1. Basic knowledge of hashMap:

We have a node<K,V> to store the key and value
The node is stored in an array called table

We uses the hash(`return (key == null) ? 0 : (h = key.hashCode()) ^ (h >>> 16))` to get the hash

```
We use the following to get the index
n = table.length;index = (n-1) & hash;
```

To optimize the performance, when a length of list equals 8 and the table.length is over 64, change the list to tree.
Likewise,change tree to list when the length comes to equals 6

Default capacity: 16

Maximum capacity: 1 G (2^30)

Loader Factor: 0.75

Treeify_thredhold: 8 // the length of a list exceeds 8, change the list to a tree
Untreeify_thredhold: 6 // If the length is shorter than 6, change a tree to a list

```
static final int MIN_TREEIFY_CAPACITY = 64;

//if the table.length < 64, the table will resize not treeify
```

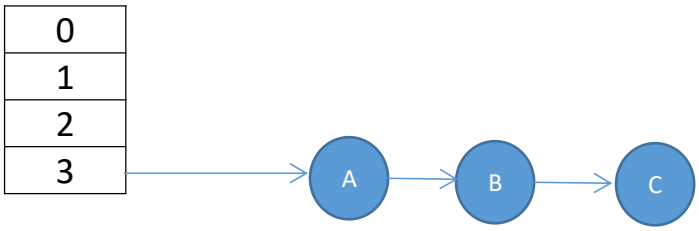
2. DeadLock Analysis

When size >= Capacity*loader Factor

```
void resize(int newCapacity) {
  Entry[] oldTable = table;
  int oldCapacity = oldTable.length;
  ... Entry[] newTable = new Entry[newCapacity];
```

```
... transfer(newTable, rehash);
  table = newTable;
  threshold = (int)Math.min(newCapacity * loadFactor,
MAXIMUM_CAPACITY + 1); }


void transfer(Entry[] newTable, boolean rehash) {
  int newCapacity = newTable.length;
for (Entry<K,V> e : table) { while(null != e) { Entry<K,V> next = e.next;
if (rehash) { e.hash = null == e.key ? 0 : hash(e.key); }
  int i = indexFor(e.hash, newCapacity);
e.next = newTable[i];
newTable[i] = e;
  e = next; } } }
```

Suppose we have 3 nodes and table.length=4
These nodes have same index

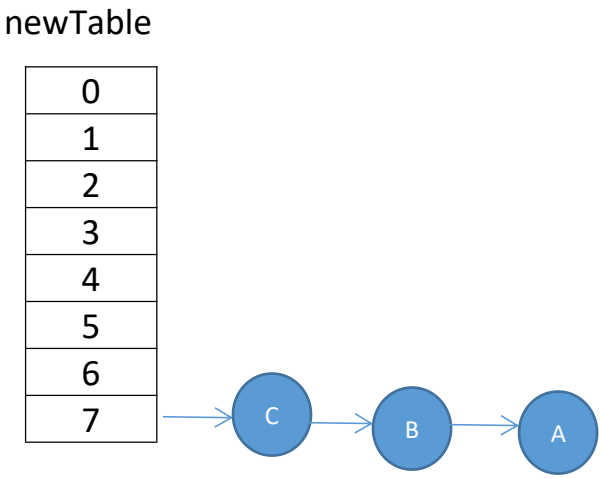


When 2 threads add a need node,the both threads need to resize the table.
Suppose thread 1 yields the cup when it completes Entry<K,V> next = e.next, thread 1 takes cpu

Thread 1 refresh the table and don't complete update:

Before table = newTable;

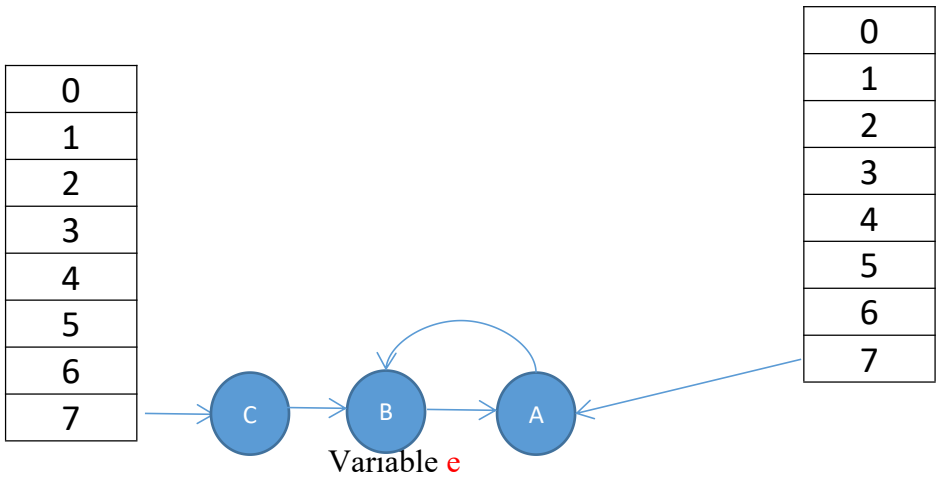
Thread 1



Then thread2 takes the cpu and continues the following instructions

```
Entry<K,V> next = e.next;

 int i = indexFor(e.hash, newCapacity);

e.next = newTable[i];

 newTable[i] = e;

e = next;
```

Now we know e=b b.next=a a.next =b



Therefore, a loop is formed between A and B