分析：

1. 设计O(1)的算法，需要三个map
2. 分别存储k-iter, k-{v, freq}, freq-keys

```
#include <iostream>
#include <list>
#include <unordered_map>

// @lc code=start
class LFUCache {
public:
  LFUCache(int capacity) {
    _minFreq = 1;
    _cap     = capacity;
  }

  int get(int key) {
    if (!kvMap.count(key) || this->_cap <= 0) {
      return -1;
    }

    increaseFreq(key);
    return kvMap[key].first;
  }

  void put(int key, int value) {
    if (this->_cap <= 0) {
      return;
    }

    // 已存在key, 更新频率
    if (kvMap.count(key)) {
      kvMap[key].first = value;
      increaseFreq(key);
      return;
    }

    // 不存在
    if (_size >= this->_cap) {
      // _cap已满, 删除一个元素
      removeKeyFromkvMap();
      _size--;
    }

    // 插入一个元素
    kvMap[key] = std::make_pair(value, 1);
    freqKeysMap[1].push_back(key);  // 每次新插入元素, 都在list的最后
    keyIter[key] = --freqKeysMap[1].end();
    // 每次新插入元素, minFreq 都重置为1
    this->_minFreq = 1;
    _size++;
  }

  void print() {
    for (auto it : kvMap) {
      std::cout << it.first << ' ' << it.second.first << ' ' <<
it.second.second
                << std::endl;
    }
    std::cout << std::endl;
  }

private:
  void increaseFreq(int key) {
    int freq = kvMap[key].second;
    kvMap[key].second++;  // 访问了一次，freq增加1

    std::list<int>::iterator it = keyIter[key];
    freqKeysMap[freq].erase(it);  // 从原freq中删除当前key

    if (freqKeysMap[freq].empty()) {
      freqKeysMap.erase(freq);
      if (this->_minFreq == freq) {  // 原freq中只有一个key，且为最小频率
        this->_minFreq++;
      }
    }
    // 在新的freq中插入key
    freqKeysMap[freq + 1].push_back(key);
    keyIter[key] = --freqKeysMap[freq + 1].end();
  }

  void removeKeyFromkvMap() {
    int key = freqKeysMap[_minFreq].front();
    freqKeysMap[_minFreq].pop_front();

    kvMap.erase(key);
    keyIter.erase(key);
    if (freqKeysMap[_minFreq].empty()) {
      freqKeysMap.erase(_minFreq);
    }
  }

  int                                             _minFreq;  // 最小的freq
  int                                             _cap;
  int                                             _size;
  std::unordered_map<int, std::pair<int, int>> kvMap;  // key-<val, frep>
map
  std::unordered_map<int, std::list<int>::iterator>
                                           keyIter;      // key-iterator
map
  std::unordered_map<int, std::list<int>> freqKeysMap;  // freq-keys map
};
```