

2024

L1-1 编程解决一切 (简单输出)

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

L1-2 再进去几个人 (简单计算)

```
1 | #include <bits/stdc++.h>
2 | using namespace std;
3 |
4 | int main() {
5 |
6 |     int a, b;
7 |     cin >> a >> b;
8 |     cout << b - a;
9 |
10 |    return 0;
11 | }
```

L1-3 帮助色盲 (if - else)

```
1 | #include <bits/stdc++.h>
2 | using namespace std;
3 |
4 | int main() {
5 |
6 |     int a, b;
7 |     cin >> a >> b;
8 |
9 |     if(b == 1) {
10 |         cout << '-' << endl;
11 |         if(a == 0 || a == 2) {
12 |             cout << "stop";
13 |         } else {
14 |             cout << "move";
15 |         }
16 |     } else {
17 |         if(a == 0) {
18 |             cout << "biii\nstop";
19 |         } else if(a == 1) {
20 |             cout << "dudu\nmove";
21 |         } else {
22 |             cout << "-\nstop";
23 |         }
24 |     }
25 |
26 |    return 0;
27 | }
```

L1-4 四项全能 (简单循环)

```
1 | #include <bits/stdc++.h>
```

```

2  using namespace std;
3
4  int main() {
5
6      int n, m;
7      cin >> n >> m;
8
9      int sum = 0;
10     for(int i = 0; i < m; ++i) {
11         int x;
12         cin >> x;
13         sum += x;
14     }
15
16     cout << max(0, sum - (m - 1) * n);
17
18     return 0;
19 }

```

L1-5 兰州牛肉面（简单数组运用）

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  vector<double> price;
5  vector<int> cnt;
6
7  int main() {
8
9      int n;
10     cin >> n;
11     price.resize(n + 1);
12     cnt.resize(n + 1);
13     for(int i = 1; i <= n; ++i) cin >> price[i];
14
15     int a, b;
16     double sum = 0.0;
17     while(cin >> a >> b) {
18         if(a == 0) break;
19         cnt[a] += b;
20         sum += price[a] * (double)b;
21     }
22
23     for(int i = 1; i <= n; ++i) cout << cnt[i] << endl;
24     cout << fixed << setprecision(2);
25     cout << sum;
26
27     return 0;
28 }

```

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

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5
6      int n;
7      cin >> n;
8      cin.ignore();
9
10     vector<string> wjc(n + 1);
11     for(int i = 1; i <= n; ++i) getline(cin, wjc[i]);
12
13     int k;
14     cin >> k;
15     cin.ignore();
16
17     string text;
18     getline(cin, text);
19
20     int cnt = 0;
21     for(int i = 1; i <= n; ++i) {
22         while(text.find(wjc[i]) != string::npos) {
23             auto pos = text.find(wjc[i]);
24             cnt++;
25             text.replace(pos, wjc[i].size(), "@#@");
26         }
27     }
28
29     if(cnt >= k) {
30         cout << cnt << endl << "He Xie Ni Quan Jia!";
31     } else {
32         while(text.find("@#@") != string::npos) {
33             auto pos = text.find("@#@");
34             text.replace(pos, 3, "<censored>");
35         }
36         cout << text;
37     }
38
39     return 0;
40 }
```

L1-7 整数的持续性 (遍历处理)

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int Func(int n) {
5      int ans = 1;
6      while(n) {
7          ans *= n % 10;
8          n /= 10;
9      }
10     return ans;
11 }
```

```

11 }
12
13 int main() {
14
15     int a, b;
16     cin >> a >> b;
17
18     int cnt, mx = 0;
19     vector<int> res;
20     for(int i = a; i <= b; ++i) {
21         cnt = 0;
22         int ii = i;
23         while(ii >= 10) {
24             cnt++;
25             ii = Func(ii);
26         }
27
28         if(cnt > mx) {
29             res.clear();
30             res.push_back(i);
31             mx = cnt;
32         } else if(cnt == mx) {
33             res.push_back(i);
34         }
35     }
36
37     cout << mx << endl;
38     for(size_t i = 0; i < res.size(); ++i) {
39         if(i) cout << ' ';
40         cout << res[i];
41     }
42
43     return 0;
44 }

```

L1-8 九宫格 (按类别判断)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int solve() {
5      int a[9][9];
6      unordered_set<int> check;
7      bool ok = true;
8
9      for(int i = 0; i < 9; ++i) {
10         for(int j = 0; j < 9; ++j) {
11             cin >> a[i][j];
12             if(a[i][j] < 1 || a[i][j] > 9) ok = false;
13         }
14     }
15
16     if(!ok) return 0;
17
18     // 行

```

```

19     for(int i = 0; i < 9; ++i) {
20         check.clear();
21         for(int j = 0; j < 9; ++j) {
22             check.insert(a[i][j]);
23         }
24         if(check.size() != 9) return 0;
25     }
26
27     // 列
28     for(int j = 0; j < 9; ++j) {
29         check.clear();
30         for(int i = 0; i < 9; ++i) {
31             check.insert(a[i][j]);
32         }
33         if(check.size() != 9) return 0;
34     }
35
36     // 格
37     for(int i = 0; i < 9; ++i) {          // 第i格
38         check.clear();
39         for(int j = 0; j < 9; ++j) {      // 格中j个
40             int x = 3 * (i / 3) + j / 3;
41             int y = 3 * (i % 3) + j % 3;
42             check.insert(a[x][y]);
43         }
44         if(check.size() != 9) return 0;
45     }
46
47     return 1;
48 }
49
50 int main() {
51
52     int t;
53     cin >> t;
54
55     while(t--) {
56         cout << solve() << endl;
57     }
58
59     return 0;
60 }

```

L2-1 鱼与熊掌 (set 查找)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5
6      int n, m;
7      cin >> n >> m;
8      vector<unordered_set<int>> num(n);
9
10     for(int i = 0; i < n; ++i) {

```

```

11     int k;
12     cin >> k;
13
14     while(k--> 0) {
15         int x;
16         cin >> x;
17         num[i].insert(x);
18     }
19 }
20
21 int q;
22 cin >> q;
23 while(q--> 0) {
24     int a, b, cnt = 0;
25     cin >> a >> b;
26
27     for(int i = 0; i < n; ++i) {
28         if(num[i].count(a) && num[i].count(b)) {
29             cnt++;
30         }
31     }
32
33     cout << cnt << endl;
34 }
35
36 return 0;
37 }

```

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

```

1  stringstream ss(s);           // 转化为字符串流
2  string word;                  // 存储从字符串流中每次读取到的单个单词
3  while(ss >> word) {
4  // 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）
5      if(!word.empty()) {       // 防止读取到空单词
6          acronym += word[0];    // 取每个单词的首字母
7      }
8  }

```

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

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  unordered_map<string, vector<string>> dic;
5
6  int main() {
7
8      ios::sync_with_stdio(false);
9      cin.tie(nullptr);
10
11     int n;
12     cin >> n;

```

```

13     cin.ignore();
14
15     for(int i = 0; i < n; ++i) {
16         string s, origin, acronym;
17         getline(cin, s);
18         origin = s;          // 保留初始字符串
19
20         // 步骤1: 分割字符串为单词 (按空格分割)
21         stringstream ss(s);
22         string word;
23         while(ss >> word) {    // 跳过空格, 读取每个单词
24             if(!word.empty()) {
25                 acronym += word[0];    // 取每个单词的首字母
26             }
27         }
28
29         dic[acronym].push_back(origin);
30     }
31
32     int m;
33     cin >> m;
34     cin.ignore();
35
36     for(int j = 0; j < m; ++j) {
37         string t, origin, acronym;
38         getline(cin, t);
39         origin = t;
40
41         stringstream tt(t);
42         string word;
43         while(tt >> word) {
44             if(!word.empty()) {
45                 acronym += word[0];
46             }
47         }
48
49         if(dic.count(acronym) && !dic[acronym].empty()) {
50             sort(dic[acronym].begin(), dic[acronym].end());
51             for(size_t i = 0; i < dic[acronym].size(); ++i) {
52                 if(i) cout << ' ';
53                 cout << dic[acronym][i];
54             }
55         } else {
56             cout << origin;
57         }
58         cout << endl;
59     }
60
61     return 0;
62 }

```

L2-3 满树的遍历 (DFS实现前序遍历)

```

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

```

```

3
4  int mx = 0;
5  bool ok = true;
6  bitset<100005> vis;
7  vector<int> res;
8  vector<vector<int>> adj;
9
10 void dfs(int root) {
11     vis.set(root);
12     res.push_back(root);
13     if(adj[root].empty()) return ;
14
15     int root_size = adj[root].size();
16
17     if(root_size != 0) {
18         if(mx == 0) mx = root_size;
19         else if(root_size != mx){
20             mx = max(mx, root_size);
21             ok = false;
22         }
23     }
24
25     for(size_t i = 0; i < adj[root].size(); ++i) {
26         if(!vis.test(adj[root][i])) {
27             dfs(adj[root][i]);
28         }
29     }
30 }
31
32 int main() {
33
34     int n;
35     cin >> n;
36     adj.resize(n + 1);
37     ok = true;
38
39     int root;
40     for(int i = 1; i <= n; ++i) {
41         int x;
42         cin >> x;
43         if(x == 0) root = i;
44
45         adj[x].push_back(i);
46     }
47
48     dfs(root);
49
50     cout << mx << ' ';
51
52     if(ok) cout << "yes\n";
53     else cout << "no\n";
54
55     for(size_t i = 0; i < res.size(); ++i) {
56         if(i) cout << ' ';
57         cout << res[i];
58     }

```



```

59
60
61     return 0;
62 }

```

L2-4 吉利矩阵 (DFS构造 + 判断检查)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4
5  int l, n;
6  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
7  ll cnt = 0;
8
9  bool check() {
10     int s1 = 0, s2 = 0;
11     // s1: 前(n-1)行最后一列的元素之和 (推导值)
12     // 前n-1行最后一列元素 = 1 - sx[i] (sx[i]是第i行前n-1列的和)
13     for (int i = 1; i <= n - 1; i++) {
14         s1 += 1 - sx[i];
15     }
16     // s2: 最后一行前(n-1)列的元素之和 (推导值)
17     // 最后一行前n-1列元素 = 1 - sy[i] (sy[i]是第i列前n-1行的和)
18     for (int i = 1; i <= n - 1; i++) {
19         s2 += 1 - sy[i];
20     }
21     // 合法条件:
22     // 1. s1 == s2 → 最后一行最后一列的元素唯一 (同时满足行和、列和)
23     // 2. s1 <= 1 → 最后一行最后一列的元素非负 (1 - s1 ≥ 0)
24     if (s1 == s2 && s1 <= 1) return 1;
25     else return 0;
26 }
27
28 // DFS枚举前(n-1)*(n-1)个元素: x=当前行, y=当前列
29 void dfs(int x, int y) {
30     // 递归终止: 枚举完前(n-1)*(n-1)个元素
31     if (x == n) {
32         if (check()) cnt++; // 验证推导结果合法则计数+1
33         return;
34     }
35
36     // 枚举当前位置(x,y)的元素值i (非负整数)
37     // 枚举上限优化:
38     // i ≤ 1 - sx[x] → 保证第x行累加和不超过1 (行和约束)
39     // i ≤ 1 - sy[y] → 保证第y列累加和不超过1 (列和约束)
40     for (int i = 0; i <= min(1 - sx[x], 1 - sy[y]); i++) {
41         sx[x] += i; // 第x行累加和 += i
42         sy[y] += i; // 第y列累加和 += i
43
44         // 计算下一个枚举位置 (按行优先遍历)
45         int nx = x, ny = y + 1;
46         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
47             nx++, ny = 1;
48         }

```

```

49
50     dfs(nx, ny); // 递归枚举下一个位置
51
52     // 回溯：撤销当前选择，恢复状态
53     sx[x] -= i;
54     sy[y] -= i;
55 }
56 }
57
58
59 int main() {
60
61     cin >> l >> n;
62
63     dfs(1, 1);
64
65     cout << cnt << endl;
66
67     return 0;
68 }

```

L3-1 夺宝大赛 (目标点看作起点, 进行BFS)

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int MAXN = 105;
5  const int INF = 0x3f3f3f3f;
6
7  int n, m;
8  int t[MAXN][MAXN];
9  int a[MAXN][MAXN];
10 bool vis[MAXN][MAXN];
11 int wk[4][2] = {{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
12
13 struct Node {
14     int x, y;
15     int time;
16 };
17
18 void bfs(int x, int y) {
19
20     queue<Node> q;
21     q.push({x, y, 0});
22     t[x][y] = 0;
23     vis[x][y] = true;
24
25     while(!q.empty()) {
26         auto [xx, yy, tt] = q.front();
27         q.pop();
28
29         for(int k = 0; k < 4; ++k) {
30             int dx = xx + wk[k][0];
31             int dy = yy + wk[k][1];
32             if(dx < 1 || dx > m || dy < 1 || dy > n

```

```

33         || a[dx][dy] == 0 || vis[dx][dy]) continue;
34
35         vis[dx][dy] = true;
36         t[dx][dy] = tt + 1;
37         q.push({dx, dy, t[dx][dy]});
38     }
39 }
40 }
41
42 int main() {
43
44     int sx, sy;
45     cin >> m >> n;
46     memset(t, 0x3f, sizeof(t));
47     for(int i = 1; i <= m; ++i) {
48         for(int j = 1; j <= n; ++j) {
49             cin >> a[i][j];
50             if(a[i][j] == 2) sx = i, sy = j;
51         }
52     }
53
54     bfs(sx, sy);
55
56     int k;
57     cin >> k;
58
59     unordered_map<int, int> cnt;
60     unordered_map<int, int> idx;
61
62     for(int i = 1; i <= k; ++i) {
63         int x, y;
64         cin >> y >> x;
65
66         if(x < 1 || x > m || y < 1 || y > n || t[x][y] == INF) continue;
67
68         int cur = t[x][y];
69         cnt[cur]++;
70         if(!idx.count(cur)) idx[cur] = i;
71     }
72
73     int mn_val = INF, mn_idx = -1;
74     for(auto& [t, c] : cnt) {
75         if(c == 1) {
76             if(t < mn_val) {
77                 mn_val = t;
78                 mn_idx = idx[t];
79             }
80         }
81     }
82
83     if(mn_idx == -1) {
84         cout << "No winner.";
85     } else {
86         cout << mn_idx << ' ' << mn_val;
87     }
88

```

```
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