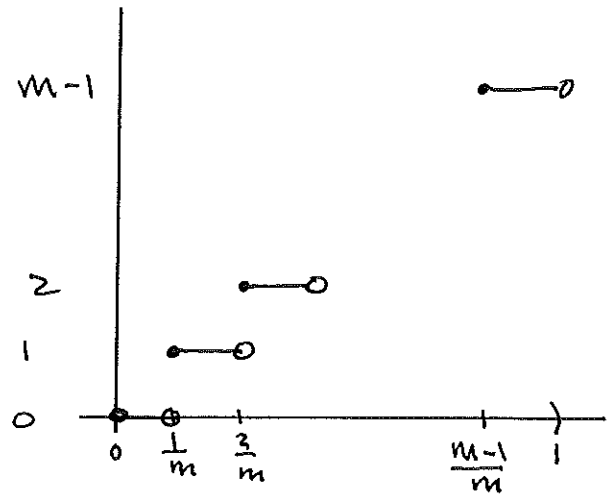
Ex.
IF THE KEYS ARE UNIFORMLY DISTRIBUTED OVER $U = \mathbb{Z}$, THEN $h(k) = k \bmod m$ WORKS.

Ex.
IF THE KEYS ARE UNIFORMLY DISTRIBUTED OVER $0 \le k < 1$, i.e. $U = [0, 1)$ THEN $h(k) = \lfloor km \rfloor$ SATISFIES SIMPLE UNIFORM HASHING.



OBSERVE THAT

$$h : \left[\frac{i}{m}, \frac{i+1}{m}\right) \longrightarrow \{i\} \qquad (0 \le i \le m-1).$$

IF THE PROBABILITY FUNCTION $P(k)$ IS NOT KNOWN, WE MUST TAKE A HEURISTIC APPROACH TO CHOOSING h.

WE WILL ASSUME FROM NOW ON THAT $U = \mathbb{N} = \{0, 1, \ldots\}$.

EX. (DIVISION METHOD)

$$h(k) = k \bmod m$$

WHEN THE DISTRIBUTION OF KEYS IS UNKNOWN, THE CHOICE OF m CAN BE CRITICAL. FOR INSTANCE, WE SHOULD NOT CHOOSE $m = 2^p$ FOR THEN $h(k)$ IS JUST THE p LOWEST ORDER BITS OF k.

e.g. $m = 2^3$, $k = 10110\,(101) = 10110000 + 101$
$\therefore\ h(k) = 101$

IT WOULD BE BETTER TO MAKE $h(k)$ DEPEND ON ALL BITS IN k.

GOOD VALUES FOR m ARE PRIMES NOT CLOSE TO EXACT POWERS OF 2.

EX. (MULTIPLICATION METHOD)
FIX A NUMBER A WITH $0 < A < 1$.
DEFINE

$$h(k) = \lfloor m\,(\underbrace{kA - \lfloor kA \rfloor}_{\text{FRACTIONAL PART OF } kA}) \rfloor$$

THE ADVANTAGE HERE IS THAT THE VALUE OF m IS NOT CRITICAL.

IF $m = 2^p$ THEN THIS FUNCTION IS EASY TO IMPLEMENT AT THE ASSEMBLY LANGUAGE LEVEL.

FOR INSTANCE, SUPPOSE THE WORD SIZE ON OUR COMPUTER IS $w = 8$, AND K FITS IN A SINGLE WORD.

SAY $m = 2^7$ AND $A = .8203125$. THEN

$A =$		$.11010010$
$K =$		10110101
$KA =$	$10010100.$	$\boxed{0111}1010$
$KA - \lfloor KA \rfloor =$		$.0111101 1$
$m(KA - \lfloor KA \rfloor) =$		01111.011
$h(K) = \lfloor m(KA - \lfloor KA \rfloor) \rfloor =$		01111

NOTICE THAT IN THIS EXAMPLE $h(K) = 01111$ BEARS NO OBVIOUS RELATIONSHIP TO $K = 10110101$ (IN TERMS OF BIT PATTERN.)

SOME VALUES OF A ARE THOUGHT TO BE BETTER THAN OTHERS.

OPEN ADDRESSING IS AN ALTERNATIVE TO CHAINING FOR RESOLUTION OF COLLISIONS.

IN THIS METHOD ALL ELEMENTS ARE STORED IN THE HASH TABLE ITSELF, SO IT IS POSSIBLE FOR THE TABLE TO FILL UP, OR "OVERFLOW".

TO PERFORM INSERTION USING OPEN ADDRESSING WE EXAMINE OR PROBE POSITIONS IN THE HASH TABLE UNTIL WE FIND AN EMPTY SLOT.

THE HASH FUNCTION IS OF THE FORM

$$h : U \times \{0, \ldots, m-1\} \to \{0, 1, \ldots, m-1\}.$$

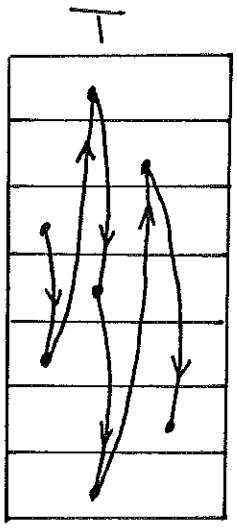
WE REQUIRE THAT FOR EACH KEY, THE PROBE SEQUENCE

$$\big(h(K,0), h(K,1), h(K,2), \ldots, h(K, m-1)\big)$$

BE A PERMUTATION OF $(0, 1, 2, \ldots, m-1)$, SO THAT EVERY SLOT IS EVENTUALLY CONSIDERED AS A POSITION FOR A NEW TABLE ENTRY.

EX.

$h(K,2) = 0$
$h(K,5) = 1$
$h(K,0) = 2$
$h(K,3) = 3$
$h(K,1) = 4$
$h(K,6) = 5$
$h(K,4) = 6$



$m = 7$

IN THE FOLLOWING WE ASSUME THAT TABLE ENTRIES ARE KEYS WITH NO SATELLITE DATA.

Insert (T, K)

```
1.) i ← 0
2.) WHILE i < m
3.)       j ← h(K, i)
4.)       If T[j] = NIL
5.)             T[j] ← K
6.)             Return j
7.)       Else
8.)             i ← i+1
9.) error "overflow"
```

NOTE THAT Insert RETURNS A VALUE, NAMELY THE SLOT j IN WHICH KEY K WAS EVENTUALLY PLACED.

Search (T, k)

1.) $i \leftarrow 0$
2.) Repeat
3.) $\quad j \leftarrow h(k, i)$
4.) $\quad$ If $T[j] = k$
5.) $\quad\quad$ Return j
6.) $\quad i \leftarrow i + 1$
7.) until $i = m$ or $T[j] = NIL$.
8.) Return NIL.

DELETIONS CAN CAUSE PROBLEMS IN AN OPEN ADDRESS TABLE, SINCE THE VALUE NIL IS USED TO CANCEL A SEARCH.

ONE SOLUTION IS TO DEFINE A SPECIAL VALUE CALLED DELETED, WHICH DOESN'T STOP A SEARCH, BUT WHICH DOES STOP THE INSERT PROCESS. STEP (4) OF Insert BECOMES: IF $T[i] = NIL$ OR $T[i] = DELETED$.

Delete (T, i)

1.) $T[i] \leftarrow$ DELETED

THERE ARE A NUMBER OF WAYS TO DESIGN THE HASH FUNCTION

$$h : U \times \{0 \ldots m-1\} \rightarrow \{0 \ldots m-1\}$$

FOR AN OPEN ADDRESS TABLE.

Ex. (Linear Probing)

$$h(k,i) = (h'(k) + ci) \mod m$$

where $h' : U \to \{0 \ldots m-1\}$ is an ordinary hash function and c is a positive integer, relatively prime to m.

Probe sequence:

$$(h'(k), h'(k)+c, h'(k)+2c, \ldots, h'(k)+(m-1)c)$$

Ex. (Quadratic Probing)

$$h(k,i) = (h'(k) + c_1 i + c_2 i^2) \mod m$$

Both of these methods suffer from clustering: long runs of occupied slots, increasing the average search time.

Ex. (Double Hashing)

$$h(k,i) = (h_1(k) + i h_2(k)) \mod m$$

where $h_1, h_2 : U \to \{0 \ldots m-1\}$ are auxiliary hash functions.

THE PROBE SEQUENCE is (MOD m)

$(h_1(K), h_1(K)+h_2(K), h_1(K)+2h_2(K), \ldots, h_1(K)+(m-1)h_2(K))$

OBSERVE: THE INITIAL PROBE AND THE INTERVAL BETWEEN SUCCESSIVE PROBES MAY VARY WITH K. THUS NO TWO PROBE SEQUENCES LOOK ALIKE.

NOTE ALSO THAT $h_2(K)$ MUST BE RELATIVELY PRIME TO m FOR THE WHOLE HASH TABLE TO BE SEARCHED.

e.g. $m=6$, $h_2(K)=3$ YIELDS THE PROBE SEQUENCE

$(h_1(K), h_1(K)+3, h_1(K)+6, h_1(K)+9, h_1(K)+12, h_1(K)+15)$

BUT (MOD 6) THERE ARE ONLY TWO TWO TERMS IN THIS SEQUENCE, NOT 6.

ONE WAY TO AVOID THIS IS TO CHOOSE m TO BE PRIME. ANOTHER WAY IS TO CHOOSE $m=2^p$, AND DESIGN h_2 SO THAT ITS VALUES ARE ALWAYS ODD.