姓名：刘佳

学号：517021911048

2018年11月24日

Documentation

LRU

```
struct Node
{
    int key;
    int value;
};
```

```
private:
    int size;
    list<Node> cacheList;
    map<int,list<Node>::iterator> mp;
```

Node 类型和LRU的私有成员如上。

其中，Node用于存储键和键值。私有成员size即LRUCache的容量。cacheList用于存储Node链，least recently used 的key存放在链的末尾，most recently used 的key存放在链的首部。mp的键即key，键值为相应key的Node在cacheList中的位置。mp的作用在于能快速找到key对应在cacheList中的位置，降低时间复杂度，空间复杂度为O(n)。

在函数int get(int key)中， cacheList.erase(it) 、cacheList.push_front(Node)、cacheList.begin()的时间复杂度都为Constant，mp.find(key)、mp[key]的时间复杂度为Logarithmic in the size of the container。且未嵌套任何循环，故时间复杂度为O(lgn)。

在函数void put(int key, int value)中，mp.end()的复杂度为Constant，其余与以上相同。且未嵌套任何循环，故时间复杂度为O(lgn)。

LFU

```
private:
    struct FreNode{
        int fre;
        list<pair<int, int>> val;//maxsubfre to minsubfre, {key,value} key is used when cache fills
            map.erase(it->first)
    };
    int cap;
    int total;
    list<FreNode> freList;//minfre to maxfre, every FreNode stores {key,value} with the same frequency
    unordered_map<int, pair<list<pair<int, int>>::iterator, list<FreNode>::iterator>> mp;//key to {FreNode
        position,freList position}
```

LFUCache的私有成员如上。

其中，FreNode为同一频率下{key,value}的集合，按时间顺序由出现距今时长最短到出现距今时长最长排列。cap即LFUCache的容量，total即LFUCache中现有的存储量（等于mp.size()）。freList按频率由小到大排列。mp记录每一个key对应在FreNode中的位置和freList中的位置，便于依值访问并更改。

在函数int get(int key)中，list.erase(it) 、list.end()、 list.insert(iter,FreNode) 、list.push_front(*it) 、list.begin()、 list.empty()的时间复杂度都为Constant，mp.count(key)、mp[key]的时间复杂度为Logarithmic in the size of the container。且未嵌套任何循环，故时间复杂度为O(lgn)。

在函数void put(int key, int value)中，list.pop_back() list.pop_front() list.front()的复杂度为Constant，其余与以上相同。且未嵌套任何循环，故时间复杂度为O(lgn)。