ASSIGNMENT 9

Aim: Store data of students with telephone number and name in the structure using hashing function for telephone number and implement chaining with and without replacement

Theory:

The organisation of the file and the order in which the keys are inserted affect the number of keys that must be searched before obtaining the desired one. Optimally we would like to have a table organisation and search technique in which there are no ~~unce~~ unnecessary comparisons. Hash table are good for doing a good quick search on things. The most efficient way to organise such a table is an array. If the records keys are integers, the key themselves can serve as indices to the array.

$$\text{Key} \xrightarrow{\text{Hash function}} \text{Index to Array}$$

A function that transforms a key into a table index is called hash function. If h is a hash function and key is key, h(key) is called the hash of key and is the index at which a record with key key should be placed.

I] Characteristics of a good hash function :

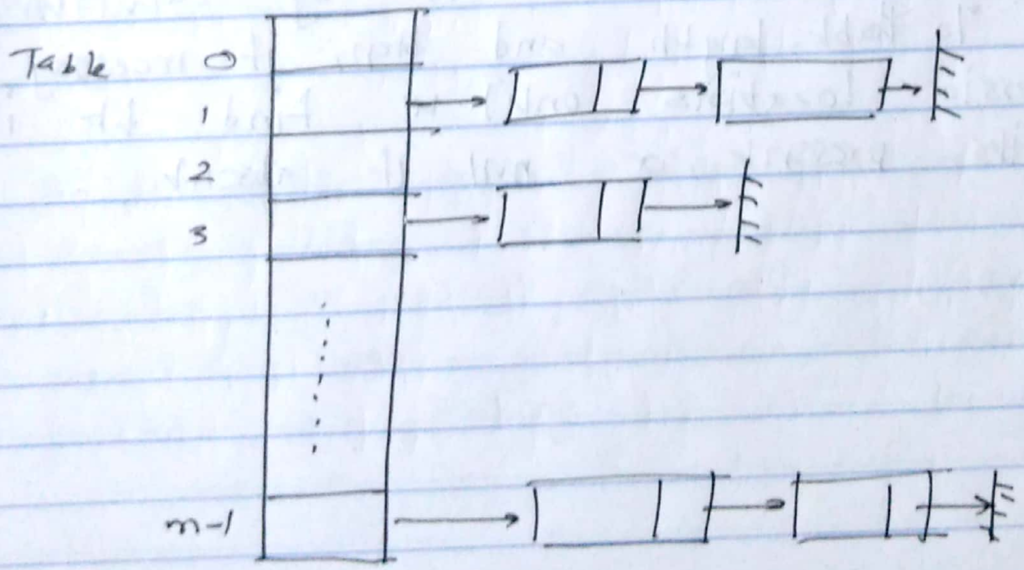
- The hash value is fully determined by the data being hashed
- The hash function uses all input data
- The hash function "uniformly" distributes the data across the entire set of possible hash values.
- The hash function generates very different hash values for similar strings.

II] - Collision : A clash or collision is a situation that occurs when two distinct pieces of data have the same hash value. Collisions are unavoidable whenever members of a very large set are maped to a relatively short bit string. The problem of collision can not be eliminated but can be minimised.

- Overflow : When a hash table becomes full then the method suggested here is to define a new table of length greater than old hash table length and then to serially re-hash into the new table of elements already defined in the original table. subsequent entries and acessing will be make on new table.

• Chaining: One simple scheme is to chain all collisions in lists attached to the appropriate slot. This allows an unlimited number of collisions to be handled and doesn't require a priori knowledge of how many elements are contained in the collection. The tradeoff is the same as with linked lists versus away implementations of collections: linked list overhead in space and, to a lesser extent, in time.

Hashing with chaining is an application of linked lists and gien an approach to collision resolution. In hashing with chaining, the hash table contains linked lists of elements or pointers to elements. The lists are referred to as chains, and technique is called chaining. This is a common technique where a straightforward implementation is desired and maximum efficiency isn't requied. Each linked list contains all elements whose key hashes to the same index.



· Linear probing : It is a scheme in computer programming for resolving hash collisions of values of hash functions by sequentially searching the hash table for a free location. In that case, since we cannot insert n at h(n), which we now call h for simplicity, we try the next slot h+1 If it is ~~vacant~~ vacant, we insert n here. Otherwise, we try h+2 and so on. The sequence of locations that we probe, as far as necessary, is therefore h, h+1, h+2, h+3, . . .

The only qualification is that we must only probe within the table array. When this sequence would run off the end of the array, we wrap around back to the beginning at 0. The sequence of probes is therefore at
h+i % table.length.
for i=0, 1, 2 . . . n

~~When~~
We simply search for it by calculating h(d) % table.length and then, if necessary, probe successive locations until we find the item d, if its present, or null if absent.

· Different hash function to ~~[illegible]~~ organise a hash table:

(i) Division method: The simplest of all methods of hashing an integer x is to divide x by M and then to use the remainder modulo M. This is called the division method of hashing.
Hash function is
$$h(x) = x \bmod M$$

Generally, this approach is quite good for just about any value of M. In this case, M is a constant, However, an advantage of the division method is that M need not be a compile-time constant -- its value can be determined at run time.
A disadvantage of this method is due to the property that consecutive keys map to consecutive hash values. While this ensures that consecutive keys do not collide, it does mean that consecutive array locations will be occupied. It may lead to degradation in performance.

(ii) Middle square method: Since integer division is usually slower than multiplication, by avoiding division we can potentially improve the running time of hashing algorithm. We can avoid division by making use of the fact that a computer does finite-precision integer arithmetic.

Scanned with CamScanner

The middle - square hasing method works as follows. First we assure that M is a power of 2, say $M = 2^k$ for some $k \geq 1$. Then to hash an integer x, we use the following hash function:

$$h(x) = \left\lfloor \frac{M}{W} (x^2 \bmod W) \right\rfloor$$

(iii) Multiplication Method: A very simple variation on the middle square method that alleviates its deficiencies is the so called multiplication hashing method. Instead of multiplying the key x by itself, we multiply the key by a carefully chosen constant a, and then extract the middle k bits from the result. In this case, the hashing function is

$$h(x) = \left\lfloor \frac{M}{W} (ax \bmod W) \right\rfloor$$

what is suitable choice for constant a? If we want to avoid problems that the middle - square method encounters with keys having a large number of leading or trailing zeroes, then we should choose an a that has neither leading nor trailing zeroes. Furthermore if we choose an a that is relatively prime to W, then ∃ another a' number such that $aa' = 1 \pmod W$.

(ii) Folding method: In the folding method, the key is divided into two parts that are then combined or folded together to create an index into the table. This done by first dividing the key into 2 parts of the key will be the same length as the desired key.
In the shift folding method, these parts are then added together to create the index
eg. No:- 987654321 we divide into 3 parts
987 654 321 and then add these to get 1962. Then use either division or extraction to get 3 digit index.

Algorithm: Create hash table and define hash function.

```
~~ste~~
(i) location = hash (key);
(ii) if hashtable (location [0]) = keyvalue;
(iii) else if hash( hashtable ([location [0]) == hash(keyvalue).
              i = ~~key~~ location;
      do step A while hashtable [i] [1] = != -1
           A : i ← hashtable [i] [1];
      ~~Hash~~ HashTable [i] [1] = position;
      HashTable [position] [0] = keyvalue;
end if
else i = location++;
      do step A, B while (i % 10) != location
          if i == 10 then i = 0;
           a : if hash(HashTable [i] [0]) == hash(key)
```

```
            Do   step   A   while   HashTable [i][1] != -1
                 A :   i = HashTable [i][1];
            end   if
                 b :   HashTable [i][1] <- position;
                 c :   break;
          B :   Increment   i;
        HashTable [position] [0] <- key Value;
      end   else;
   end   else;
(iv)   Stop
```

Scanned with CamScanner

Conclusion: Implementation of hashing and using a hashing for telephone number and chaining with and without replacement was done successfully.