HASHING - APPROACHING DIRECT ACCESS BASED ON THE KEY VALUE

- A simple hashing scheme

Name	ASCII Code for first two chars		Product	Home Address
BALL	66	65	66*65=4290	290
LOWELL	76	79	76*79=6004	004
TREE	84	82	84*82=6888	888

Problems:

- Many names have the same first two characters => collisions
- Some combinations of two characters never occur in any name => skew distribution

HASHING

- A better hashing scheme:

1) Fix the length and represent the key in numerical form

	1	2	3	4	5	6	7	8	9	10	11	12
LOWELL =	L	O	W	E	L	L						
	76	79	87	69	76	76	32	32	32	32	32	32

2) Fold and add

76 79 | 87 69 | 76 76 | 32 32 | 32 32 | 32 32 |

76 79 + 87 69 + 76 76 + 32 32 + 32 32 + 32 32

HASHING

Problem: What if overflow occurs?

Solution: Take prime 19937, do modulo 19937 after each addition.

7679+8769 => 16448 % 19937 => 16448
16448+7676 => 24124 % 19937 => 4187
4187+3232 => 7419 % 19937 => 7419
7419+3232 => 10651 % 19937 => 10651
10651+3232 => 13883 % 19937 => 13883

3) Do modulo by the size of the addressing space (e.g. 101)

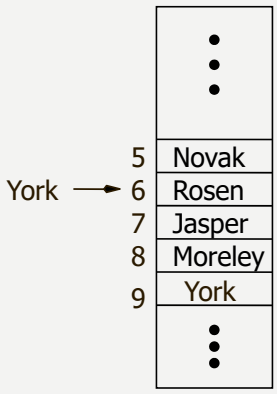
a = 13883 % 101
= 46

HASHING

```
int Hash ( char Key[12], int MaxAddress )
    {
     int sum = 0;
     for ( int j=0, j<12, j+=2 )
         sum = (sum + 100*Key[j] + key[j+1]) % 19937;
     return sum % MaxAddress;
    }
```

RESOLVING COLLISIONS

- Progressive overflow: if collisions occur, check the subsequent position for space.

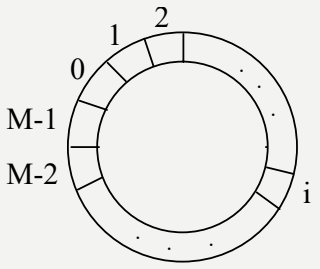


York has the home address of 6 which is occupied.
Thus York is stored in the first empty space after 6, namely 9.

RESOLVING COLLISIONS

Problem: What if the end of the array is reached?

Solution: Do (i + 1) % M, where M is the size of the array, that is look at the array as a circle.



HASHING WITH PROGRESSIVE OVERFLOW

On Search:

If key is found in the home address, OK.

Otherwise sequentially scan the subsequent positions until:

1) Key is found, or

2) an empty space is encountered, or

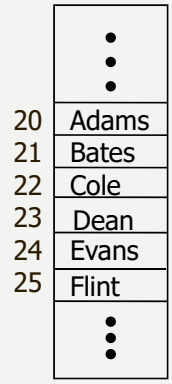
3) come back to the home address.

The last two cases indicate that the key is not there.

SEARCH LENGTH

- Example:

Key	Home Address
Adams	20
Bates	21
Cole	21
Dean	22
Evans	20
Flint	21



Key	Adams	Bates	Cole	Dean	Evans	Flint
#of Accesses	1	1	2	2	5	5

On the average (1 + 1 + 2 + 2 + 5 +5)/6 = 2.67 accesses.

HASHING WITH PROGRESSIVE OVERFLOW

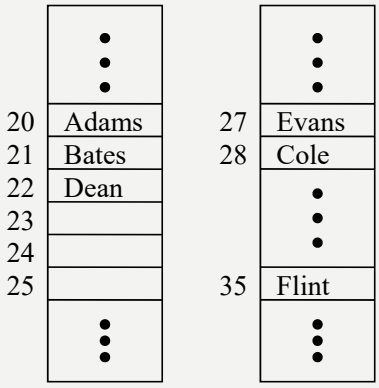
- On deletion – Tombstones: Marked the record as deleted.

- Implications of tombstones for search and insertion:
  1) Search should continue when tombstones are encountered.
  2) Insertion should do a search to make sure the key does not exist and then come back to insert at the first tombstone or empty space.

ALTERNATIVE SOLUTIONS TO COLLISIONS

1) Double hashing: when collisions occur, do (i + c)%M instead of (i + 1)%M. e.g. when c = 7.

Key	Home Address
Adams	20
Bates	21
Cole	21
Dean	22
Evans	20
Flint	21

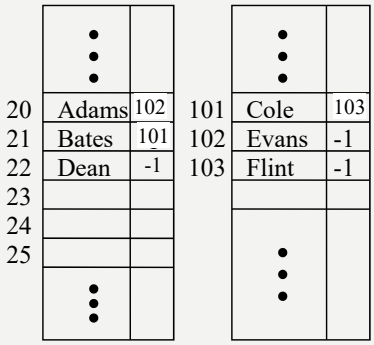


What is the number of comparisons on average?

ALTERNATIVE SOLUTIONS TO COLLISIONS

2) Chained progressive overflow: Overflow records are stored as linked lists in the addresses starting from M on, i.e. M, M+1, M+2, …. e.g. when M = 101.

Key	Home Address
Adams	20
Bates	21
Cole	21
Dean	22
Evans	20
Flint	21

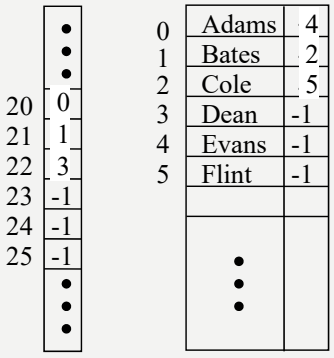


What is the number of comparisons on average?

ALTERNATIVE SOLUTIONS TO COLLISIONS

- 3) Scatter tables

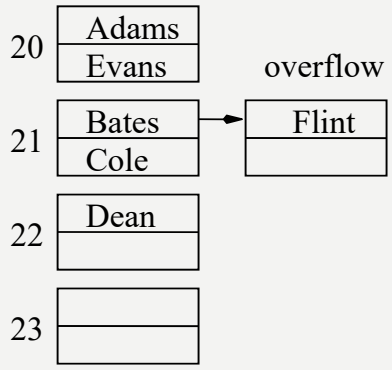
Key	Home Address
Adams	20
Bates	21
Cole	21
Dean	22
Evans	20
Flint	21



What is the number of comparisons on average?

BUCKETS – STORING MORE THAN ONE RECORD IN AN ADDRESS

Key	Home Address
Adams	20
Bates	21
Cole	21
Dean	22
Evans	20
Flint	21



What is the number of comparisons on average?