, and 9 hashed to bucket 0

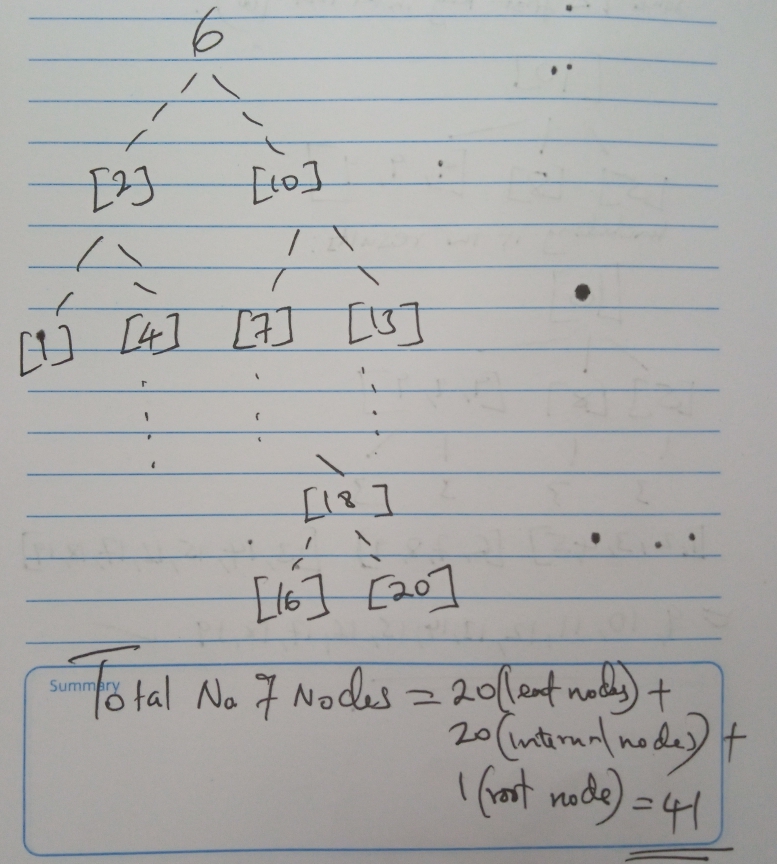
4, 5, 6, 7, 12, and 13 hashed to bucket 1

16, 17, 18, 19, 24, and 25 hashed to bucket 2

* If we insert a key that hashes to bucket 2 (using h1), then we will need to split that bucket.

The largest key less than 25 that hashes to bucket 2 using h1 is **19**.

**Question 4**



**Question 5**

Plan I is more efficient than Plan II.

Because in Plan I, we requires less processing in carrying out the join operations as compared to Plan II which requires more processing.

**Question 6**

The statement is true.

The statement is true, as multi-threaded execution is required for operators that receive input from multiple children in the vectorized processing model.

**Question 7**

Use parallel processing: Parallel processing techniques such as multi-threading and distributed processing can be used to improve the performance of the hash join algorithm by exploiting multiple processing resources.

Use hybrid join algorithms: Hybrid join algorithms such as sort-merge join and index-join can be used in combination with hash join to optimize performance for certain scenarios.

**Question 8**

i. Scan Applicants table with the city attribute equals to 'Seattle':

- Number of pages=500

- Number of matched tuples=50

- Cost=500 (sequential scan) + 50 (50 I/Os to read matched tuples)

ii. Join the result with Schools table on sid attribute:

- Number of pages in Schools table=200

- Number of matched tuples=5

- Cost=200 (sequential scan) + 5 (5 I/Os to read matched tuples)

iii. Join the result with Major table on id attribute using the index:

- Number of leaf pages in the index=100

- Number of matched tuples=2

- Cost=100 (2 I/Os to search for the matching keys in the index) + 2 (2 I/Os to read matched tuples from Applicants table)

Total cost of the query plan= 657 page I/Os.

**Question 9**

A.) Use a different hash function that distributes the tuples more evenly across the hash buckets. Another approach is to use a hybrid hash join that combines hashing with another join algorithm, such as sort-merge join, to handle the overflow pages more efficiently.

B.)

M + ⌈M/(B-1)⌉ \* N

M = 2,400 pages

N = 1,200 pages

B = 75 buffer pages

2400+[2400/(75-1)]\*1200

2400+[2400/74]\*1200

2400+32.43\*1200

2400+38916

41316

**Question 10**

A full binary tree has either 0 or 2 children for each internal node. Therefore, the number of leaf nodes L is equal to the number of internal nodes plus 1:

L = 2n + 1

**Question 11**

1. Insert 13, Insert 12