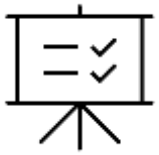
Advanced Algorithms & Data Structures

AIM OF THE SESSION

To familiarize students with the basic concept of Double Hashing, Extendible hashing, Rehashing

INSTRUCTIONAL OBJECTIVES



This Session is designed to:

1. Demonstrate Double Hashing, Extendible hashing, Rehashing
2. List out the advantages and applications of hashing techniques.

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Define Double Hashing, Extendible hashing, Rehashing.
2. Describe each method.
3. Summarize definition, types of hashing and its applications

Double Hashing

Double hashing

- Double hashing is a collision resolution technique that utilizes two hash functions to handle collisions when multiple keys map to the same index in a hash table.
- First hash function is typically **hash1(key) = key % TABLE_SIZE**
- Whenever collision occurs, second hash function is

  hash2(key) = PRIME – (key % PRIME)

  where PRIME is a prime smaller than the TABLE_SIZE.
- Then calculate **h(key, i) = (hash1(key) + i * hash2(key)) % tableSize**

  Where, i is the probe count, starting from 1 and incrementing for each failed attempt.

Double Hashing

Example: Consider a sequence of keys as 20, 34, 45, 70 with table_size 11

- $h_1(20) = 20 \bmod 11=9$
- $h_1(34) = 34 \bmod 11=1$
- $h_1(45) = 45 \bmod 11=1$

  Collision Occurs at index 1, so calculate second hash function

  $h_2(45) = 7 - (45 \bmod 7) = 3$

  Now, calculate $h(45, 1) = (1 + 1 * 3) \bmod 11=4$, where i=1
- $h_1(70) = 70 \bmod 11=4$

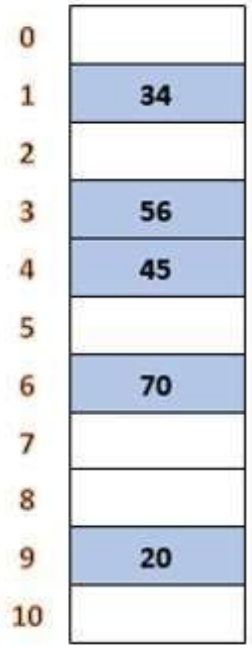
  Collision Occurs at index 4, so calculate second hash function

  h2(70) = 7 - (70 mod 7) = 7

  h(70, 1) = (4 + 1 * 7) mod 11=1

  Again, Collision Occurs, increment i, h(70, 2) = (1 + 2 * 7) mod 11=4, where i=2

Again, Collision Occurs, increment i, h(70, 3) = (1 + 3 * 7) mod 11=0, where i=3



Rehashing

Rehashing:

- Rehashing method is used to prevent collisions by increasing the size of a hashmap and redistributing the elements into new buckets based on their new hash values.

How Rehashing is done?

- For each new entry to the map, check the load factor.
- If it's greater than its pre-defined value (or default value of 0.75 if not given), then Rehash.
- For Rehash, make a new array of double the previous size and make it the new bucket array.
- Then traverse to each element in the old bucket Array and call the insert() for each so, as to insert it into the new larger bucket array.

Load Factor Calculation

Load factor can be calculated by using the following formula:

- The initial capacity of the HashMap * Load factor of the HashMap.
- Good load factor value is 75, while HashMap's default beginning capacity is 16, when the number of elements hits or exceeds 0.75 times the capacity, the complexity rises.
- To combat this, the array's size is doubled, and all the values are hashed again and saved in the new double-sized array.
- After Rehashing items can fall into same bucket or different bucket into new array.

Ex: when the number of items reaches 12, rehashing begins. (as 12 = 0.75 * 16).

Extendible Hashing

- Extendible Hashing is a dynamic hashing method wherein directories, and buckets are used to hash data. It is an aggressively flexible method in which the hash function also experiences dynamic changes.

Frequently used terms in Extendible Hashing

- **Directories:** Instead of using a single hash table, extendible hashing employs a multi-level directory structure.
- **Buckets:** Each directory entry points to a bucket, which stores a subset of key-value pairs.
- **Local Depth:** Each bucket has a local depth, indicating the number of bits from the hash value used for addressing within that bucket.
- **Global Depth:** The directory's global depth determines the maximum number of bits used for addressing.
- **Bucket Splitting:** When the number of elements in a bucket exceeds a particular size, then the bucket is split into two parts.
- **Directory Expansion:** Directory Expansion Takes place when a bucket overflows. Directory Expansion is performed when the local depth of the overflowing bucket is equal to the global depth.

Extendible Hashing

Example: Consider a prominent example of hashing the following elements: 16,4,6,22,24,10,31,7,9,20,26 with bucket Size 3.

Hash Function: Suppose the global depth is X. Then the Hash Function returns X LSBs.

First, calculate the binary forms of each of the given numbers.

- 16- 10000
- 4- 00100
- 6- 00110
- 22- 10110
- 24- 11000
- 10- 01010

Extendible Hashing

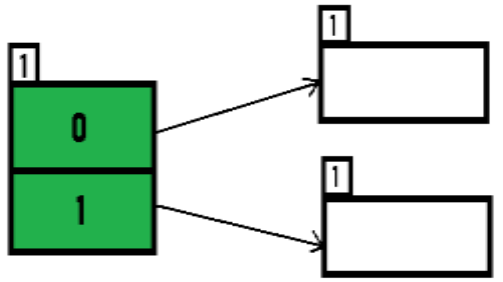
31- 11111

7- 00111

9- 01001

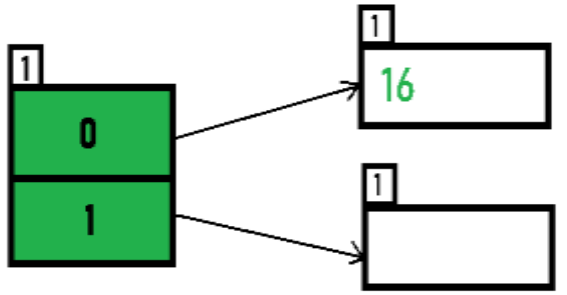
20- 10100

26- 11010



- Initially, the global-depth and local-depth is always 1. Thus, the hashing frame looks like this:

Inserting 16:



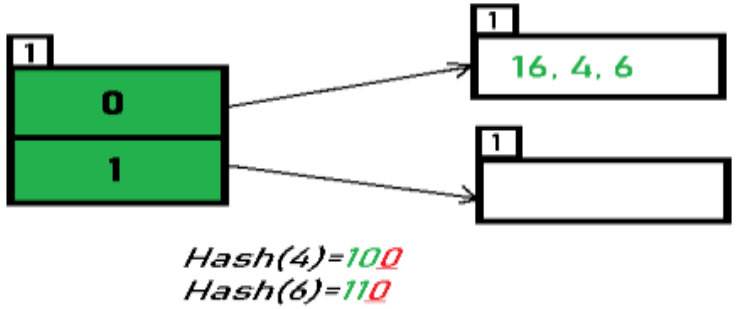
Hash(16)= 10000

- The binary format of 16 is 10000 and global-depth is 1. The hash function returns 1 LSB of 10000 which is 0. Hence, 16 is mapped to the directory with id=0.

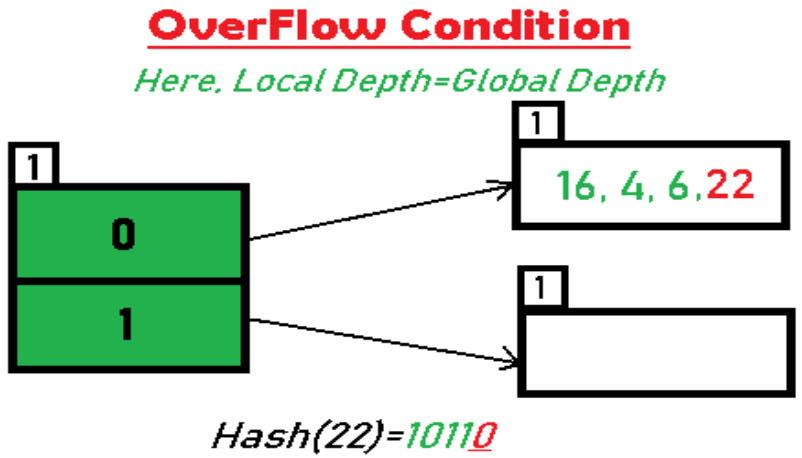
Extendible Hashing

- Inserting 4 and 6:

Both 4(100) and 6(110)have 0 in their LSB. Hence, they are hashed as follows:

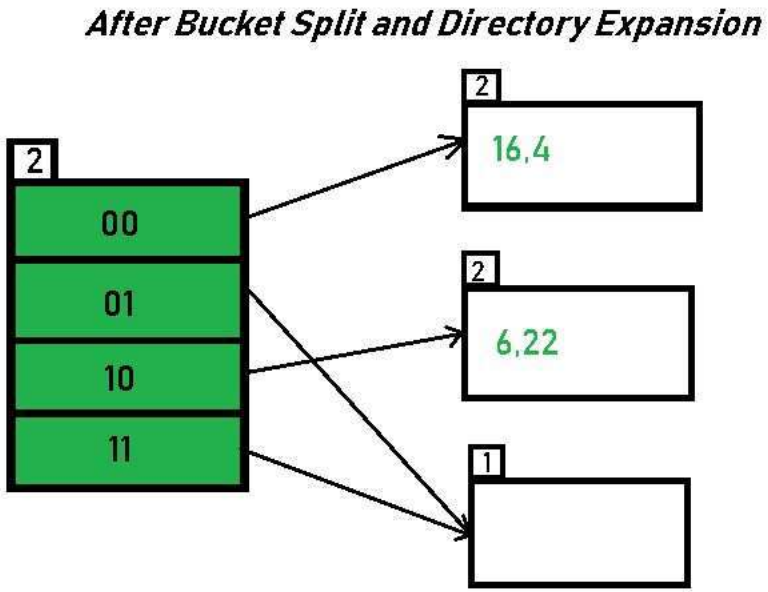


- Inserting 22: The binary form of 22 is 10110. Its LSB is 0. The bucket pointed by directory 0 is already full. Hence, overflow occurs.



Extendible Hashing

Since Local Depth = Global Depth, the bucket splits and directory expansion takes place. Also, rehashing of numbers present in the overflowing bucket takes place after the split. And, since the global depth is incremented by 1, now,the global depth is 2. Hence, 16,4,6,22 are now rehashed w.r.t 2 LSBs.
[16(10000),4(100),6(110),22(10110)]



IMPORTANT FACTS RELATED TO THE SESSION

Advantages of Double hashing

- It can reduce clustering by using a second hash function to determine the next probe location.
- It can handle a larger number of collisions than linear probing.

Disadvantages of Double hashing

- It can be more complex to implement and understand than linear or quadratic probing

Advantages of Rehashing

- Handling collisions: As mentioned above, rehashing is primarily used to handle collisions in hash tables. ...
- Improving performance: Rehashing can also be used to improve the performance of hash tables

Advantages of Extendible Hashing

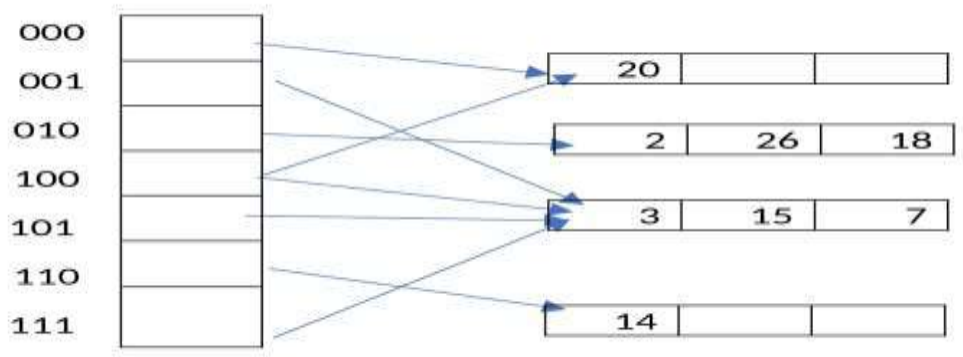
- Data retrieval is less expensive (in terms of computing).
- Very flexible in choosing size of bucket and alllows storage on disks/remote memory access.

Disadvantages of Extendible Hashing

- Increased algorithm complexity
- Extra memory overhead to store index inside the bucket.

EXAMPLES

- Consider an extendible hashing structure that, each bucket can hold up to 2 records. Is initially empty (only one empty bucket). Consider the result after inserting keys 8, 16, 4, 3, 11, 12 in order, using the lowest bits for the hash function. That is records in a bucket of local depth d agree on their rightmost d bits. For example, key 4 (0100) and key 12(1100) agree on their rightmost 3 bits (100). What is the global depth of the resulting directory? Now insert key 18. What is the local depth of the bucket that contains key 18?



IMPORTANT FACTS RELATED TO THE SESSION

Applications of rehashing in real life

- Password Verification
- Data Structures
- File System.

Applications of Extendible Hashing

- All modern file systems use extendible hashing.

Applications of Double Hashing

- To prevent the collision of two keys ,the idea of Double Hashing is used. In case any collision occurs when we just use traditional hash code evaluating function, another hash code is generated by some other function and both the hash codes are used to find a free space by probing through array elements.

SUMMARY

- **Double hashing**:It is a technique in which two hash function are used when there is an occurrence of collision.
- The hash functions for this technique are:

  H1(key)= (key % table size + i * H2(key))% table size

  Whereas H2(key)=P-(key mod P); i=0,1..<size
- **Rehashing** is the process of increasing the size of a hashmap and redistributing the elements to new buckets based on their new hash values. It is done to improve the performance of the hashmap and to prevent collisions caused by a high load factor.
- **Extendible Hashing** is a dynamic hashing method wherein directories, and buckets are used to hash data. It is an aggressively flexible method in which the hash function also experiences dynamic changes.
- Number of Directories = 2^Global Depth.

SELF-ASSESSMENT QUESTIONS

1. What is the hash function used in Double Hashing?

a) (h1(k) – i*h2(k))mod m
b) h1(k) + h2(k)
c) (h1(k) + i*h2(k))mod m
d) (h1(k) + h2(k))mod m

2. Double hashing is one of the best methods available for open addressing

a) True
b) Falseny point

TERMINAL QUESTIONS

1. What is Load factor?

2. When is rehashing done?

3. What is local depth, global depth?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books:

1. Mark Allen Weiss, Data Structures and Algorithm Analysis in C, 2010 , Second Edition, PearsonEducation.
2. 2. Ellis Horowitz, Fundamentals of Data Structures in C: Second Edition, 2015
3. A.V.Aho, J. E. Hopcroft, and J. D. Ullman, “Data Structures And Algorithms”, Pearson Education, First Edition Reprint2003.

Sites and Web links:

1. https://nptel.ac.in/courses/106102064
2. https://in.udacity.com/course/intro-to-algorithms--cs215
3. https://www.coursera.org/learn/data-structures?action=enroll

THANK YOU