ADVANCED DATA STRUCTURE (R16)

Unit 2 (Hashing)

Syllabus: Introduction-Static Hashing- Hash Table- Hash Functions- Secure Hash Function- Overflow Handling- Theoretical Evaluation of Overflow Techniques, Dynamic Hashing- Motivation for Dynamic Hashing -Dynamic Hashing Using Directories- Directory less Dynamic Hashing.

Important Questions

1. Define Hash function? Explain about different hashing functions with example?
2. Differentiate between Static and Dynamic hashing?
3. Define the following
    1. Hash Table & Hash Function
    2. Probing
    3. Collision and Overflow problem while hashing
    4. Dynamic Hashing with directory
    5. Dynamic Hashing with directory-less
4. Define Collision? How to handle collisions while hashing and explain them with examples?
5. Differentiate between closed hashing and open hashing?
6. Explain the characteristics of good hashing function?
7. Give importance of secure hash function with neat sketch?
8. Discuss theoretical Evaluation of Overflow Techniques?
9. Explain about dynamic hashing using directories with examples?
10. Explain about Directory less dynamic hashing with examples

Previous Questions

1. What is a hash table? What is hash function? What is bucket and home bucket?
2. Explain with example, explain folding and rotation hashing methods.
3. When collision will occur? Explain quadratic probing with example.
4. Explain the characteristics of good hashing function?
5. Consider the given values 72, 27, 36, 24, 63, 81, 92, 101 and perform linear, quadratic-Probing operations in a given hash table of size 10.
6. With example, explain modulo division and digit extraction hashing methods.
7. What do you mean by collision and how can you handle it by using linear probing.
8. What do you mean by a hash table and a hash function. Explain the following hash functions with an example (i). Division method (ii). Mid square (iii). Digit analysis

B. Dinesh Reddy ***Associate Professor***

Ch. Sudhakar ***Assistant Professor***

9. What is collision? Explain different collision resolution methods
10. Explain how open hashing and closed hashing is done with examples.
11. Differentiate between double Hashing and Rehashing
12. Discuss the Problems associated with Quadratic probing.
13. How will you handle overflow and collision detection in a hash table? Discuss methods.
14. Construct the open hash table and closed hash table for the input:
    30, 20, 56, 75, 31, 19 using the hash function h(k)= k mod 11
15. Discuss the Advantages of hashing?
16. Explain the linear probing method in hashing. Discuss its performance analysis.
17. What is hashing with chains? Explain. Compare this with linear probing
18. Compare Closed hashing Vs Open hashing
19. With a procedure and a relevant example discuss separate chaining in hashing.
20. Explain separate chaining and open addressing in detail?
21. Explain how open hashing and closed hashing is done with examples.
22. What is collision? Explain different collision resolution methods.
23. What are Hash functions? List some techniques that are used to implement Hash functions?
24. What is rehashing?
25. The Keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function linear probing. What is the resultant hash table?
26. What do you mean by hashing? Why do we need it?
27. List the methods of the Hashing function & what are Heap Properties?
28. Explain about hash table restructuring with examples.
29. Explain about the analysis of closed hashing for successful search and deletion
30. Explain the different collision resolution strategies for hashing. State the advantages and disadvantages of each techniques.
31. Following elements are inserted into an empty hash table with hash function f(x) = x% 17 and quadratic probing 20,10,5,30,40,57,35,25,18,22,21.
32. What is Hash function and Hash table? Explain
33. Explain Rotation method in a combination with Folding and pseudorandom hashing method with example.
34. Discuss how collision can be resolved using quadratic probing while inserting following keys in Hash table of size 10. 97, 40, 15, 22, 17, 89, 67

Perfect Hash Function?

Answer:

There are four main characteristics of a good hash function:

1) Perfect hash table that has **no collisions**.
2) The hash function uses all the input data.
3) The hash function "uniformly" distributes the data across the entire set of possible hash values.
4) The hash function generates very different hash values for similar strings.

Discuss theoretical Evaluation of Overflow Techniques?

Answer:

The experimental evaluation of hashing techniques indicates a very good performance over conventional techniques such as balanced trees. However, the worst-case performance of hashing can be O(n).

Let ht [0 .. b – I] be a hash table with b buckets. each bucket having one slot. Let h be a uniform hash function with range [0. b – 1]. If n identifiers (keys) X1, X2 …. Xn are entered into the hash table. Then there are 'b' distinct hash sequences h (x1), h (X2)….. h(xn). Let Sn denote the expected number of identifier comparisons needed to locate a randomly chosen Xi. 1 S is n. Then Sn is the average number of comparisons needed to find the jth key Xj. Let U be the expected number of identifier comparisons when a search is made for an identifier not in the hash table.

Theoretical Evaluation of Overflow Techniques

- Theorem 8.1

  Let α=n/b be the loading density of a hash table using a uniform hashing function h. Then

- for linear open addressing

$$U_n \approx \frac{1}{2}[1+\frac{1}{(1-\alpha)^2}] \qquad S_n \approx \frac{1}{2}[1+\frac{1}{1-\alpha}]$$

- for rehashing, random probing, and quadratic probing

$$U_n \approx 1/(1-\alpha) \qquad S_n \approx -[\frac{1}{\alpha}]\log_e(1-\alpha)$$

- for chaining

$$U_n \approx \alpha \qquad S_n \approx 1+\alpha/2$$

Explain about Dynamic Hashing with examples?

Dynamic Hashing

- The purpose of dynamic hashing (extendible hashing) is to retain the fast retrieval time of conventional hashing while extending the technique so that it can accommodate dynamically increasing and decreasing file size without penalty.
- Assume a file F is a collection of records R. Each record has a key field K by which it is identified. Records are stored in pages or buckets whose capacity is p.
  - The goal of dynamic hashing is to minimize access to pages.
  - The measure of space utilization is the ratio of the number of records, n, divided by the total space, mp, where m is the number of pages.

Dynamic Hashing

- Three Kinds of the method
  - Trie Tree
  - Directory
  - Directoryless

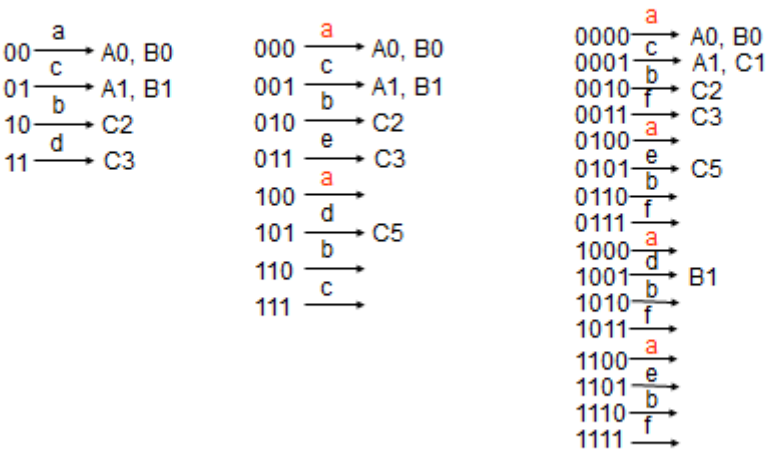
Hashing With Directory

- Fagin et al. present a method, called *extensible hashing*, for solving the above issues.
  - A hash function is used to avoid skewed distribution. The function takes the key and produces a random set of binary digits.
  - To avoid long search down the trie, the trie is mapped to a directory
    - directory is a table of pointers.
    - If k bits are needed to distinguish the identifiers, the directory has 2^k entries indexed 0, 1, ..., 2^k-1
    - Each entry contains a pointer to a page.

Hashing With Directory

- Using a directory to represent a trie allows table of identifiers to grow and shrink dynamically.
- Accessing any page only requires two steps:
  - First step: use the hash function to find the address of the directory entry.
  - Second step: retrieve the page associated with the address
- Since the keys are not uniformly divided among the pages, the directory can grow quite large.
- To avoid the above issue, translate the bits into a random sequence using a uniform hash function. So identifiers can be distributed uniformly across the entries of directory. In this case, multiple hash functions are needed.

Trie Collapsed Into Directories



(a) 2 bits (b) 3 bits (C) 4 bits

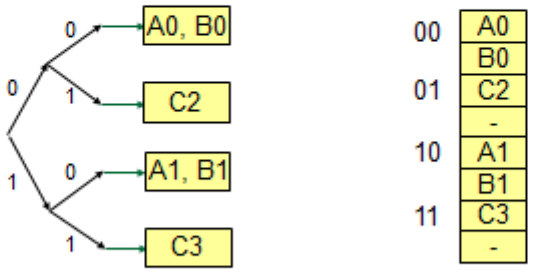
Analysis of Directory-Based Dynamic Hashing

- The most important of the directory version of extensible hashing is that it guarantees only two disk accesses for any page retrieval.
- We get the performance at the expense of space. This is because a lot of pointers could point to the same page.
- One of the criteria to evaluate a hashing function is the space utilization.
- Space utilization is defined as the ratio of the number of records stored in the table divided by the total number of space allocated.
- Research has shown that dynamic hashing has 69% of space utilization if no special overflow mechanisms are used.

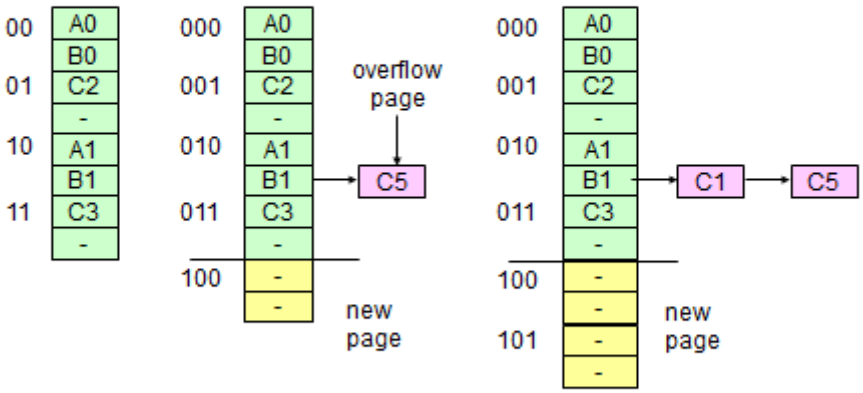
Directoryless

- Directoryless hashing (or linear hashing) assume a continuous address space in the memory to hold all the records. Therefore, the directory is not needed.
- Thus, the hash function must produce the actual address of a page containing the key.
- Contrasting to the directory scheme in which a single page might be pointed at by several directory entries, in the directoryless scheme one unique page is assigned to every possible address

Directoryless



Directoryless Scheme Overflow Handling



Analysis of Directoryless Hashing

- The advantage of this scheme is that for many retrievals the time is one access for those identifiers that are in the page directly addressed by the hash function.
- The problem is that for others, substantially more than two accesses might be required as one moves along the overflow chain.
- Also when a new page is added and the identifiers split across the two pages, all identifiers including the overflows are rehashed.
- Hence, the space utilization is not good, about 60%. (shown by Litwin).

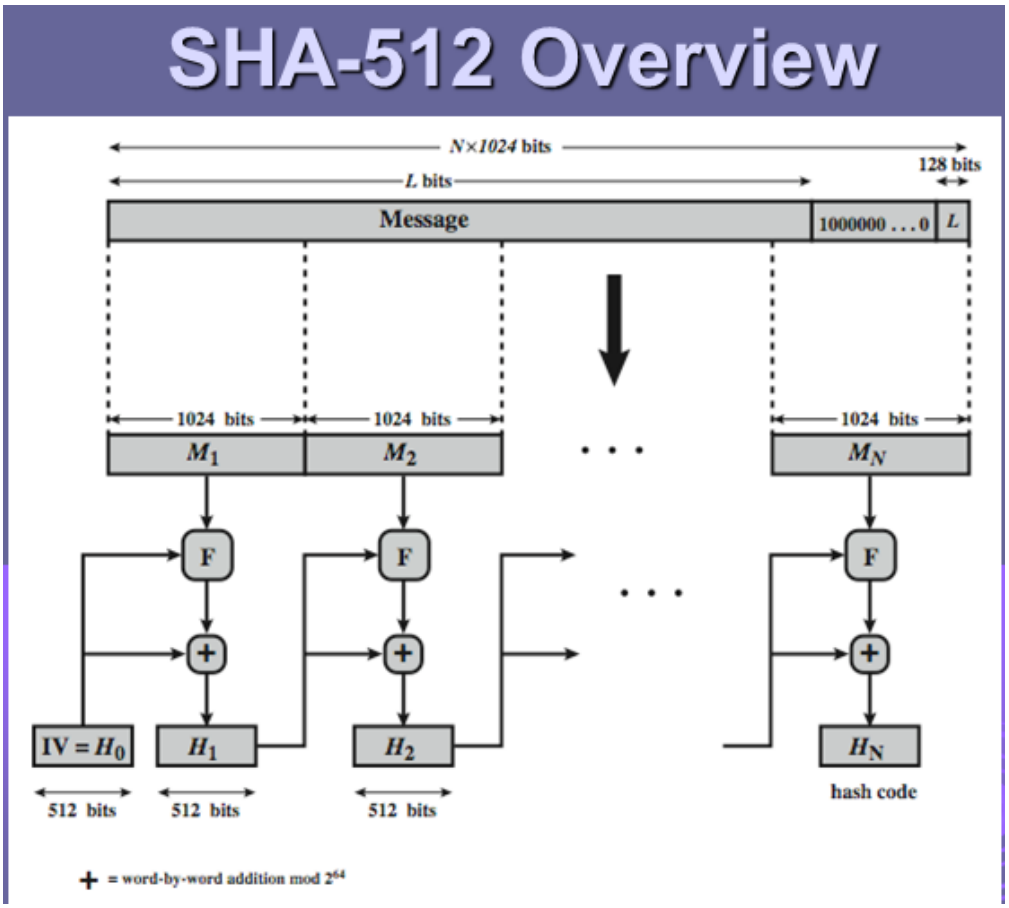
Give importance of secure hash function with neat sketch?

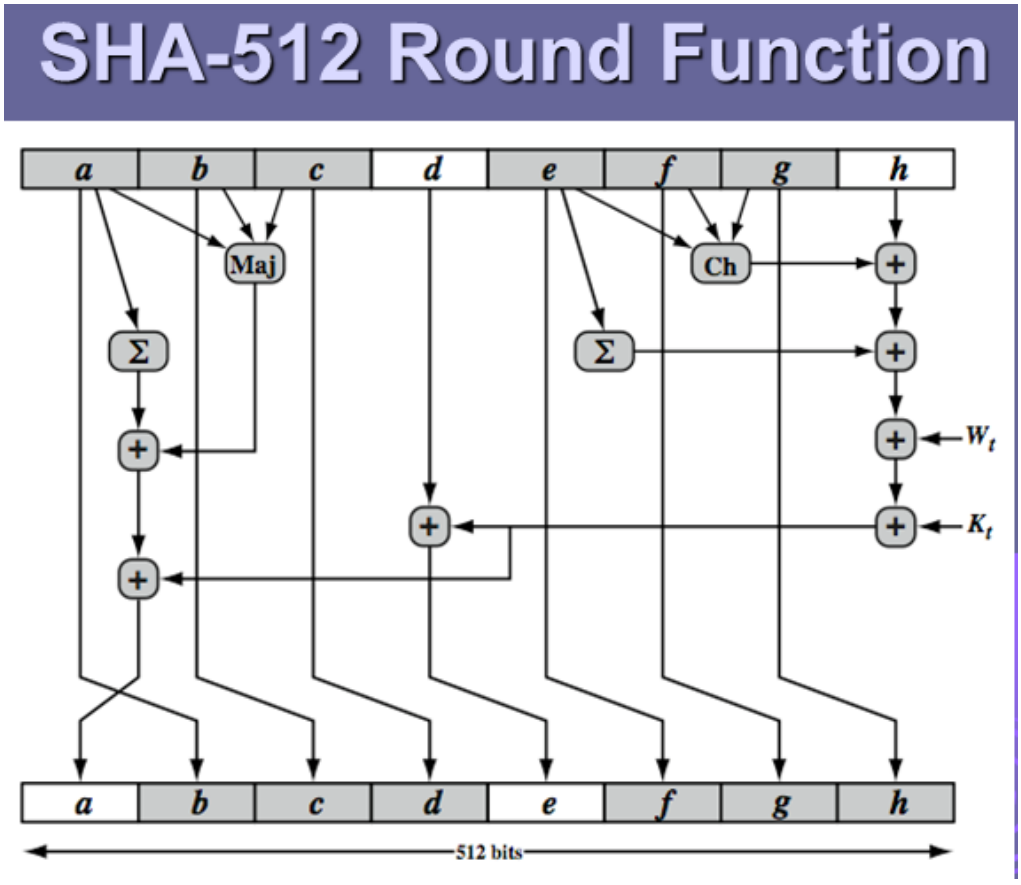
Answer:

In cryptography, **SHA-1** is a cryptographic hash function designed by the National Security Agency (NSA) and published by the NIST as a U.S. Federal Information Processing Standard. SHA stands for **Secure Hash Algorithm**. The three SHA algorithms are structured differently and are distinguished as *SHA-0*, *SHA-1*, and *SHA-2*. SHA-1 is very similar to SHA-0, but corrects an error in the original SHA hash specification that led to significant weaknesses. The SHA-0 algorithm was not adopted by many applications. SHA-2 on the other hand significantly differs from the SHA-1 hash function.

Secure Hash Algorithm (SHA)

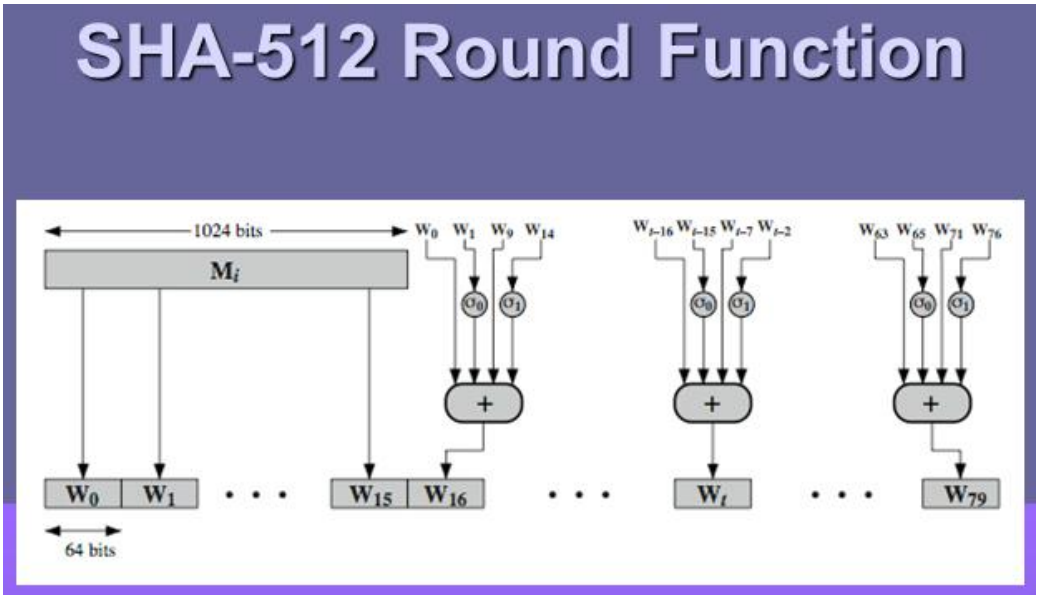
- Successor to and similar to MD5 (by Ron Rivest)
- SHA-0: FIPS PUB 180, 1993. Withdrawn shortly after publ.
- SHA-1: FIPS PUB 180-1, 1995. 160 bit hash
- SHA-2: FIPS PUB 180-2, 2002
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- SHA-1 is used in TLS, SSL, PGP, SSH, S/MIME, and IPsec
  - Required by law in US Govt applications
  - Used in Digital Signature Standard
- Pseudo-codes for SHA algorithms are available.
- NIST certifies implementations.





B. Dinesh Reddy *Associate Professor*

Ch. Sudhakar *Assistant Professor*



Applications

SHA-1 forms part of several widely-used security applications and protocols, including TLS and SSL, PGP, SSH, S/MIME, and IPsec. Those applications can also use MD5; both MD5 and SHA-1 are descended from MD4. SHA-1 hashing is also used in distributed revision control systems such as Git, Mercurial, and Monotone to identify revisions, and to detect data corruption or tampering. The algorithm has also been used on Nintendo console Wii for signature verification during boot but a significant implementation flaw allowed an attacker to bypass the security scheme.[6]

Differentiate between static and dynamic hashing?

	STATIC HASHING	DYNAMIC HASHING
1	Numbers of buckets are fixed.	Numbers of Buckets are not fixed.
2	As the file grows, performance decreases.	Performance doesn't degrade as the file grows.
3	Space overhead is more.	Minimum space lies overhead.
4	Here we do not use Bucket Address table.	Bucket address table is used.
5	Open hashing and Closed hashing are forms of it.	Extendable hashing and Linear hashing are forms of it.
6	No complex implementation.	Implementation is complex.
7	It is a less attractive technique.	It is a highly attractive technique.
8	Here system directly accesses the Bucket.	Here the Bucket address table is accessed before accessing the Bucket.
9	Chaining used is overflow chaining.	Overflow chaining is not used.
	Block Diagram	Block Diagram

Differentiate between open and closed hashing?

	Open Hashing	Closed Hashing
1	No additional data structure except hash table is used for storing the elements of hash table.	To avoid collision of data, additional data structure is used.
2	Unique key must be obtained.	Unique key may or may not be obtained.
3	Use space efficiently.	Space overhead.
4	Determining size of the hash table, adequate enough for storing all the data is difficult.	Performance deterioration of closed addressing much slower as compared to Open addressing.
5	State need to be maintained for additional data.	No space need to be maintained
6	Block Diagram	Block Diagram