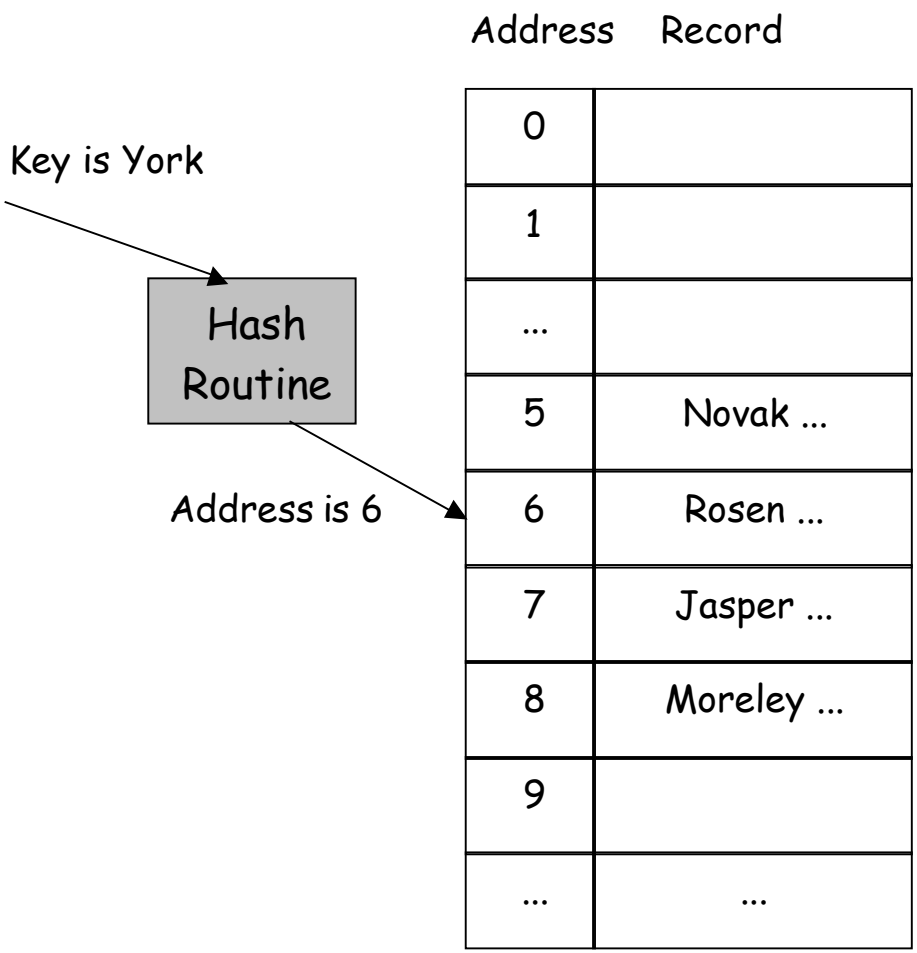
Hashing—Collision Resolution – Progressive Overflow

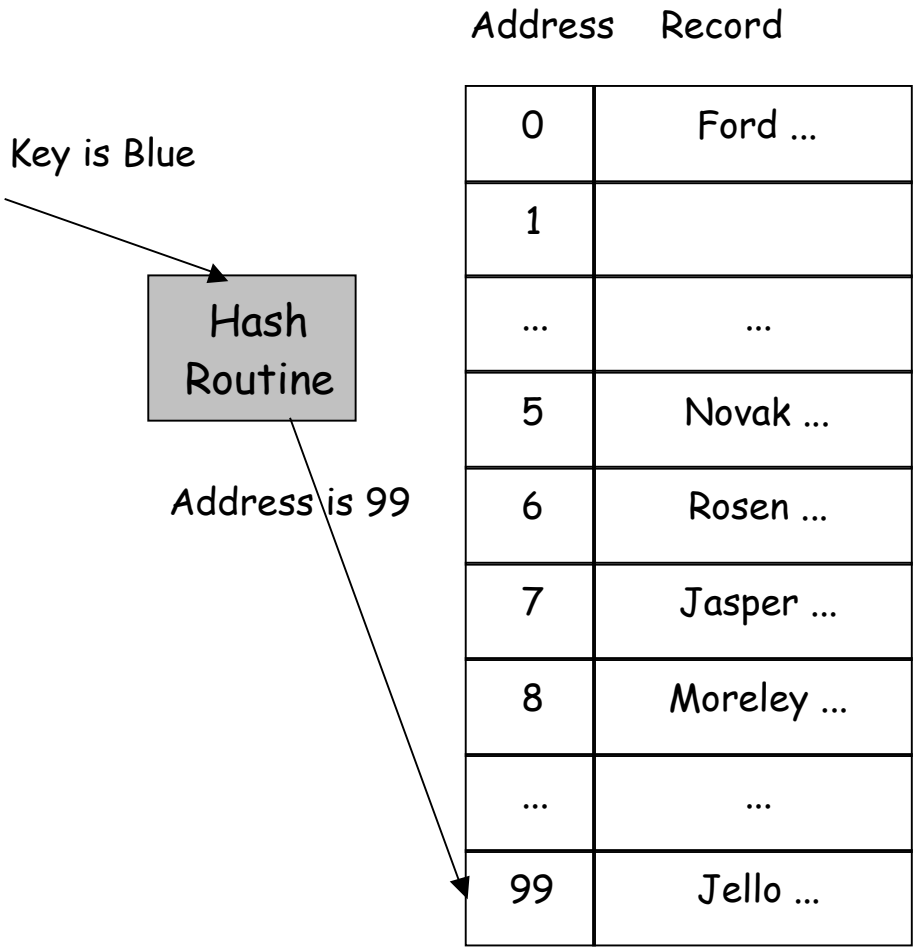


York hashes to 6, which is already in use.

Next we check 7. It is also in use.

We check subsequent spots until we find an empty record.

So, York ends up in position 9.



In this case 99 is busy.

So we start again at 0 until we find an empty record for Blue.

In this example it ends up in position 1.

So how do searches work?

Starting at the hash address look at it and subsequent addresses until

- you find the record: return found information
- you come to the end of the file: start looking at the beginning again
- you come to an empty record: return record not present
- you come to the original address: return record not present

How long will this search be?

Packing density can be used to predict search length.

$$\text{average search length} = \frac{\text{total search length}}{\text{total number of records}}$$

Average search length increases rapidly with increase in packing density

We can tolerate search length of 2 or less.

At 10 and 20 the average search length is 1

Between 20 and 70 it moves to 2

At 80 it is at 3 and at 90 it is on its way past 5

So unless you can afford the extra space this isn't a very good technique.

Collision Resolution – Buckets

Extend the idea of address to an address of a group of records.

This group is the size of a physical block of the file.

Each of these is called a **Bucket**

If we use this technique we can with a bucket size of 3 most of the collisions disappear.

The packing density is now dependent on the bucket size as well as the number of records to be stored and the number of records possible

$$\text{Packing Density} = \frac{r}{bN}$$

Suppose we want to store 750 records in 1000 spaces.

If we use 1000 addresses without buckets the packing density is $\frac{750}{1000} = 75\%$

If we use 500 addresses with a bucket size of 2 we still have a packing density of

$$\frac{750}{2 \times 500} = \frac{750}{1000} = 75\% \quad \text{and} \quad \frac{r}{N} = \frac{750}{500} = 1.5$$

And the Poisson distribution gives us.

p(x)	p(0)	p(1)	p(2)	p(3)	p(4)	p(5)	p(6)	p(7)
Without Buckets	0.472	0.354	0.133	0.033	0.006	0.001	--	--
With Buckets	0.223	0.335	0.251	0.126	0.047	0.014	0.004	0.001

How many addresses are assigned one or two records?

These are the ones that do not have overflow.

So the number that result in overflow with a bucket size of 2 is

$$500 \times [1 \times p(3) + 2 \times p(4) + 3 \times p(5) + 4 \times p(6) + ...] = 140$$

This is $\frac{140}{1000} = 18.7\%$ of the records.

For comparison the with a bucket size of 1 the number of overflow records is

$$1000 \times [1 \times p(2) + 2 \times p(3) + 3 \times p(4) + 4 \times p(5) + 5 \times p(6) + ...] = 222$$

This is $\frac{222}{1000} = 29.6\%$ of the records.

With the same amount of space we can considerably reduce collisions.

Charts in the book indicate that with

- a bucket size of 2
- a packing density of 80
- progressive overflow

The average number of accesses required in a successful search is ≈ 2

Making Deletions

- The slot freed by the deletion must not be allowed to hinder later searches
- It should be possible to reuse the freed slot for later additions

Remember the search routine

We can't have empty addresses because they will signal too soon that a record is not present.

Tombstones for Handling Deletions

We need a solution that has the following characteristics.

- The freed space does not break a sequence of searches for a record
- The freed space is obviously available and may be reclaimed for later additions

Use key that can't otherwise appear.

If the slot following the deleted record is empty you needn't use a tombstone.

What difficulties do tombstones cause?

What must change when we do an insertion?

- You can replace a tombstone with the new record
- But you have to check past the tombstone to the next empty record to make sure you don't have two copies of the record

Especially if there are lots of deletions and insertions!

- The hashed file will reach equilibrium
- But the search length will deteriorate
    - Use local reorganization
    - Use complete reorganization
    - Use another collision resolution technique