Hashing / Linear & Binary Search

Agenda

- Hashing or Hash Addressing
- Hash Function
- Collision Resolution

Hashing

- In searching algorithm the search time depends on the number of elements in the collection of data.
- Hashing or hash addressing is a searching technique which is independent of the number of elements in the collection of data.
- ‘F’ is a file of ‘n’ records with a set ‘K’ of keys which uniquely determine the records in file.
- ‘F’ is maintained in memory by a table ‘T’ of ‘m’ memory location and that ‘L’ is the set of memory address of the location in the table.

Cont…

- Function ‘H’ from the set ‘K’ of keys into the set ‘L’ of memory address is called a hash function or hashing function.

  $H: K \rightarrow L$

- Unfortunately such a function wont yield distinct values, it is possible that two different keys k1 & k2 will yield the same hash address, this situation is called **Collision** and methods to be used to resolve this.
- Thus we have two things to discuss **Hashing function** & **Collision resolutions**

Hash Function

- Two principal criteria are used in selection a hash function.
  - The function(H) should be very easy and quick on compute.
  - The function(H) should as far as possible uniformly distribute the hash addresses throughout the set(L) of memory addresses so that there are a minimum number of collisions.
- There is no guarantee that the Second condition can be completely fulfilled without actually knowing beforehand the keys and addresses.

Cont…

- Certain general techniques do help in achieving the principal.
  - One technique is to chop a key ‘K’ into pieces and combine the pieces in some way to form the hash address H(k), the term hashing comes from this technique of chopping a key into pieces.
- Lets look at come popular hash functions .
  - **Division method :** Choose a number m large than the number n of keys in K. The number m is usually chosen to be a prime number or a number without small divisors, since this frequently minimizes the number of collisions. Hash function is defined by

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

Cont…

- **Midsquare method :** The key k is squared. Where l is obtained by deleting the digits from both ends of k^2

  H(k)= L

- **Folding method :** The key k is partitioned into a number of parts, k1,…kn , where each part, expect possibly the last, has the same number of digits as the required address. Then the parts are added together, ignoring the last carry.

  H(k)= k1+k2+…kn

Collision Resolution

- Suppose we want to add a new record R with key k to our file F, but if the memory location address H(k) is already occupied. This situation is called collision.
- **Load factor(λ) :** The ratio of the number of keys (n of k) to the number of hash addresses(m in L).

$$\lambda = n/m$$

- The efficiency of a hash function with a collision resolution procedure is measured by the average number of probes needed to find the location of the record with a given key k. Efficiency depends on the load factor. S(λ) & U(λ) are two important factors.

Open Addressing : Linear probing & Modifications

- Suppose we want to add a new record R with key k to our file F, but if the memory location address H(k)=h is already occupied. This situation is called collision.
- One natural way to resolve the collision is to assign R to the first available location following T[h].
- Accordingly , with such a collision procedure, we will search for the record R in the table T by linearly searching the locations T[h],T[h+1],…. Until finding R or meeting an empty location , which indicates an unsuccessful search. This method of collision resolution is called **linear probing.**

Cont…

- One major disadvantage of linear probing is that records tend to cluster, that is, appear next to one another, when the load factor is greater than 50%.
- Such a clustering substantially increases the average search time for a record. We have two techniques to minimize clustering.
  - **Quadratic probing :** Instead of searching the locations with address h, h+1… , we linearly search the address with h, h+4, h+9….h+i^2
  - **Double hashing :** Here a second hash function is used for resolving a collision . Then we linearly search the locations with addresses h, h+h'. H+2h' …

Chaining

- Chaining involves maintaining two tables in memory.
- First table T as before except that T now has an additional field LINK which is used so that all the records in T with the same hash address h may be linked together to form a linked list.
- Second , there is a hash address table LIST which contains pointers to the linked list in T.
- Suppose a new record R with key k is added to the file F. We place R in the first available location in the table T and then add R to the linked list with pointer LIST[H(k)]. This would be similar to LL.

Linear Search

- Traverses the array sequentially to locate the item.
- Algorithm LINEAR(DATA,N,ITEM,LOC)

  DATA is a linear array with N elements and ITEM is a given item of information. This algorithm finds the location LOC of ITEM in DATA or set s LOC:=0 if the search is unsuccessful

  – Algorithm LINEAR(DATA,N,ITEM,LOC)

    1. [Insert ITEM at the end of DATA] Set DATA[N+1]:=ITEM
    2. [Initialize counter] Set LOC:=1
    3. [Search for ITEM]
    4. Repeat while DATA[LOC] != ITEM,
    5. Set LOC:=LOC+1
    6. [Successful?] If LOC=N+1, then set LOC:=0
    7. Exit

Binary Search

- Searching algorithm used when the elements of the array is sorted.
- Algorithm BINARY(DATA,LB,UB,ITEM,LOC)

  DATA is a sorted array with lower bound LB and upper bound UB, and ITEM is a given item of information. The variables BEG, END and MID denote, respectively, the beginning , end and middle location of a segment of elements of DATA. This algorithm finds the location LOC of ITEM in DATA or set s LOC=NULL if the search is unsuccessful

Cont…

- Algorithm BINARY(DATA,LB,UB,ITEM,LOC)
  1. [Initialize Segment variables] Set FIRST=LB, LAST=UB & MID=INT(FIRST+LAST/2)
  2. Repeat step 3 & 4 while BEG<=LAST and DATA[MID] != ITEM
  3. If ITEM<DATA[MID], then
     Set LAST:=MID-1
     Else Set FIRST:=MID+1
  4. Set MID:=INT(FIRST+LAST/2)
  5. If DATA[MID]=ITEM, then
     Set LOC:=MID
     Else Set LOC:=NULL
  6. Exit

Thank You