

UNIT 3: HASH FUNCTION

A hash function is any well defined procedure or mathematical function that converts a large, possibly variably sized amount of data into small datum usually a single integer that may serve as an index to an array. The value returns by hash function are called Hash value, Hash Codes, Hash sums or simply Hashes.

The function $H = K \rightarrow L$ is called a hash function.

The two principal criteria used for selecting a hash functions $H = K \rightarrow L$ are as follows:

- (1) The hash function H should be very easy and quick to compute.
- (2) The function H should as far as possible, uniformly distribute the hash addresses throughout the set L so that there are minimum number of collision.

Hash Function Techniques

1. Division method
2. Mid-square method
3. Folding method

1. Division Method

Choose a number m larger than the number n of keys in K (the number m is usually chosen to be a prime number or a number without small division, since these frequently minimizes the number of collision). Then the hash function H is denoted by;

$$H(k) = k \pmod{m} \text{ or } H(k) = k \pmod{m} + 1$$

The first formula $k \pmod{m}$ denotes the remainder when k is divided by m while the second formular is used when we want the hash address to range from 1 to m rather than from 0 to $m-1$.

Example:

Suppose a company with 68 employees assign a 4 - digit employee number to each employee which is used as the primary key in the company's employee file. Suppose L consist of 100 two-digit addresses 00, 01, 02, ..., 99. Applying the hash function to each of the following employee numbers: 3205, 7148, 2345.

Solution:

Using division method, choose a prime number in which is close to 99 such as $m = 97$. Then

$$H(k) = k \pmod{m}$$

a) $H(3205) = 3205 \pmod{97} = 3205/97 = 4 \quad H(3205) = 4$

That is, dividing 3205 by 97 gives a remainder of 4.

b) $H(k) = k \pmod{m}$

$$H(7148) = 7148 \pmod{97} = 7148/97 = 67 \quad H(7148) = 64$$

That is, dividing 7148 by 97 gives a remainder of 64.

c) $H(k) = k \pmod{m}$

$$H(2345) = 2345 \pmod{97} = 2345/97 = 17 \quad H(2345) = 17$$

That is, dividing 2345 by 97 gives a remainder of 17.

2. Midsquare

The key k is square. Then the hash function H is defined by

$$H(k) = l$$

where l is obtained by deleting digits from both ends of k^2 . Note that the same position of k^2 must be used for all of the keys.

Example

Using the above equation

Solution

The following calculations are performed:

| | | | |
|-------|----------|----------|----------|
| k | 3205 | 7148 | 2345 |
| K^2 | 10272025 | 51093904 | 05499025 |
| H(k) | 72 | 93 | 99 |

Observe that the fourth and fifth digits, counting from the right, are chosen for the hash address .

3. Folding Method

The key k is partitioned into a number of parts k_1, \dots, k_r , where each part, except possibly the last, has the same number of digits as the required address. Then the parts are added together, ignoring the last carry. That is,

$$H(k) = k_1 + k_2 + k_3 + \dots + k_r$$

where the leading-digit carries, if any, are ignored. Sometimes, for extra “milling”, the even-numbered parts k_2, k_4, \dots , are each reversed before the addition.

Example:

Chopping the key k into two parts and adding yields of the following hash addresses:

$$H(3205) = 32 + 05 = 37$$

$$H(7148) = 71 + 48 = 119 = 19$$

$$H(2345) = 23 + 45 = 68$$

Observe that the the leading digit **1** of $H(7148)$ is ignored. Alternatively, one may want to reverse the second parts before adding, this producing the following hash addresses:

$$H(3205) = 32 + 50 = 82$$

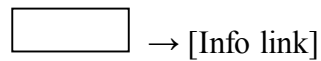
$$H(7148) = 71 + 84 = 155 = 55$$

$$H(2345) = 23 + 54 = 77$$

UNIT 4: LINKED LIST

Basic Concepts

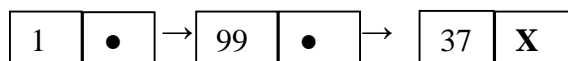
This is a data structure that consist of a sequence of data record such that in each record there is a field that contain a reference to the next field



A node is made up of two parts which are the data field and link-list.

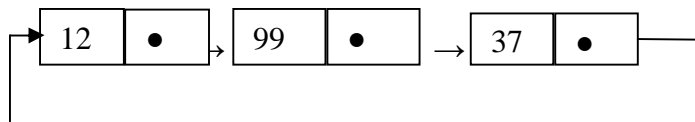
Each record of a link-list is called a NODE which is made up of two parts the information part and the pointer part.

Linear List



In linear linked list, the components are all linked together in some sequential manner.

Circular List



In circular linked list, the component has no beginning and end.

Singly, doubly and multiply linked list are example of a linked list:

Singly-linked list contain nodes which have a data field as well as a next field, which points to the next node in the linked list.

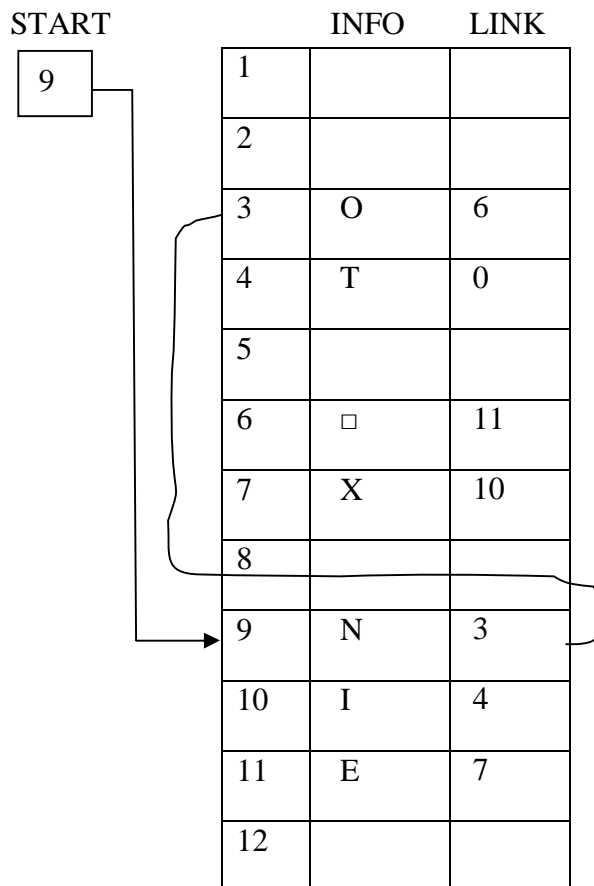
In a doubly-linked list, each node contains, besides the next-node link, a second link filed pointing to the previous node in the sequence. The two links may be called forwars(s) and backwards.

A Multiply Linked List

Representation of Link List in Memory

Let LIST be a linked list. LIST require two linear arrays called INFO and LINK, such that INFO [K] and LINK [K] contain, respectively, the information part and the next pointer field of a node of LIST. It should be noted that, LIST requires a variable name such as START which indicate the beginning of the list and a next-pointer sentinel – denoted by NULL which indicate the end of the list.

Example



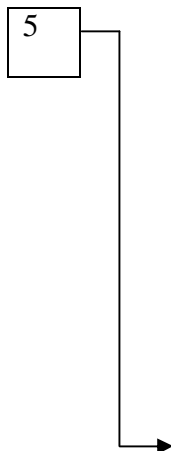
Interpreted as:

START = 9, so INFO [9] = N (is the first character)
LINK [9] = 3, so INFO [3] = O (is the second character)
LINK [3] = 6, so INFO [6] = □ (blank) is the third character
LINK [6] = 11, so INFO [11] = E is the fourth character
LINK [11] = 7, so INFO [7] = X is the fifth character
LINK [7] = 10, so INFO [10] = I is the sixth character
LINK [10] = 4, so INFO [4] = T is the seventh character
LINK [4] = 0 INFO [0] = NULL value, so the list has ended

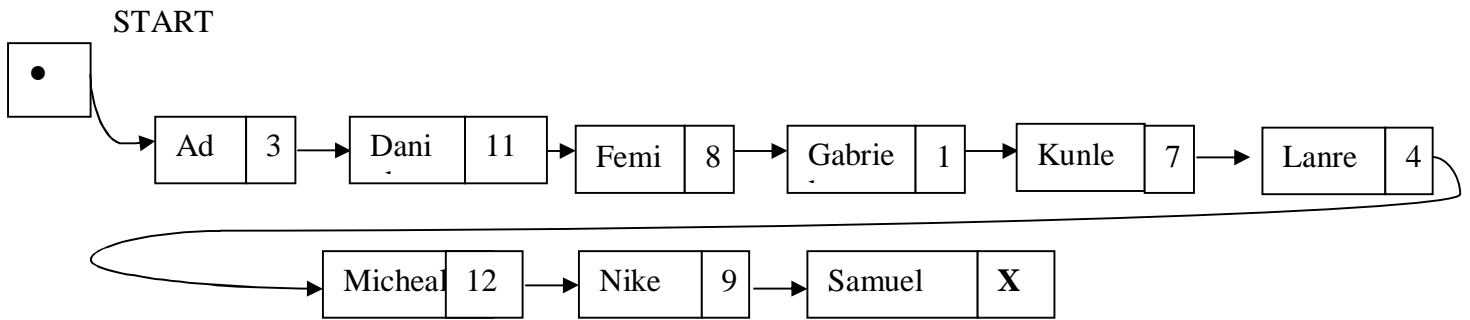
In other words, NO EXIT is the character string

Example:

A hospital ward contains 12 beds of which 9 are occupied. The listing is given by pointer field START

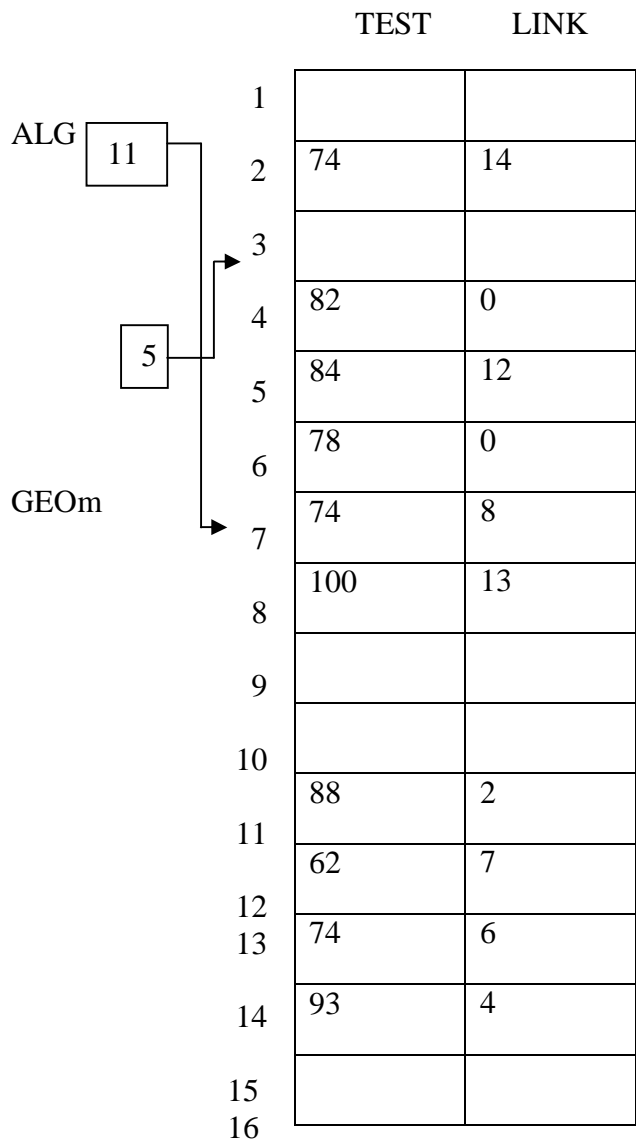


| Bed Number | Patient | |
|------------|---------|----|
| 1 | Kunle | 7 |
| 2 | | |
| 3 | Daniel | 11 |
| 4 | Micheal | 12 |
| 5 | Ade | 3 |
| 6 | | |
| 7 | Lanre | 4 |
| 8 | Gabriel | 1 |
| 9 | Samuel | 0 |
| 10 | | |
| 11 | Femi | 8 |
| 12 | Nike | 9 |



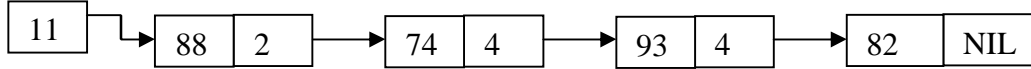
Example 2:

This figure shows the test score in Algebra & geometry stored in the same linked list.



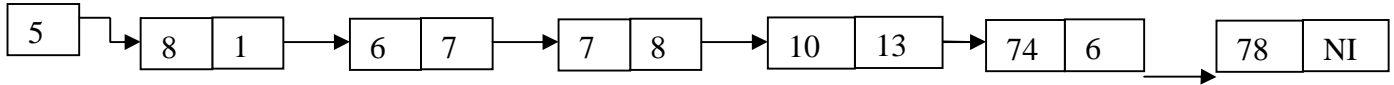
FOR ALGEBRA (ALG)

ALG



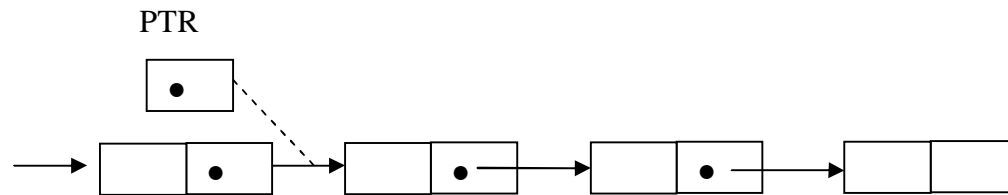
The information for ALG is 88, 74, 93, 82

FOR GEOM



The information for geom. is 84, 62, 74, 100, 74, and 78

Transversing a Link List



$PTR = LINK [PTR]$

Algorithm to access each element exactly once in the list

- (1) Set $PTR = START$ [initiate pointer PTR]
- (2) Repeat step 3 and 4 while $PTR = NULL$
- (3) Apply process to $INFO [PTR]$
- (4) Set $PTR := LINK [PTR]$ [PTR now points to the next node] [End of step 2 loop]
- (5) Exit

Searching

Algorithm 2:

List is a linked list in memory. This algorithm finds the location LOC of node where ITEM first appear in LIST or sets $LOC = NULL$

- (1) Set $PTR := START$
- (2) Repeat step 3 while $PTR \neq NULL$
- (3) If $ITEM = INFO [PTR]$, then

Set $LOC := PTR$ and EXIT