1. 旅行商问题

d[i][j]: 城市i到城市j的距离

状态变量：dp[i][j]:从0出发访问过i中的城市，现在位于j位置

1. Dijisitra最小路径算法，维护最小堆，算的每个点到终点的最短路径
2. 麻将游戏
3. 【最大/小频率】字符串（只包含大写字母或只包含小写字母），最大出现频率次数在记录过程中就能找到，最小频率次数在搜索完后遍历26个字母对于的数就能找到。记录O（n）,找到最大/最小频率是O(1)
4. 给一堆pair说明点连线关系。
   1. 可以规定<a,b> a<b，在map中记录pair
   2. 映射关系，如果a，b都是小于10^5, <a,b>连接可以映射成10^5\*a+b, 这样保留了所有信息，起到了去重统计作用
5. 有序数组查找和大于q的所有数对：双指针，从两头开始，一直移动左指针到满足q

对数为s+=r-l;

            while(l<r)

            {

              while(l<r&&ds[l]+ds[r]<=q) l++;

              if(l<r) S3+=(r-l);

               r--;

            }

1. 【容斥+连接映射】**问题1782**，点对一共拥有的连接线。记录每个点拥有的线O(n), 点对<a,b>之间连接线O(m), 如何查找点对连接线满足要求的点。
   1. 搜索所有连接O(m)。有链接且满足要求的。
   2. 无连接且满足要求的。（容斥） (双指针查找有序数组满足和大于q的点对)
2. 0-1背包问，完全背包问题，多重背包问题。

每个物品最多可以放2个，那就把物品变成两个，然后问题转化为0-1背包问题

1. 一个数组nums中，枚举出每个子序列（子集）的和。对于每个元素只有放和不放。

Vector<int> res;

For(int i=0;i<n;i++)

{  
 int n=res.size(); //这句别忘了，不然会陷入无限死循环

For(int j=0;j<n;j++)

Res.push\_back(nums[i]+res[j]); //复杂度是2^n; 当n<10; 1024比较小的时候可以用

}

1. **1774 0-1背包问题**，注意思考角度。0-1背包问题主要思想是，第i个物体放和不放两种情况。动态规划是基于上依次结果，更新的时候需要从后往前！！！一定要注意
2. **5最长连续回文子串问题**。动态规划优化内存的时候，涉及到多次重写更新。一定要把逻辑完整写出来，切记不要省略。需要特殊考虑两个相等相连的情况
3. 1**771 注意动态规划循环顺序**

dp[i][j]=max(dp[i+1][j],dp[i][j-1]);

同一层dp[i][j]=dp[i][j-1]说明先有j-1再有j, 所以j循环是从前往后

优化时候顺序注意：

dp[i][j]=dp[i+1][j-1]+2;

由于i和i+1相邻层，所以可以优化，该层j取决于上一层j-1.

所以空间优化后的循环顺序是j->j-1,从后往前

1. 最大整数2^31-1, 最大负数-2^32.补码：保持符号位不变。其他位取反+1就是负数的绝对值大小。
2. 记录**新**出现的值，去重并记录。数据量大的时候要用unordered\_set取代vector<bool>

千万级别就算很大了。用Vector会超时，这个时候应该用unordered\_set

1. **Leecode4寻找两个正序数组的中位数**。O（long(n+m)）思想值得学习，

寻找第k小的数据，寻找第1小很好找，于是想到迭代删除。

迭代法，更新k，idx=k/2-1; 更新头指向

如果数组索引超出长度，min解决

如果数列空了，直接返回非空数列第k小；

非空但是k==1的时候，返回头中较小的

1. 寻找最长重复子序列
2. **LRU问题**list和map，记录最近访问排序和key-val等信息

注意，访问的时候要更改排序，放到头

放入数据时候：如果以及存在的key（修改value和排序）；不存在但是满了，删除一个，再插入头；没满直接插入

1. **Leetcode1044**寻找字符串中是否包含长度为L的重复子串。O(n)。编码法+二分应用

**二分最大特点是logN，对于低于目标值都具有的特性，高于目标值都具有另一个特性。这种情况可以用二分，二分的l，和r是目标范围，在二分过程中不断缩小范围**

字母最多26个

长度为3的字符串”abc”: (1\*27+2)\*27+4

注意0对于编码是没有体现信息的，所以a-z对应-26，进制取27

这种编码方式，编码值会非常大🡪根据 生日悖论，模数一般需要和被编码的信息数量的平方根的数量级相同(字符串长度为10^5，长度为L的子串最多也就10^5个)，在本题中，相同长度的子串的数量不会超过 N 个，因此选取模数是 2^{32}（无符号整型数的最大值）是足够的。

为了保险起见，在编码相同的时候还是要对字符串进行再次比较判断。

abcd滑动一次，结合hashmap就能知道是否存在

1. Leetcode68 有效数判断逻辑题。

今日任务：背包问题，合并集问题；链表排序问题

1. **Leetcode7整数反转。**不能取绝对值。带符号的str2int,int2str转换和大小比较如果将

s+=(s[0]=='-')?('0'-x%10):('0'+x%10);

写成

s+=(s[0]=='-')?( -x%10+'0'):( x%10+'0');

会报错，( -x%10+'0')当x=INT\_MIN 前面添加负数会越界

( '0'-x%10) 这样就会会对x取负号

这个方法烂透了。较好的思路

每次反转一位前都确定不会溢出，再更新

1. 有序数组寻找和为q的所有可能。O（n）。双指针，不断靠近。

**15三数之和** 三数之和的时间复杂度是O(n^2)

1. 删除倒数第n个节点。建立一个头节点；快慢指针都从头节点开始走。需要找到倒数第n+1个，也就是说快指针先走n+1步
2. **42接雨水问题**。O(n), 前后指针，更新水位
3. **815公交路线问题。**BFS，从小到大遍历所有情况, 找到最小值。【不难但是速度太慢
4. **72最小编辑代价。**DP[i][j] 代表从word1前i个字母便哼word2前j个字母的编辑代价。

If(word1[i]==word2[i]) dp[i][j]=dp[i-1][j-1];

Else dp[i][j]=min(dp[i-1][j-1]替换，dp[i-1][j]删除，dp[i][j-1]添加)+1；【当时没思路】

1. **⭕10正则匹配**。用逻辑法递归做很耗时间，准确率不高。

动态规划dp[i][j]前i个和前j个是否能匹配，转移方程有点难想加油

出现’\*’的处理方式值得学习

1. **⭕**26K个一组反转链表。，链表的题目多做几遍。【速度太慢了】
2. **124二叉树的最短路径和。【耗时太久，思路不够敏捷】**

返回值：单侧路径最大（包含根）；结果是单侧路径最大，单侧路径+根和最大记录在res，不断更新，后序遍历

1. 32最长有效括号子串。DP。虽然说栈也能做，但是很多细节会考虑不到。凡事能DP请DP。
2. **⭕312戳气球**。考察逆向思维。分治。记忆法递归实现分治。虽然通过但是耗时太长了

**从暴力角度看是O(n!)级别复杂度，通过分治降低到**

**好好理解分治为什么快!!!** 对于n区间问题，第i层有2^i个规模为(n/2^i)的子问题

如果每个子问题复杂度为O(n), 每层的时间复杂度为n, 一共有lgn层。解决这个问题需要O(nlgn)

**DP也能做？？？不太好想**

1. **239滑动窗口最大值**。优先队列。线段树nlgn(虽然过了但是内存开销太大) 【线段树代码不熟练！！】定成区间最大问题，非定常区间最大 多次访问问题。不定区间用线段树；定区间用优先队列nlgn。维护前n个的最大值，如果最大值查过了窗口左边界弹出。

**单调队列。**

**单调栈数据是有先后顺序的，同时保证了这个栈是递增或则递减的。【掌握不熟练】**

什么问题适合用单调栈解答：前/后面第一个比当前值大，滑窗中的最大，最小值，（涉及到先后顺序和最大最小值的）

1. **√879 0-1背包问题改编**。三维。做的不对

完成集中训练【416，474，**1049**，**377**，494】

**两个约束条件就是三维度。**

0-1：前i个作为状态量一部分

利润高于：有个状态量就是至少利润。。。

不多于：有个状态量也是不多于

最接近targ的值：既是约束条件又是求解值？？？🡪

约束条件：最接近j的

求解值：满足该约束条件的值

1. 数组中是否存在和为sum的子集。罗列出所有子集的复杂度为指数倍。

每个元素在或者不在0-1背包问题：前i个

约束条件和为sum: 和为j

时间复杂度为n²

TopK

int rankK(vector<vector<int>> nums,vector<int> P,int k)

{

//10^5次方

priority\_queue<int, vector<int>, less<int>> q;

for (auto& i : P)

{

//查找 nums[i]

for (int j = 0;j < nums[i].size();j++)

{

if (q.size() < k) q.push(nums[i][j]);

else if (q.top() > nums[i][j])

{

q.pop();

q.push(nums[i][j]);

}

else break;

}

}

return q.top();

}

# 图

# 分糖果问题

## 135. 分发糖果

评分更高的比相邻两侧孩子获得糖果数多

每个孩子至少获得一颗

左侧的分数高的糖果多；右侧的分数高糖果多

取两者max

## 最大中位数

# √跳跃问题

## 55跳跃游戏

贪心思想，递归思想

**从后后往前降低目标**

**最后判断目标是否为0**

## √1696. 跳跃游戏 VI

动态规划，不优化会超时

第二个循环可以优化，

使用了单调递减栈实现O（1）去最值

## √1306. 跳跃游戏 III

层序遍历

## ⭕LCP 09. 最小跳跃次数

贪心很容易出错，动态规划

i->n最少到达次数

从i位置跳到>=n位置至少需要dp[i]次

Dp[i]=dp[i+jump[i]]

第二个循环的条件约束需要注意，不然会超时

for(int j=i+1;j<min(n,i + jump[i])&&dp[j]>dp[i];j++)

## √1345. 跳跃游戏 IV

左跳右跳，同值跳

**层序遍历减枝法**

## 1340. 跳跃游戏 V

动态规划，填表顺序从低到高

**975. ⭕奇偶跳**

单调栈和动态规划，但是只是过了一半

# 单调栈和优先队列的应用

求最大值：front放着最大，所以维护的是单调递减

求最大：front放的是最小，所以维护的是单调递增

deque双向队列 front放着最大值/最小值（根据时间需要维护单调递增和单调递减栈）

放入新元素。以下是维护单调递减栈

       while(q.size()&&dp[i]>=q.back().first)

            q.pop\_back();

        q.push\_back({dp[i],i});

读取最值

         while(q.size()&&q.front().second>i+k) q.pop\_front();

寻找后面第一个小于当前值的数所在索引，比较的都是back

       for(int i=0;i<n;i++)

       {

          while(q.size()&&A[i]<=A[q.back()])

          {

                smaller[q.back()]=i;

                q.pop\_back();

           }

            q.push\_back(i);

       }

# √层序遍历及其优化

## 55跳跃游戏

从前往后层序遍历，超时

优化：跳过的地方不要重复（用visited数组做记录）

# √并查集用在分类问题上

## √ 分类

49同母异位词分词

分类数cnt，属于哪个类

## √ 1202连通集合

查集是由一个数组pre[]，和两个函数构成的，一个函数为**findgp()函数，用于寻找前导点的最后返回根**，第二个函数是**join()用于合并路线的**

**//以较小的作为组编号**

 void join(vector<int> p)

    {

       int a=min(p[0],p[1]);

       int b=(p[0]==a)?p[1]:p[0];

       int r1=findgp(a);

       int r2=findgp(b);

       pre[r2]=r1;

    }

    int findgp(int i)

    {

        if(pre[i]==i) return i;

        return pre[i]=findgp(pre[i]); //进行了路径压缩

    }

注意连接完之后对所有数进行findgp，统一分组编号

然后用unordered\_map进行分组

对分组进行操作

# 递归回溯和分治

递归回溯，数组中有重复元素，求满足要求的所有不同子集。

先sort排序，set容器去重

## √50 pow（x,n）分治算法

求解一半就能知道整个结果的可以用分治。O(lgn)

## 312戳气球

分别求解两部分可得全局，如果每个递归内部复杂度是n，则分治复杂度是nlogn。

# √旋转数组和二分搜索

**二分要确保输入的hi>=lo**

有重复一定也要先去重(while(n>=3&&nums[n-1]==nums[0]) n--;

不然无法二分

## √二分搜索三种写法

寻找第一个target

最后一个target

含有等号的写法（while(l<=h)）

**二分要确保输入的hi>=lo**

## √33搜索旋转排序中的目标值

（无重复元素的）

先找出旋转点，再二分

**二分要确保输入的hi>=lo**

## √旋转矩阵

Visted矩阵; 方向数组dx,dy; 注意走到头回退，往新方向走一步

**方位别写错了**！**按边走不是按点走！每次递归走一边**

按顺时针/逆时针返回矩阵内部数据

顺时针：右，下，左，上递归按边访问

注意x：指排：上下相关

Y：指列，左右相关

     int dx[4]={0,1,0,-1};

**int dy[4]={1,0,-1,0};**

**逆时针：**左下右上按边访问

## √面试10.03搜索旋转数组

有重复，寻找第一个target索引

后面的和前面相同就去掉后面重复的（**去重前提是n>=3**），更新范围位n范围（范围不要忘记更新）

查找旋转点（二分查找）

确定target范围进行正常二分（确保二分输入的h>=l）

## √ 189 旋转数组的三种方法

将数组选装k个，空间复杂度O(1)

* 1. K步，数组长度n 空间复杂度O(nk) 每次走1步，走k次
  2. 旋转数组，走K步，把后面K个数字平移到前面（涉及到内存的删除和开辟）
  3. 1234567

走3步567|1234

整个倒置7654321

765|4321 然后各部分内部反转(0-k-1,k-n-1)

旋转图像

# 曼哈顿距离问题

√1162. 地图分析 求最近最大 层序遍历

？？求距离所有核站曼哈顿距离最大/最小 并输出最远的点

# √动态规划问题

## 核心总结：

## 变量几个，变量个数=状态量个数+所求量个数

如何确定状态量？从题目中涉及到的量的个数。如果递推关系用到一个，变量就少一个。

如果是背包问题就直接把这些量原封不懂替换成状态，不改变语义；

如果不是背包问题：根据实际对象，修改状态量含义（上升子序列🡪第i个结尾才能利用上升回文 🡪第i个开头，j结尾）

修改还是要看经验和积累

## 状态转移依据是什么？背包依据（选择/方案/子集）；所求对象特性（回文，上升子序列，连续上升子串）决定

-------------------------------------------------------------------------------

1. **1049 背包问题+脑筋急转弯**。如果问剩下最大，怎么解答？？？从最大的数字开始，剩下的求最小。复杂度不超过n^2
2. **518完全背包问题不能内存优化!!**
3. **879【至少】**状态量中有这层含义的，当状态量<0应该取0而不是舍弃
4. **377【有无顺序】**背包问题解题总结：

如果是组合问题，是否需要考虑元素之间的顺序。需要考虑顺序有顺序的解法，不需要考虑顺序又有对应的解法。（不考虑排序：前i个）

排序数：状态量是背包容量为j时的排序数量

到达容量j上一步是任何一个物品，状态量只有一个。

排序数字可能会非常大，用unsigned long long

**如果求组合数就是外层for循环遍历物品，内层for遍历背包**。

**如果求排列数就是外层for遍历背包，内层for循环遍历物品**。

1. **【涉及到正负】**完全背包问题，都是正数，寻找和为target的排序数，组合数 377 青蛙跳从后往前

正号和负号的遍历方向不一样

1. **【含负号】**含负数的01背包问题

**poj2184**

5头牛，对于分别的情商和智商

主人从中选几只参加展览，要求智商和>=0,情商和>=0

求情商智商和最大为多少？？？

情商和+智商和在 -1000~1000范围内

5

-5 7

8 -6

6 -3

2 1

-8 -5

输出结果：

8

分析：放和不放影响结果

三维变量：前i只牛，智商大于等于s,情商大于等于f

问题1，三维内存消耗太大了；情商和智商还存在负数；

Dp[i][j]: 前i头牛智商为j时候，总商最大值

三个变量确定2个递推求第三个量

递推关系：0-1放和不放

dp[i][j]: 前i头奶牛，智商为j时候的最大总商;

dp[i][j]=max(dp[i-1][j],dp[i-1][j-smart[i-1]]+funny[i-1];

问题2：当前状态用到了未来状态（还没获得呀）

解决，判断元素正负决定循环方向

问题3：初始化问题。不能用INT\_MIN, 会溢出导致负数变正数；总商下线的好几倍

表示不存在。

vector<int> dp(tol+1, -999999); //不能写INT\_MIN 因为过程中再增加负数就变成正数了

const int N=200010;

for(int i=0;i<=N;i++) dp[i]=-INF;

int mid=100000; //负数索引解决方案，将零位前移到mid

dp[mid]=0;

for(int i=1;i<=n;i++){

if(a[i]>0)

for(int j=N;j>=a[i];j--){ //二维会看得更清楚

dp[j]=max(dp[j],dp[j-a[i]]+b[i]); //

}

else

for(int j=0;j<=N+a[i];j++){ //纯负数的更新

dp[j]=max(dp[j],dp[j-a[i]]+b[i]);//负数序列，要从后往前

}

}

1. 背包问题集训

<https://blog.csdn.net/xiaoleiacmer/article/details/10242365?utm_medium=distribute.pc_relevant.none-task-blog-baidujs_title-1&spm=1001.2101.3001.4242>

1. 求和方案问题

N个数的数组，数不重复，全正，求和为tag的子集总数

分析：01背包问题，无排序，全正数

前i个；和为j；转移求子集最大数目（定2，求1最优）

递推关系: 每个元素放与不放

1. N个数的数组，数不重复，全正，求和为tag的组合排序数，可重复使用

分析：完全背包；有排序

和为i的 排序数

递推关系：N个元素任意一个放到i-1就能得到i的表达.（N个数这个量用作了递推关 系，所以变量只有两个），状态量只有一个和为j对应的排序数

1. 最长上升子序列的长度

变量长度，n个数

前i个，最长上升子序列长度-🡪放与不放无法影响长度（不属于01背包问题）

修正：以第i个结尾的最长上升子序列 （）

# ⭕数据结构题LRU和LFU

## LRU最近访问的前capacity个值

分析：最近🡪list 如果<key,val>

Key，value，所在链表位置

Unordered\_map: 记录<key,<value,pos>> 通过key可以查询到所有信息

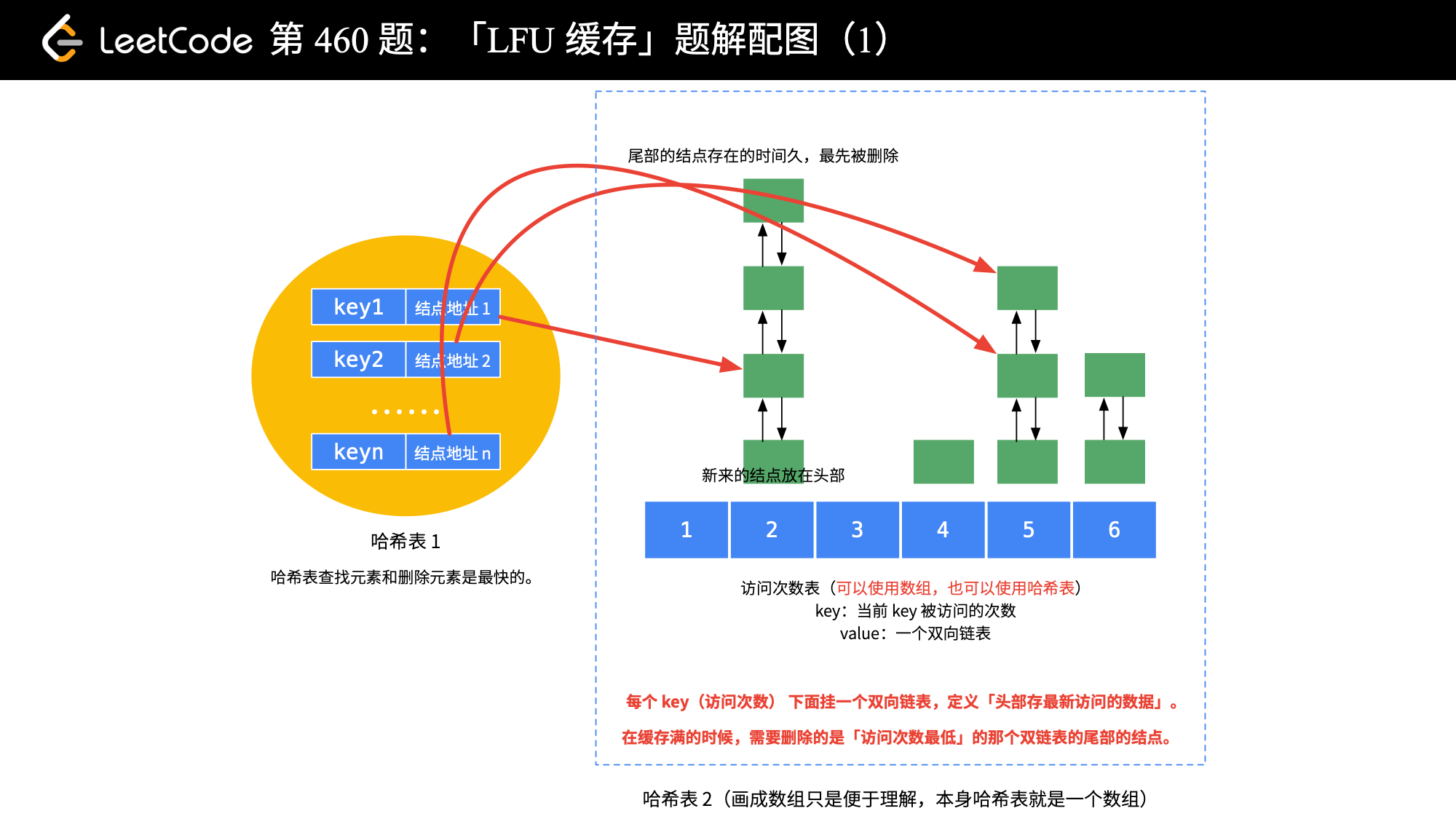
当缓冲区满的时候，可以需要删除list尾巴，这个时候需要索引到对应的key

所以list中装key 实现了两者相互索引

LFU: 最少/最多访问频率

Leetcode460

建立node节点：key,value,frq;信息



键值访问: unorderd\_map<key,list::iterator> keymap;

不同频率对于一个链表unorderd\_map<int,list<node>> frqmap;

注意: cap==0的情况

需要记录minrq。增加新值为1

Get的时候判断更新

1. **什么邻接表**
2. **并查集的题目**
3. **完全背包问题和不完全背包问题**

【低级错误】

1. 回溯一定要还原
2. Memset(v,0,sizeof(v))；sizeof返回的单位是byte，用于string和数组，不能用于vector
3. vector的resize是会保留原有数据
4. priority\_queue <int,vector<int>,greater<int> > q
5. set使用迭代器时候，尤其\*(--it) 首先要确保it!=st.begin(); 不然数不对
6. leetcode205同构字符串。所谓同构就是出现的字母最多被映射一次和映射别人一次。需要两个hash表记录。题意不好理解啊
7. 计算结果可能很大的话，里面涉及到的变量都要设置成long long
8. 注意变量更新的先后顺序。使用的时候看一下是否被改变了
9. 创建新变量一定要初始化。
10. 循环变量没有++或者--。没长记性。。。
11. S+=c1+c2; ❌ s是string，c1,c2是char. S=S+c1+c2✔
12. 注意动态规划的初始化，和状态量索引和在数组/字符串索引之间是否差1

动态规划的时候循环初始值错误；

循环索引使用错乱

状态转移中i-1写成i

1. 数组初始化vector。 Int nums={1,2,3,4} vector<int> v(nums, nums+size(nums))
2. Stac没有clear。自居while手动清空谢谢
3. 线段树查找[l,r]区间最大值的代码写错

  void getmax(vector<int>& tr,int i,int start,int end,int l,int r,int& res)

    {

        if(start==l&&r==end)

        {

            res=max(res,tr[i]);

            return;

        }

        int mid=(start+end)>>1;

        if(l<=mid) getmax(tr,i\*2+1,start,mid,l,min(r,mid),res);

        if(r>mid) getmax(tr,i\*2+2,mid+1,end,max(l,mid+1),r,res);

        return ;

    }

# 概率期望问题

<https://blog.csdn.net/hutwuguangrong/article/details/102066794>

<https://blog.csdn.net/dreaming__ldx/article/details/84824173>

**codeforce 28C bath que**

**n个人，m个食堂，食堂里ai条队伍，队伍人数均分；求最长队伍的期望值。**

**Dp[i][j][k]: i个人，j个澡堂，最长队伍长度为k对应的概率**

**转移方程：**

## codeforces 442B

## codeforces 515D

## codeforces 148D

## codeforces 846F

## codeforces 678E

## codeforces 540D

## codeforces 912D

## codeforces 859D

# 容斥原理&组合数学

# 笔试细节问题

1.头文件包含

#include <bits/stdc++.h> //包含了std所有

using namespace std;

2. 当不知道cin数据个数怎么读进vector？



Cout内容会被cin吗？