Hashing

Method of searching in which searching is independent of no. of elements

Hash Table:

It's an array addressed via Hash function.

Hash Function:

Converts the key into corr. location in Hash Table

~~Two~~ Two types of Hash function:

1) Distribution-dependent

2) Distribution-independent

⇒ Distr. dependent:
Obtained by examining subset of key values corr. to desired ~~results~~ records.

⇒ Distr. independent:
Does not utilize distr. of keys in Hash Table to compute location of desired record.

Most pop. distribution-independent Functions:

1) Division method 2) Mid-square

3) Folding 4) Digit analysis

5) Length dependent method

Main Idea:

→ Find 1-1 correspondence betn key value & index

Criteria to decide Hash funcn:

→ Easy & quick to compare

→ Achieve even distr. of keys, that occur across range of indices

1) Division Method:

→ Convert key to integer

→ mod size of index range

→ Remainder as result

Hash funcn:

$H(K) = K \% i$ if index start from 0

$H(K) = (K \% i) + 1$ index start from 1

E.g. Size of Hash table = 97
key = 2763

$H(2763) = 2763 \% 97$
$= 48$ index starts from 0
$= 49$ " — " — 1

2) Mid-Square method

$H(k) = x$, where x is obtained by selecting appr. no. of bits from middle of square of key-value K.

E.g. Select 3 digits at even pos. starting from rightmost digit in the sq.

K:	1234	2345	3456
K^2:	1522756	5499025	11943936
$H(k)$:	525	492	933

Drawback: Time consuming computation
Advantage: Uniform Distr. of Keys

3) Folding method:

→ Partition the key into no. of parts $K_1, K_2, K_3 \cdots K_n$, where each part, except possibly the last, has possibly same no. of bits as req^d addr. width.
→ The parts are added together ignoring the last carry

$\boxed{H(k) = k_1 + k_2 + k_3 - - - - k_n}$

Last carry if any, is ignored

If keys are in binary form, EX-OR operation may be subs. for addⁿ.

Fold shifting → Even parts are $k_2, k_4 \ldots$ are each reversed before addⁿ.

Fold boundary → Two boundary parts k_1 & k_n each reversed & added with all other parts.

E.g.

K:	1522756	5499025	11943936
~~Chopping~~:			
Chopping:	01 + 52 + 27 + 56	05 + 49 + 90 + 25	11 + 94 + 39 + 36
Pure folding	: 136	169	= 180
Fold shifting	: 01+25+27+65 = 118	05+94+90+52 = 241	11+49+39+63 = 162
Fold boundary	: 10+52+27+65 = 154	50+49+90+52 = 241	11+94+39+63 = 207

Useful in converting multi-word keys into a single word, so that other hashing funcⁿ.

4) Digit Analysis Method

Form hash addresses by extracting and/or shifting the extracted digits or bits of the original key.

E.g.: Key value: 6732541

can be transformed to hash add. 427 by extracting the digits in even pos. & then reversing it.

For a given set of keys, the posⁿ in the keys & same rearrangement pattern must be used consistently

Useful in case of static files where the keys values of all the records.

Collision Resolⁿ

Size = 10 indices: 0, 1, 2, ... 8, 9

Keys: 10, 19, 35, 43, 62, 59, 31, 49, 77, 33

Allotment of more than one key values in one values in one location in Hash table is collision.

Two methods to resolve ~~col~~ collision:

1) Closed Hashing (also called Open Addr. ~~linear~~ probing)
2) Open Hashing (chaining)

Closed Hashing:
→ Linear Probing

→ Quadratic Probing

→ Double Hashing

Linear Probing:

Size of table = h
Addr. of mapping = i

Start ~~with~~ at locⁿ where collision occ. let it be i, then do sequential search until,

i, i+1, i+2 - - - h, 1, 2, - - - i-1

→ Desired index is empty
→ If des. locⁿ not empty ~~then~~ & empty locⁿ is encountered
→ It reaches locⁿ where search st.

Hash table is considered to be circular, hence technique is called closed hashing. Probe means key comparison

~~Size of Hash Table~~ =

Size of Hash Table: 10

No. of elements: 8

Keys:
16, 66, 77, 31, 42, 52, 76, 67

8. Hash function: $H(K) = K \% size$
~~S-1(16)~~

Index:	S-1(16)	S-2(66)	S-3(77)	S-4(31)	S-5(42)
0					
1				31	31
2					42
3					
4					
5					
6	16	16	16	16	16
7		66 *	66	66	66
8			77 *	77	77
9					

~~8~~
S-2: Collision at index: 6
S-3: Collision at index: 7

Index	S-6 (52)	S-7 (76)	~~S-8 (67)~~	S-8 (67)
0			~~67~~	67*
1	31	31	~~67~~	31
2	42	42		42
3	52*	52		52
4				
5				
6	16	16		16
7	66	66		66
8	77	77		77
9		76*		76

S-6: Collision at index: 2
S-7: Collision at index: 6, 7, 8
S-8: Collision at index: 7, 8, 9

c) Quadratic Probing

→ Quadratic probing is a coll. resol. method that eliminates primary clustering.

→ For linear probing, if there is a coll. at loc. i then next i+1, i+2, i+3 etc. locn are to probed, but in quad., the next loc. to be probed are $i+1^2$, $i+2^2$, $i+3^2$

Date

If $H(K) \% 5$ is full

then try $H(k) = (H(k) + i^2) \% 5$
where $(i = 0, 1, 2 \cdots n)$

Primary Clustering

→ Linear probing

→ ∴ Create long runs of filled slots near the hash posⁿˢ

→ If primary index is x, subseq. probes go to $x+1, x+2, x+3 \cdots$

→ Reduces searching time & perf.

Secondary Clustering:

→ Quadratic probing

→ ∴ Create long runs of filled slots away from the hash fⁿ. pos. of keys.

→ primary index = i, subsequent searches go to $i+1, i+4, i+9, i+16$.