2024

L1-1 编程解决一切（简单输出）

```python
print("Problem? The Solution: Programming.")
```

L1-2 再进去几个人（简单计算）

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {

    int a, b;
    cin >> a >> b;
    cout << b - a;

    return 0;
}
```

L1-3 帮助色盲（if - else）

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {

    int a, b;
    cin >> a >> b;

    if(b == 1) {
        cout << '-' << endl;
        if(a == 0 || a == 2) {
            cout << "stop";
        } else {
            cout << "move";
        }
    } else {
        if(a == 0) {
            cout << "biii\nstop";
        } else if(a == 1) {
            cout << "dudu\nmove";
        } else {
            cout << "-\nstop";
        }
    }

    return 0;
}
```

L1-4 四项全能（简单循环）

```cpp
#include <bits/stdc++.h>
```

```cpp
using namespace std;

int main() {

    int n, m;
    cin >> n >> m;

    int sum = 0;
    for(int i = 0; i < m; ++i) {
        int x;
        cin >> x;
        sum += x;
    }

    cout << max(0, sum - (m - 1) * n);

    return 0;
}
```

L1-5 兰州牛肉面（简单数组运用）

```cpp
#include <bits/stdc++.h>
using namespace std;

vector<double> price;
vector<int> cnt;

int main() {

    int n;
    cin >> n;
    price.resize(n + 1);
    cnt.resize(n + 1);
    for(int i = 1; i <= n; ++i) cin >> price[i];

    int a, b;
    double sum = 0.0;
    while(cin >> a >> b) {
        if(a == 0) break;
        cnt[a] += b;
        sum += price[a] * (double)b;
    }

    for(int i = 1; i <= n; ++i) cout << cnt[i] << endl;
    cout << fixed << setprecision(2);
    cout << sum;

    return 0;
}
```

L1-6 别再来这么多猫娘了！

1. 读入整行字符串 `getline(cin, str);`
2. 注意忽略换行符 `cin.ignore();`

3. 先用三个特殊字符 `如：@#@` 替代，避免使用 `<censored>` 导致超时

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {

    int n;
    cin >> n;
    cin.ignore();

    vector<string> wjc(n + 1);
    for(int i = 1; i <= n; ++i) getline(cin, wjc[i]);

    int k;
    cin >> k;
    cin.ignore();

    string text;
    getline(cin, text);

    int cnt = 0;
    for(int i = 1; i <= n; ++i) {
        while(text.find(wjc[i]) != string::npos) {
            auto pos = text.find(wjc[i]);
            cnt++;
            text.replace(pos, wjc[i].size(), "@#@");
        }
    }

    if(cnt >= k) {
        cout << cnt << endl << "He Xie Ni Quan Jia!";
    } else {
        while(text.find("@#@") != string::npos) {
            auto pos = text.find("@#@");
            text.replace(pos, 3, "<censored>");
        }
        cout << text;
    }

    return 0;
}
```

L1-7 整数的持续性（遍历处理）

```cpp
#include <bits/stdc++.h>
using namespace std;

int Func(int n) {
    int ans = 1;
    while(n) {
        ans *= n % 10;
        n /= 10;
    }
    return ans;
```

```cpp
}

int main() {

    int a, b;
    cin >> a >> b;

    int cnt, mx = 0;
    vector<int> res;
    for(int i = a; i <= b; ++i) {
        cnt = 0;
        int ii = i;
        while(ii >= 10) {
            cnt++;
            ii = Func(ii);
        }

        if(cnt > mx) {
            res.clear();
            res.push_back(i);
            mx = cnt;
        } else if(cnt == mx) {
            res.push_back(i);
        }
    }

    cout << mx << endl;
    for(size_t i = 0; i < res.size(); ++i) {
        if(i) cout << ' ';
        cout << res[i];
    }

    return 0;
}
```

L1-8 九宫格（按类别判断）

```cpp
#include <bits/stdc++.h>
using namespace std;

int solve() {
    int a[9][9];
    unordered_set<int> check;
    bool ok = true;

    for(int i = 0; i < 9; ++i) {
        for(int j = 0; j < 9; ++j) {
            cin >> a[i][j];
            if(a[i][j] < 1 || a[i][j] > 9) ok = false;
        }
    }

    if(!ok) return 0;

    // 行
```

```
    for(int i = 0; i < 9; ++i) {
        check.clear();
        for(int j = 0; j < 9; ++j) {
            check.insert(a[i][j]);
        }
        if(check.size() != 9) return 0;
    }

    // 列
    for(int j = 0; j < 9; ++j) {
        check.clear();
        for(int i = 0; i < 9; ++i) {
            check.insert(a[i][j]);
        }
        if(check.size() != 9) return 0;
    }

    // 格
    for(int i = 0; i < 9; ++i) {        // 第i格
        check.clear();
        for(int j = 0; j < 9; ++j) {    // 格中j个
            int x = 3 * (i / 3) + j / 3;
            int y = 3 * (i % 3) + j % 3;
            check.insert(a[x][y]);
        }
        if(check.size() != 9) return 0;
    }

    return 1;
}

int main() {

    int t;
    cin >> t;

    while(t--) {
        cout << solve() << endl;
    }

    return 0;
}
```

L2-1 鱼与熊掌（set 查找）

```
#include <bits/stdc++.h>
using namespace std;

int main() {

    int n, m;
    cin >> n >> m;
    vector<unordered_set<int>> num(n);

    for(int i = 0; i < n; ++i) {
```

```
        int k;
        cin >> k;

        while(k--) {
            int x;
            cin >> x;
            num[i].insert(x);
        }
    }

    int q;
    cin >> q;
    while(q--) {
        int a, b, cnt = 0;
        cin >> a >> b;

        for(int i = 0; i < n; ++i) {
            if(num[i].count(a) && num[i].count(b)) {
                cnt++;
            }
        }

        cout << cnt << endl;
    }

    return 0;
}
```

L2-2 懂蛇语

1. 按空格(你不知道有几个)分割字符串，并读取每个单词的首字母

```
stringstream ss(s);                 // 转化为字符串流
string word;                        // 存储从字符串流中每次读取到的单个单词
while(ss >> word) {
// 跳过字符串中所有的空白字符（包括单个空格、多个连续空格、制表符\t、换行符\n等）
    if(!word.empty()) {             // 防止读取到空单词
        acronym += word[0];         // 取每个单词的首字母
    }
}
```

2. 查找【字符串 -> 字符串数组】`unordered_map<string, vector<string>> dic;`

```
#include <bits/stdc++.h>
using namespace std;

unordered_map<string, vector<string>> dic;

int main() {

    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
```

```
    cin.ignore();

    for(int i = 0; i < n; ++i) {
        string s, origin, acronym;
        getline(cin, s);
        origin = s;          // 保留初始字符串

        // 步骤1:分割字符串为单词（按空格分割）
        stringstream ss(s);
        string word;
        while(ss >> word) {      // 跳过空格，读取每个单词
            if(!word.empty()) {
                acronym += word[0];      // 取每个单词的首字母
            }
        }

        dic[acronym].push_back(origin);
    }

    int m;
    cin >> m;
    cin.ignore();

    for(int j = 0; j < m; ++j) {
        string t, origin, acronym;
        getline(cin, t);
        origin = t;

        stringstream tt(t);
        string word;
        while(tt >> word) {
            if(!word.empty()) {
                acronym += word[0];
            }
        }

        if(dic.count(acronym) && !dic[acronym].empty()) {
            sort(dic[acronym].begin(), dic[acronym].end());
            for(size_t i = 0; i < dic[acronym].size(); ++i) {
                if(i) cout << '|';
                cout << dic[acronym][i];
            }
        } else {
            cout << origin;
        }
        cout << endl;
    }

    return 0;
}
```

L2-3 满树的遍历（DFS实现前序遍历）

```
#include <bits/stdc++.h>
using namespace std;
```

```cpp

int mx = 0;
bool ok = true;
bitset<100005> vis;
vector<int> res;
vector<vector<int>> adj;

void dfs(int root) {
    vis.set(root);
    res.push_back(root);
    if(adj[root].empty()) return ;

    int root_size = adj[root].size();

    if(root_size != 0) {
        if(mx == 0) mx = root_size;
        else if(root_size != mx){
            mx = max(mx, root_size);
            ok = false;
        }
    }

    for(size_t i = 0; i < adj[root].size(); ++i) {
        if(!vis.test(adj[root][i])) {
            dfs(adj[root][i]);
        }
    }
}

int main() {

    int n;
    cin >> n;
    adj.resize(n + 1);
    ok = true;

    int root;
    for(int i = 1; i <= n; ++i) {
        int x;
        cin >> x;
        if(x == 0) root = i;

        adj[x].push_back(i);
    }

    dfs(root);

    cout << mx << ' ';

    if(ok) cout << "yes\n";
    else cout << "no\n";

    for(size_t i = 0; i < res.size(); ++i) {
        if(i) cout << ' ';
        cout << res[i];
    }
```

```

    return 0;
}
```

L2-4 吉利矩阵（DFS构造 + 判断检查）

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long

int l, n;
int sx[5], sy[5]; // sx[x]：第x行已枚举列的累加和；sy[y]：第y列已枚举行的累加和
ll cnt = 0;

bool check() {
    int s1 = 0, s2 = 0;
    // s1：前(n-1)行最后一列的元素之和（推导值）
    // 前n-1行最后一列元素 = l - sx[i]（sx[i]是第i行前n-1列的和）
    for (int i = 1; i <= n - 1; i++) {
        s1 += l - sx[i];
    }
    // s2：最后一行前(n-1)列的元素之和（推导值）
    // 最后一行前n-1列元素 = l - sy[i]（sy[i]是第i列前n-1行的和）
    for (int i = 1; i <= n - 1; i++) {
        s2 += l - sy[i];
    }
    // 合法条件：
    // 1. s1 == s2 → 最后一行最后一列的元素唯一（同时满足行和、列和）
    // 2. s1 <= l → 最后一行最后一列的元素非负（l - s1 ≥ 0）
    if (s1 == s2 && s1 <= l) return 1;
    else return 0;
}

// DFS枚举前(n-1)*(n-1)个元素：x=当前行，y=当前列
void dfs(int x, int y) {
    // 递归终止：枚举完前(n-1)*(n-1)个元素
    if (x == n) {
        if (check()) cnt++; // 验证推导结果合法则计数+1
        return;
    }

    // 枚举当前位置(x,y)的元素值i（非负整数）
    // 枚举上限优化：
    // i ≤ l - sx[x] → 保证第x行累加和不超过l（行和约束）
    // i ≤ l - sy[y] → 保证第y列累加和不超过l（列和约束）
    for (int i = 0; i <= min(l - sx[x], l - sy[y]); i++) {
        sx[x] += i;  // 第x行累加和 += i
        sy[y] += i;  // 第y列累加和 += i

        // 计算下一个枚举位置（按行优先遍历）
        int nx = x, ny = y + 1;
        if (ny == n) { // 当前列到了n（前n-1列枚举完），切换到下一行第1列
            nx++, ny = 1;
        }
```

```
        dfs(nx, ny); // 递归枚举下一个位置

        // 回溯：撤销当前选择，恢复状态
        sx[x] -= i;
        sy[y] -= i;
    }
}


int main() {

    cin >> l >> n;

    dfs(1, 1);

    cout << cnt << endl;

    return 0;
}
```

L3-1 夺宝大赛（目标点看作起点，进行BFS）

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 105;
const int INF = 0x3f3f3f3f;

int n, m;
int t[MAXN][MAXN];
int a[MAXN][MAXN];
bool vis[MAXN][MAXN];
int wk[4][2] = {{0, 1}, {0, -1}, {1, 0}, {-1, 0}};

struct Node {
    int x, y;
    int time;
};

void bfs(int x, int y) {

    queue<Node> q;
    q.push({x, y, 0});
    t[x][y] = 0;
    vis[x][y] = true;

    while(!q.empty()) {
        auto [xx, yy, tt] = q.front();
        q.pop();

        for(int k = 0; k < 4; ++k) {
            int dx = xx + wk[k][0];
            int dy = yy + wk[k][1];
            if(dx < 1 || dx > m || dy < 1 || dy > n
```

```
                || a[dx][dy] == 0 || vis[dx][dy]) continue;

            vis[dx][dy] = true;
            t[dx][dy] = tt + 1;
            q.push({dx, dy, t[dx][dy]});
        }
    }
}

int main() {

    int sx, sy;
    cin >> m >> n;
    memset(t, 0x3f, sizeof(t));
    for(int i = 1; i <= m; ++i) {
        for(int j = 1; j <= n; ++j) {
            cin >> a[i][j];
            if(a[i][j] == 2) sx = i, sy = j;
        }
    }

    bfs(sx, sy);

    int k;
    cin >> k;

    unordered_map<int, int> cnt;
    unordered_map<int, int> idx;

    for(int i = 1; i <= k; ++i) {
        int x, y;
        cin >> y >> x;

        if(x < 1 || x > m || y < 1 || y > n || t[x][y] == INF) continue;

        int cur = t[x][y];
        cnt[cur]++;
        if(!idx.count(cur)) idx[cur] = i;
    }

    int mn_val = INF, mn_idx = -1;
    for(auto& [t, c] : cnt) {
        if(c == 1) {
            if(t < mn_val) {
                mn_val = t;
                mn_idx = idx[t];
            }
        }
    }

    if(mn_idx == -1) {
        cout << "No winner.";
    } else {
        cout << mn_idx << ' ' << mn_val;
    }

```

```
89      return 0;
90  }
```

L3-2 工业园区建设

前缀和 + 双指针，没写出来QAQ

L3-3 攀岩

九条可怜还是太权威了