缘起

- 刷题的链接，https://blog.csdn.net/goldarmour/article/details/129807099?spm=1001.2014.3001.5502
- 2023-10-28花时间整理了一下
- 感觉**滑窗**不难，但其实自己还不太难做得出来。

内容

-

题量比较大的（滑窗、自定义排序、字符串、逻辑分析）

字符串

- 03敏感字段加密
    - *好像并不能，case全过，要再check一下*

```
void od03_impl(int k,string input) {
    string d = input;
    while (k) {
        size_t pos = d.find("_");
        k--;
        d = d.substr(pos+1, d.size());
    }
    //cout << d << endl;
    size_t pos = d.find("_");
    string replace = d.substr(0, pos);

    pos = input.find(replace);
    string first = input.substr(0, pos);
    string last = input.substr(pos + replace.size(), input.size());
    string res = first + "******" + last;
    cout << res << endl;
}
```

- 07IPv4地址转换为整数

```
string num2hex(int num) {
    std::ostringstream ss;
    if (num >= 0 && num <= 255) {
        ss << std::hex <<setw(2) <<setfill('0') << num;//没有头文件iomainip，所以不能用setw
        //string tmp = ss.str();
    }
    return ss.str();
}

void od07_impl(string input) {
    vector<int> data;
```

```
    string res;
    int start = 0;
    for (int i = 0; i < input.size(); i++) {
        if (input[i] == '#') {
            //cout << start <<"," << i << endl;
            data.emplace_back(stoi(input.substr(start, i)));
            start = i+1;//lionel，这个地方写错了，应该是i+1是start，调试出来的
        }
    }
    data.emplace_back(stoi(input.substr(start, input.size())));
    for (auto i : data) {
        res += num2hex(i);
    }
    string newres("0x");
    newres += res;
    //cout << newres << endl;
    cout << stoll(newres, NULL, 16) << endl;//lionel，要stoll才行，stoi和stol都越界了
}
```

- 16最远足迹
    - *感觉是不是想复杂了啊*

```
bool cmp16(pair<int, int> m1, pair<int, int> m2) {
    int sum1 = (m1.first - 0) * (m1.second - 0);
    int sum2 = (m2.first - 0) * (m2.second - 0);

    if (sum1 > sum2) {
        return 1;
    }
    return 0;
}

void od16_impl(string input) {
    vector <pair<int, int>> res;
    //lionel，用啥去过滤，先找(，然后固定长度即可

    //查找所有的话，用while
    char flagL = '(';
    char flagR = ')';
    int pos = 0;
    int i=0;
    vector<int> left;
    while ((pos = input.find(flagL, pos)) != string::npos) {
        left.emplace_back(pos);
        pos++;
        i++;
    }

    vector<int> right;
    pos = 0;
    i = 0;
    while ((pos = input.find(flagR, pos)) != string::npos) {
        right.emplace_back(pos);
        pos++;
```

```
        i++;
    }

    vector<string> val;
    //lionel，这个风险就在左右如果不匹配怎么办？
    for (int i = 0; i < right.size(); i++) {
        string tmp = input.substr(left[i] + 1, right[i] - left[i] - 1);
        val.emplace_back(tmp);
    }

    for (auto s : val) {
        auto pos = s.find(',');
        int left = stoi(s.substr(0, pos-0)); //substr的用法是啥，最后一个是长度
        int right = stoi(s.substr(pos+1, s.size()-pos-1));
        //cout << left << "," << right << endl;
        res.emplace_back(make_pair(left, right));
    }

    sort(res.begin(), res.end(), cmp16);

    //lionel，我觉得更简单的方式用栈即可
    cout << res[0].first << "," << res[0].second << endl;
}
```

- 17需要打开多少监控器
  - *这个自己不会*，最开始没啥思路，有点dfs的意思，但又没这么写，*这个学一下*
  - 参数为啥要多个(0,0)，*当然，这个是实现的区别，关键还在于思想*
  - *这个怎么算字符串的了*，也算了**逻辑分析**的，我更觉得是后者

```
void od17_impl(vector<vector<int>> input, int m, int n) {
    vector<vector<int>> park(m + 2, vector<int>(n + 2, 0));
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            park[i][j] = input[i - 1][j - 1];
        }
    }

    int dx[5] = { -1,0,1,0,0 };
    int dy[5] = { 0,-1,0,1,0 };//lionel，为啥是5呢？

    for (int i = 1; i <m; i++) {
        for (int j = 1; j <n; j++) {
            if (park[i][j] == 1) {
                for (int k = 0; k < 5; k++) {
                    int nx = i + dx[k];
                    int ny = j + dy[k];

                    if (nx >= 1 && nx <= m && ny >= 1 && ny <= n) {
                        park[nx][ny] = 1;
                    }
                }
            }
        }
    }
```

```
    int res = 0;
    for (int i = 1; i <=m; i++) { //lionl，啥时候要等于，啥时候要不等于，这是个问题，要看下
        for (int j = 1; j <= n; j++) {
            if (park[i][j] == 1)
                res++;
        }
    }

    cout << res << endl;
}
```

- 19求字符串中所有整数的最小和
    - 感觉最开始也没做出来，*最后也没debug出来啊*，最后sum前面的if里面，没写内容。

```
//bb1234aa,输出1+2+3+4=10
void od19_impl(string val) {
    int sum = 0;
    int first = 0, last = 0;
    bool minFlag = false;
    for (int i = 0; i < val.size(); i++) {
        if (val[i] == '+') {
            break;
        }
        if (val[i] == '-') {
            minFlag = true;
            first = i;
        }
        if (isdigit(val[i])) {
            if()
            sum += (val[i] - '0');
        }
    }
    cout << sum << endl;
}
```

- 21TLV解析
    - 我没做出来，*还得再学下*，**核心是，定义了一个`map<int, vector<int>> tagOffset;`结构体**

```
void od21() {
    string s = "0F04ABABABAB";
    //getline(cin, s);
    int n = 1;
    int tag = 15;
    //int n=2,tag就有2个值，15，17

    //lionel，不会，用map来记录需要匹配的tag和其解析后的[起始点，结束点]
    map<int, vector<int>> tagOffset;
    for (int i = 0; i < n; i++) {
        vector<int> tmp = { 0,0 };
        tagOffset[tag] = tmp;
```

```
    }

    int bound = s.size();
    int left = 0;
    int right = 2;
    int offset = 2;
    while (left < bound) {  //lionel,这个地方，作者写的是left<=bound，我改成了
left<bound
        //一开始匹配的就是tag
        //https://blog.csdn.net/ma_minmin/article/details/126908825
[stoi(tmp,0,16)]
        string tmp = s.substr(left,
right);//https://blog.csdn.net/zjl2222/article/details/112135497  [C++的16进制，转
在10进制]
        int tag = stoi(tmp,0,16);  //lionel，get到了这个

        int length = stoi(s.substr(left + 2, right + 2));

        //判断tag是否匹配
        if (tagOffset.find(tag) != tagOffset.end() && ((offset + length) * 2 <=
bound)) {  //lionel，这个地方，作者写的是<bound，我改成<=了
            tagOffset[tag] = { length,offset };
        }

        //根据长度来更新偏移量
        offset = offset + length + 2;
        left += (length * 2) + 4;
        right += (length * 2) + 4;
    }

    for (auto c : tagOffset) {
        cout << c.second[0] << " " << c.second[1] << endl;
    }
}
```

- 22拼接URL
    - 输入`string input1 = "acm,/bb";`，输出`/acm/bb`

```
//用,分割
string check(string value) {
    string str;
    if (value[0] != '/')
        str = "/" + value;
    if (value[value.size()] == '/' && value[value.size() - 1] == '/')
        str = value.substr(0, value.size() - 1);

    if (value[value.size()] == '/') {
        str = value.substr(0, value.size() - 1);
    }

    if (str.size()>0) {
        return str;
    }
    return value;
}
```

```
void od22_impl(string input) {
    size_t pos = input.find(",");
    if (pos != input.npos) {
        string first = input.substr(0, pos);
        string last = input.substr(pos+1, input.size());
        string a = check(first);
        string b = check(last);
        string tmp = check(first) + check(last);
        cout << tmp << endl;
    }
}
```

- 42寻找相同子串
  - 直接用 `find` 或 `find_first_of()` 是不是不行?
  - **扩展学一下KMP吧**

```
void od42_impl(string source, string target) {
    //auto res = find_first_of(source.begin(), source.end(), target.begin(),
target.end());  //lionel，用法不对
    cout << source.find_first_of(target) + 1 << endl;  //lionel，这个地方要加1吗？
    //cout << *res << endl;
}
```

- 44报文重排序
  - 自己当时2个思路，用vector存结构体，用map
  - **把字符串拆成字符串和数字**（用的正则，其实可以string从后往前）
  - 当时想的是`map()`是不是就默认按key排序，自己定义的`cmp`，其实还没有用上
    - **C++中map的默认排序，就是从小到大**，https://www.zhihu.com/question/518791037，*当然也可以自己指定*
    - `map<string, int, cmp> strInt;`可以这样定义

```
//rolling3 stone4 like1 a2  输出:like a rolling stone
struct cmp {
    bool operator()(const int &num1, const int &num2) const {  //lionel，这个都需要
const，不管是参数，还是 函数，都要是const  【这个还是会报错，const _Tky转成const int】
        return num1 > num2;
    }
};

void od44_impl(int num, vector<string>input) {
    //map<string, int, cmp> strInt;  //我写成这样，当然也是可以的
    map<int,string> intStr;//lionel，相当于自定义了一个排序规则
    for (auto c : input) {
        string s1 = c;
        string s2 = c;
        regex reg1 ("[0-9]") ;
        regex reg2("[a-zA-Z]") ;
        string value = regex_replace(s1, reg1, "");
        string num = regex_replace(s2, reg2, "");
        //cout << value << "," << num << endl;
        //strInt.insert(make_pair(value, stoi(num)));
```

```
        intStr.insert(make_pair(stoi(num), value));
    }

    string res;
    //for (auto c : strInt) {
    for (auto c : intStr) {
        res += c.second;
        res += " ";
    }
    cout << res << endl;
}
```

- 59最长公共后缀
- 66计算最大乘积
    - 目前是逐个比较，用的是2层for循环
    - *但应该还有更好的方法*，比如看到了用`set()`，但我没看懂
    - *lionel，判断，集合没有交集，两个子字符串没有重复元素*

```
int mul(string s1, string s2) {
    for (int i = 0; i < s1.size(); i++) {
        for (int j = 0; j < s2.size(); j++) {
            if (s1[i] == s2[j]) {
                return 0;
            }
        }
    }
    return s1.size() * s2.size();
}

void od66_impl(vector<string> inputs) {
    int res=0;
    for (int i = 0; i < inputs.size() - 1; i++) {
        res = max(res, mul(inputs[i], inputs[i + 1]));
    }
    cout << res << endl;
}
```

- 73按单词下标区间翻转文章内容
    - 最开始不需要用正则，转化为acm模式下的输入输出即可

```
//string inputs1{ "I am a developer" };,start1是1和2
void od73_impl(string input, int start, int end) {
    regex ws_re("\\s+");//lionel，不是太明白这个正则
    vector<string> res_helper(sregex_token_iterator(input.begin(), input.end(),
ws_re, -1), sregex_token_iterator());

    while(start<end){
        swap(res_helper[start], res_helper[end]);
        start++;
        end--;
    }
```

```
    string res;
    for (auto val : res_helper) {
        res += val;
        res += " ";
    }
    cout << res << endl;
}
```

- 84关联子串
    - *这个题，不太会啊*【也是抄的】
    - *看用到了回溯算法，但解题没用这种*

```
//input:  abc; efghicbaiii  Ouput: 5  【相当于abc的全排列，然后找位置？】
void od84_impl(string s1, string s2) {
    //求s1的全排列,使用next_permutation
    vector<char> res_helper;
    vector<char> res;
    vector<vector<char>> resout;
    res_helper.assign(s1.begin(), s1.end());
    sort(res_helper.begin(), res_helper.end());
    do {
        for (int i = 0; i < res_helper.size(); i++) {
            //cout << res_helper[i];
            res.emplace_back(res_helper[i]);
        }

    } while (next_permutation(res_helper.begin(), res_helper.end()));
    resout.emplace_back(res);

    vector<string> sVect;
    for (auto h : resout) {
        string s{ h.begin(), h.end() };
        for (int i = 0; i < s.size(); i+=3) {
            sVect.emplace_back(s.substr(i, 3)); //lionel,substr的用法，还是不熟啊
        }
    }

    for (auto c : sVect) {
        //cout << c << endl;
        auto pos = s2.find(c);
        if (pos != s2.npos) {
            cout << pos << endl;
        }
        //auto it = find(s2.begin(), s2.end(), c);//lionel，太习惯用find算法后，不太会用string里面的find了
        //if (it != s2.end())
            //cout << *it << endl;
    }
}
```

- 88一种字符串压缩表示的解压

- *不一定是最优的*

```
//In:4dff，Out：  ddddff
void od88_impl(string input) {
    vector<char> res;
    for (int i = 0; i < input.size(); i++) {
        if (isdigit(input[i])) {
            if (input[i] - '0' <= 2) {
                //cout << input[i] - '0' << endl;
                cout << "!error" << endl;
                break;
            } else {
                int num = input[i] - '0';
                while (num-1 > 0) {
                    res.push_back(input[i + 1]);
                    num--;
                }
            }
        }else if (isalpha(input[i])) {
            res.push_back(input[i]);
        } else {
            cout << "!error" << endl;
            break;
        }
    }

    //cout << res << endl;
    string str(res.begin(), res.end());
//https://blog.csdn.net/liuzehn/article/details/107606163  把vector<char>转成string
    cout << str << endl;  //lionel，写了还不够健壮，但意思到了
}
```

- 90单词重量

```
void od90_impl(string input) {
    //https://zhuanlan.zhihu.com/p/426939341  正则，我是没想到的，字符串split，用find的话，要不断循环
    std::string s;
    char delim = ' ';
    s.append(1, delim);
    std::regex reg(s);
    std::vector<std::string> elems(std::sregex_token_iterator(input.begin(), input.end(), reg, -1),
        std::sregex_token_iterator());

    vector<int> res;
    for (auto c : elems) {
        res.emplace_back(c.size());
    }

    int sum = 0.0;
    sum = accumulate(res.begin(), res.end(), sum);
```

```
    cout << fixed<<setprecision(2)<<(sum*1.0) / res.size() << endl;
//https://blog.csdn.net/abilix_tony/article/details/106956204  lionel，2个处问题，setprecison(2)和fixed不太熟悉
}
```

字符串（undo-2）

- 69响应报文时间
- 87相对开音节
- 93

自定义排序

- 09组成最大数
    - **难点，在于组成最大数的规则**，《剑指offer》上有这个理论
        - 换成string后，s1+s2，s2+s1进行比较即可

```
bool cmp(int n1, int n2) {   // lionel，这个比较规则，是我没有想到的
    string s1 = to_string(n1);
    string s2 = to_string(n2);

    string tmp1 = s1 + s2;
    string tmp2 = s2 + s1;
    if (tmp1 < tmp2) {
        return 0;
    }

    return 1;
}

//In: 4589,101,41425,9999
//Out:9999 4589 41425 101
void od09_impl(vector<int> inputs) {
    sort(inputs.begin(), inputs.end(), cmp);
    string s;
    for (auto c : inputs) {
        s += to_string(c);
    }
    cout << stoll(s) << endl; //lionel，是不是要考虑个越界的问题？
}
```

- 13选修课
    - *自己感觉也没弄明白*
    - 属于**栈**，但我没用这个思想

```
bool cmp13(string s1, string s2) {
    //http://c.biancheng.net/view/1447.html compare函数
    int m = s1.compare(0, 5, s2, 0, 5);
    if (m > 1)
        return 1;
    else {
```

```
        // m==0，班级名称一样
        return 0;
    }

}

void od13_impl(vector<pair<string, int>> input1, vector<pair<string, int>>
input2) {
    vector<string> res;
    //同时选修了2门，就是在1，和2都有一样的学号
    for (int i = 0; i < input1.size(); i++) {
        //cout << input1[i].first << endl;
        for (int j = 0; j < input2.size(); j++) {
            if (input1[i].first == input2[j].first) {
                //cout << input1[i].first << endl;
                res.emplace_back(input1[i].first);
            }
        }
    }
    sort(res.begin(), res.end(), cmp13);

    for (int i = 0; i < res.size()-1; i++) {
        if (res[i].compare(0, 5, res[i + 1], 0, 5) == 0) {
            cout << res[i].substr(0, 5) << endl;
            cout << res[i] <<"," << res[i + 1] << endl;
        } else {
            cout << res[i].substr(0, 5)<<endl;
            cout << res[i] << endl;
        }
    }
}
```

- 28按身高和体重排队
    - **核心是，定义结构体，【同时加上index字段】**，中间的`equal()`函数没啥用吧

```
typedef struct person {
    int index;  //lionel，自定义排序，核心还是加个索引
    int height;
    int weight;
}Person;

bool comp(Person p1, Person p2) {
    if (p1.height < p2.height) {
        return 1;
    } else if (p1.height == p2.height && p1.weight < p2.weight) {
        return 1;
    }
    return 0;
}

bool equal(Person p1, Person p2) {
    if (p1.height == p2.height && p1.weight == p2.weight) {
        return 1;
    }
    return 0;
```

```
}

void od28() {
    //把输入的2行当成结构体好了
    Person p[4] = { {1,100,40},{2,100,30},{3,120,60},{4,130,50} };
    vector<Person> vp;
    vp.emplace_back(p[0]);
    vp.emplace_back(p[1]);
    vp.emplace_back(p[2]);
    vp.emplace_back(p[3]);
    vector<Person> old_vp{ vp };
    vector<int> num;

    sort(vp.begin(), vp.end(),comp);  //lionel，问题是，如何输出序号？

    for (auto c : vp) {
        cout << c.index << endl;
    }
}
```

- 32寻找身高相近的小朋友
    - 当时写的思考是，*用map存与vector存，有啥区别?*

```
int g_height = 0;
bool cmp32(pair<int, int> m1, pair<int, int> m2) {
    int diff1 = m1.second - g_height;
    int diff2 = m2.second - g_height;
    if (abs(diff1) < abs(diff2)) {
        return 1;
    } else if (abs(diff1) == abs(diff2)) {
        if (m1.first < m2.first)
            return 1;
    }
    return 0;
}

void od32_impl(int height, int count, vector<int> inputs) {
    g_height = height;
    vector<pair<int, int>> res;
    for (int i = 0; i < count; i++) {
        res.emplace_back(make_pair(i, inputs[i]));
    }

    sort(res.begin(), res.end(), cmp32);

    for (auto c : res) {
        cout << c.second << endl;
    }
}
```

- 34数组组成的最小数字
    - **核心是全排列算法**

```
bool cmp(int i1, int i2) {//如果lambda的话，我不知道sort里面的参数是啥  ,【排序是对的，排出来的是，5，21，30，然后组合】
    //cout << i1 << "input," << i2<<endl;
    string s1 = to_string(i1);
    string s2 = to_string(i2);
    if (s1.size() < s2.size()) {
        return 1;
    } else if (s1.size() == s2.size()) {
        for (int i = 0; i < s1.size(); i++) {
            //cout << s1[i] << "," << s2[i] << endl;
            if (s1[i] < s2[i]) {
                return 1;
            } else
            {
                return 0;
            }
        }
    }
    return 0;
}

void od34_impl(vector<int> inputs) {
    while(inputs.size() < 3) {
        inputs.emplace_back(0);
    }
    sort(inputs.begin(), inputs.end(), cmp);  //lionel，sort的用法
    //for (auto c : inputs) {
    //  cout << c << endl;
    //}
    string s1 = to_string(inputs[0]);
    string s2 = to_string(inputs[1]);
    string s3 = to_string(inputs[2]);

    //回到另外的问题上，3个string，如果全排列
    vector<string> v1{ s1,s2,s3 };
    sort(v1.begin(), v1.end());
    vector<string>v2;
    do {
        //cout << v1 << endl;
        v2.push_back(v1[0] + v1[1] + v1[2]);
    } while (next_permutation(v1.begin(), v1.end()));


    cout << stoi(v2[0]) << endl;
}
```

- 36数组去重和排序
  - **重复文件如何插入map**
    - 1、用pair类型，`map.insert(make_pair(1,1));map.insert(make_pair(1,2));`
    - 2、用下标，`map[1]=2;map[1]=3;`
  - pair的头文件是`#include<utility>`

```
bool cmp36(pair<int, pair<int, int>> m1, pair<int, pair<int, int>> m2) {
```

```
    if (m1.second.second > m2.second.second)
        return 1;
    else if (m1.second.second == m2.second.second) {
        if (m2.first > m1.first)
            return 1;
    }
    return 0;
}


void od36_impl(vector<int> inputs) {
    unordered_map<int, int> val;  //因为存的时候，数据顺序变掉了，把map改成了unordered_map，lionel
    for (int i = 0; i < inputs.size(); i++)
        val[inputs[i]]++;

    vector<pair<int, pair<int, int>>> res1;  //lionel，用双层给解决了一下

    vector<pair<int, int>> res{ val.begin(),val.end() };  //lionel，当时不知道不知道map里面是不能自定义排序的，要用vector去保存一下
    for (int i = 0; i < val.size(); i++) {
        //int i = val[i].second();
        pair<int, int> tmp = res[i];
        pair<int, pair<int, int>> tmp2 = make_pair(i, tmp);
        //res1[i] = tmp;
        //res1.insert(i, tmp);
        res1.emplace_back(tmp2);
    }

    sort(res1.begin(), res1.end(),cmp36);
    for (auto c : res1) {
        cout << c.second.first << endl;
    }
}
```

- 47字符串摘要
    - *自己不太会，估计也是抄的*，题也没全部做完，*连续的怎么办?*
    - 算的是**字符串、滑窗**的两类

```
bool cmp47(pair<char, int> m1, pair<char, int> m2) {
    if (m1.second > m2.second)
        return 1;
    return 0;
}

//aabbcc，输出a2b2c2
void od47_impl(string input) {
    unordered_map<char, int> res_helper;
    for (int i = 0; i < input.size(); i++) {
        //lionel，这个要连续的，所以需要滑动窗口，这个没写，包括大小写算连续，非字符要break掉
        res_helper[input[i]]++;
    }

    vector<pair<char, int>> res{ res_helper.begin(),res_helper.end() };
```

```
    sort(res.begin(), res.end(),cmp47);

    string result;
    for (auto c : res) {
        //result.push_back(c.first);
        result += c.first;
        result += to_string(c.second);
    }
    cout << result<<endl;
}
```

- 55拔河比赛
    - **核心在于，定义一个结构体**

```
typedef struct person {
    int heigth;
    int weight;
}Person;

bool cmp(Person p1, Person p2) {
    if (p1.heigth > p2.heigth) {
        return 1;
    } else if (p1.heigth == p2.heigth) {
        if (p1.weight > p2.weight) {
            return 1;
        }
    }

    return 0;
}
void od55_impl(vector<vector<int>> inputs) {
    vector<Person> res;
    for (auto c : inputs) {
        Person tmp;
        tmp.heigth = c[0];
        tmp.weight = c[1];
        res.push_back(tmp);
    }
    sort(res.begin(), res.end(), cmp);

    for (int i = 0; i < 10; i++) {
        cout << res[i].heigth << "," << res[i].weight << endl;
    }
}
```

- 60支持优先级的队列
    - **结构体中增加一个index下标**，用for的时候去把i的值赋值给它即可

```
typedef struct que {
    int index;
    int value;
    int weight;
```

```
}Que;

bool cmp(Que q1, Que q2) {
    if (q2.weight > q1.weight) {
        return 0;
    } else if (q2.weight == q1.weight) {
        if (q2.index < q1.index)
            return 0;
    }
    return 1;
}

void od60_impl() {
    //lionel，我是没想好，怎么处理输入  【2023-10-28,for的时候，用i赋值给index】
    vector<Que> v1{ {1, 10, 1}, { 2,20,1 }, { 3,30,2 }, { 4,40,3 }};  //输出40，
30，10，20
    sort(v1.begin(), v1.end(), cmp);

    for (auto c : v1) {
        cout << c.value << endl;
    }
}
```

- 77奖牌排行榜
    - 用结构体，用**字符串的compare方法**

```
struct prize {
    string nation;
    int gold;
    int silver;
    int bronze;
};

bool cmp77(struct prize p1, struct prize p2) {
    if (p1.gold > p2.gold) {
        return 1;
    } else if (p1.gold == p2.gold && p1.silver > p2.silver) {
        return 1;
    } else if (p1.silver == p2.silver && p1.bronze > p2.bronze) {
        return 1;
    } else if (p1.bronze == p2.bronze) {
        if (p1.nation.compare(p2.nation) > 1)
            return 1;
    }
    //lionel,第一遍（0912）写的时候，cmp写的有问题，这时用结构体存的，我0922再写的时候，想到
可以用vector<pair<string,vector<int>>>这样的形式，

    return 0;
}

void od77_impl(int n, vector<prize> input) {
    sort(input.begin(), input.end(), cmp77);
    for (int i = 0; i < 3; i++) {
        cout << input[i].nation << endl;
    }
```

```
}
```

- 79数字最低位排序
    - 多了一个函数，计算**最低位**，要再看一下

```
int lowbit(int num) {
    num = abs(num);  //lionel，最开始用了abs(num)，没用num去接
    string tmp = to_string(num);
    int len = tmp.size();
    while (num > 10) {
        int val = pow(10, len - 1);
        len--;
        num -= (num / val) * val;
    }
    return num;
}

bool cmp79(pair<int, int> m1, pair<int, int> m2) {
    //取最低位
    int num1 = lowbit(m1.second);
    int num2 = lowbit(m2.second);
    //cout << num1 << "," << num2 << endl;
    if (num1 < num2) {
        return 1;
    } else if (num1 == num2) {
        if (m1.first > m2.first) {
            return 0;
        } else
            return 1;
    }
    return 0;
}

void od79_impl(vector<int> input) {
    vector<pair<int, int>> res;
    for (int i = 0; i < input.size(); i++) {
        res.emplace_back(make_pair(i, input[i]));
    }

    sort(res.begin(), res.end(), cmp79);

    for (auto c : res) {
        cout << c.second << endl;
    }
}
```

- 86统计射击成绩
    - *也不太确定，最终有没有做出来了*

```
bool cmp86(pair<int, int> m1, pair<int, int> m2) {
    if (m1.second > m2.second)
        return 1;
    else if (m1.second == m2.second && m1.first>m2.first) {
        return 1;
    }
```

```
    return 0;
}

void od86_impl(int count, vector<int> player, vector<int> score) {
    vector<pair<int, int>>res_helper;
    for (int i = 0; i < count; i++) {
        res_helper.emplace_back(make_pair(player[i], score[i]));
    }

    //先统计次数
    map<int, int> res_map;
    for (auto c : player) {
        res_map[c]++;
    }

    vector<pair<int, int>> res;
    vector<pair<int, int>> res_vector(res_map.begin(), res_map.end());
    for (auto c : res_vector) {
        if (c.second < 3) {
            //res_vector.erase(c.first);
            cout << c.first << endl;  //lionel，如何取消，这个地方应该遗留了一点
        }

        int sum = 0;
        int val = c.first;
        vector<int> top3;
        for (int i = 0; i < res_helper.size(); i++) {
            if (val == res_helper[i].first) {
                //sum += res_helper[i].second;
                top3.emplace_back(res_helper[i].second);
            }
        }
        sort(top3.begin(), top3.end());
        int len = top3.size();
        sum += top3[len - 1];
        sum += top3[len - 2];
        sum += top3[len - 3];  //lionel，第一次忘记算3的成绩了
        res.emplace_back(make_pair(val, sum));
    }

    sort(res.begin(), res.end(),cmp86);

    for (auto c : res) {
        cout << c.first << endl;
    }
}
```

自定义排序（undo-1）

- 74比赛

滑窗、单双指针

- 20求满足条件的最长子串长度
  - *这也是抄的，其实没弄懂*
  - 大概猜了一下，**用滑动窗口，找到值，然后check一下，符合条件的，求出size，更新上去**

```
//只含(a-z,A-Z)，其它是数字，   abC124ACb，输出是4（abC1,4Acb）
bool check(string str) {
    regex r("[0-9]");
    string replace = regex_replace(str,r, "");
    //cout << str << "," << replace << endl;
    /*
    *
    ab, ab
    bC1, bC
    C124, C
C124A, CA
124ACb, ACb
24ACb, ACb
4ACb, ACb
ACb, ACb
    */
    return replace.size() != str.size() && replace.size() <= 1; //lionel，这个也没
懂
}

void od20_impl(string input) {
    int left = 0, right = 1;
    int res = -1;
    while (left < input.size() && right < input.size()) {
        right++;
        string subStr = input.substr(left, right); //lionel，求这个啥意思
        //cout << subStr << endl;
        if (check(subStr)) {
            res = max(res, (int)subStr.size());
        } else {
            left++;
        }
    }
    cout << res << endl;
}
```

- 40滑动窗口最大和
  - 每次的窗口是k

```
void od40_impl(vector<int> inputs, int k) {
    int res = 0;
    int first = 0, last = first + k;
    while (first < last) {
        if (last > inputs.size()) {
            break;
        }

        int sum = 0;
```

```
        sum = accumulate(inputs.begin() + first, inputs.begin() + last, sum);
        res = max(res, sum);
        first++; //lionel，会忘记这个++
        last++;
    }
    cout << res << endl;
}
```

- 45阿里巴巴找宝箱V
    - *跟40题是一样的?*

```
void od45_impl(vector<int> inputs, int k) {
    int res = 0;
    int first = 0, last = first + k;
    while (first < last) {
        if (last > inputs.size()) {
            break;
        }

        int sum = 0;
        sum = accumulate(inputs.begin() + first, inputs.begin() + last, sum);
        res = max(res, sum);
        first++; //lionel，会忘记这个++
        last++;
    }
    cout << res << endl;
}
```

- 这4道是单双指针
- 04阿里巴巴找黄金宝箱I

```
void od04_impl(vector<int> nums) {
    for (int i = 0; i < nums.size(); i++) {
        int first = accumulate(nums.begin(), nums.begin() + i,0);
        int last = accumulate(nums.begin()+i+1, nums.end(), 0);
        //cout << first << "," << last << endl;
        if (first == last) {
            cout << i << endl;
        }
    }
    //不然，返回-1
}
```

- 10最大花费金额
    - **类似于3数之和**（第一层for，然后再用2数之和的方式）

```
void od10_impl(vector<int> input, int target) {
    //09-12重新想了一下，类似于three sum的写法
    vector<int>res;
```

```
    sort(input.begin(), input.end());
    for (int i = 0; i < input.size(); i++) {
        int first = i + 1;
        int last = input.size() - 1;

        target -= input[i];

        while (first < last) {
            if (input[first] + input[last] < target) {
                res.emplace_back(input[first] + input[last] + input[i]);
                first++;//lionel，这个顺序写错了，应该先emplace_back，然后再first++;
            } else {
                last--;
            }
        }
    }

    sort(res.begin(), res.end());
    for (auto c : res) {
        cout << c << " ";
    }
    cout << res[res.size()-1] << endl;
}
```

- 11太阳能面板最大面积
  - 有2个点
    - 1是**哪个值小，移左，还是右**
    - 2是**每次去max一下（res,area)**
  - LC84柱状图中的最大矩形

```
void od11_impl(vector<int> input) {
    int res = 0;
    int first = 0, last = input.size() - 1;
    while (first < last) {
        //lionel，怎么移呢，【琢磨出来了，哪个短，哪个移
        int tmp = min(input[first], input[last]);
        //cout << tmp << endl;
        if (tmp == input[first]) {
            first++;
        }
        if (tmp == input[last]) {
            last--;
        }
        int area = (last - first + 1) * tmp; //lionel，这个地方是不是得能+1才行
        res = max(res, area);
    }
    cout << res << endl;
}
```

- 35求最多可派出多少支团队
  - 排序，大于能力的可以组队，最多2个人一组，2个之和大于能力也算一支

```
void od35_impl(int sum, vector<int> input, int ability) {
    sort(input.begin(), input.end());
    int left = 0, right = input.size() - 1;
    int res = 0;
    while (left < right) {
        if (input[right] >= ability) {
            res++;
            right--;
        } else {
            if (input[right] + input[left] >= ability) {
                res++;
                right--;
                left++;
            } else
            {
                left++;
            }
        }
    }
    cout << res << endl;
}
```

- 下面这3道，只是没写
- 81
- 82
- 91

滑窗、单双指针（undo-4）

- 41
- 46
- 51
- 56

逻辑分析

动规：27

回溯：54

栈：76

- 06斗地主之顺子
    - 确实是逻辑题，但**难点在于，至少5张怎么判断**，用了一个`vector.back()`

```
int Str2Int(string val) {
    if (val == "J")
        return 11;
    else if (val == "Q")
        return 12;
    else if (val == "K")
        return 13;
    else if (val == "A")
        return 14;
    else
        return stoi(val);
}
```

```
string Int2Str(int num) {
    switch (num){
    case 11:
        return "J";
    case 12:
        return "Q";
    case 13:
        return "K";
    case 14:
        return "A";
    default:
        return to_string(num);
    }
}

void od06() {
    vector<int> nums;
    vector<string> inputs{ "2","9","J","10","3","4","K","A","7","Q","A","5","6"
};
    for (auto c : inputs) {
        int val = Str2Int(c);
        if (val > 2) {
            nums.push_back(val);
        }
    }

    sort(nums.begin(), nums.end());

    vector<vector<int>> res;
    vector<int> path;
    while (nums.size() >= 5) {
        path.push_back(nums[0]);
        nums.erase(nums.begin());

        for (int i = 0; i < nums.size(); i++) {
            if (nums[i] == path.back()) {  //lionel，这个没懂
                continue;
            }

            //lionel，我就这个地方没懂，根因是啥？
            if (nums[i] == path.back() + 1) {  //lionel，还是用了这个规则，back()，我
也是搜的
                path.push_back(nums[i]);
                nums.erase(nums.begin() + i);
                i--; //i--是表示nums[i]的下标
            }
        }

        //表示一轮结束，存一下
        if (path.size() >= 5) {
            res.push_back(path);
        }
        path.clear();
    }

    if (res.empty()) {
        cout << "No" << endl;
```

```
        return;
    }

    sort(res.begin(), res.end());
    for (int i = 0; i < res.size(); i++) {
        for (auto c : res[i]) {
            string val = Int2Str(c);
            cout << val << " ";
        }
        cout << endl;
    }

}
```

- 12坐位调整
    - *自己不会*，也可以归为**滑窗问题**
    - LC605种花问题

```
void od12_impl(vector<int> input) {
    int res = 0;
    int len = input.size();
    int prev = -1;
    for (int i = 0; i < len; i++) {
        if (input[i] == 1) {
            if (prev < 0) {
                res += i / 2;
            } else {
                res += (i - prev - 2) / 2;//lionel，没懂啥意思
            }
            prev = i; //lionel，为啥是放在里面？
        }
    }

    if (prev < 0) {
        res += (len + 1) / 2;
    } else {
        res += (len - prev + 1) / 2;
    }

    cout << res << endl;
}
```

- 27猴子爬山（done）
- 37快递运输
    - *没有全部AC*

```
void od37_impl(vector<int> input, int target) {
    sort(input.begin(), input.end());
    int first = 0, last = first + 1;
    int sum = 0;
    sum += input[first];
```

```
    int res = 0;
    while (first < last) {
        if (sum >= target) {
            res = (last - first)-1;
            break;
        }

        sum += input[last];
        last++;
    }
    cout << res << endl;
}
```

- 38停车场车辆统计
    - *lionel，应该也是看的*，自己没做出来
    - **核心是用replace替换**

```
string subString(string src, string oldStr, string newStr) {
    string dest_str = src;
    string::size_type pos = 0;
    while ((pos = dest_str.find(oldStr)) != string::npos) {
        dest_str.replace(pos, oldStr.size(), newStr);
    }
    return dest_str;
}

void od38_impl(vector<int> input) {
    string res_helper;
    for (auto n : input) {
        res_helper += to_string(n);
    }

    string tmp1 = subString(res_helper, "111", "x");
    string tmp2 = subString(tmp1, "11", "x");
    string tmp3 = subString(tmp2, "1", "x");
    int res = 0;
    for (auto c : tmp3) {
        if (c == 'x')
            res++;
    }

    cout << res << endl;
}
```

- 52分割数组最大差值
    - 也是逻辑做的，就是暴力

```
void od52_impl(int num, vector<int> input) {
    int total = 0;
    total = accumulate(input.begin(), input.end(), total);

    int res = 0;
```

```
    int sum = 0;

    for (int i = 0; i < input.size(); i++) {
        sum += input[i];
        int left = sum, right = total - sum;
        res = max(res, abs(right - left));
    }
    cout << res << endl;
}
```

- 54保密电梯（done）
  - *这是抄的*，自己并没有完全消化
- 63数字涂色
  - *不一定对*，重新学一下

```
void od63_impl(int n, vector<int> inputs) {
    vector<int> res;
    int min = inputs[0];
    res.push_back(min);
    for (int i = 1; i < inputs.size(); i++) {
        if (inputs[i] % min != 0) {
            res.push_back(inputs[i]);
        } else
        {
            break;
        }

    }
    cout << res.size() << endl;
}
```

- 70食堂供餐

```
//hour是开餐开长，count是盒饭份数，person是排队人数
void od70_impl(int hour, int count, vector<int> person) {
    int res = 0;
    int sum = 0;
    sum = accumulate(person.begin(), person.end(), sum);  //lionel，这个sum，还得给个返回值来接一下，不然sum还是0

    while ((hour - 1)*res + count < sum) {
        res++;
    }
    cout << res << endl;
}
```

- 76荒岛求生

```
void od76_impl(vector<int> input) {
```

```
    stack<int> s;
    int res = 0;
    for (auto c : input) {
        if (c > 0) {
            s.push(c);
        } else if (c < 0) {  //lionel，现在是正数多一个，如果是负数多一个怎么办？s.top()取不到值，但c还有，也是要++的，这个地方不够健壮，要判断一下 s.top()
            if (s.top() + c != 0) {
                res++;
            }
            s.pop();
        }
    }
    if (s.size() > 0)
        res += s.size();
    cout << res << endl;
}
```

- 78报数游戏
  - 是不是也属于约瑟夫环的一种?
  - *lionel，应该也没有做出来*

```
void od78_impl(int n) {
    //list<int> input;
    vector<int> input;
    for (int i = 1; i <= 100; i++)
        input.emplace_back(i);

    int index = 0;
    while (input.size() >= n) {
        index = (index + (n - 1)) % (input.size());  //lionel，核心还是这个公式，当然还是要用vector，当然用vector时，用erase(v.begin()+pos)
        //input.remove(index);
        input.erase(input.begin()+index);
        //cout << input.size() << endl;
    }

    for (auto c : input) {
        cout << c << endl;
    }
}
```

- 80CPU算力
  - 逻辑题
    - 除表示次数
    - 取余表示剩下还可以用

```
void od80_impl(vector<int> input, int taskNum) {
    int time = 0;
    int remain = 0;
    for (auto count : input) {
        if (count + remain > taskNum) {  //remain第一次只是顺带，后面的话，要代入带来处理的
```

```
            remain = count + remain - taskNum;
        } else {
            remain = 0;
        }
        time++;
    }

    time += remain / taskNum; //剩下的还要几次

    if (remain % taskNum > 0) //如果剩下的还有余下的，说明还需要一次
        time++;

    cout << time << endl;
}
```

- 83字符串序列II
    - *这个自己不会*

```
//判断 abc 是不是 abcayber的子序列，Out:3【最后一个子序列的起始位置】
void od83_impl(string target, string source) {
    //for (int i = target.size(); i > 0; i--) {  //lionel，本来想搞2层for循环的，发现可以优化，也是参考的
    int cur = target.size();
        for (int j = source.size(); j > 0; j--) {
            if (source[j] == target[cur]) {
                cur--;
                if (cur < 0)
                {
                    cout << j << endl;
                    break;
                }

            }
        }
    //}
}
```

- 85字符串变最小字符串
    - *也是抄的*

```
void od85() {
    string str = "bcdefa";//找到a,跟b换一下，输出acdefb

    string s2 = str;
    sort(s2.begin(), s2.end());

    char min = s2[0];
    int place = 0, change = 0;

    for (int i = 0; i < str.size(); i++) {
        if (str[i] == min) {//找到原串中最小的位置
            place = i;
            break;
        }
```

```
    }

    for (int j = 0; j < str.size(); j++) {
        if (str[j] != min) {  //这个地方取原串，未排序的
            change = j;//记下排序后，不等的地方
            break;
        }
    }

    if (place > change) {
        swap(str[place], str[change]);
    }

    cout << str << endl;
}
```

逻辑分析（undo-13）

- 14
- 18
- 26
- 31
- 48
- 49
- 61
- 67
- 68
- 75
- 89
- 92
- 94

重要看的（之前不会）

dfs

- 50文件目录大小
    - **这个难点，在于构造`file`结构体**，这确实是我没想到的，dfs的思想反而简单了些
    - *知道要有dfs()函数，便参数怎么确定，不太明白*
    - LC113：路径总和II，*是这个吗？* https://www.jianshu.com/p/c2e1851164eb
    - LC690：

```
//参考LC690
typedef struct f {
    int id;
    int size;
    vector<int> subordinates;
}file;

void dfs(int id, unordered_map<int, file>& um, int& sum) {
    for (auto id : um[id].subordinates) {
        sum += um[id].size;
        dfs(id, um, sum);
```

```
    }
}

void od_50_impl(int n, vector<file> input) {
    unordered_map<int, file> um;
    for (auto c : input) {
        um[c.id] = c;
    }

    int sum = um[n].size;
    dfs(n, um, sum);
    cout << sum << endl;
}

//lionel，相当于，我要把每一行放到 file 结构体里

void od50() {
    int m = 4;
    int n = 2;
    vector<int> tmp{ 4,5 };
    vector<int> emp;
    vector<file> inputs{ {4,20,emp},{5,30,emp},{2,10,tmp},{1,40,emp} };  //输出就是60
    od_50_impl(n, inputs);  //lionel，如何用指针？
}
```

递归

- 15分糖果
    - *自己做出来了*，但不确定是不是就是100分的答案

```
int od15_impl(int input) {
    int res = 0;
    if (input == 1) {
         res++;
         return res;//lionel，这个地方有问题，return res++（只是返回4），拆成2句就是5，**其实左加和右加还是有区别的**
    }

    if (input % 2) {
        input += 1;
    } else{
        input /= 2;
        res++;
    }

    cout << input << endl;
    res += od15_impl(input);
}
```

贪心

- 24最大股票收益

```
void od24_impl(vector<string> input) {
    vector<int> nums;
    string tmp;
    for (auto s : input) {
        int len = s.size();
        char last = s[len - 1];
        tmp = s.substr(0, len - 1);
        if (last == 'Y') {
            nums.emplace_back(stoi(tmp));
        }
        if (last == 'S') {
            nums.emplace_back(stoi(tmp) * 7);
        }
    }

    int sum = 0;
    int maxValue = 0;
    int first = nums[0];
    for (int i = 1; i < nums.size(); i++) {
        if (nums[i] - first > maxValue) {
            sum -= maxValue;
            maxValue = nums[i] - first;  //lionel，当时用了一个补丁方案，当i=len-1时，
sum+=maxValue，解决最后一个maxValue不能加的问题
            sum += maxValue;
        } else {
            maxValue = 0;
            first = nums[i];//lionel，第一版时，写在这的sum += maxValue;
        }
    }
    cout << sum << endl;
}
```

- 29阿里巴巴找黄金宝箱II

```
bool cmp(pair<int, int> m1, pair<int, int> m2) {
    if (m1.second > m2.second)
        return 1;
    return 0;
}

//1,1,1,1,3,3,3,6,6,8  Out:2
void od29_impl(vector<int> inputs) {
    map<int, int> val;
    for (int i = 0; i < inputs.size(); i++) {
        val[inputs[i]]++;
    }

    int len = inputs.size();
    vector<pair<int, int>> res{ val.begin(),val.end() };

    sort(res.begin(), res.end(), cmp);
```

```
    //cout << res[0].first << res[0].second << endl;
    int remain = len / 2 - 0;

    while (remain > 0) { //lionel，这个是临时想到的
        remain -= res[0].second;
        res.erase(res.begin());  //lionel，这个是不会，搜的
    }
    cout << res.size() << endl;
}
```

- 30玩牌高手
    - *lionel，每次与前面的比最大值*，也不知道对不对

```
//1,-5,-6,4,3,6,-2  Out:11  【不选本轮的话，选3轮前的】
void od30_impl(vector<int> input) {
    //vector<int>result(input.size());//lionel，本意是把input.size()的大小，都赋值成0，vector<int>result { 0 };这种写法是错误的
    //int res = 0;
    int maxValue = 0;
    vector<int> score;
    for (int i = 0; i < input.size(); i++) {
        if (i < 3) {
            maxValue = max(input[i], 0);
            score.emplace_back(0);
        } else {
            maxValue = max(input[i]+maxValue, score[i - 3]);
            score.emplace_back(maxValue);  //lionel，也是突然之间改对了，境加了一个score的vector类型，把每次的比较，result[i-3]改成了score[i-3]
            //res += maxValue;
        }
        //cout << maxValue << endl;
    }
    cout << maxValue << endl;
}
```

单调栈

- 25找朋友

动规（我自己列的）

- 27猴子爬山
    - 参考LC70
    - 每次跳一步，或者跳3步
        - 第1个台阶，只有1种
        - 第2个台阶，也只有1种
        - 第3个台阶，就有2种
        - 第4个台阶，就可以用`f(n)=f(n-1)+f(n-3)`了，**记得从4开始**

```
void od27_impl(int input) {
```

```
    const int n = 100;//lionel，先定义为100吧
    int dp[n] = {0};
    dp[1] = 1;
    dp[2] = 1;
    dp[3] = 2;
    //dp[3]=2,dp[2]+dp[1] = 2;
    // dp[4] =dp[3]+dp[2]
    for (int i = 4; i <= input; i++) {  //lionel，这个点要从4开始，我写成了3了
        dp[i] = dp[i - 1] + dp[i - 3];
    }
    cout << dp[input] << endl;
    //动态规划的公式不会算，dp[i]=dp[i-3]
}
```

- 在

不会的知识点

区间交并集

- 2路灯照明问题
    - *不一定是自己做出来的，也不一定对*

```
int odO2_impl(vector<int> input, int N) {
    constexpr int len = 100;
    int res = 0;
    int first = 0, last = 1;
    while (first < last) {
        while (last >= input.size())
            //break;
            return res;  //lionel，这是个临时方案，为何break; 在这个地方不行了呢

        int distance = input[first] + input[last];
        if (distance == len) {
            res += 0;
        } else {
            if (distance > len)
                res += 0;//lionel，大于100的时候，不用管，其实 这里面可以缩小到只判断一
种，多了一些无用的判断语句
            else
                res += abs(len - distance);
        }
        first++;
        last++;
    }
    return res;
}
```

位运算

- 08分苹果
  - 最开始没做出来，感觉是题意都没读懂，当然*也有后面的技术问题*（比如怎么逐个算2进制）
  - *还是有一些疑问的*，比如**二进制累加**，为何可以直接用**异或**，而不需要单独拆成二进制？【当然这可能属于经典知识】
    - **异或，就是不带进位的二进制加法**（题目的意思，就是2进制累加时，不计算进位）-*本质上是这个知识点不会*

```
//输入数量3，每个苹果质量是3,5,6  [输出11，因为3的二进制是1，5和6的二进制之和也是1]
void od08_impl(int num, vector<int> input) {   //lionel，可能最开始题目都理解错了
    //lionel，问题是，1个数的二进制怎么算？ 跟谁去异或？【这个问题，没解答出来】

    // 先排个序？lionel，其实无所谓
    sort(input.begin(), input.end());

    int res = 0;
    //10-28解答，是因为要算left【至少从0到1】，算right也是【i,到最后一个】
    for (int i = 1; i < input.size()-1; i++) { //lionel，这个地方为何是从1开始？还有只到input.size()-1 ？
        int binA = 0, binB = 0;
        int sumA = 0, sumB = 0;  //lionel，这2处赋初值，是不是只能在for里面赋值，啥意思？
        //left
        for (int left = 0; left < i; left++) {
            binA ^= input[left];
            sumA += input[left];
        }

        //right
        for (int right = i; right < input.size(); right++) {
            binB ^= input[right];
            sumB += input[right];
        }

        if (binA == binB) {
            res = max(sumA, sumB);
        }
    }
    cout << res << endl;

}
```

- 72数据分类

```
int add(int value){  //lionel，是我这个理解错了，https://blog.csdn.net/yuhaomogui/article/details/125341750
    int sum = 0;
    while (value > 0) {
        sum += value & 0xff;
        value >>= 8;
    }
    return sum;
```

```
}

void od72_impl(vector<int> input) {
    int modNum = input[1];
    int val = input[0];
    vector<int>nums(input.begin() + 2, input.end());
    vector<int> res;
    for (auto c : nums) {
        int v = add(c);
        //cout << v << endl;
        res.push_back(v % modNum);
    }

    map<int, int>m;
    for (auto c : res) {  //lionel，应该是1 2 3 0 1 2 3 0 12，我输出是 0 1 2 3 0 1 2
3 0 1
        m[c]++;
    }

    int min = 0;
    for (auto c : m) {
        //cout << c.first << endl;
        if (c.first < val) {
            min = c.second > min ? c.second : min;
        }
    }
    cout << min << endl;
}
```

正则表达式

- 58增强的strstr
    - *但考查的应该不是这个*

```
//"a[bc]"匹配"ab"或"ac"
void od58_impl(string input_src, string input_target) {
    regex r(input_target);
    smatch m; //lionel，这是个啥
    regex_search(input_src, m, r);
    int res = 0;
    if (m.size()) {
        res = m.position();
    } else
        res = -1;

    cout << res << endl;
}
```

数学运算（debug-1，undo-1）

- 23、
- 64勾股数元组
    - 核心**两两互斥**怎么实现

```
//输入，1和20，输出 3组  【3，4,5 ； 5,12,13； 8,15,17】
int gcd(int a, int b) {
    return b == 0 ? a : gcd(b, a % b);
}

void od64() {
    int n, m;
    while (cin >> n >> m) {
        int found = 0;
        for (int i = n; i <= m; i++) {
            for (int j = i + 1; j <= m; j++) {
                int k = (int)sqrt(i * i + j * j);
                if (k > m) {
                    break;
                }
                if (k * k == i * i + j * j) {
                    if (gcd(i, j) == 1 && gcd(j, k) == 1) {
                        cout << i << " " << j << " " << k << endl;
                        found = 1;
                    }
                }
            }
        }
        if (!found) {
            cout << "Na" << endl;
        }
    }
}
```

- 65工号不够怎么办
    - 标准的数学问题，没想出来，网上找的答案

```
void od65() {
    //n=26的a次方+10的b次方
    int x = 2600, y = 1;  //26,1输出1   2600，1输出2

    //总人数
    int sum = pow(26, y);

    //数字个数
    int dig = 1;
    //y个字母
    while (sum * pow(10, dig) < x) {
        dig++;
    }

    cout << dig << endl;
```

```
}
```

- 71、（undo）

数据结构（栈）

- 13选修课（*但我没用栈做，在**字符串**里*）
- 33消消乐游戏

```
//Input:mMbccbc  Out:3(mMc)
void od33_impl(string inputs) {
    stack<char> s;
    for (int i = 0; i < inputs.size(); i++) {
        s.push(inputs[i]);
        if (s.top() == inputs[i+1]) { //i++和++i还是有蛮大区别的，写一行，和分两行写，还是有说法的
            s.pop();
            i++;
            break; //lionel，break还是调试出来的；
        }
    }
    cout << s.size() << endl;
}
```

- 36数组去重和排序（*但我没用栈做，在**字符串**里*）
- 39查找众数及中位数
    - *估计还没有debug出来，会core掉*

```
bool cmp39(pair<int, int> m1, pair<int, int> m2) {
    if (m1.second < m2.second)  //lionel，因为存在重复的元素，我就聪明的写个=号，会发生断言错误if (m1.second <= m2.second)，这种不支持
        return 1;
    return 0;
}

void od39_impl(vector<int> inputs) {
    map<int, int> countMap;
    for (int i = 0; i < inputs.size(); i++) {
        //lionel，不知道怎么用map去存重复的值了？
        countMap[inputs[i]]++; //lionel,0922突然想起来的
    }

    vector<pair<int, int>> res{ countMap.begin(),countMap.end() };
    sort(res.begin(), res.end(), cmp39);

    vector<int> result;
    int len = res.size();
    int max = res[len - 1].second;
    for (int i = len; i > 0; i--) {
        if (res[i - 1].second == max)
            result.push_back(res[i - 1].first);
        else
```

```
            break;
    }

    int middle = 0;
    sort(result.begin(), result.end());
    len = result.size();
    if (len % 2 != 0) {
        middle = result[len / 2];
    } else {
        middle = (result[len / 2] + result[len / 2 - 1]) / 2;
    }
    cout << middle << endl;
}
```

二分查找

- 53生日礼物

```
//10,20,5 蛋糕单价 5,5,2小礼物单， 预算15
void od53_impl(vector<int> cake, vector<int> gift, int value) {
    int res = 0;
    //先排个序
    sort(cake.begin(), cake.end());
    sort(gift.begin(), gift.end());
    for (int i = cake.size() - 1; i >= 0; i--) { //lionel，i>=0，少判断一位
        if (cake[i] > value) {
            continue; //lionel，这个还是没分清，break与contiue的区别，还是调试出来的
        }
        for (int j = gift.size() - 1; j >= 0; j--) {
            if (cake[i] + gift[j] <= value) {
                res++;
            }
        }
    }
    cout << res << endl;
}
```

约瑟夫环（undo-1）

- 05喊7的次数重排

自己觉得未消化的知识

回溯

- 54保密电梯

```
void od54() {
    // 5,3
    // 1 2 6
    //输出6 2 1  【1，2 6和6，2，1都是，按先处理大值，6，2，1】

    //典型的回溯，(lionel，我怎么看不出来典型）
```

```
    int target=5, n=3;
    //cin >> target >> n;
    //1 2 6 怎么写到vector中
    int tmp;
    vector<int> nums{1,2,6};
#if 0
    //其实可以在while（）里用n--的方式
    while (cin >> tmp) {
        nums.push_back(tmp);
        if (cin.get() == '\n')
            break;
    }
#endif

    //按大值处理，先排序
    sort(nums.begin(), nums.end(), greater<int>());
    vector<int> path(n,0);
    vector<bool>visited(100,false);  //lionel，记错了，前面是数量，后面才是初始化的值

    dfs(nums, 1, 0, 0, target, path, visited);

    if (!res.empty()) {   //lionel，为啥这个地方是6呢
        string s = to_string(res[0]);
        for (int i = 1; i < res.size(); i++){
            s += ",";
            s += to_string(res[i]);
        }
        cout << s << endl;
    }
}

/*
* 数组序列
* 向上还是下
* 目前所在楼层
* 第几次楼层移动
* 目标楼层
* 在楼层上下过程中的记录
* 记录哪些移动楼层数字已经被使用
*/
void dfs(vector<int> nums, int direction, int level, int i, int target,
vector<int> &path, vector<bool> &visitied) {
    //使用完所有次数
    if(i==nums.size()){
        //到达楼层，或者小于该楼层最近
        if (target - level >= 0 && target - level < min1) {
            min1 = target - level;
            res = path;
        }
        return;
    }

    //遍历序列，
    for (int j = 0; j < nums.size(); j++) {
        if (!visitied[j]) {
            visitied[j] = true;
            path.push_back(nums[j]);
```

```
            //上还是下
            if (direction == 1) {
                dfs(nums, 0, level + nums[j], i + 1, target, path, visitied);
            }else {
                //下的话，就要减
                dfs(nums, 1, level - nums[j], i + 1, target, path, visitied);
            }
            //回溯
            path.pop_back();//删除元素
https://blog.csdn.net/qq_39451578/article/details/115015639
            visitied[j] = false;
        }
    }
}
```

之前不会的知识点

位运算

- **异或，就是不带进位的二进制加法**（题目的意思，就是2进制累加时，不计算进位）-*本质上是这个知识点不会*
  - 来自题08

字符串

- 题07
  - 如果用`stol()`越界的话，可以考试用`stoll()`
  - 16进制转型，`string hexstr; stoi(hexstr,0,16);`，这个后面也遇到过，没做对
  - C++20的，`std::format`，*lionel，这个没会，没学呢*
  - 把整型转成16进制字符串（ostringstream对象、iomainip）
    - 因为0的时候，只转成1个0，所以要强制一下?

```
string num2hex(int num) {
    std::ostringstream ss;
    if (num >= 0 && num <= 255) {
        ss << std::hex <<setw(2) <<setfill('0') << num;//没有头文件iomainip，所以不能用setw
    }
    return ss.str();
}
```

- 题19
  - 字符转整型，`sum += (val[i] - '0');`，**直接用`-'0'`解决**
  - `stoi()`是可以判断正负的，https://www.jianshu.com/p/cdd95f5eaebb
- 题90
  - 怎么拆分空格（没有好好理解呢）**感觉有点整复杂了**

```
//Input: who love solo  这种其实可以在输入的时候直接处理掉，直接转换成acm模式的输入输出问题了
std::string s;
char delim = ' ';
s.append(1, delim);
std::regex reg(s);
std::vector<std::string> elems(std::sregex_token_iterator(input.begin(),
input.end(), reg, -1),             std::sregex_token_iterator());
```

  - 输出格式化，`cout << fixed<<setprecision(2)<<(sum*1.0) / res.size() << endl;`
- 题13
  - **字符串比较compare函数**，`int m = s1.compare(0, 5, s2, 0, 5);`

正则表达式

- 题20
  - 要包含头文件`#include<regex>`

```
bool check(string str) {
    regex r("[0-9]");
    string replace = regex_replace(str,r, "");//把数字替换掉
    //cout << str << "," << replace << endl;
    /*
    *
    ab, ab
    bC1, bC
    C124, C
C124A, CA
124ACb, ACb
24ACb, ACb
4ACb, ACb
ACb, ACb
    */
    return replace.size() != str.size() && replace.size() <= 1; //lionel，这个也没懂
}
```

- 题44

```
//把字符串拆成数字和字符，比如stone4这样，当然也可以不用这么复杂
regex reg1 ("[0-9]") ;
regex reg2("[a-zA-Z]") ;
string value = regex_replace(s1, reg1, "");
string num = regex_replace(s2, reg2, "");
```

- 题73

```
regex ws_re("\\s+");//lionel，不是太明白这个正则
    vector<string> res_helper(sregex_token_iterator(input.begin(), input.end(),
ws_re, -1), sregex_token_iterator());
```

- 在

其它

- `accumlate()`的使用，**头文件是`#include<numic>`，具体使用是`int first = accumulate(nums.begin(), nums.begin() + i,0);`
- 题34，**全排列**

```
    vector<string> v1{ s1,s2,s3 };
    sort(v1.begin(), v1.end());
    vector<string>v2;
    do {
        //cout << v1 << endl;
        v2.push_back(v1[0] + v1[1] + v1[2]);
    } while (next_permutation(v1.begin(), v1.end()));
```

- 题36，map的一些使用
  - **重复文件如何插入map**
    - 1、用pair类型，`map.insert(make_pair(1,1));map.insert(make_pair(1,2));`
    - 2、用下标，`map[1]=2;map[1]=3;`
  - pair的头文件是`#include<utility>`
  - `sort()`不支持map，要把map赋值到vector中才行
    - csdn，Tianqinse，摸森堡
  - csdn, weixin_41588502，Daniel_tmz，C++中的sort自定义排序函数
- `return res++;`和`return ++res;`还是有区别的
- 09组成最大数的**理论依据**
- **自定义排序时**，不能**有等于的**情况（大于，小于都可以）
- 在

遗留问题

做得不一定对

- 03
- 63
- 84
- 86

知识点

- C++20的，`std::format`
- 12、
  - LC，种花