Open Addressing

Linear Probing:

h(k), (h(k)+1) **mod** N, (h(k)+2) **mod** N, (h(k)+3) **mod** N, ...

Open Addressing: put Method (linear probing)

Algorithm put (k,data,N)

In: record (k,data) to insert, size N of hash table

Out: {add record (k,data) to table, or ERROR if insertion not allowed}

```
pos ← h(k)
count ← 0
while (T[pos] != NULL) and (T[pos] != DELETED) do {
    if T[pos].getKey() = k then ERROR
    pos ← (pos + 1) mod N
    count ← count + 1
    if count = N then ERROR
}
T[pos] ← (k,data)
```

Open Addressing

Linear Probing:

h(k), (h(k)+1) **mod** N, (h(k)+2) **mod** N, (h(k)+3) **mod** N, ...

Double Hashing:

h(k), (h(k) + h'(k)) **mod** N, (h(k) + 2×h'(k)) **mod** N,

(h(k) + 3×h'(k)) **mod** N, ...

Open Addressing: put Method (double hashing)

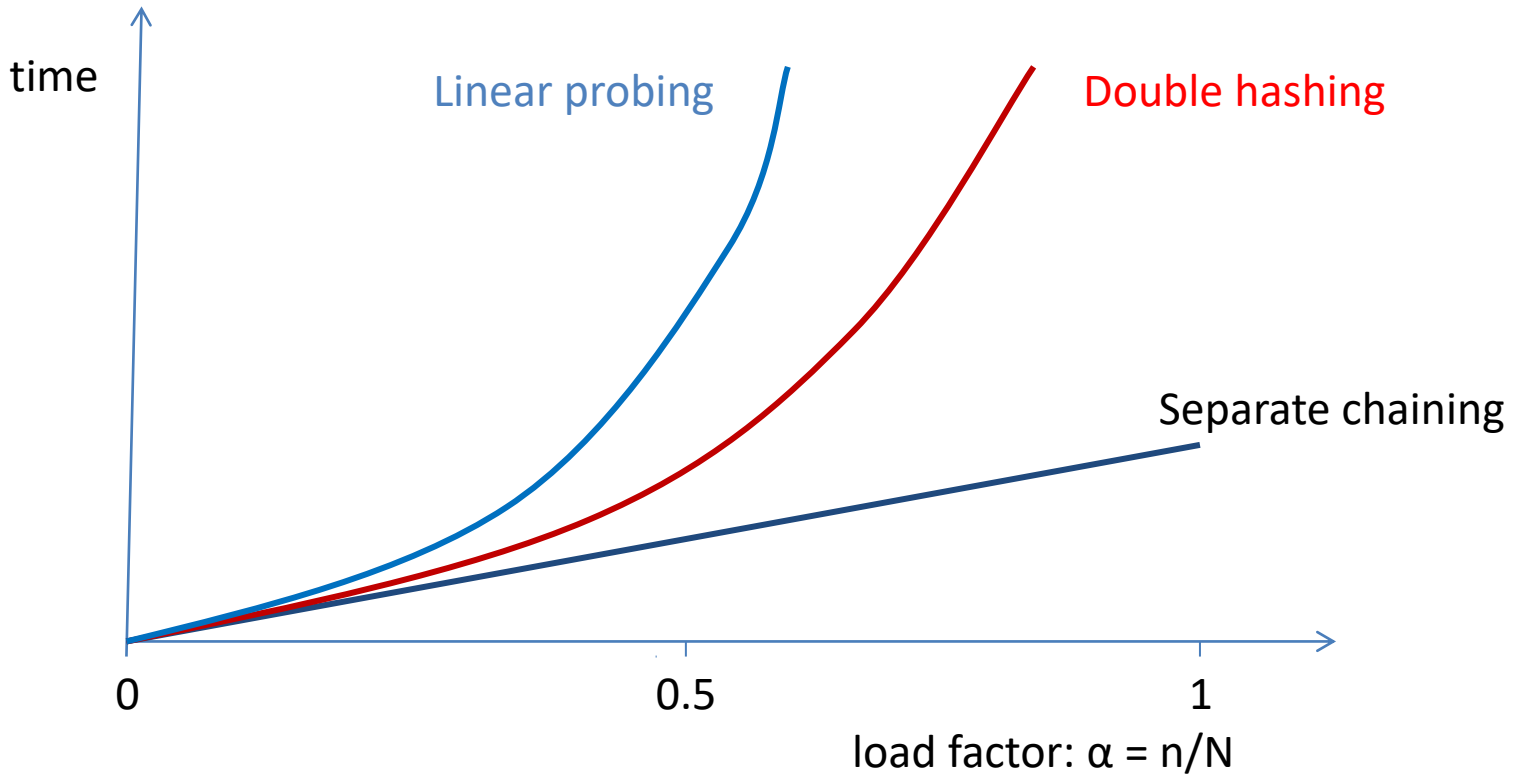
Algorithm put (k,data,N)

In: record (k,data) to insert, size N of hash table

Out: {add record (k,data) to table, or ERROR if insertion not allowed}

```
pos ← h(k)
count ← 0
while (T[pos] != NULL) and (T[pos] != DELETED) do {
    if T[pos].getKey() = k then ERROR
    pos ← (pos + h'(k)) mod N
    count ← count + 1
    if count = N then ERROR
}
T[pos] ← (k,data)
```

Average Time Complexity of get Operation



	Average number of key comparisons
Separate chaining	$1 + \alpha$
Linear Probing	$\frac{1}{2} + 1/(2(1 - \alpha)^2)$
Double Hashing	$1/(1 - \alpha)$