

Index

英文字母/数字开头

• A*B问题	4
• AVL Tree	97
• B树的插入遍历和查找	89
• Entropy	35
• FBI 树	48
• Intervals	27
• islands 打炉石传说（二进制枚举）	8
• Looploop 可能过不了	76
• n个最小和（STL）	18
• Robotic Sort	73
• Root of AVL Tree	88
• Shaolin	72
• The order of a Tree	

	64
• Trees on the level	
	70
A	
• 阿里天池的新任务	
	117
B	
• 布设光纤	
	109
C	
• 插入排序	
	21
• 闯关游戏	
	99
D	
• 打印锯齿矩阵 (STL)	
	6
• 得到整数 X	
	5
• 得到整数 X	
	10
• 丁香之路	
	110
• 对称二叉树	
	86
• 多机调度问题	
	24
E	

• 二叉搜索树	63
• 二叉树	66
• 二叉树遍历	54
• 二叉树中的最低公共祖先	46
F	
• 斐波那契数列	1
G	
• 灌溉机器人	41
H	
• 汉诺塔	11
• 猴子打字	118
• 画图游戏	95
• 回文串	41
• 活动安排	25
J	
• 接龙	60

• 节点的最近公共祖先	107
• 计算集合的并集 (STL)	7
• 金字塔数独	13
• 矩阵旋转	2
L	
• 两个整数a,b之和	1
• 邻接表的使用	94
• 邻接矩阵的使用	94
M	
• 迷瘴	25
N	
• 逆序对	26
P	
• 判定欧拉回路	113
• 朋友	56
• 普通平衡树	83

Q

• 求二叉树高度	51
----------	----

S

• 删除最少的元素	39
• 商业信息共享	115
• 圣诞树	102
• 食物链	49
• 首尾相接	122
• 水果店 (STL)	14
• 四平方和	4
• 算法提高 选择排序	9

T

• 踏青	12
• 逃跑	28

W

• 完全二叉树的权值	86
------------	----

• 网络交友	58
• 网络延时	104
• 网页跳转	16
• 威虎山上的分配	112
X	
• 消除字符串	45
• 小明回家	32
• 小明面试 (STL)	15
• 小明的积木	44
• 小明的购物袋1	36
• 小明的购物袋2	37
• 小明的购物袋3	38
• 小明的训练室	105
• 小明跳木桩	38
• 小明学英语 (STL)	

7

- 修建大桥

98

- 新二叉树

53

- 线索二叉树的中序遍历

67

- 旋转数字

124

- 循环比赛日程表

34

Y

- 一维坐标的移动

23

- 一元三次方程求解

34

- 约瑟夫环问题（循环链表）

18

- 幼儿园买玩具（二进制枚举）

17

Z

- 在二叉树上移动

62

- 糟糕的 Bug

120

- 找出所有谎言

59

- 找出星型图的中心节点

97

- 字符串的冒泡排序

20

- 最长公共子序列

40

- 最大子阵

3

Data Structure and Algorithms - All Codes

Week 1 20250221

A 两个整数a,b之和

```
1 // 这题用C, CPP会超时
2 // https://blog.csdn.net/00FFrankDura/article/details/79093578
3
4 #include <iostream>
5
6 using namespace std;
7
8 int main(void) {
9     int n;
10    cin >> n;
11
12    int a, b;
13    for (int i = 0; i < n; i++) {
14        cin >> a >> b;
15        cout << a + b << endl;
16    }
17 }
```

B 斐波那契数列

```
1 #include <iostream>
2
3 using namespace std;
4
5 int num(int n);
6
7 int main(void) {
8     int n;
9     cin >> n;
10    cout << num(n);
11 }
12
13 int num(int n) {
14     if (n == 1) {
```

```

15         return 1;
16     }
17     if (n == 2) {
18         return 1;
19     }
20
21     int mod = 1000000007;
22     int a = 1, b = 1, c;
23     for (int i = 3; i <= n; i++) {
24         c = (a + b) % mod;
25         a = b;
26         b = c;
27     } // 防超时
28
29     return c;
30 }

```

C 矩阵旋转

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int n, m;
7      cin >> n >> m;
8
9      int v[n][m];
10     for (int i = 0; i < n; i++) {
11         for (int j = 0; j < m; j++) {
12             cin >> v[i][j];
13         }
14     }
15
16     for (int i = 0; i < m; i++) {
17         for (int j = n - 1; j >= 0; j--) {
18             if (j == n - 1) {
19                 cout << v[j][i];
20                 continue;
21             }
22             cout << " " << v[j][i];
23         }
24         cout << endl;
25     }
26 }

```

D 最大子阵

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int m, n;
7      cin >> m >> n;
8
9      vector<vector<int>> v(m, vector<int>(n));
10     for (int i = 0; i < m; i++) {
11         for (int j = 0; j < n; j++) {
12             cin >> v[i][j];
13         }
14     }
15
16     for (int i = 0; i < m; i++) {
17         for (int j = 1; j < n; j++) {
18             v[i][j] += v[i][j - 1];
19         }
20     }
21
22     long long b[m];
23     long long ans = INT_MIN;
24     for (int i = 0; i < n; i++) {
25         for (int j = i; j < n; j++) {
26             for (int k = 0; k < m; k++) {
27                 if (j == 0) {
28                     b[k] = v[k][0];
29                 } else if (i != 0) {
30                     b[k] = v[k][j] - v[k][i - 1];
31                 } else if (i == 0) {
32                     b[k] = v[k][j];
33                 }
34             }
35
36             vector<long long> dp(m, b[0]);
37             long long mmax = dp[0];
38             for (int k = 1; k < m; k++) {
39                 dp[k] = max(dp[k - 1] + b[k], b[k]);
40                 mmax = max(mmax, dp[k]);
41             }
42
43             ans = max(mmax, ans);
44     }
```

```

45     }
46
47     cout << ans << endl;
48 }

```

E 四平方和

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int n;
7      cin >> n;
8      for (int a = 0; a * a <= n; a++) { // 没什么好说的，暴力枚举
9          for (int b = a; b * b <= n - a * a; b++) {
10             for (int c = b; c * c <= n - a * a - b * b; c++) {
11                 double d = sqrt(n - a * a - b * b - c * c);
12                 if (!(d - (int)d)) {
13                     cout << a << " " << b << " " << c << " " << (int)d <<
14                     " " << endl;
15                     return 0;
16                 }
17             }
18         }
19     }

```

F A*B问题

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  vector<int> multi(vector<int> A, vector<int> B) {
6      vector<int> C(A.size() + B.size(), 0);
7      for (int i = 0; i < A.size(); i++) {
8          for (int j = 0; j < B.size(); j++) {
9              C[i + j] += A[i] * B[j];
10             }
11         }
12
13         int t = 0;
14         for (int i = 0; i < C.size(); i++) {
15             t += C[i];

```

```

16         C[i] = t % 10;
17         t /= 10;
18     }
19
20     while (C.size() > 1 && C.back() == 0) {
21         C.pop_back();
22     }
23
24     return C;
25 }
26
27 int main(void) {
28     string a, b;
29     cin >> a >> b;
30
31     vector<int> A, B;
32     for (int i = a.size() - 1; i >= 0; i--) {
33         A.push_back(a[i] - '0');
34     }
35     for (int i = b.size() - 1; i >= 0; i--) {
36         B.push_back(b[i] - '0');
37     }
38
39     vector<int> C = multi(A, B);
40
41     for (int i = C.size() - 1; i >= 0; i--) {
42         cout << C[i];
43     }
44     cout << endl;
45 }

```

G 得到整数 X

```

1  // 二进制枚举?
2  // 实际上也是动态规划
3  #include <bits/stdc++.h>
4
5  using namespace std;
6
7  int main(void) {
8      int n, x;
9      cin >> n >> x;
10     vector<int> v(n);
11
12     for (int i = 0; i < n; i++) {
13         cin >> v[i];

```

```

14     }
15
16     vector<int> dp(x + 1, 0);
17     dp[0] = 1;
18     for (int i = 0; i < n; i++) {
19         for (int j = x; j >= v[i]; j--) {
20             dp[j] += dp[j - v[i]];
21         }
22     }
23
24     cout << dp[x];
25 }

```

Week 2 20250228

A 打印锯齿矩阵 (STL)

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int n, m;
7      cin >> n >> m;
8      vector<vector<int>> v(n);
9      for (int i = 0; i < m; i++) {
10         int a, b;
11         cin >> a >> b;
12         v[a - 1].push_back(b);
13     }
14
15     for (int i = 0; i < n; i++) {
16         if (v[i].size() == 0) {
17             cout << " " << endl;
18             continue;
19         }
20
21         for (int j = 0; j < v[i].size(); j++) {
22             cout << v[i][j] << " ";
23         }
24         cout << endl;
25     }
26 }

```

C 计算集合的并集 (STL)

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  bool search(vector<int> v, int target) {
6      for (int i = 0; i < v.size(); i++) {
7          if (v[i] == target) {
8              return true;
9          }
10     }
11
12     return false;
13 }
14
15 int main(void) {
16     int n, m;
17     cin >> n >> m;
18     vector<int> v1(n);
19     vector<int> v2(m);
20
21     for (int i = 0; i < n; i++) {
22         cin >> v1[i];
23     }
24     for (int i = 0; i < m; i++) {
25         cin >> v2[i];
26     }
27
28     for (int i = 0; i < m; i++) {
29         if (search(v1, v2[i])) {
30             continue;
31         } else {
32             v1.push_back(v2[i]);
33         }
34     }
35
36     sort(v1.begin(), v1.end());
37     for (int i = 0; i < v1.size(); i++) {
38         cout << v1[i] << " ";
39     }
40     cout << endl;
41 }
```

D 小明学英语 (STL)

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  bool search(vector<string> v, string target) {
6      for (int i = 0; i < v.size(); i++) {
7          if (v[i] == target) {
8              return true;
9          }
10     }
11     return false;
12 }
13
14 int main(void) {
15     int n;
16     cin >> n;
17     vector<string> v1;
18
19     for (int i = 0; i < n; i++) {
20         int status;
21         cin >> status;
22         if (status == 0) {
23             string name;
24             cin >> name;
25             transform(name.begin(), name.end(), name.begin(), ::tolower);
26             // 转为小写
27             v1.push_back(name);
28         } else {
29             string name;
30             cin >> name;
31             transform(name.begin(), name.end(), name.begin(), ::tolower);
32             if (search(v1, name)) {
33                 cout << "Yes" << endl;
34             } else {
35                 cout << "No" << endl;
36             }
37         }
38     }
39 }

```

J islands 打炉石传说（二进制枚举）

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4

```



```

5  struct Card {
6      int cost;
7      bool d;
8      int w;
9  };
10
11 int main(void) {
12     int n;
13     cin >> n;
14
15     Card cards[n];
16     for (int i = 0; i < n; i++) {
17         cin >> cards[i].cost >> cards[i].d >> cards[i].w;
18     }
19
20     vector<int> dp(11, 0); // i点法力值时的最大攻击力
21     for (int i = 0; i < n; i++) {
22         if (!cards[i].d) {
23             for (int j = 10; j >= cards[i].cost; j--) {
24                 dp[j] = max(dp[j], dp[j - cards[i].cost] + cards[i].w);
25             }
26         }
27     }
28
29     for (int i = 0; i < n; i++) {
30         if (cards[i].d) {
31             for (int j = 10; j >= cards[i].cost; j--) {
32                 dp[j] = max(dp[j], dp[j - cards[i].cost] + cards[i].w);
33             }
34         }
35     }
36
37     int max = INT_MIN;
38     for (int i = 0; i < 11; i++) {
39         if (dp[i] > max) {
40             max = dp[i];
41         }
42     }
43
44     cout << max << endl;
45 }

```

Q 算法提高 选择排序

```

1  #include <bits/stdc++.h>
2

```

```

3  using namespace std;
4
5  int main(void) {
6      int n;
7      cin >> n;
8      vector<int> v(n);
9      for (int i = 0; i < n; i++) {
10         cin >> v[i];
11     }
12
13     bool sorted = false;
14     int index = 0;
15     while (!sorted) {
16         int a = v[index];
17         int minNum = INT_MAX;
18         int minIndex;
19         for (int i = index; i < n; i++) {
20             if (v[i] < minNum) {
21                 minNum = v[i];
22                 minIndex = i;
23             }
24         }
25         if (minNum == a) {
26             cout << "swap(a[" << index << "], a[" << index << "]):";
27             for (int i = 0; i < n; i++) {
28                 cout << v[i] << " ";
29             }
30             cout << endl;
31             index++;
32         } else {
33             swap(v[index], v[minIndex]);
34             cout << "swap(a[" << index << "], a[" << minIndex << "]):";
35             for (int i = 0; i < n; i++) {
36                 cout << v[i] << " ";
37             }
38             cout << endl;
39             index++;
40         }
41         if (index == n) {
42             sorted = true;
43         }
44     }
45 }

```

R 得到整数 X

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int n, m;
7      cin >> n >> m;
8      vector<int> v(n);
9      for (int i = 0; i < n; i++) {
10         cin >> v[i];
11     }
12
13     vector<int> dp(m + 1, 0);
14     dp[0] = 1;
15     for (int i = 0; i < n; i++) {
16         for (int j = m; j >= v[i]; j--) {
17             dp[j] += dp[j - v[i]];
18         }
19     }
20
21     cout << dp[m] << endl;
22 }

```

Week 3 20250308

A 汉诺塔

```

1  #include<iostream>
2  using namespace std;
3
4  long long moveCount(int n) {
5      if (n == 1) {
6          return 1;
7      }
8      return 1 + 2 * moveCount(n - 1);
9  }
10
11 long long strengthCost(int n) {
12     if (n == 1) {
13         return 1;
14     }
15     return n + 2 * strengthCost(n - 1);
16 }

```

```

17
18  int main() {
19      int n;
20      cin >> n;
21      cout << moveCount(n) << " " << strengthCost(n);
22      return 0;
23  }

```

B 踏青

```

1  #include<iostream>
2  using namespace std;
3
4  int px[4] = {-1, 1, 0, 0};
5  int py[4] = {0, 0, -1, 1};
6  char map[100][100];
7  int vst[100][100] = { 0 };
8  int n, m;
9
10 void grass(int x, int y) {
11     vst[x][y] = 1;
12     for (int i = 0; i < 4; i++) {
13         int nx = x + px[i];
14         int ny = y + py[i];
15
16         if (nx >= 0 && nx < n && ny >= 0 && ny < m && vst[nx][ny] == 0 &&
map[nx][ny] == '#') {
17             grass(nx, ny);
18         }
19     }
20 }
21
22 int main() {
23     cin >> n >> m;
24     int count = 0;
25     for (int i = 0; i < n; i++) {
26         for (int j = 0; j < m; j++) {
27             cin >> map[i][j];
28         }
29     }
30     for (int i = 0; i < n; i++) {
31         for (int j = 0; j < m; j++) {
32             if (map[i][j] == '#' && vst[i][j] == 0) {
33                 grass(i, j);
34                 count++;
35             }

```

```

36         }
37     }
38     cout << count;
39     return 0;
40 }

```

C 金字塔数独

```

1  #include<iostream>
2  using namespace std;
3
4  const int N = 9;
5  int grid[N][N];
6  int scores[N][N] = {
7      {6, 6, 6, 6, 6, 6, 6, 6, 6},
8      {6, 7, 7, 7, 7, 7, 7, 7, 6},
9      {6, 7, 8, 8, 8, 8, 8, 7, 6},
10     {6, 7, 8, 9, 9, 9, 8, 7, 6},
11     {6, 7, 8, 9, 10, 9, 8, 7, 6},
12     {6, 7, 8, 9, 9, 9, 8, 7, 6},
13     {6, 7, 8, 8, 8, 8, 8, 7, 6},
14     {6, 7, 7, 7, 7, 7, 7, 7, 6},
15     {6, 6, 6, 6, 6, 6, 6, 6, 6}
16 };
17 int maxScore = -1;
18
19 bool isValid(int x, int y, int num) {
20     for (int i = 0; i < N; i++) {
21         if (grid[i][y] == num || grid[x][i] == num) {
22             return false;
23         }
24     }
25
26     int rowStart = 3 * (x / 3);
27     int colStart = 3 * (y / 3);
28     for (int i = 0; i < 3; i++) {
29         for (int j = 0; j < 3; j++) {
30             if (grid[rowStart + i][colStart + j] == num) {
31                 return false;
32             }
33         }
34     }
35
36     return true;
37 }
38

```

```

39 void solve(int x, int y, int totalScore) {
40     if (x == N) {
41         maxScore = max(maxScore, totalScore);
42         return;
43     }
44
45     int nx = y == N - 1 ? x + 1 : x;
46     int ny = y == N - 1 ? 0 : y + 1;
47     if (grid[x][y] == 0) {
48         for (int i = 1; i < 10; i++) {
49             if (!isValid(x, y, i)) {
50                 continue;
51             }
52             grid[x][y] = i;
53             solve(nx, ny, totalScore + i * scores[x][y]);
54         }
55         grid[x][y] = 0;
56     }
57     else {
58         solve(nx, ny, totalScore + grid[x][y] * scores[x][y]);
59     }
60 }
61
62 int main() {
63     for (int i = 0; i < N; i++) {
64         for (int j = 0; j < N; j++) {
65             cin >> grid[i][j];
66         }
67     }
68     solve(0, 0, 0);
69     if (maxScore == 0) {
70         cout << -1 << endl;
71     }
72     else {
73         cout << maxScore << endl;
74     }
75     return 0;
76 }
77

```

D 水果店 (STL)

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4

```

```

5  int main(void) {
6      int n;
7      cin >> n;
8      map<string, map<string, int>> m;
9      for (int i = 0; i < n; i++) {
10         string origin;
11         string fruit;
12         int sales;
13         cin >> fruit >> origin >> sales;
14         m[origin][fruit] += sales;
15     }
16
17     for (auto& p1 : m) {
18         cout << p1.first << endl;
19         for (auto& p2 : p1.second) {
20             cout << "    |---" << p2.first << '(' << p2.second << ')' <<
endl;
21         }
22     }
23 }

```

E 小明面试 (STL)

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int n;
7      cin >> n;
8      unordered_map<int, int> m;
9      for (int i = 0; i < n; i++) {
10         int a;
11         cin >> a;
12         if (m.find(a) != m.end()) {
13             m[a]++;
14         } else {
15             m.insert({a, 1});
16         }
17     }
18
19     int imax = INT_MIN;
20     for (auto& [key, value] : m) { // 注意迭代器
21         imax = max(value, imax);
22     }
23 }

```

```

24     int mmax = INT_MIN;
25     for (auto& [key, value] : m) {
26         if (value == imax) {
27             mmax = max(mmax, key);
28         }
29     }
30     cout << mmax << ' ' << imax << endl;
31 }

```

F 网页跳转

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int n;
7      cin >> n;
8      stack<string> backStack, forwardStack;
9      string current = "Ignore";
10     for (int i = 0; i < n; i++) {
11         string action;
12         cin >> action;
13         if (action == "VISIT") {
14             string url;
15             cin >> url;
16             cout << url << endl;
17             if (current != "Ignore") {
18                 backStack.push(current);
19             }
20             current = url;
21             while (!forwardStack.empty()) {
22                 forwardStack.pop();
23             }
24         } else if (action == "BACK") {
25             if (backStack.empty()) {
26                 cout << "Ignore" << endl;
27             } else {
28                 forwardStack.push(current);
29                 current = backStack.top();
30                 backStack.pop();
31                 cout << current << endl;
32             }
33         } else if (action == "FORWARD") {
34             if (forwardStack.empty()) {

```



```

36         cout << "Ignore" << endl;
37     } else {
38         backStack.push(current);
39         current = forwardStack.top();
40         forwardStack.pop();
41         cout << current << endl;
42     }
43 }
44 }
45 }

```

G 幼儿园买玩具（二进制枚举）

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int n, m, k;
6      int result = 0;
7      int list[101][16];
8      cin >> n >> m >> k;
9
10     for (int i = 0; i < n; i++) {
11         cin >> list[i][0];
12         for (int j = 1; j <= list[i][0]; j++) {
13             cin >> list[i][j];
14         }
15     }
16
17     for (int i = 0; i < (1 << k); i++) {
18         int flag[16] = {0};
19         int count = 0;
20         int num = 0;
21         for (int j = 0; j < k; j++) {
22             if (i & (1 << j)) {
23                 num++;
24                 flag[j + 1] = 1;
25             }
26         }
27         for (int x = 0; x < n; x++) {
28             bool a = true;
29             for (int y = 1; y <= list[x][0]; y++) {
30                 if (flag[list[x][y]] == 0) {
31                     a = false;
32                 }
33             }

```

```

34         if (a)
35             count++;
36     }
37     if (count > result && num <= m) {
38         result = count;
39     }
40 }
41
42 cout << result;
43 return 0;
44 }

```

H 约瑟夫环问题（循环链表）

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int n, m;
7      cin >> n >> m;
8      list<int> l;
9      for (int i = 1; i <= n; i++) {
10         l.push_back(i);
11     }
12
13     auto it = l.begin();
14     while (l.size() > 1) {
15         for (int i = 1; i < m; i++) {
16             it++;
17             if (it == l.end()) {
18                 it = l.begin();
19             }
20         }
21
22         cout << *it << " ";
23         it = l.erase(it);
24         if (it == l.end()) {
25             it = l.begin();
26         }
27     }
28     cout << *l.begin() << endl;
29 }

```

I n个最小和（STL）

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  struct Node {
6      int sum, i, j;
7      bool operator>(const Node &other) const {
8          return sum > other.sum;
9      }
10 };
11
12 vector<int> kSmallestSums(vector<int> &A, vector<int> &B, int n) {
13     sort(A.begin(), A.end());
14     sort(B.begin(), B.end());
15
16     priority_queue<Node, vector<Node>, greater<Node>> minHeap;
17     set<pair<int, int>> visited;
18
19     minHeap.push({A[0] + B[0], 0, 0});
20     visited.insert({0, 0});
21
22     vector<int> result;
23
24     while (n--) {
25         Node cur = minHeap.top();
26         minHeap.pop();
27         result.push_back(cur.sum);
28
29         int i = cur.i, j = cur.j;
30
31         if (i + 1 < A.size() && !visited.count({i + 1, j})) {
32             minHeap.push({A[i + 1] + B[j], i + 1, j});
33             visited.insert({i + 1, j});
34         }
35
36         if (j + 1 < B.size() && !visited.count({i, j + 1})) {
37             minHeap.push({A[i] + B[j + 1], i, j + 1});
38             visited.insert({i, j + 1});
39         }
40     }
41
42     return result;
43 }
44
45 int main() {
46     int n;
```

```

47     cin >> n;
48     vector<int> A(n), B(n);
49
50     for (int i = 0; i < n; i++) cin >> A[i];
51     for (int i = 0; i < n; i++) cin >> B[i];
52
53     vector<int> result = kSmallestSums(A, B, n);
54
55     for (int i = 0; i < result.size(); i++) {
56         if (i > 0) cout << " ";
57         cout << result[i];
58     }
59     cout << endl;
60
61     return 0;
62 }

```

J 字符串的冒泡排序

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  vector<int> bubbleSortTransform(string sh, const string &ch) {
5      int n = sh.size();
6      vector<int> steps;
7
8      unordered_map<char, int> countSh, countCh;
9      for (char c : sh) {
10         countSh[c]++;
11     }
12     for (char c : ch) {
13         countCh[c]++;
14     }
15     if (countSh != countCh) {
16         return {-1};
17     }
18
19     for (int i = 0; i < n; i++) {
20         if (sh[i] == ch[i]) {
21             continue;
22         }
23
24         int pos = i;
25         while (sh[pos] != ch[i]) pos++;
26
27         for (int j = pos; j > i; j--) {

```

```

28         swap(sh[j], sh[j - 1]);
29         steps.push_back(j);
30     }
31 }
32
33 steps.insert(steps.begin(), steps.size());
34 return steps;
35 }
36
37 int main() {
38     int n;
39     string sh, ch;
40     cin >> n >> sh >> ch;
41
42     vector<int> result = bubbleSortTransform(sh, ch);
43
44     if (result[0] == -1) {
45         cout << -1 << endl;
46     } else {
47         for (int i = 0; i < result.size(); i++) {
48             if (i > 0) cout << " ";
49             cout << result[i];
50         }
51         cout << endl;
52     }
53     return 0;
54 }

```

K 插入排序

```

1  #include <bits/stdc++.h>
2  #define MAXN 8010
3  using namespace std;
4
5  struct node {
6      int val;
7      int num;
8      bool operator>(const node b) const {
9          if (this->val != b.val) return this->val > b.val;
10         return this->num > b.num;
11     }
12     bool operator<(const node b) const {
13         if (this->val != b.val) return this->val < b.val;
14         return this->num < b.num;
15     }
16 };

```

```
17
18 node a[MAXN];
19 int n, q, type, x, v;
20 int order[MAXN];
21
22 void get_order() {
23     for (int i = 1; i <= n; i++) {
24         order[a[i].num] = i;
25     }
26 }
27
28 void update(int x, int v) {
29     a[order[x]].val = v;
30     for (int i = order[x]; i < n; i++) {
31         if (a[i] > a[i + 1]) {
32             swap(a[i], a[i + 1]);
33         }
34     }
35     for (int i = order[x]; i > 1; i--) {
36         if (a[i] < a[i - 1]) {
37             swap(a[i], a[i - 1]);
38         }
39     }
40     get_order();
41 }
42
43 int main() {
44     freopen("sort.in", "r", stdin);
45     freopen("sort.out", "w", stdout);
46     scanf("%d%d", &n, &q);
47     for (int i = 1; i <= n; i++) {
48         scanf("%d", &a[i].val);
49         a[i].num = i;
50     }
51     sort(a + 1, a + n + 1);
52     get_order();
53     while (q--) {
54         scanf("%d", &type);
55         if (type == 1) {
56             scanf("%d%d", &x, &v);
57             update(x, v);
58         } else {
59             scanf("%d", &x);
60             printf("%d\n", order[x]);
61         }
62     }
```

```
63     return 0;
64 }
```

Week 4 20250314

A 一维坐标的移动

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int minSteps(int n, int start, int end) {
6      if (start == end)
7          return 0;
8
9      vector<bool> visited(n + 1, false);
10     queue<pair<int, int>> q;
11     q.push({start, 0});
12
13     while (!q.empty()) {
14         int pos = q.front().first;
15         int steps = q.front().second;
16         q.pop();
17
18         if (pos == end) return steps;
19
20         if (pos + 1 <= n && !visited[pos + 1]) {
21             q.push({pos + 1, steps + 1});
22             visited[pos + 1] = true;
23         }
24         if (pos - 1 >= 0 && !visited[pos - 1]) {
25             q.push({pos - 1, steps + 1});
26             visited[pos - 1] = true;
27         }
28         if (pos * 2 <= n && !visited[pos * 2]) {
29             q.push({pos * 2, steps + 1});
30             visited[pos * 2] = true;
31         }
32     }
33
34     return -1;
35 }
```

```

37  int main() {
38      int n, start, end;
39      cin >> n >> start >> end;
40
41      int result = minSteps(n, start, end);
42      cout << result << endl;
43
44      return 0;
45  }

```

B 多机调度问题

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main() {
6      int n, m;
7      vector<int> taxt;
8      cin >> n >> m;
9      vector<int> dp(m);
10
11     for (int i = 0; i < n; i++) {
12         int tmp;
13         cin >> tmp;
14         taxt.push_back(tmp);
15     }
16     sort(taxt.begin(), taxt.end(), greater<int>());
17     if (n <= m) {
18         cout << taxt[n - 1];
19         return 0;
20     }
21     else {
22         for (int j = 0; j < m; j++) {
23             dp[j] = taxt[j];
24         }
25         for (int j = m; j < n; j++) {
26             sort(dp.begin(), dp.end());
27             dp[0] += taxt[j];
28         }
29     }
30
31     sort(dp.begin(), dp.end(), greater<int>());
32     cout << dp[0];
33     return 0;
34 }

```


C 迷瘴

```
1  #include <cstdio>
2  #include <bits/stdc++.h>
3  using namespace std;
4
5  int main() {
6      int v, n, i;
7      double w, sum = 0, p = 0, an[105] = {0};
8      scanf("%d%d%lf", &n, &v, &w);
9      for (i = 0; i < n; i++) {
10         scanf("%lf", &an[i]);
11     }
12     sort(an, an + n);
13     for (i = 0; i < n; i++) {
14         sum += an[i];
15         if (sum / (i + 1) <= w) {
16             p = sum / (i + 1);
17         } else {
18             break;
19         }
20     }
21     if (i == 0)
22         printf("0 0.00\n");
23     else
24         printf("%d %.2f\n", i * v, p / 100);
25     return 0;
26 }
```

D 活动安排

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  struct Program {
6      int start;
7      int end;
8  };
9
10 bool cmp(Program a, Program b) {
11     return a.end < b.end;
12 }
13
14 int main() {
15     int n;
```

```

16     cin >> n;
17     vector<Program> programs(n);
18
19     for (int i = 0; i < n; i++) {
20         cin >> programs[i].start >> programs[i].end;
21     }
22     sort(programs.begin(), programs.end(), cmp);
23
24     int count = 1;
25     int tmp = programs[0].end;
26     for (int i = 1; i < n; i++) {
27         if (programs[i].start >= tmp) {
28             tmp = programs[i].end;
29             count++;
30         }
31     }
32
33     cout << count;
34     return 0;
35 }

```

E 逆序对

```

1  #include<iostream>
2  using namespace std;
3
4  int nums[100000];
5  long long reverseCount = 0;
6
7  void merge(int sta, int mid, int end) {
8      int tmp[100000];
9      int i = sta;
10     int j = mid + 1;
11     int k = sta;
12
13     while (i <= mid && j <= end) {
14         if (nums[i] > nums[j]) {
15             reverseCount += mid + 1 - i;
16             tmp[k++] = nums[j++];
17         }
18         else {
19             tmp[k++] = nums[i++];
20         }
21     }
22     while (i <= mid) {
23         tmp[k++] = nums[i++];

```

```

24     }
25     while (j <= end) {
26         tmp[k++] = nums[j++];
27     }
28
29     for (int i = sta; i <= end; i++) {
30         nums[i] = tmp[i];
31     }
32 }
33
34 void mergeSort(int sta, int end) {
35     if (sta < end) {
36         int mid = sta + (end - sta) / 2;
37         mergeSort(sta, mid);
38         mergeSort(mid + 1, end);
39         merge(sta, mid, end);
40     }
41 }
42
43 int main() {
44     int n, k;
45     cin >> n >> k;
46     for (int i = 0; i < n; i++) {
47         cin >> nums[i];
48     }
49
50     mergeSort(0, n - 1);
51     cout << max((long long)0, reverseCount - k);
52     return 0;
53 }

```

G Intervals

```

1  #include<iostream>
2  #include<vector>
3  #include<algorithm>
4  using namespace std;
5
6  struct Interval {
7      int start;
8      int end;
9  };
10
11 bool cmp(Interval a, Interval b) {
12     return a.start < b.start;
13 }

```

```

14
15 int main() {
16     int n;
17     int count = 0;
18     cin >> n;
19     vector<Interval> intervals(n);
20     vector<Interval> result;
21
22     for (int i = 0; i < n; i++) {
23         cin >> intervals[i].start >> intervals[i].end;
24     }
25     sort(intervals.begin(), intervals.end(), cmp);
26
27     result.push_back(intervals[0]);
28     for (int i = 1; i < n; i++) {
29         if (intervals[i].start <= result[count].end) {
30             result[count].end = max(intervals[i].end, result[count].end);
31         }
32         else {
33             result.push_back(intervals[i]);
34             count++;
35         }
36     }
37
38     for (int i = 0; i <= count; i++) {
39         cout << result[i].start << " " << result[i].end << endl;
40     }
41     return 0;
42 }

```

H 逃跑

```

1  #include<iostream>
2  #include<vector>
3  #include<queue>
4  #include<bitset>
5  using namespace std;
6
7  const int N = 110;
8  int dx[] = { 0, 0, 1, -1, 0 };
9  int dy[] = { 1, -1, 0, 0, 0 };
10 int pretime = -1;
11 int n, m, k, d;
12 bitset<N> vis[N];
13 bitset<N> sold[N];
14 bitset<N> go[N][N];

```

```
15
16 struct Role {
17     int x;
18     int y;
19     int time;
20     int d;
21 };
22
23 struct Soldier {
24     char c;
25     int t;
26     int v;
27     int x;
28     int y;
29 }soldiers[N];
30
31 struct Bullet {
32     int x;
33     int y;
34     int v;
35     char c;
36 };
37
38 queue<Role> roles;
39 vector<Bullet> bullets;
40
41 bool isValid(int x, int y) {
42     return x >= 0 && x <= n && y >= 0 && y <= m;
43 }
44
45 void bulletUpdate(Bullet& bullet) {
46     switch (bullet.c) {
47         case 'N':
48             bullet.x -= bullet.v;
49             break;
50         case 'W':
51             bullet.y -= bullet.v;
52             break;
53         case 'E':
54             bullet.y += bullet.v;
55             break;
56         case 'S':
57             bullet.x += bullet.v;
58             break;
59     }
60 }
```

```

61
62 void update(int time) {
63     if (pretime == time) {
64         return;
65     }
66     else {
67         pretime = time;
68         for (Bullet& bullet : bullets) {
69             if (!isValid(bullet.x, bullet.y)) {
70                 continue;
71             }
72             vis[bullet.x][bullet.y] = 0;
73             bulletUpdate(bullet);
74             if (!isValid(bullet.x, bullet.y)) {
75                 continue;
76             }
77             vis[bullet.x][bullet.y] = 1;
78         }
79         for (int i = 1; i <= k; i++)
80         {
81             if (time % soldiers[i].t == 0) {
82                 bullets.push_back({ soldiers[i].x, soldiers[i].y,
soldiers[i].v, soldiers[i].c });
83                 vis[soldiers[i].x][soldiers[i].y] = 1;
84             }
85         }
86     }
87 }
88
89 void bfs(int x, int y, int time, int d) {
90     bool flag = false;
91     roles.push({ x, y, time, d });
92     go[x][y] = true;
93
94     while (roles.size()) {
95         Role role = roles.front();
96         roles.pop();
97
98         if (role.d < m + n - role.x - role.y) {
99             continue;
100         }
101         if (go[n][m][role.time] == 1) {
102             cout << roles.back().time;
103             flag = true;
104             return;
105         }

```

```
106         for (int i = 0; i < 5; i++) {
107             int tx = role.x + dx[i];
108             int ty = role.y + dy[i];
109             update(role.time + 1);
110             if ((!isValid(tx, ty)) || vis[tx][ty] == 1 || go[tx][ty]
[role.time] || sold[tx][ty] == 1) {
111                 continue;
112             }
113             if (!vis[tx][ty]) {
114                 go[tx][ty][role.time] = 1;
115             }
116             roles.push({ tx, ty, role.time + 1, role.d - 1 });
117         }
118     }
119
120     if (!flag) {
121         cout << "Bad luck!";
122     }
123 }
124
125 void solve() {
126     cin >> n >> m >> k >> d;
127     for (int i = 1; i <= k; i++) {
128         char c;
129         int t, v, x, y;
130         cin >> c >> t >> v >> x >> y;
131         soldiers[i] = { c, t, v, x, y };
132         sold[x][y] = 1;
133     }
134     for (int i = 1; i <= k; i++) {
135         bullets.push_back({ soldiers[i].x, soldiers[i].y, soldiers[i].v,
soldiers[i].c });
136         vis[soldiers[i].x][soldiers[i].y] = 1;
137     }
138
139     bfs(0, 0, 0, d);
140 }
141
142 int main() {
143     int q = 1;
144     for (int i = 0; i < q; i++) {
145         solve();
146     }
147     return 0;
148 }
```

I 小明回家

```
1  #include<iostream>
2  #include<vector>
3  #include<queue>
4  using namespace std;
5
6  struct Point {
7      int x;
8      int y;
9      int step;
10 };
11
12 char map[2000][2000];
13 int px[4] = {-1, 1, 0, 0};
14 int py[4] = {0, 0, -1, 1};
15 int n, m;
16
17 int stepCount(Point start, Point end) {
18     int vst[2000][2000] = { 0 };
19     queue<Point> points;
20     points.push(start);
21     vst[start.x][start.y] = 1;
22
23     while (!points.empty()) {
24         Point tmp = points.front();
25         int x = tmp.x;
26         int y = tmp.y;
27         int step = tmp.step;
28         points.pop();
29
30         if (x == end.x && y == end.y) {
31             return step;
32         }
33
34         for (int i = 0; i < 4; i++) {
35             int nx = x + px[i];
36             int ny = y + py[i];
37
38             if (nx >= 0 && nx < n && ny >= 0 && ny < m && vst[nx][ny] ==
0 && map[nx][ny] != '#') {
39                 Point next;
40                 next.x = nx;
41                 next.y = ny;
42                 next.step = step + 1;
43                 points.push(next);
```



```

44         vst[nx][ny] = 1;
45     }
46 }
47 }
48
49     return -1;
50 }
51
52 int main() {
53     Point start, key, end;
54     vector<Point> keys;
55     int keyCount = 0;
56     int minStep = -1;
57     cin >> n >> m;
58     for (int i = 0; i < n; i++) {
59         for (int j = 0; j < m; j++) {
60             cin >> map[i][j];
61             if (map[i][j] == 'S') {
62                 start.x = i;
63                 start.y = j;
64                 start.step = 0;
65             }
66             if (map[i][j] == 'P') {
67                 key.x = i;
68                 key.y = j;
69                 keyCount++;
70                 keys.push_back(key);
71             }
72             if (map[i][j] == 'T') {
73                 end.x = i;
74                 end.y = j;
75             }
76         }
77     }
78
79     for (int i = 0; i < keyCount; i++) {
80         if (stepCount(start, keys[i]) == -1)
81             continue;
82         keys[i].step = stepCount(start, keys[i]);
83         if (stepCount(keys[i], end) == -1)
84             continue;
85         minStep = minStep == -1 ? stepCount(keys[i], end) :
min(stepCount(keys[i], end), minStep);
86     }
87     cout << minStep;
88     return 0;

```

Week 5 20250321

A 一元三次方程求解

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4
5  double A, B, C, D;
6
7  double f(double x) {
8      return A*x*x*x + B*x*x + C*x + D;
9  }
10
11 int main() {
12     cin >> A >> B >> C >> D;
13     for (int i = -10000; i <= 10000; i++) {
14         if (f((i - 0.5) / 100) * f((i + 0.5) / 100) < 0 || f((i - 0.5) /
15             100) == 0)
16             printf("%.2lf ", i / 100.0);
17     }
18     return 0;
19 }
```

B 循环比赛日程表

```
1  #include <iostream>
2  #include <vector>
3  #include <iomanip>
4  using namespace std;
5
6  int main() {
7      int M;
8      cin >> M;
9      int N = 1 << M;
10     vector<vector<int>> table(N, vector<int>(N, 0));
11
12     table[0][0] = 1;
13     int half = 1;
14
15     for (int k = 0; k < M; k++) {
```

```

16         for (int i = 0; i < (1 << k); i++) {
17             for (int j = 0; j < (1 << k); j++) {
18                 table[i][j + half] = table[i][j] + half;
19                 table[i + half][j] = table[i][j] + half;
20                 table[i + half][j + half] = table[i][j];
21             }
22         }
23         half *= 2;
24     }
25
26     for (int i = 0; i < N; i++) {
27         for (int j = 0; j < N; j++) {
28             cout << setw(3) << table[i][j];
29         }
30         cout << endl;
31     }
32
33     return 0;
34 }

```

C Entropy

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main()
5  {
6      int freq_arr[27];
7      string s;
8      while (cin >> s && s != "END")
9      {
10         int total = 0;
11         int sum = 0;
12         priority_queue<int, vector<int>, greater<int> > q;
13         memset(freq_arr, 0, sizeof(freq_arr));
14         int n = s.size();
15         for (int i = 0; i < n; i++)
16         {
17             if (s[i] == '_')
18             {
19                 freq_arr[26]++;
20             }
21             else
22             {
23                 freq_arr[s[i] - 'A']++;
24             }

```

```

25     }
26     for (int i = 0; i < 27; i++)
27     {
28         if (freq_arr[i])
29         {
30             q.push(freq_arr[i]);
31         }
32     }
33     if (q.size() == 1)
34     {
35         total = q.top();
36         printf("%d %d %.1lf\n", n * 8, n * 1, (double)8/1);
37     }
38     else
39     {
40         while (q.size() > 1)
41         {
42             sum = 0;
43             sum += q.top();
44             q.pop();
45             sum += q.top();
46             q.pop();
47             q.push(sum);
48             total += sum;
49         }
50         printf("%d %d %.1lf\n", n * 8, total, (double)n * 8 / total);
51     }
52 }
53
54 return 0;
55 }

```

D 小明的购物袋1

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int V;
7      int n;
8      cin >> V >> n;
9
10     vector<int> v(n);
11     for (int i = 0; i < n; i++) {
12         cin >> v[i];

```

```

13     }
14
15     vector<vector<int>> dp(n + 1, vector<int>(V + 1, 0));
16     for (int i = 1; i < n + 1; i++) {
17         for (int j = 0; j <= V; j++) {
18             if (v[i - 1] <= j) {
19                 dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - v[i - 1]] +
v[i - 1]);
20             } else {
21                 dp[i][j] = dp[i - 1][j];
22             }
23         }
24     }
25     cout << V - dp[n][V] << endl;
26 }

```

E 小明的购物袋2

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int V, n;
7      cin >> V >> n;
8      vector<pair<int, int>> v(n);
9      for (int i = 0; i < n; i++) {
10         cin >> v[i].first >> v[i].second;
11     }
12
13     vector<vector<int>> dp(n + 1, vector<int>(V + 1, 0));
14
15     for (int i = 1; i < n + 1; i++) {
16         for (int j = 0; j < V + 1; j++) {
17             if (j < v[i - 1].first) {
18                 dp[i][j] = dp[i - 1][j];
19             } else {
20                 dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - v[i -
1].first] + v[i - 1].second);
21             }
22         }
23     }
24
25     cout << dp[n][V] << endl;
26 }

```

F 小明的购物袋3

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int V, n;
7      cin >> n >> V;
8      vector<pair<int, int>> v(n);
9      for (int i = 0; i < n; i++) {
10         cin >> v[i].first >> v[i].second;
11     }
12
13     vector<int> dp(V + 1, 0);
14
15     for (int i = 0; i < n; i++) {
16         for (int j = v[i].first; j <= V; j++) {
17             dp[j] = max(dp[j], dp[j - v[i].first] + v[i].second);
18         }
19     }
20
21     cout << dp[V] << endl;
22 }
```

Week 6 20250328

A 小明跳木桩

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int n;
7      cin >> n;
8      vector<int> h(n);
9      for (int i = 0; i < n; i++) {
10         cin >> h[i];
11     }
12
13     vector<int> dp(n, 1);
14     for (int i = 0; i < n; i++) {
```

```

15         for (int j = 0; j < i; j++) {
16             if (h[j] >= h[i]) {
17                 dp[i] = max(dp[i], dp[j] + 1);
18             }
19         }
20     }
21
22     int mmax = INT_MIN;
23     for (int i = 0; i < n; i++) {
24         mmax = max(mmax, dp[i]);
25     }
26
27     cout << mmax << endl;
28 }

```

B 删除最少的元素

```

1  #include<iostream>
2  using namespace std;
3
4  int main(){
5      int n;
6      cin >> n;
7      int dp1[1009];
8      int dp2[1009];
9      int a[1009];
10     int ans = 0;
11
12     for(int i = 0; i < n; i++){
13         cin >> a[i];
14     }
15
16     for(int i = 0; i < n; i++){
17         dp1[i] = 1;
18         dp2[i] = 1;
19     }
20
21     for(int i = 1; i < n; i++){
22         for(int j = 0; j < i; j++){
23             if(a[i] <= a[j]){
24                 dp1[i] = max(dp1[i], dp1[j] + 1);
25             }
26         }
27     }
28
29     for(int i = n - 2; i >= 0; i--){

```

```

30         for(int j = n - 1; j > i; j--){
31             if(a[i] <= a[j]){
32                 dp2[i] = max(dp2[i], dp2[j] + 1);
33             }
34         }
35     }
36
37     for(int i = 0; i < n; i++){
38         ans = max(ans, dp1[i] + dp2[i] - 1);
39     }
40
41     cout << n - ans;
42
43     return 0;
44 }

```

C 最长公共子序列

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      string a, b;
7      cin >> a >> b;
8      vector<vector<int>> dp(a.length() + 1, vector<int>(b.length() + 1,
9      0));
10
11     for (int i = 1; i <= a.size(); i++) {
12         for (int j = 1; j <= b.size(); j++) {
13             if (a[i - 1] == b[j - 1]) {
14                 dp[i][j] = dp[i - 1][j - 1] + 1;
15             } else {
16                 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
17             }
18         }
19     }
20
21     int mmax = INT_MIN;
22     for (int i = 0; i < a.size() + 1; i++) {
23         for (int j = 0; j < b.size() + 1; j++) {
24             mmax = max(mmax, dp[i][j]);
25         }
26     }
27

```



```
28     cout << mmax << endl;
29 }
```

D 回文串

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      string s;
7      cin >> s;
8      string s2 = s;
9      reverse(s2.begin(), s2.end());
10
11     vector<vector<int>> dp(s.size() + 1, vector<int>(s.size() + 1));
12     for (int i = 1; i <= s.size(); i++) {
13         for (int j = 1; j <= s.size(); j++) {
14             if (s[i - 1] == s2[j - 1]) {
15                 dp[i][j] = dp[i - 1][j - 1] + 1;
16             } else {
17                 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
18             }
19         }
20     }
21
22     // l = len - lcs
23     cout << s.length() - dp[s.size()][s.size()] << endl;
24 }
```

E 灌溉机器人

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define max_Heap(x) priority_queue<x, vector<x>, less<x>>
4  #define min_Heap(x) priority_queue<x, vector<x>, greater<x>>
5  typedef long long ll;
6  typedef unsigned long long ull;
7  typedef pair<int, int> PII;
8  typedef pair<long long, long long> PLL;
9  const double PI = acos(-1);
10
11 int n, m;
12 char field[106][16];
13 vector<int> s[106];
```

```

14  int dp[106][106][106];
15
16  int main()
17  {
18      ios::sync_with_stdio(0);
19      cin.tie(0);
20      cout.tie(0);
21
22      unordered_map<int, int> mp;
23
24      cin >> n >> m;
25      for (int i = 1; i <= n; i++)
26      {
27          for (int j = 0; j < m; j++)
28          {
29              cin >> field[i][j];
30          }
31      }
32
33      for (int i = 1; i <= n; i++)
34      {
35          for (int j = 0; j < (1 << m); j++)
36          {
37              bool ok = 1;
38              for (int k = 0; k < m; k++)
39              {
40                  if (((j >> k) & 1) && (field[i][k] == 'H'))
41                  {
42                      ok = 0;
43                      break;
44                  }
45              }
46              if ((j & (j << 1)) || (j & (j << 2)) || (j & (j >> 1)) || (j
& (j >> 2)))
47              {
48                  ok = 0;
49              }
50              if (ok)
51                  s[i].push_back(j);
52          }
53      }
54
55      for (int i = 0; i < (1 << m); i++)
56      {
57          int cnt = 0;
58          for (int j = 0; j < m; j++)

```

```

59         {
60             if ((i >> j) & 1)
61                 cnt++;
62         }
63         mp[i] = cnt;
64     }
65
66     for (int i = 0; i < s[1].size(); i++)
67     {
68         dp[1][i][0] = mp[s[1][i]];
69     }
70
71     s[0].push_back(0);
72
73     for (int i = 1; i <= n; i++)
74     {
75         for (int num3 = 0; num3 < s[i].size(); num3++)
76         {
77             int s3 = s[i][num3];
78             for (int num2 = 0; num2 < s[i - 1].size(); num2++)
79             {
80                 int s2 = s[i - 1][num2];
81                 for (int num1 = 0; num1 < s[i - 2].size(); num1++)
82                 {
83                     int s1 = s[i - 2][num1];
84                     if (!(s1 & s2) && !(s1 & s3) && !(s2 & s3))
85                         dp[i][num3][num2] = max(dp[i][num3][num2], dp[i -
1][num2][num1] + mp[s3]);
86                 }
87             }
88         }
89     }
90
91     int ans = 0;
92     for (int i = 0; i < s[n].size(); i++)
93     {
94         for (int j = 0; j < s[n - 1].size(); j++)
95         {
96             ans = max(ans, dp[n][i][j]);
97         }
98     }
99     cout << ans;
100
101     return 0;
102 }

```

F 小明的积木

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define max_Heap(x) priority_queue<x, vector<x>, less<x>>
4  #define min_Heap(x) priority_queue<x, vector<x>, greater<x>>
5  typedef long long ll;
6  typedef unsigned long long ull;
7  typedef pair<int, int> PII;
8  typedef pair<long long, long long> PLL;
9  const double PI = acos(-1);
10
11 int main()
12 {
13     ios::sync_with_stdio(0);
14     cin.tie(0);
15     cout.tie(0);
16
17     int capacity, n;
18     cin >> capacity >> n;
19     int dp[(1 << n)];
20     memset(dp, 0x3f, sizeof(dp));
21     vector<int> s(n);
22     vector<int> w(n);
23     for (int i = 0; i < n; i++)
24     {
25         cin >> s[i] >> w[i];
26     }
27     for (int i = 0; i < (1 << n); i++)
28     {
29         int max_s = 0;
30         int sum_w = 0;
31         for (int j = 0; j < n; j++)
32         {
33             if ((i >> j) & 1)
34             {
35                 max_s = max(max_s, s[j]);
36                 sum_w += w[j];
37             }
38         }
39         if (sum_w <= capacity)
40         {
41             dp[i] = max_s;
42         }
43         else
44         {
```

```

45         for (int j = ((i - 1) & i); j > 0; j = ((j - 1) & i))
46         {
47             dp[i] = min(dp[i], dp[j] + dp[j ^ i]);
48         }
49     }
50 }
51 cout << dp[(1 << n) - 1];
52
53 return 0;
54 }

```

G 消除字符串

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  string str;
5
6  bool isReverse(int i) {
7      string str1;
8      string str2;
9
10     for (int j = 0; j < 16; j++) {
11         if ((i >> j) & 1) {
12             str1 += str[j];
13         }
14     }
15     str2 = str1;
16     reverse(str1.begin(), str1.end());
17
18     return str1 == str2;
19 }
20
21 int main() {
22     cin >> str;
23     int length = str.size();
24     vector<int> dp(1 << length, 0);
25
26     for (int i = 1; i < (1 << length); i++) {
27         dp[i] = isReverse(i) ? 1 : 16;
28         for (int j = i; j; j = (j - 1) & i) {
29             dp[i] = min(dp[i], dp[j] + dp[j ^ i]);
30         }
31     }
32     cout << dp[(1 << length) - 1];
33     return 0;

```

Week 8 20250411

A 二叉树中的最低公共祖先

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4  struct Node {
5      int val;
6      Node* lchild;
7      Node* rchild;
8      Node(int val) : val(val), lchild(NULL), rchild(NULL) {}
9  };
10
11 vector<int> inorder;
12 vector<int> preorder;
13 map<int, Node*> record;
14
15 Node* createTree(vector<int> preorder, vector<int> inorder, int preL, int
preR, int inL, int inR) {
16     if (preL > preR) {
17         return NULL;
18     }
19
20     Node* root = new Node(preorder[preL]);
21     record[preorder[preL]] = root;
22     int k = inL;
23     while (inorder[k] != preorder[preL]) {
24         k++;
25     }
26     int numLeft = k - inL;
27
28     root->lchild = createTree(preorder, inorder, preL + 1, preL +
numLeft, inL, k - 1);
29     root->rchild = createTree(preorder, inorder, preL + numLeft + 1,
preR, k + 1, inR);
30     return root;
31 }
32
33 Node* LCA(Node* root, Node* first, Node* second) {
34     if (root == NULL || root == first || root == second) {
```

```

35         return root;
36     }
37
38     Node* left = LCA(root->lchild, first, second);
39     Node* right = LCA(root->rchild, first, second);
40     if (left && right) return root;
41     else if (left) return left;
42     else return right;
43 }
44
45 int main() {
46     int M, N;
47     cin >> M >> N;
48     for (int i = 0; i < N; i++) {
49         int order;
50         cin >> order;
51         inorder.push_back(order);
52     }
53     for (int i = 0; i < N; i++) {
54         int order;
55         cin >> order;
56         preorder.push_back(order);
57     }
58
59     Node* root = createTree(preorder, inorder, 0, N - 1, 0, N - 1);
60     while (M--) {
61         int u, v;
62         cin >> u >> v;
63         if (record.count(u) && record.count(v)) {
64             Node* lca = LCA(root, record[u], record[v]);
65             if (lca->val == u) {
66                 cout << u << " is an ancestor of " << v << "." << endl;
67             }
68             else if (lca->val == v) {
69                 cout << v << " is an ancestor of " << u << "." << endl;
70             }
71             else {
72                 cout << "LCA of " << u << " and " << v << " is " << lca-
>val << "." << endl;
73             }
74         }
75         else if (record.count(u)) {
76             cout << "ERROR: " << v << " is not found." << endl;
77         }
78         else if (record.count(v)) {
79             cout << "ERROR: " << u << " is not found." << endl;

```

```

80         }
81     else {
82         cout << "ERROR: " << u << " and " << v << " are not found."
<< endl;
83     }
84 }
85 return 0;
86 }

```

B FBI 树

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  struct Node {
6      char val;
7      Node* lchild;
8      Node* rchild;
9      Node(char val) : val(val), lchild(NULL), rchild(NULL) {}
10 };
11
12 char classify(string str) {
13     bool contain1 = false;
14     bool contain0 = false;
15
16     for (int i = 0; i < str.size(); i++) {
17         if (str[i] == '0') {
18             contain0 = true;
19         }
20         else {
21             contain1 = true;
22         }
23     }
24     if (contain0 && contain1) {
25         return 'F';
26     }
27     else if (contain0){
28         return 'B';
29     }
30     else {
31         return 'I';
32     }
33 }
34
35 Node* creatTree(string str) {

```



```

36     char val = classify(str);
37     int len = str.size();
38
39     if (len == 0) {
40         return NULL;
41     }
42
43     Node* node = new Node(val);
44     if (len >= 2) {
45         node->lchild = creatTree(str.substr(0, len / 2));
46         node->rchild = creatTree(str.substr(len / 2, len / 2));
47     }
48     return node;
49 }
50
51 void postorder(Node* node) {
52     if (node) {
53         postorder(node->lchild);
54         postorder(node->rchild);
55         cout << node->val;
56     }
57 }
58
59 int main() {
60     int N;
61     string str;
62     cin >> N >> str;
63
64     Node* root = creatTree(str);
65     postorder(root);
66
67     return 0;
68 }

```

C 食物链

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  class FoodUnion {
6  private:
7      int parent[50000];
8      int relation[50000];
9      int fakeCount;
10     int total;

```

```
11
12 public:
13     FoodUnion(int n) {
14         for (int i = 1; i <= n; ++i) {
15             parent[i] = i;
16             relation[i] = 0;
17         }
18         fakeCount = 0;
19         total = n;
20     }
21
22     int find(int x) {
23         if (x != parent[x]) {
24             int root = find(parent[x]);
25             relation[x] += relation[parent[x]];
26             parent[x] = root;
27         }
28         return parent[x];
29     }
30
31     void unite(int x, int y, int relType) {
32         relType--;
33         if (x > total || y > total) {
34             fakeCount++;
35             return;
36         }
37         if (relType == 1 && x == y) {
38             fakeCount++;
39             return;
40         }
41
42         int rootX = find(x);
43         int rootY = find(y);
44
45         if (rootX == rootY) {
46             int delta = ((relation[x] - relation[y]) % 3 + 3) % 3;
47             if (delta != relType) {
48                 fakeCount++;
49             }
50         } else {
51             parent[rootX] = rootY;
52             relation[rootX] = relation[y] - relation[x] + relType;
53         }
54     }
55
56     int getFakeCount() const {
```

```

57         return fakeCount;
58     }
59 };
60
61 int main() {
62     int n, k;
63     cin >> n >> k;
64     FoodUnion uf(n);
65
66     for (int i = 0; i < k; ++i) {
67         int type, a, b;
68         cin >> type >> a >> b;
69         uf.unite(a, b, type);
70     }
71
72     cout << uf.getFakeCount() << endl;
73     return 0;
74 }

```

D 求二叉树高度

```

1  #include<string>
2  #include<iostream>
3  #include <queue>
4  #include <algorithm>
5  #include <sstream>
6  using namespace std;
7
8  struct TreeNode {
9      int val;
10     TreeNode *lchild;
11     TreeNode *rchild;
12     TreeNode(int x) : val(x), lchild(NULL), rchild(NULL) {}
13 };
14
15 class Solution {
16 public:
17     int maxDepth(TreeNode* p) {
18         int lh,rh,hi;
19         lh = rh = hi = 0;
20         if(p != nullptr){
21             if(p->lchild == NULL) lh = 0 ; else lh= maxDepth(p->lchild);
22             if(p->rchild == NULL) rh= 0 ; else rh= maxDepth(p->rchild);
23             if (lh > rh) hi = lh + 1; else hi = rh + 1;
24         }
25         else hi = 0;

```

```

26         return hi;
27     }
28 };
29
30 void trimLeftTrailingSpaces(string &input) {
31     input.erase(input.begin(), find_if(input.begin(), input.end(), [](int
ch) {
32         return !isspace(ch);
33     }));
34 }
35
36 void trimRightTrailingSpaces(string &input) {
37     input.erase(find_if(input.rbegin(), input.rend(), [](int ch) {
38         return !isspace(ch);
39     }).base(), input.end());
40 }
41
42 TreeNode* stringToTreeNode(string input) {
43     trimLeftTrailingSpaces(input);
44     trimRightTrailingSpaces(input);
45     input = input.substr(1, input.length() - 2);
46     if (!input.size()) {
47         return nullptr;
48     }
49
50     string item;
51     stringstream ss;
52     ss.str(input);
53
54     getline(ss, item, ',');
55     TreeNode* root = new TreeNode(stoi(item));
56     queue<TreeNode*> nodeQueue;
57     nodeQueue.push(root);
58
59     while (true) {
60         TreeNode* node = nodeQueue.front();
61         nodeQueue.pop();
62
63         if (!getline(ss, item, ',')) {
64             break;
65         }
66
67         trimLeftTrailingSpaces(item);
68         if (item != "null") {
69             int leftNumber = stoi(item);
70             node->lchild = new TreeNode(leftNumber);

```

```

71         nodeQueue.push(node->lchild);
72     }
73
74     if (!getline(ss, item, ',')) {
75         break;
76     }
77
78     trimLeftTrailingSpaces(item);
79     if (item != "null") {
80         int rightNumber = stoi(item);
81         node->rchild = new TreeNode(rightNumber);
82         nodeQueue.push(node->rchild);
83     }
84 }
85 return root;
86 }
87
88 int main() {
89     string line;
90     getline(cin, line);
91     TreeNode* root = stringToTreeNode(line);
92     int ret = Solution().maxDepth(root);
93     string out = to_string(ret);
94     cout << out << endl;
95     return 0;
96 }

```

E 新二叉树

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  struct TreeNode {
6      char val;
7      TreeNode *left;
8      TreeNode *right;
9      TreeNode(char val) : val(val), left(nullptr), right(nullptr) {}
10 };
11
12 TreeNode *traversal(TreeNode *node, char val) {
13     if (node == nullptr) {
14         return nullptr;
15     }
16     if (node->val == val) {
17         return node;

```

```

18     }
19
20     TreeNode *p = traversal(node->left, val);
21     if (p == nullptr) {
22         return traversal(node->right, val);
23     }
24     return p;
25 }
26
27 void traversal2(TreeNode *node) {
28     if (node == nullptr) {
29         return;
30     }
31     cout << node->val;
32     traversal2(node->left);
33     traversal2(node->right);
34 }
35
36 int main(void) {
37     int n;
38     cin >> n;
39
40     TreeNode *root = nullptr;
41     TreeNode *p = root;
42     for (int i = 0; i < n; i++) {
43         char a, b, c;
44         cin >> a >> b >> c;
45         if (i == 0) {
46             root = new TreeNode(a);
47         }
48         p = traversal(root, a);
49         if (b != '*') {
50             p->left = new TreeNode(b);
51         }
52         if (c != '*') {
53             p->right = new TreeNode(c);
54         }
55     }
56     traversal2(root);
57     cout << endl;
58 }

```

F 二叉树遍历

```

1  #include <bits/stdc++.h>
2

```

```
3 using namespace std;
4
5 struct TreeNode {
6     int val;
7     TreeNode *left;
8     TreeNode *right;
9     TreeNode(int val) : val(val), left(nullptr), right(nullptr) {}
10 };
11
12 TreeNode* traversal(vector<int>& pre, vector<int>& middle) {
13     if (pre.size() == 0) {
14         return nullptr;
15     }
16     int rootVal = pre[0];
17     TreeNode *root = new TreeNode(rootVal);
18
19     if (pre.size() == 1) {
20         return root;
21     }
22
23     int index = -1;
24     for (int i = 0; i < middle.size(); i++) {
25         if (middle[i] == rootVal) {
26             index = i;
27             break;
28         }
29     }
30
31     vector<int> leftMiddle(middle.begin(), middle.begin() + index);
32     vector<int> rightMiddle(middle.begin() + index + 1, middle.end());
33
34     vector<int> leftPre(pre.begin() + 1, pre.begin() + leftMiddle.size()
35 + 1);
36     vector<int> rightPre(pre.begin() + leftMiddle.size() + 1, pre.end());
37
38     root->left = traversal(leftPre, leftMiddle);
39     root->right = traversal(rightPre, rightMiddle);
40
41     return root;
42 }
43
44 void traversal2(TreeNode *root) {
45     if (root == nullptr) {
46         return;
47     }
```

```

48     traversal2(root->left);
49     traversal2(root->right);
50     cout << root->val << ' ';
51 }
52
53 int main(void) {
54     int n;
55     cin >> n;
56     vector<int> pre(n);
57     vector<int> middle(n);
58
59     for (int i = 0; i < n; i++) {
60         cin >> pre[i];
61     }
62
63     for (int i = 0; i < n ; i++) {
64         cin >> middle[i];
65     }
66
67     TreeNode *root = traversal(pre, middle);
68
69     traversal2(root);
70     cout << endl;
71 }

```

G 朋友

```

1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  class UnionFind {
6  private:
7      int father[5000];
8      int height[5000];
9
10 public:
11     UnionFind(int n) {
12         for (int i = 1; i <= n; i++) {
13             father[i] = i;
14             height[i] = 1;
15         }
16     }
17
18     int get(int x) {
19         if (x == father[x]) {

```



```
20         return x;
21     }
22     return father[x] = get(father[x]);
23 }
24
25 void merge(int x, int y) {
26     x = get(x);
27     y = get(y);
28
29     if (x == y) {
30         return;
31     }
32     else {
33         if (height[x] == height[y]) {
34             height[x]++;
35             father[y] = x;
36         }
37         else if (height[x] > height[y]) {
38             father[y] = x;
39         }
40         else {
41             father[x] = y;
42         }
43     }
44 }
45 };
46
47 int main() {
48     int n, m, p;
49     cin >> n >> m >> p;
50     UnionFind friends(n);
51
52     for (int i = 0; i < m; i++) {
53         int x, y;
54         cin >> x >> y;
55         friends.merge(x, y);
56     }
57     for (int i = 0; i < p; i++) {
58         int x, y;
59         cin >> x >> y;
60         if (friends.get(x) == friends.get(y)) {
61             cout << "Yes" << endl;
62         }
63         else {
64             cout << "No" << endl;
65         }
66     }
```

```
66     }
67
68     return 0;
69 }
```

H 网络交友

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  class DisjointSet {
6  private:
7      unordered_map<string, string> parent;
8      unordered_map<string, int> size;
9      unordered_map<string, int> depth;
10
11 public:
12     void insert(const string& person) {
13         if (parent.count(person)) return;
14         parent[person] = person;
15         size[person] = 1;
16         depth[person] = 1;
17     }
18
19     string find(const string& person) {
20         if (parent[person] != person) {
21             parent[person] = find(parent[person]);
22         }
23         return parent[person];
24     }
25
26     void unite(const string& a, const string& b) {
27         string rootA = find(a);
28         string rootB = find(b);
29
30         if (rootA == rootB) return;
31
32         if (depth[rootA] < depth[rootB]) {
33             parent[rootA] = rootB;
34             size[rootB] += size[rootA];
35         } else {
36             parent[rootB] = rootA;
37             size[rootA] += size[rootB];
38             if (depth[rootA] == depth[rootB]) {
39                 depth[rootA]++;
```

```

40         }
41     }
42 }
43
44 int getSize(const string& person) {
45     return size[find(person)];
46 }
47 };
48
49 int main() {
50     int relationCount;
51     cin >> relationCount;
52
53     DisjointSet network;
54
55     while (relationCount-->0) {
56         string person1, person2;
57         cin >> person1 >> person2;
58         network.insert(person1);
59         network.insert(person2);
60         network.unite(person1, person2);
61         cout << network.getSize(person1) << endl;
62     }
63
64     return 0;
65 }

```

Week 9

A 找出所有谎言

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int MAXN = 50010;
5  int parent[MAXN], rel[MAXN];
6
7  int findf(int x) {
8      if (parent[x] != x) {
9          int p = parent[x];
10         parent[x] = findf(p);
11         rel[x] = (rel[x] + rel[p]) % 3;
12     }

```

```

13     return parent[x];
14 }
15
16 int main() {
17     ios::sync_with_stdio(false);
18     cin.tie(NULL);
19
20     int n, k;
21     cin >> n >> k;
22     for (int i = 1; i <= n; i++) {
23         parent[i] = i;
24         rel[i] = 0;
25     }
26     int ans = 0;
27     while (k--) {
28         int d, x, y;
29         cin >> d >> x >> y;
30         if (x < 1 || x > n || y < 1 || y > n || (d == 2 && x == y)) {
31             ans++;
32             continue;
33         }
34         int fx = findf(x), fy = findf(y);
35         if (fx == fy) {
36             if (d == 1) {
37                 if ((rel[x] - rel[y] + 3) % 3 != 0) ans++;
38             } else {
39                 if ((rel[x] - rel[y] + 3) % 3 != 1) ans++;
40             }
41         } else {
42             if (d == 1) {
43                 parent[fx] = fy;
44                 rel[fx] = (rel[y] - rel[x] + 3) % 3;
45             } else {
46                 parent[fx] = fy;
47                 rel[fx] = (rel[y] + 1 - rel[x] + 3) % 3;
48             }
49         }
50     }
51     cout << ans << endl;
52     return 0;
53 }

```

B 接龙

```

1  #include <cstdio>
2  #include <cstdlib>

```

```
3  #include <algorithm>
4  using namespace std;
5
6  const int MAXN = 30001;
7
8  int parent[MAXN];
9  int distance_[MAXN];
10 int size_[MAXN];
11
12 void init() {
13     for (int i = 1; i <= 30000; ++i) {
14         parent[i] = i;
15         distance_[i] = 0;
16         size_[i] = 1;
17     }
18 }
19
20 int find(int x) {
21     if (parent[x] != x) {
22         int old_parent = parent[x];
23         find(old_parent);
24         distance_[x] += distance_[old_parent];
25         parent[x] = parent[old_parent];
26     }
27     return parent[x];
28 }
29
30 void merge(int i, int j) {
31     int x = find(i);
32     int y = find(j);
33     if (x != y) {
34         parent[x] = y;
35         distance_[x] = size_[y];
36         size_[y] += size_[x];
37     }
38 }
39
40 int main() {
41     init();
42     int T;
43     scanf("%d", &T);
44     char op;
45     int a, b;
46     while (T--) {
47         scanf(" %c %d %d", &op, &a, &b);
48         if (op == 'M') {
```

```

49         merge(a, b);
50     } else {
51         int x_root = find(a);
52         int y_root = find(b);
53         if (x_root != y_root) {
54             printf("-1\n");
55         } else {
56             int diff = abs(distance[a] - distance[b]);
57             printf("%d\n", diff - 1);
58         }
59     }
60 }
61 return 0;
62 }

```

C 在二叉树上移动

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  string handle(string &s) {
6      string result;
7      int i = 0;
8      while (i < s.size()) {
9          if (i == 0) {
10             result.push_back(s[i]);
11             i++;
12             continue;
13         }
14         if (s[i] == 'U') {
15             if (result.back() == 'L' || result.back() == 'R') {
16                 result.pop_back();
17                 i++;
18                 continue;
19             }
20         }
21
22         result.push_back(s[i]);
23         i++;
24     }
25
26     return result;
27 }
28
29 int main(void) {

```

```

30     ios::sync_with_stdio(false);
31     cin.tie(0);
32     unsigned long long N, X;
33     cin >> N >> X;
34     string s;
35     cin >> s;
36     s = handle(s);
37     for (char c : s) {
38         if (c == 'U'){
39             X /= 2;
40         } else if (c == 'L') {
41             X *= 2;
42         } else {
43             X = X * 2 + 1;
44         }
45     }
46     cout << X << endl;
47 }

```

D 二叉搜索树

```

1  #include <iostream>
2  using namespace std;
3
4  struct Node {
5      int value;
6      Node* left;
7      Node* right;
8      Node(int v) : value(v), left(nullptr), right(nullptr) {}
9  };
10
11 Node* createBST(const string& sequence) {
12     if (sequence.empty()) return nullptr;
13
14     Node* root = new Node(sequence[0] - '0');
15
16     for (int i = 1; i < sequence.size(); ++i) {
17         int num = sequence[i] - '0';
18         Node* current = root;
19         Node* previous = nullptr;
20
21         while (current != nullptr) {
22             previous = current;
23             if (num < current->value)
24                 current = current->left;
25             else

```

```

26         current = current->right;
27     }
28
29     Node* newNode = new Node(num);
30     if (num < previous->value)
31         previous->left = newNode;
32     else
33         previous->right = newNode;
34 }
35
36 return root;
37 }
38
39 bool areIdentical(Node* a, Node* b) {
40     if (!a && !b) return true;
41     if (!a || !b) return false;
42
43     return (a->value == b->value) &&
44           areIdentical(a->left, b->left) &&
45           areIdentical(a->right, b->right);
46 }
47
48 int main() {
49     int m;
50     string baseSequence;
51     cin >> m >> baseSequence;
52
53     Node* referenceTree = createBST(baseSequence);
54
55     while (m--) {
56         string testSequence;
57         cin >> testSequence;
58
59         Node* testTree = createBST(testSequence);
60         cout << (areIdentical(referenceTree, testTree) ? "YES" : "NO") <<
endl;
61     }
62
63     return 0;
64 }

```

E The order of a Tree

```

1  #include <iostream>
2  #include <cstring>
3  using namespace std;

```



```
4
5  const int N = 100010;
6
7  int value[N], leftChild[N], rightChild[N], preorderResult[N];
8  int nodeCount, totalNodes;
9
10 void insertNode(int rootIndex, int val) {
11     if (val <= value[rootIndex]) {
12         if (leftChild[rootIndex])
13             insertNode(leftChild[rootIndex], val);
14         else
15             leftChild[rootIndex] = totalNodes;
16     } else {
17         if (rightChild[rootIndex])
18             insertNode(rightChild[rootIndex], val);
19         else
20             rightChild[rootIndex] = totalNodes;
21     }
22 }
23
24 void preorderTraversal(int index) {
25     preorderResult[nodeCount++] = value[index];
26     if (leftChild[index]) preorderTraversal(leftChild[index]);
27     if (rightChild[index]) preorderTraversal(rightChild[index]);
28 }
29
30 int main() {
31     int n;
32     while (cin >> n) {
33         memset(leftChild, 0, sizeof leftChild);
34         memset(rightChild, 0, sizeof rightChild);
35         int rootIndex = -1, x;
36         totalNodes = 0;
37
38         for (int i = 0; i < n; ++i) {
39             cin >> x;
40             if (rootIndex == -1) {
41                 value[++rootIndex] = x;
42             } else {
43                 value[++totalNodes] = x;
44                 insertNode(rootIndex, x);
45             }
46         }
47
48         nodeCount = 0;
49         preorderTraversal(rootIndex);
```

```

50
51     cout << preorderResult[0];
52     for (int i = 1; i < nodeCount; ++i)
53         cout << " " << preorderResult[i];
54     cout << endl;
55 }
56
57 return 0;
58 }

```

F 二叉树

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  struct TreeNode {
6      int val;
7      TreeNode *left;
8      TreeNode *right;
9      TreeNode(int val) : val(val), left(nullptr), right(nullptr) {}
10 };
11
12 TreeNode *traversal(vector<int>& in, vector<int>& pre, int inLeft, int
inRight, int preLeft, int preRight) {
13     if (inLeft >= inRight) {
14         return nullptr;
15     }
16     int rootVal = pre[preLeft];
17     TreeNode *root = new TreeNode(rootVal);
18     if (inRight - inLeft == 1) {
19         return root;
20     }
21     int i = 0;
22     for (i = inLeft; i < inRight; i++) {
23         if (in[i] == rootVal) {
24             break;
25         }
26     }
27
28     root->left = traversal(in, pre, inLeft, i, preLeft + 1, preLeft + i -
inLeft);
29     root->right = traversal(in, pre, i + 1, inRight, preLeft + i - inLeft
+ 1, preRight);
30     return root;
31 }

```

```

32
33     vector<int> v;
34
35     void traversal2(TreeNode *node) {
36         if (node == nullptr) {
37             return;
38         }
39         traversal2(node->left);
40         traversal2(node->right);
41         v.push_back(node->val);
42     }
43
44     int main(void) {
45         ios::sync_with_stdio(false);
46         cin.tie(0);
47
48         int n;
49         while (cin >> n) {
50             v.clear();
51             vector<int> pre(n);
52             for (int i = 0; i < n; i++) {
53                 cin >> pre[i];
54             }
55             vector<int> in(n);
56             for (int i = 0; i < n; i++) {
57                 cin >> in[i];
58             }
59
60             TreeNode *root = traversal(in, pre, 0, in.size(), 0, pre.size());
61             traversal2(root);
62             for (int i = 0; i < n - 1; i++) {
63                 cout << v[i] << " ";
64             }
65
66             cout << v[n - 1] << endl;
67         }
68     }

```

G 线索二叉树的中序遍历

```

1     #include<string>
2     #include<iostream>
3     #include <queue>
4     #include <algorithm>
5     #include <sstream>
6     using namespace std;

```

```

7
8 struct TreeNode {
9     int val;
10    TreeNode *lchild;
11    TreeNode *rchild;
12    int ltag;
13    int rtag;
14    TreeNode(int x) : val(x), ltag(0), rtag(0), lchild(NULL), rchild(NULL)
15    {}
16    };
17
18 class Solution {
19 public:
20     void inorderthread(TreeNode * thr) {
21         TreeNode * p = thr;
22         while (p) {
23             while(p->ltag == 0 && p->lchild) p = p->lchild;
24             cout << p->val << endl;
25             while (p->rtag == 1 && p->rchild) {
26                 p = p->rchild;
27                 cout << p->val << endl;
28             }
29             p = p->rchild;
30         }
31     }
32 };
33
34
35
36
37 void trimLeftTrailingSpaces(string &input) {
38     input.erase(input.begin(), find_if(input.begin(), input.end(), [](int
ch) {
39         return !isspace(ch);
40     }));
41 }
42
43 void trimRightTrailingSpaces(string &input) {
44     input.erase(find_if(input.rbegin(), input.rend(), [](int ch) {
45         return !isspace(ch);
46     }).base(), input.end());
47 }
48
49
50 void inThread(TreeNode *p, TreeNode *&pre){

```

```

51     if(p){
52         inThread(p->lchild, pre); //线索化左子树
53         if(p->lchild == NULL){
54             p->lchild = pre;
55             p->ltag = 1;
56         }
57         if(pre != NULL && pre->rchild == NULL){
58             pre->rchild = p;
59             pre->rtag = 1;
60         }
61         pre = p;
62         inThread(p->rchild, pre);
63     }
64 }
65
66 void createThreadTree(TreeNode * root){
67     TreeNode * pre = NULL;
68     if(root){
69         inThread(root, pre);
70     }
71 }
72
73
74 TreeNode* stringToTreeNode(string input) {
75     trimLeftTrailingSpaces(input);
76     trimRightTrailingSpaces(input);
77     input = input.substr(1, input.length() - 2);
78     if (!input.size()) {
79         return nullptr;
80     }
81
82     string item;
83     stringstream ss;
84     ss.str(input);
85
86     getline(ss, item, ',');
87     TreeNode* root = new TreeNode(stoi(item));
88     queue<TreeNode*> nodeQueue;
89     nodeQueue.push(root);
90
91     while (true) {
92         TreeNode* node = nodeQueue.front();
93         nodeQueue.pop();
94
95         if (!getline(ss, item, ',')) {
96             break;

```

```

97         }
98
99         trimLeftTrailingSpaces(item);
100        if (item != "null") {
101            int leftNumber = stoi(item);
102            node->lchild = new TreeNode(leftNumber);
103            nodeQueue.push(node->lchild);
104        }
105
106        if (!getline(ss, item, ',')) {
107            break;
108        }
109
110        trimLeftTrailingSpaces(item);
111        if (item != "null") {
112            int rightNumber = stoi(item);
113            node->rchild = new TreeNode(rightNumber);
114            nodeQueue.push(node->rchild);
115        }
116    }
117    return root;
118 }
119
120 int main() {
121     string line;
122     getline(cin, line);
123     TreeNode* root = stringToTreeNode(line);
124     createThreadTree(root);
125     Solution().inorderthread(root);
126     //TravIn_Thread_BT(root);
127     return 0;
128 }

```

Week 10 课上

A Trees on the level

```

1  #include<iostream>
2  #include<cstring>
3  #include<queue>
4  #include<stdio.h>
5  #include<string>
6  using namespace std;

```

```

7  #define maxn 300
8
9  struct node {
10     int val;
11     bool haveVal;
12     int l,r;
13 } tree[300];
14 int root,len;
15 bool ok;
16
17 bool read_input() {
18     string s;
19     ok = true;
20     root = 0,len=1,tree[0].l=-1,tree[0].r=-1,tree[0].haveVal=false;
21     for (;;) {
22         if (!(cin>>s))return false;
23         if (s=="()")return true;
24         int v,now=root;
25         sscanf(&s[1], "%d", &v);
26         int i=s.find(',')+1;
27         while (s[i] != ')') {
28             if (s[i] == 'L') {
29                 if (tree[now].l == -1) {
30                     tree[now].l = len++;
31
32                     tree[tree[now].l].l=-1,tree[tree[now].l].r=-1,tree[tree[now].l].haveVal=false;
33                     now = tree[now].l;
34                 }
35                 else{
36                     if (tree[now].r == -1){
37                         tree[now].r = len++;
38
39                         tree[tree[now].r].l=-1,tree[tree[now].r].r=-1,tree[tree[now].r].haveVal=false;
40                         now = tree[now].r;
41                     }
42                     i++;
43                 }
44             if (tree[now].haveVal) {
45                 ok = false;
46             }
47             tree[now].val = v;
48             tree[now].haveVal = true;
49         }
50     }

```

```

51
52 int main() {
53     while (read_input()) {
54         if (!ok) {
55             cout << "not complete" << endl;
56             continue;
57         }
58         string s="";
59         queue<int> q;
60         q.push(root);
61         int now;
62         while (q.size()) {
63             now = q.front(); q.pop();
64             if (!tree[now].haveVal) {
65                 s = "not complete";
66                 break;
67             }
68             else {
69                 if (s == "")
70                     s += to_string(tree[now].val);
71                 else
72                     s += " "+to_string(tree[now].val);
73             }
74             if(tree[now].l!=-1)q.push(tree[now].l);
75             if(tree[now].r!=-1)q.push(tree[now].r);
76         }
77         cout << s << endl;
78     }
79     return 0;
80 }

```

C Shaolin

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <vector>
5  #include <set>
6  #include <algorithm>
7  #include <cmath>
8  #include <map>
9  using namespace std;
10
11 #define MAX_N 100006
12 int n;
13 map<int, int> distanceMap;

```



```

14 map<int, int>::iterator lowerIt, prevIt;
15
16 int main() {
17     while (scanf("%d", &n) == 1 && n) {
18         distanceMap.clear();
19         distanceMap[1000000000] = 1;
20
21         for (int i = 0; i < n; i++) {
22             int x, y;
23             scanf("%d %d", &x, &y);
24
25             lowerIt = distanceMap.lower_bound(y);
26
27             if (lowerIt == distanceMap.begin()) {
28                 printf("%d %d\n", x, lowerIt->second);
29             } else {
30                 prevIt = lowerIt;
31                 prevIt--;
32
33                 if (abs(y - prevIt->first) <= abs(y - lowerIt->first)) {
34                     printf("%d %d\n", x, prevIt->second);
35                 } else {
36                     printf("%d %d\n", x, lowerIt->second);
37                 }
38             }
39
40             distanceMap[y] = x;
41         }
42     }
43     return 0;
44 }

```

D Robotic Sort

```

1  #include<stdio.h>
2  #include<algorithm>
3  #include<string.h>
4  using namespace std;
5  #define N 500006
6  #define lc (tr[id].c[0])
7  #define rc (tr[id].c[1])
8  #define key (tr[tr[root].c[1]].c[0])
9  struct tree
10 {
11     int fa,sum,c[2],lz,v;
12 }tr[N];

```

```
13 struct point
14 {
15     int v,id;
16     bool operator<(const point a)const
17     {
18         if(a.v==v)return id<a.id;
19         else return v<a.v;
20     }
21 }so[N/5];
22 int tot,root,n;
23 int xia[N];
24 int newpoint(int d,int f)
25 {
26     tr[tot].sum=1;
27     tr[tot].v=d;
28     tr[tot].c[0]=tr[tot].c[1]=-1;
29     tr[tot].lz=0;
30     tr[tot].fa=f;
31     return tot++;
32 }
33 void push(int id)
34 {
35     int lsum,rsum;
36     if(lc==-1)lsum=0;
37     else lsum=tr[lc].sum;
38     if(rc==-1)rsum=0;
39     else rsum=tr[rc].sum;
40     tr[id].sum=lsum+rsum+1;
41 }
42 int build(int l,int r,int v)
43 {
44     if(r<l)return -1;
45     int mid=(r+l)>>1;
46     int ro=newpoint(mid,v);
47     xia[mid]=ro;
48     tr[ro].c[0]=build(l,mid-1,ro);
49     tr[ro].c[1]=build(mid+1,r,ro);
50     push(ro);
51     return ro;
52 }
53 void lazy(int id)
54 {
55     if(tr[id].lz)
56     {
57         swap(lc,rc);
58         tr[lc].lz^=1;
```

```
59         tr[rc].lz^=1;
60         tr[id].lz=0;
61     }
62 }
63
64 void xuanzhuan(int x,int k)
65 {
66     if(tr[x].fa==-1)return ;
67     int fa=tr[x].fa;
68     int w;
69     lazy(fa);
70     lazy(x);
71     tr[fa].c[!k]=tr[x].c[k];
72     if(tr[x].c[k]!=-1)tr[tr[x].c[k]].fa=fa;
73     tr[x].fa=tr[fa].fa;
74     tr[x].c[k]=fa;
75     if(tr[fa].fa!=-1)
76     {
77         w=tr[tr[fa].fa].c[1]==fa;
78         tr[tr[fa].fa].c[w]=x;
79     }
80     tr[fa].fa=x;
81     push(fa);
82     push(x);
83 }
84
85 void splay(int x,int goal)
86 {
87     if(x==-1)return ;
88     lazy(x);
89     while(tr[x].fa!=goal)
90     {
91         int y=tr[x].fa;
92         lazy(tr[y].fa);
93         lazy(y);
94         lazy(x);
95         bool w=(x==tr[y].c[1]);
96         if(tr[y].fa!=goal&&w==(y==tr[tr[y].fa].c[1]))xuanzhuan(y,!w);
97         xuanzhuan(x,!w);
98     }
99     if(goal==-1)root=x;
100    push(x);
101 }
102 int next(int id)
103 {
104     lazy(id);
```

```

105     int p=tr[id].c[1];
106     if(p==-1)return id;
107     lazy(p);
108     while(tr[p].c[0]!=-1)
109     {
110         p=tr[p].c[0];
111         lazy(p);
112     }
113     return p;
114 }
115 int main()
116 {
117     while(scanf("%d",&n),n)
118     {
119         for(int i=1;i<=n;i++)
120         {
121             scanf("%d",&so[i].v);
122             so[i].id=i;
123         }
124         sort(so+1,so+n+1);
125         so[0].id=0;
126         tot=0;
127         int d,l;
128         root=build(0,n+1,-1);
129         for(int i=1;i<=n;i++)
130         {
131             int ro=xia[so[i].id];
132             int ne;
133             splay(ro,-1);
134             d=tr[tr[root].c[0]].sum;
135             l=xia[so[i-1].id];
136             ne=next(ro);
137             splay(l,-1);
138             splay(ne,root);
139             lazy(root);
140             lazy(tr[root].c[1]);
141             tr[key].lz^=1;
142             if(i!=1)printf(" ");
143             printf("%d",d);
144         }
145         printf("\n");
146     }
147     return 0;
148 }

```

E Looploop 可能过不了

```
1  #include <functional>
2  #include <iostream>
3  #include <cstdio>
4  #include <cstdlib>
5  #include <cstring>
6  #include <algorithm>
7  #include <map>
8  #include <cmath>
9  using namespace std;
10 typedef long long ll;
11 typedef long double ld;
12 #define key_value ch[ch[root][1]][0]
13 const int maxn = 200010;
14
15 int ch[maxn][2];
16 int pre[maxn],siz[maxn],num[maxn];
17 int rev[maxn],key[maxn];
18 int add[maxn];
19 int Min[maxn],a[maxn];
20 int tot,tp;
21 int root,n;
22 void push_up(int r)
23 {
24     int lson = ch[r][0],rson = ch[r][1];
25     siz[r] = siz[lson] + siz[rson] + 1;
26     Min[r] = min(key[r],min(Min[lson],Min[rson]));
27 }
28
29 void update_add(int r,int val)
30 {
31     if(!r) return;
32     key[r] += val;
33     add[r] += val;
34     Min[r] += val;
35 }
36
37 void inOrder(int r)
38 {
39     if(!r)
40         return;
41     inOrder(ch[r][0]);
42     printf("%d ",key[r]);
43     inOrder(ch[r][1]);
44 }
45
46 void debug()
```

```
47 {
48     inOrder(root);
49     cout <<endl;
50 }
51
52 void update_rev(int r)
53 {
54     if(!r)return ;
55     swap(ch[r][0],ch[r][1]);
56     rev[r] ^= 1;
57 }
58
59 void push_down(int r)
60 {
61     if(rev[r])
62     {
63         update_rev(ch[r][0]);
64         update_rev(ch[r][1]);
65         rev[r] = 0;
66     }
67     if(add[r])
68     {
69         update_add(ch[r][0],add[r]);
70         update_add(ch[r][1],add[r]);
71         add[r] = 0;
72     }
73 }
74
75
76 void NewNode(int &r,int far,int k)
77 {
78     r = ++tot;
79     pre[r] = far;
80     ch[r][0] = ch[r][1] = 0;
81     siz[r] = 1;
82     Min[r] = k;
83     key[r] = k;
84     rev[r] = 0;
85     add[r] = 0;
86 }
87
88
89 void rotat(int x,int kind)
90 {
91     int y = pre[x];
92     push_down(y);
```

```

93     push_down(x);
94     ch[y][!kind] = ch[x][kind];
95     pre[ch[x][kind]] = y;
96     if(pre[y])
97         ch[pre[y]][ch[pre[y]][1]==y] = x;
98     pre[x] = pre[y];
99     ch[x][kind] = y;
100    pre[y] = x;
101    push_up(y);
102 }
103
104 void build(int &x,int l,int r,int far)
105 {
106     if(l > r) return ;
107     int mid = (l+r) >>1;
108     NewNode(x,far,a[mid]);
109     build(ch[x][0],l,mid-1,x);
110     build(ch[x][1],mid+1,r,x);
111     push_up(x);
112 }
113
114 void splay(int r,int goal)
115 {
116     push_down(r);
117     while(pre[r] != goal)
118     {
119         if(pre[pre[r]] == goal)
120         {
121             push_down(pre[r]);
122             push_down(r);
123             rotat(r,ch[pre[r]][0] == r);
124         }
125         else
126         {
127             push_down(pre[pre[r]]);
128             push_down(pre[r]);
129             push_down(r);
130             int y = pre[r];
131             int kind = ch[pre[y]][0] == y;
132             if(ch[y][kind] == r)
133             {
134                 rotat(r,!kind);
135                 rotat(r,kind);
136             }
137             else
138             {

```

```

139         rotat(y,kind);
140         rotat(r,kind);
141     }
142 }
143 }
144 push_up(r);
145 if(goal == 0)
146     root = r;
147 }
148
149
150 int get_kth(int r,int k)
151 {
152     push_down(r);
153     int t = siz[ch[r][0]] + 1;
154     if(k == t)return r;
155     if(t > k) return get_kth(ch[r][0],k);
156     else return get_kth(ch[r][1],k-t);
157 }
158
159 int get_next(int r)
160 {
161     push_down(r);
162     if(ch[r][1] == 0)return -1;
163     r = ch[r][1];
164     while(ch[r][0])
165     {
166         r = ch[r][0];
167         push_down(r);
168     }
169     return r;
170 }
171
172 void Reverse(int l,int r)
173 {
174     splay(get_kth(root,l),0);
175     splay(get_kth(root,r+2),root);
176     update_rev(key_value);
177     push_up(ch[root][1]);
178     push_up(root);
179 }
180
181 void Add(int l,int r,int val)
182 {
183     splay(get_kth(root,l),0);
184     splay(get_kth(root,r+2),root);

```



```
185     update_add(key_value, val);
186     push_up(ch[root][1]);
187     push_up(root);
188 }
189
190 void ini(int n)
191 {
192     tot = root = 0;
193     ch[root][0] = ch[root][1] = pre[root] = siz[root] = num[root] = 0;
194     Min[root] = 0x3f3f3f3f;
195     rev[root] = add[root] = 0;
196     NewNode(root, 0, -1);
197     NewNode(ch[root][1], root, -1);
198     for(int i=1; i <= n; i++)
199     {
200         scanf("%d", &a[i]);
201     }
202     build(key_value, 1, n, ch[root][1]);
203
204     push_up(ch[root][1]);
205     push_up(root);
206 }
207
208 int get_min(int r)
209 {
210     push_down(r);
211     while(ch[r][0])
212     {
213         r = ch[r][0];
214         push_down(r);
215     }
216     return r;
217 }
218
219 int Delete(int r)
220 {
221     int t = get_kth(root, r+1);
222     splay(t, 0);
223     if(ch[root][0] == 0 || ch[root][1] == 0)
224     {
225         root = ch[root][0] + ch[root][1];
226         pre[root] = 0;
227         return key[t];
228     }
229     int k = get_min(ch[root][1]);
230     splay(k, root);
```

```

231     ch[ch[root][1]][0] = ch[root][0];
232     root = ch[root][1];
233     pre[ch[root][0]] = root;
234     pre[root] = 0;
235     push_up(root);
236     n--;
237     return key[t];
238 }
239
240 void Insert(int x,int y)
241 {
242     splay(get_kth(root,x),0);
243     splay(get_kth(root,x+1),root);
244     NewNode(key_value,ch[root][1],y);
245     push_up(ch[root][1]);
246     push_up(root);
247     n++;
248 }
249
250 void Move(int x)
251 {
252     if(x == 1)
253     {
254         int t = Delete(n);
255         Insert(1,t);
256         // debug();
257     }
258     else
259     {
260         int t = Delete(1);
261         Insert(n+1,t);
262     }
263 }
264
265
266 int main()
267 {
268     int p,k1,k2;
269     int cas = 1;
270     while(scanf("%d%d%d%d",&n,&p,&k1,&k2) != EOF)
271     {
272         if(!n && !p && !k1 && !k2)
273             break;
274         printf("Case #%d:\n",cas++);
275         ini(n);
276         char opr[10];

```

```

277         int x;
278         for(int i =1; i <= p; i++)
279         {
280             scanf("%s",opr);
281             if(opr[0] == 'a')
282             {
283                 scanf("%d",&x);
284                 Add(1,k2,x);
285             }
286             else if(opr[0] == 'm')
287             {
288                 scanf("%d",&x);
289                 Move(x);
290             }
291             else if(opr[0] == 'r')
292             {
293                 Reverse(1,k1);
294             }
295             else if(opr[0] == 'q')
296             {
297                 printf("%d\n",key[get_kth(root,2)]);
298             }
299             else if(opr[0] == 'i')
300             {
301                 scanf("%d",&x);
302                 Insert(2,x);
303             }
304             else if(opr[0] == 'd')
305             {
306                 Delete(1);
307             }
308         }
309     }
310     return 0;
311 }

```

Week 10 课下

B 普通平衡树

```

1  #include <iostream>
2  #include <cstring>
3  using namespace std;

```

```

4
5  const int N = 1e5 + 10, INF = 1e8;
6  struct Node {
7      int son[2];
8      int key, val;
9      int cnt, size;
10 } tr[N];
11 int n, root, idx;
12
13 int get_node(int key) {
14     tr[++idx].key = key;
15     tr[idx].val = rand();
16     tr[idx].cnt = tr[idx].size = 1;
17     return idx;
18 }
19
20 void pushup(int p) {
21     tr[p].size = tr[tr[p].son[0]].size + tr[tr[p].son[1]].size +
tr[p].cnt;
22 }
23
24 void rotate(int& p, int d) {
25     int q = tr[p].son[d ^ 1];
26     tr[p].son[d ^ 1] = tr[q].son[d], tr[q].son[d] = p, p = q;
27     pushup(tr[p].son[d]);
28 }
29
30 void insert(int& p, int key) {
31     if (!p) p = get_node(key);
32     else if (key < tr[p].key) {
33         insert(tr[p].son[0], key);
34         if (tr[tr[p].son[0]].val > tr[p].val) rotate(p, 1);
35     } else if (key > tr[p].key) {
36         insert(tr[p].son[1], key);
37         if (tr[tr[p].son[1]].val > tr[p].val) rotate(p, 0);
38     } else tr[p].cnt++;
39
40     pushup(p);
41 }
42
43 void remove(int& p, int key) {
44     if (!p) return;
45     if (key < tr[p].key) remove(tr[p].son[0], key);
46     else if (key > tr[p].key) remove(tr[p].son[1], key);
47     else if (tr[p].cnt > 1) tr[p].cnt--;
48     else if (!tr[p].son[0] && !tr[p].son[1]) p = 0;

```

```

49     else if (!tr[p].son[1] || (tr[p].son[0] && tr[tr[p].son[0]].val >
tr[tr[p].son[1]].val)) {
50         rotate(p, 1);
51         remove(tr[p].son[1], key);
52     } else {
53         rotate(p, 0);
54         remove(tr[p].son[0], key);
55     }
56
57     pushup(p);
58 }
59
60 int get_rank_by_key(int p, int key) {
61     if (!p) return 0;
62     if (key < tr[p].key) return get_rank_by_key(tr[p].son[0], key);
63     else if (key > tr[p].key) return tr[tr[p].son[0]].size + tr[p].cnt +
get_rank_by_key(tr[p].son[1], key);
64     else return tr[tr[p].son[0]].size + 1;
65 }
66
67 int get_key_by_rank(int p, int rank) {
68     if (!p) return INF;
69     else if (rank <= tr[tr[p].son[0]].size) return
get_key_by_rank(tr[p].son[0], rank);
70     else if (rank > tr[tr[p].son[0]].size + tr[p].cnt) return
get_key_by_rank(tr[p].son[1], rank - tr[tr[p].son[0]].size - tr[p].cnt);
71     else return tr[p].key;
72 }
73
74 int get_prev(int p, int key) {
75     if (!p) return -INF;
76     if (key <= tr[p].key) return get_prev(tr[p].son[0], key);
77     else return max(tr[p].key, get_prev(tr[p].son[1], key));
78 }
79
80 int get_next(int p, int key) {
81     if (!p) return INF;
82     if (key >= tr[p].key) return get_next(tr[p].son[1], key);
83     else return min(tr[p].key, get_next(tr[p].son[0], key));
84 }
85
86 int main() {
87     scanf("%d", &n);
88     while (n--) {
89         int op, x;
90         scanf("%d%d", &op, &x);

```

```

91         if (op == 1) insert(root, x);
92         else if (op == 2) remove(root, x);
93         else if (op == 3) printf("%d\n", get_rank_by_key(root, x));
94         else if (op == 4) printf("%d\n", get_key_by_rank(root, x));
95         else if (op == 5) printf("%d\n", get_prev(root, x));
96         else if (op == 6) printf("%d\n", get_next(root, x));
97     }
98
99     return 0;
100 }

```

D 对称二叉树

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N=1000010;
4  long long a[N],l[N],r[N],n,ans=0;
5  inline bool dfs(long long l2,long long r2){
6      if(l2==--1&&r2==--1)
7          return true;
8      if(l2==--1||r2==--1||a[l2]!=a[r2])
9          return false;
10     return dfs(l[l2],r[r2])&&dfs(l[r2],r[l2]);
11 }
12 inline long long get(long long x){
13     if(x==--1)
14         return 0;
15     else
16         return get(l[x])+get(r[x])+1;
17 }
18 int main(){
19     cin>>n;
20     for(long long i=1;i<=n;i++)
21         cin>>a[i];
22     for(long long i=1;i<=n;i++)
23         cin>>l[i]>>r[i];
24     for(long long i=1;i<=n;i++)
25         if(dfs(i,i))
26             ans=max(ans,get(i));
27     cout<<ans;
28     return 0;
29 }

```

E 完全二叉树的权值

```

1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
4  #include <cstring>
5
6  using namespace std;
7
8  typedef long long LL;
9
10 const int N = 1e5 + 5;
11
12 LL tr[N];
13 LL cnt[50];
14
15 int main()
16 {
17     int n, k = 1; scanf("%d", &n);
18     for(int i = 1; i <= n; ++ i) scanf("%lld", &tr[i]);
19
20     for(int i = 1, j = 1; i <= n; ++ i)
21     {
22         cnt[k] += tr[i];
23
24         if(i - j >= (1 << (k - 1)) - 1)
25         {
26             j = i + 1;
27             k ++;
28         }
29     }
30
31     int ans = 0, maxx = 1-1e6;
32     for(int i = 1; i <= k ; ++ i)
33         if(cnt[i] > maxx)
34         {
35             maxx = cnt[i];
36             ans = i;
37         }
38     cout << ans << endl;
39     return 0;
40
41 }

```

A Root of AVL Tree

```
1  #include <iostream>
2  using namespace std;
3
4  struct Node {
5      int key;
6      Node *left, *right;
7      int height;
8      Node(int k) : key(k), left(nullptr), right(nullptr), height(1) {}
9  };
10
11 int height(Node* n) {
12     return n ? n->height : 0;
13 }
14
15 void updateHeight(Node* n) {
16     n->height = max(height(n->left), height(n->right)) + 1;
17 }
18
19 int balanceFactor(Node* n) {
20     return height(n->left) - height(n->right);
21 }
22
23 Node* rightRotate(Node* y) {
24     Node* x = y->left;
25     Node* T2 = x->right;
26     x->right = y;
27     y->left = T2;
28     updateHeight(y);
29     updateHeight(x);
30     return x;
31 }
32
33 Node* leftRotate(Node* x) {
34     Node* y = x->right;
35     Node* T2 = y->left;
36     y->left = x;
37     x->right = T2;
38     updateHeight(x);
39     updateHeight(y);
40     return y;
41 }
42
43 Node* insert(Node* node, int key) {
44     if (!node)
```



```

45         return new Node(key);
46     if (key < node->key)
47         node->left = insert(node->left, key);
48     else
49         node->right = insert(node->right, key);
50
51     updateHeight(node);
52
53     int bf = balanceFactor(node);
54
55     if (bf > 1 && key < node->left->key)
56         return rightRotate(node);
57     if (bf < -1 && key > node->right->key)
58         return leftRotate(node);
59     if (bf > 1 && key > node->left->key) {
60         node->left = leftRotate(node->left);
61         return rightRotate(node);
62     }
63     if (bf < -1 && key < node->right->key) {
64         node->right = rightRotate(node->right);
65         return leftRotate(node);
66     }
67
68     return node;
69 }
70
71 int main() {
72     ios::sync_with_stdio(false);
73     cin.tie(nullptr);
74
75     int N;
76     cin >> N;
77     Node* root = nullptr;
78     for (int i = 0; i < N; i++) {
79         int x;
80         cin >> x;
81         root = insert(root, x);
82     }
83     if (root)
84         cout << root->key << "\n";
85     return 0;
86 }

```

B B树的插入遍历和查找

```
1 #include<iostream>
```

```

2  #include<vector>
3  using namespace std;
4
5  struct BTreeNode
6  {
7      int *keys; // 元素数组
8      int t; // 最小度t
9      BTreeNode **C; // 子节点数组
10     int n; // 当前节点元素个数
11     bool leaf; // 是否是叶子节点
12
13     BTreeNode(int _t, bool _leaf); // 构造函数
14     void insertNonFull(int k); // 当前节点未满，插入
15     void splitChild(int i, BTreeNode *y); // 分裂节点
16     void traverse(); // 遍历
17     BTreeNode *search(int k); // 查找
18 };
19
20 class BTree
21 {
22     BTreeNode *root; // B树根节点
23     int t; // 最小度t
24 public:
25     BTree(int _t)
26     { root = NULL; t = _t; }
27
28     void traverse()
29     { if (root != NULL) root->traverse(); }
30
31     BTreeNode* search(int k)
32     { return (root == NULL)? NULL : root->search(k); }
33
34     void insert(int k);
35 };
36
37 BTreeNode::BTreeNode(int t1, bool leaf1)
38 {
39     t = t1;
40     leaf = leaf1;
41
42     keys = new int[2*t-1];
43     C = new BTreeNode *[2*t];
44
45     n = 0;
46 }
47

```

```

48 void BTreeNode::traverse()
49 {
50     int i;
51     for (i = 0; i < n; i++)
52     {
53         if (leaf == false)
54             C[i]->traverse();
55         cout << keys[i] << endl;
56     }
57     if (leaf == false)
58         C[i]->traverse();
59 }
60
61 BTreeNode *BTreeNode::search(int k)
62 {
63     int i = 0;
64     while (i < n && k > keys[i])
65         i++;
66     if (i < n && keys[i] == k)
67         return this;
68     if (leaf)
69         return NULL;
70     return C[i]->search(k);
71 }
72
73 void BTree::insert(int k)
74 {
75     if (root == NULL)
76     {
77         root = new BTreeNode(t, true);
78         root->keys[0] = k;
79         root->n = 1;
80     }
81     else
82     {
83         if (root->n == 2*t-1)
84         {
85             BTreeNode *s = new BTreeNode(t, false);
86             s->C[0] = root;
87             s->splitChild(0, root);
88             int i = 0;
89             if (s->keys[0] < k)
90                 i++;
91             s->C[i]->insertNonFull(k);
92             root = s;
93         }

```

```

94         else
95             root->insertNonFull(k);
96     }
97 }
98
99 void BTreeNode::insertNonFull(int k)
100 {
101     int i = n-1;
102
103     if (leaf == true)
104     {
105         while (i >= 0 && keys[i] > k)
106         {
107             keys[i+1] = keys[i];
108             i--;
109         }
110
111         keys[i+1] = k;
112         n = n+1;
113     }
114     else
115     {
116         while (i >= 0 && keys[i] > k)
117             i--;
118
119         if (C[i+1]->n == 2*t-1)
120         {
121             splitChild(i+1, C[i+1]);
122
123             if (keys[i+1] < k)
124                 i++;
125         }
126         C[i+1]->insertNonFull(k);
127     }
128 }
129
130 void BTreeNode::splitChild(int i, BTreeNode *y)
131 {
132     BTreeNode *z = new BTreeNode(y->t, y->leaf);
133     z->n = t - 1;
134
135     for (int j = 0; j < t-1; j++)
136         z->keys[j] = y->keys[j+t];
137
138     if (y->leaf == false)
139     {

```

```
140         for (int j = 0; j < t; j++)
141             z->C[j] = y->C[j+t];
142     }
143
144     y->n = t - 1;
145     for (int j = n; j >= i+1; j--)
146         C[j+1] = C[j];
147
148     C[i+1] = z;
149
150     for (int j = n-1; j >= i; j--)
151         keys[j+1] = keys[j];
152
153     keys[i] = y->keys[t-1];
154
155     n = n + 1;
156 }
157
158 int main()
159 {
160     int t, n, m;
161     cin>>t>>n>>m;
162     BTree tree(t);
163     vector<int> a(n);
164     for (int i = 0; i < n; i++)
165     {
166         cin>>a[i];
167         tree.insert(a[i]);
168     }
169
170     tree.traverse();
171
172     for (int i = 0; i < m; i++)
173     {
174         int q;
175         cin>>q;
176         BTreeNode *res = tree.search(q);
177         if (res)
178         {
179             for (int j = 0; j < res->n; j++)
180             {
181                 cout << res->keys[j];
182                 if (j < res->n - 1) cout << ' ';
183             }
184             cout << endl;
185         }
```

```
186     }
187     return 0;
188 }
```

C 邻接矩阵的使用

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main() {
6      ios::sync_with_stdio(false);
7      cin.tie(nullptr);
8      int n, m;
9      cin >> n >> m;
10
11     vector<vector<int>> adj_matrix(n, vector<int>(n, 0));
12
13     for (int k = 0; k < m; ++k) {
14         int type, u, v;
15         cin >> type >> u >> v;
16         if (type == 0) {
17             adj_matrix[u][v] = 1;
18         } else {
19             adj_matrix[u][v] = 1;
20             adj_matrix[v][u] = 1;
21         }
22     }
23
24     for (int i = 0; i < n; ++i) {
25         for (int j = 0; j < n; ++j) {
26             cout << adj_matrix[i][j] << (j == n - 1 ? "" : " ");
27         }
28         cout << "\n";
29     }
30
31     return 0;
32 }
```

D 邻接表的使用

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int MAXN = 100 + 5;
```

```

5  vector<int> G[MAXN];
6
7  int main() {
8      ios::sync_with_stdio(false);
9      cin.tie(nullptr);
10
11     int n, m;
12     cin >> n >> m;
13     int a, x, y;
14     for (int i = 0; i < m; i++) {
15         cin >> a >> x >> y;
16         if (a == 0) {
17             G[x].push_back(y);
18         } else {
19             G[x].push_back(y);
20             G[y].push_back(x);
21         }
22     }
23
24     for (int i = 0; i < n; i++) {
25         cout << i << ":";
26         for (int j = (int)G[i].size() - 1; j >= 0; j--) {
27             cout << " " << G[i][j];
28         }
29         cout << "\n";
30     }
31
32     return 0;
33 }

```

E 画图游戏

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  int main() {
4      ios::sync_with_stdio(false);
5      cin.tie(nullptr);
6      int n;
7      cin >> n;
8      vector<int> deg(n);
9      long long sum = 0;
10     for (int i = 0; i < n; i++) {
11         cin >> deg[i];
12         sum += deg[i];
13     }
14     if (sum % 2) {

```

```

15         cout << "None\n";
16         return 0;
17     }
18     vector<vector<int>> adj(n, vector<int>(n, 0));
19     vector<pair<int,int>> v(n);
20     for (int i = 0; i < n; i++) {
21         v[i] = {deg[i], i};
22     }
23     for (int round = 0; round < n; round++) {
24         sort(v.begin() + round, v.end(), [&](auto &a, auto &b) {
25             return a.first > b.first;
26         });
27         int d = v[round].first;
28         int u = v[round].second;
29         if (d < 0 || d > n - 1 - round) {
30             cout << "None\n";
31             return 0;
32         }
33         for (int k = 1; k <= d; k++) {
34             int vi = v[round + k].second;
35             if (v[round + k].first <= 0) {
36                 cout << "None\n";
37                 return 0;
38             }
39             adj[u][vi] = adj[vi][u] = 1;
40             v[round + k].first--;
41         }
42         v[round].first = 0;
43     }
44     for (int i = 0; i < n; i++) {
45         if (v[i].first != 0) {
46             cout << "None\n";
47             return 0;
48         }
49     }
50     for (int i = 0; i < n; i++) {
51         for (int j = 0; j < n; j++) {
52             cout << adj[i][j] << (j + 1 < n ? ' ' : '\n');
53         }
54     }
55     return 0;
56 }

```


也就多了一题

A AVL Tree

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int a[46], n;
5
6  int main() {
7      a[0] = 1;
8      a[1] = 2;
9      for (int i = 2; i <= 45; i++)
10         a[i] = a[i - 1] + a[i - 2] + 1;
11     while (~scanf("%d", &n), n) {
12         int ans = 0;
13         while (a[ans] <= n) ans++;
14         printf("%d\n", --ans);
15     }
16     return 0;
17 }
```

Week 13 课上

A 找出星型图的中心节点

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main(void) {
6      int n, m;
7      int count[100055] = {0};
8      cin >> n >> m;
9
10     for (int i = 0; i < m; i++) {
11         int a, b;
12         cin >> a >> b;
13         count[a]++;
14         count[b]++;
15         if (count[a] == m) {
16             cout << a;
17             return 0;
18         }
19     }
```

```

18         }
19         if (count[b] == m) {
20             cout << b;
21             return 0;
22         }
23     }
24     for (int i = 1; i <= n; i++) {
25         if (count[i] == n - 1) {
26             cout << i;
27             return 0;
28         }
29     }
30 }

```

B 修建大桥

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  vector<int> graph[1000];
6  bool vst[1000];
7
8  void dfs(int node) {
9      vst[node] = true;
10
11     for (int i = 0; i < graph[node].size(); i++) {
12         int nextNode = graph[node][i];
13
14         if (!vst[nextNode]) {
15             dfs(nextNode);
16         }
17     }
18 }
19
20 int main() {
21     int n, m;
22     memset(vst, false, sizeof(vst));
23     cin >> n >> m;
24
25     for (int i = 0; i < m; i++) {
26         int a, b;
27         cin >> a >> b;
28         graph[a].push_back(b);
29         graph[b].push_back(a);
30     }

```

```

31
32     int components = 0;
33     for (int i = 1; i <= n; i++) {
34         if (!vst[i]) {
35             components++;
36             dfs(i);
37         }
38     }
39     cout << components - 1;
40
41     return 0;
42 }

```

C 闯关游戏

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  struct Node {
5      int ki;
6      vector<int> adj;
7  };
8
9  int main() {
10     int n;
11     cin >> n;
12     vector<Node> nodes(n);
13     for (int i = 0; i < n; ++i) {
14         int ki, m;
15         cin >> ki >> m;
16         vector<int> adj(m);
17         for (int j = 0; j < m; ++j) {
18             cin >> adj[j];
19         }
20         nodes[i] = {ki, adj};
21     }
22
23     if (n == 1) {
24         cout << "Yes" << endl;
25         return 0;
26     }
27
28     vector<vector<int>> adj_forward(n + 1);
29     for (int u = 1; u <= n; ++u) {
30         for (int v : nodes[u - 1].adj) {
31             adj_forward[u].push_back(v);

```

```

32     }
33 }
34
35 vector<vector<int>> adj_backward(n + 1);
36 for (int u = 1; u <= n; ++u) {
37     for (int v : adj_forward[u]) {
38         adj_backward[v].push_back(u);
39     }
40 }
41
42 bool in_reachable_from_start[101] = {false};
43 queue<int> q;
44 q.push(1);
45 in_reachable_from_start[1] = true;
46 while (!q.empty()) {
47     int u = q.front();
48     q.pop();
49     for (int v : adj_forward[u]) {
50         if (!in_reachable_from_start[v]) {
51             in_reachable_from_start[v] = true;
52             q.push(v);
53         }
54     }
55 }
56
57 bool in_can_reach_end[101] = {false};
58 queue<int> q2;
59 q2.push(n);
60 in_can_reach_end[n] = true;
61 while (!q2.empty()) {
62     int v = q2.front();
63     q2.pop();
64     for (int u : adj_backward[v]) {
65         if (!in_can_reach_end[u]) {
66             in_can_reach_end[u] = true;
67             q2.push(u);
68         }
69     }
70 }
71
72 vector<pair<int, int>> edges;
73 for (int u = 1; u <= n; ++u) {
74     if (in_reachable_from_start[u] && in_can_reach_end[u]) {
75         for (int v : nodes[u - 1].adj) {
76             if (in_can_reach_end[v]) {
77                 edges.emplace_back(u, v);

```

```

78         }
79     }
80 }
81 }
82
83 int initial = 100 + nodes[0].ki;
84 if (initial <= 0) {
85     cout << "No" << endl;
86     return 0;
87 }
88
89 vector<int> max_energy(n + 1, INT_MIN);
90 max_energy[1] = initial;
91
92 for (int i = 0; i < n; ++i) {
93     bool updated = false;
94     for (auto& e : edges) {
95         int u = e.first, v = e.second;
96         if (max_energy[u] > 0) {
97             int new_energy = max_energy[u] + nodes[v - 1].ki;
98             if (new_energy > max_energy[v] && new_energy > 0) {
99                 max_energy[v] = new_energy;
100                 updated = true;
101             }
102         }
103     }
104     if (!updated) break;
105 }
106
107 bool has_positive_cycle = false;
108 for (auto& e : edges) {
109     int u = e.first, v = e.second;
110     if (max_energy[u] > 0) {
111         int new_energy = max_energy[u] + nodes[v - 1].ki;
112         if (new_energy > max_energy[v] && new_energy > 0) {
113             has_positive_cycle = true;
114             break;
115         }
116     }
117 }
118
119 if (has_positive_cycle) {
120     cout << "Yes" << endl;
121     return 0;
122 }
123

```

```

124         if (max_energy[n] > 0) {
125             cout << "Yes" << endl;
126         } else {
127             cout << "No" << endl;
128         }
129
130     return 0;
131 }

```

D 圣诞树

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  using i64 = long long;
6  typedef long long ll;
7
8  const int maxn = 5e4 + 6;
9  const int maxm = 5e4 + 6;
10
11 struct edge {
12     int to, len;
13 };
14
15 vector<edge> e[maxn];
16
17 struct node {
18     i64 dis;
19     int num;
20     bool operator>(const node &a) const {
21         return dis > a.dis;
22     }
23 };
24
25 i64 minDis[maxn];
26 bool vis[maxn];
27 priority_queue<node, vector<node>, greater<node>> pq;
28
29 void dijkstra(int n, int s) {
30     for (int i = 1; i <= n; i++) {
31         minDis[i] = 1e10;
32     }
33     minDis[s] = 0;
34     pq.push({0, s});
35     while (!pq.empty()) {

```

```

36         int u = pq.top().num;
37         pq.pop();
38         if (vis[u]) continue;
39         vis[u] = 1;
40         for (edge eg : e[u]) {
41             int v = eg.to;
42             int w = eg.len;
43             if (minDis[v] > minDis[u] + w) {
44                 minDis[v] = minDis[u] + w;
45                 pq.push({minDis[v], v});
46             }
47         }
48     }
49 }
50
51 int main() {
52     ios::sync_with_stdio(false);
53     cin.tie(nullptr);
54
55     int n, m, s;
56     cin >> n >> m;
57     vector<int> a(n + 1);
58     for (int i = 1; i <= n; i++) {
59         cin >> a[i];
60     }
61     s = 1;
62     int u, v, w;
63     while (m--) {
64         cin >> u >> v >> w;
65         e[u].push_back({v, w});
66         e[v].push_back({u, w});
67     }
68     dijkstra(n, s);
69     i64 ans = 0;
70     bool ok = 1;
71     for (int i = 1; i <= n; i++) {
72         if (minDis[i] == 1e10) {
73             ok = 0;
74             break;
75         }
76         ans += a[i] * minDis[i];
77     }
78     if (ok)
79         cout << ans << '\n';
80     else
81         cout << "No Answer\n";

```

```
82
83     return 0;
84 }
```

E 网络延时

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  vector<int> graph[1000];
6  int dist[1000];
7
8  void shortestPath(int start) {
9      memset(dist, -1, sizeof(dist));
10     queue<int> nodes;
11     nodes.push(start);
12     dist[start] = 0;
13
14     while (!nodes.empty()) {
15         int cur = nodes.front();
16         nodes.pop();
17
18         for (int i = 0; i < graph[cur].size(); i++) {
19             int next = graph[cur][i];
20
21             if (dist[next] == -1) {
22                 nodes.push(next);
23                 dist[next] = dist[cur] + 1;
24             }
25         }
26     }
27 }
28
29 int main() {
30     int N;
31     int maxNode;
32     int maxPath = 0;
33     cin >> N;
34
35     for (int i = 0; i < N - 1; i++) {
36         int a, b;
37         cin >> a >> b;
38         graph[a].push_back(b);
39         graph[b].push_back(a);
40     }
```



```

41
42     shortestPath(1);
43     for (int i = 1; i <= N; i++) {
44         if (dist[i] > maxPath) {
45             maxPath = dist[i];
46             maxNode = i;
47         }
48     }
49     shortestPath(maxNode);
50     for (int i = 1; i <= N; i++) {
51         if (dist[i] > maxPath) {
52             maxPath = dist[i];
53         }
54     }
55
56     cout << maxPath << endl;
57
58     return 0;
59 }

```

Week 13 课下

Week 14 课上

A 小明的训练室

```

1  #include <iostream>
2  #include <vector>
3  #include <queue>
4  #include <limits>
5  using namespace std;
6
7  const int INF = numeric_limits<int>::max();
8
9  struct Edge {
10     int to;
11     int h;
12 };
13
14 void dijkstra(int s, int v, const vector<vector<Edge>>& g, vector<int>&
mh) {

```

```

15     priority_queue<pair<int, int>, vector<pair<int, int>>,
greater<pair<int, int>>> pq;
16     pq.push({0, s});
17     mh[s] = 0;
18
19     while (!pq.empty()) {
20         int u = pq.top().second;
21         int curr_h = pq.top().first;
22         pq.pop();
23
24         if (curr_h > mh[u]) continue;
25
26         for (const auto& e : g[u]) {
27             int next_v = e.to;
28             int next_h = e.h;
29
30             if (mh[next_v] > max(mh[u], next_h)) {
31                 mh[next_v] = max(mh[u], next_h);
32                 pq.push({mh[next_v], next_v});
33             }
34         }
35     }
36 }
37
38 int main() {
39     int n, m, t;
40     cin >> n >> m >> t;
41
42     vector<vector<Edge>> g(n + 1);
43
44     for (int i = 0; i < m; ++i) {
45         int s, e, h;
46         cin >> s >> e >> h;
47         g[s].push_back({e, h});
48     }
49
50     while (t--) {
51         int a, b;
52         cin >> a >> b;
53
54         vector<int> mh(n + 1, INF);
55
56         dijkstra(a, n, g, mh);
57
58         int res = mh[b];
59         if (res == INF) cout << "-1" << endl;

```

```

60         else cout << res << endl;
61     }
62
63     return 0;
64 }

```

B 节点的最近公共祖先

```

1  #include <iostream>
2  #include <vector>
3  #include <cmath>
4  using namespace std;
5
6  const int MAXN = 10001;
7  const int MAXLOG = 15;
8
9  vector<int> adj[MAXN];
10 int parent[MAXN];
11 int depth[MAXN];
12 int ancestor[MAXN][MAXLOG];
13
14 void dfs(int node, int par, int dep) {
15     parent[node] = par;
16     depth[node] = dep;
17     for (int i = 0; i < adj[node].size(); ++i) {
18         int child = adj[node][i];
19         if (child != par) {
20             dfs(child, node, dep + 1);
21         }
22     }
23 }
24
25 void binaryLift(int N) {
26     for (int i = 1; i <= N; ++i) {
27         ancestor[i][0] = parent[i];
28     }
29
30     for (int j = 1; (1 << j) <= N; ++j) {
31         for (int i = 1; i <= N; ++i) {
32             if (ancestor[i][j - 1] != -1) {
33                 ancestor[i][j] = ancestor[ancestor[i][j - 1]][j - 1];
34             }
35         }
36     }
37 }
38

```

```

39  int findLCA(int u, int v) {
40      if (depth[u] < depth[v]) swap(u, v);
41
42      int log;
43      for (log = 1; (1 << log) <= depth[u]; ++log);
44      log--;
45
46      for (int i = log; i >= 0; --i) {
47          if (depth[u] - (1 << i) >= depth[v]) {
48              u = ancestor[u][i];
49          }
50      }
51
52      if (u == v) return u;
53
54      for (int i = log; i >= 0; --i) {
55          if (ancestor[u][i] != -1 && ancestor[u][i] != ancestor[v][i]) {
56              u = ancestor[u][i];
57              v = ancestor[v][i];
58          }
59      }
60
61      return parent[u];
62  }
63
64  int main() {
65      int N, M, S;
66      cin >> N >> M >> S;
67
68      for (int i = 1; i <= N; ++i) {
69          adj[i].clear();
70          parent[i] = -1;
71          depth[i] = 0;
72          for (int j = 0; j < MAXLOG; ++j) {
73              ancestor[i][j] = -1;
74          }
75      }
76
77      for (int i = 1; i < N; ++i) {
78          int x, y;
79          cin >> x >> y;
80          adj[x].push_back(y);
81          adj[y].push_back(x);
82      }
83
84      dfs(S, -1, 0);

```

```

85     binaryLift(N);
86
87     for (int i = 0; i < M; ++i) {
88         int a, b;
89         cin >> a >> b;
90         int lca = findLCA(a, b);
91         cout << lca << endl;
92     }
93
94     return 0;
95 }

```

C 布设光纤

```

1  #include <iostream>
2  #include <vector>
3  #include <climits>
4
5  using namespace std;
6
7  int main() {
8      int n;
9      cin >> n;
10     vector<vector<int>> adj(n, vector<int>(n));
11     for (int i = 0; i < n; ++i) {
12         for (int j = 0; j < n; ++j) {
13             cin >> adj[i][j];
14         }
15     }
16
17     vector<int> lowcost(n);
18     vector<bool> visited(n, false);
19     for (int i = 0; i < n; ++i) {
20         lowcost[i] = adj[0][i];
21     }
22     visited[0] = true;
23     int sum = 0;
24
25     for (int cnt = 1; cnt < n; ++cnt) {
26         int min_val = INT_MAX;
27         int min_idx = -1;
28         for (int j = 0; j < n; ++j) {
29             if (!visited[j] && lowcost[j] < min_val) {
30                 min_val = lowcost[j];
31                 min_idx = j;
32             }

```

```

33         }
34         if (min_idx == -1) {
35             break;
36         }
37         sum += min_val;
38         visited[min_idx] = true;
39         for (int j = 0; j < n; ++j) {
40             if (!visited[j] && adj[min_idx][j] < lowcost[j]) {
41                 lowcost[j] = adj[min_idx][j];
42             }
43         }
44     }
45
46     cout << sum << endl;
47
48     return 0;
49 }

```

D 丁香之路

```

1  #include<bits/stdc++.h>
2  #define ll long long
3  using namespace std;
4  const int MAXN = 2505;
5  struct Edge { int u, v, w; };
6  vector<Edge> G;
7  int f[MAXN], degree[MAXN], n, m, s, u, v, sum;
8  int ff[MAXN];
9
10 void init(int n) {
11     for (int i = 1; i <= n; ++i) f[i] = i;
12 }
13
14 int findf(int i) {
15     return f[i] == i ? f[i] : f[i] = findf(f[i]);
16 }
17
18 void merge(int i, int j) {
19     f[findf(i)] = findf(j);
20 }
21
22 bool cmp(const Edge &a, const Edge &b) {
23     return a.w < b.w;
24 }
25
26 int main() {

```

```

27 ios::sync_with_stdio(0);
28 cin.tie(0), cout.tie(0);
29 cin >> n >> m >> s;
30 init(n);
31 for (int i = 1; i <= m; ++i) {
32     cin >> u >> v;
33     degree[u]++, degree[v]++;
34     sum += abs(u - v);
35     merge(u, v);
36 }
37 degree[s]++;
38 for (int i = 1; i <= n; ++i) ff[i] = f[i];
39 for (int i = 1; i <= n; ++i) {
40     for (int j = 1; j <= n; ++j) f[j] = ff[j];
41     degree[i]++;
42     int ans = sum, pre = 0;
43     vector<int> V;
44     for (int j = 1; j <= n; ++j) {
45         if (degree[j]) V.push_back(j);
46     }
47     for (int j = 1; j <= n; ++j) {
48         if (degree[j] & 1) {
49             if (!pre) pre = j;
50             else {
51                 ans += (j - pre);
52                 for (int k = pre + 1; k <= j; ++k) merge(k, k - 1);
53                 pre = 0;
54             }
55         }
56     }
57     G.clear();
58     for (int j = 0; j + 1 < V.size(); ++j) {
59         if (findf(V[j]) != findf(V[j + 1])) {
60             G.push_back({V[j], V[j + 1], V[j + 1] - V[j]});
61         }
62     }
63     sort(G.begin(), G.end(), cmp);
64     for (const auto &e : G) {
65         if (findf(e.u) != findf(e.v)) {
66             merge(e.u, e.v); ans += 2 * e.w;
67         }
68     }
69     degree[i]--;
70     cout << ans << " ";
71 }
72 return 0;

```

E 威虎山上的分配

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  const int MAXN = 10001;
5  const int MAXM = 20001;
6
7  int cnt, money[MAXN];
8  bool vis[MAXN];
9  struct Edge {
10     int v, next;
11     int len;
12 } E[MAXM];
13
14 int p[MAXN], eid = 1;
15
16 void insert(int u, int v) {
17     E[eid].v = v;
18     E[eid].next = p[u];
19     p[u] = eid++;
20 }
21
22 int n, m;
23 int indegree[MAXN];
24
25 void topo() {
26     queue<int> q;
27     for (int i = 1; i <= n; i++) {
28         if (indegree[i] == 0) {
29             q.push(i);
30             vis[i] = true;
31         }
32     }
33     while (!q.empty()) {
34         int now = q.front();
35         q.pop();
36         cnt++;
37         for (int i = p[now]; i; i = E[i].next) {
38             int v = E[i].v;
39             if (--indegree[v] == 0) {
40                 q.push(v);
41                 vis[v] = true;
42                 money[v] = money[now] + 1;
```



```

43         }
44     }
45 }
46 }
47
48 int main() {
49     memset(indegree, 0, sizeof(indegree));
50     memset(money, 0, sizeof(money));
51     cin >> n >> m;
52     for (int i = 1; i <= m; i++) {
53         int u, v;
54         cin >> u >> v;
55         insert(v, u);
56         indegree[u]++;
57     }
58
59     topo();
60     int ans = 0;
61     for (int i = 1; i <= n; i++) {
62         if (!vis[i]) {
63             cout << "Unhappy!" << endl;
64             return 0;
65         }
66         ans += money[i];
67     }
68
69     cout << ans + n * 100 << endl;
70     return 0;
71 }

```

F 判定欧拉回路

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  const int MAXN = 1001;
5
6  vector<int> adj[MAXN];
7  int degree[MAXN];
8  bool visited[MAXN];
9
10 void dfs(int v) {
11     visited[v] = true;
12     for (int u : adj[v]) {
13         if (!visited[u]) {
14             dfs(u);

```

```

15         }
16     }
17 }
18
19 bool isConnected(int N) {
20     memset(visited, false, sizeof(visited));
21     int start = -1;
22     for (int i = 1; i <= N; ++i) {
23         if (degree[i] > 0) {
24             start = i;
25             break;
26         }
27     }
28     if (start == -1) return true;
29     dfs(start);
30
31     for (int i = 1; i <= N; ++i) {
32         if (degree[i] > 0 && !visited[i]) {
33             return false;
34         }
35     }
36     return true;
37 }
38
39 int main() {
40     int N, M;
41     cin >> N >> M;
42
43     for (int i = 0; i < M; ++i) {
44         int u, v;
45         cin >> u >> v;
46         adj[u].push_back(v);
47         adj[v].push_back(u);
48         degree[u]++;
49         degree[v]++;
50     }
51
52     if (!isConnected(N)) {
53         cout << 0 << endl;
54         return 0;
55     }
56
57     for (int i = 1; i <= N; ++i) {
58         if (degree[i] % 2 != 0) {
59             cout << 0 << endl;
60             return 0;

```

```

61         }
62     }
63
64     cout << 1 << endl;
65
66     return 0;
67 }

```

G 商业信息共享

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5      int n;
6      cin >> n;
7      vector<vector<int>> adj(n), adj_t(n);
8      for (int i = 0; i < n; ++i) {
9          int j;
10         while (cin >> j && j != 0) {
11             j--; // 转换为0-based索引
12             adj[i].push_back(j);
13             adj_t[j].push_back(i);
14         }
15     }
16
17     vector<bool> visited(n, false);
18     vector<int> order;
19     function<void(int)> dfs1 = [&](int u) {
20         visited[u] = true;
21         for (int v : adj[u]) {
22             if (!visited[v]) {
23                 dfs1(v);
24             }
25         }
26         order.push_back(u);
27     };
28
29     for (int i = 0; i < n; ++i) {
30         if (!visited[i]) {
31             dfs1(i);
32         }
33     }
34
35     reverse(order.begin(), order.end());
36

```

```

37     vector<int> component(n, -1);
38     int current_component = 0;
39     visited.assign(n, false);
40     function<void(int)> dfs2 = [&](int u) {
41         visited[u] = true;
42         component[u] = current_component;
43         for (int v : adj_t[u]) {
44             if (!visited[v]) {
45                 dfs2(v);
46             }
47         }
48     };
49
50     for (int u : order) {
51         if (!visited[u]) {
52             dfs2(u);
53             current_component++;
54         }
55     }
56
57     int k = current_component;
58     vector<int> indegree(k, 0), outdegree(k, 0);
59     vector<vector<bool>> added(k, vector<bool>(k, false));
60
61     for (int u = 0; u < n; ++u) {
62         for (int v : adj[u]) {
63             if (component[u] != component[v]) {
64                 int cu = component[u];
65                 int cv = component[v];
66                 if (!added[cu][cv]) {
67                     added[cu][cv] = true;
68                     outdegree[cu]++;
69                     indegree[cv]++;
70                 }
71             }
72         }
73     }
74
75     int c1 = 0, c2 = 0;
76     for (int i = 0; i < k; ++i) {
77         if (indegree[i] == 0) c1++;
78         if (outdegree[i] == 0) c2++;
79     }
80
81     int ans1 = c1;
82     int ans2 = (k == 1) ? 0 : max(c1, c2);

```

```
83
84     cout << ans1 << endl;
85     cout << ans2 << endl;
86
87     return 0;
88 }
```

Week 14 课下

Week 15 课上

A 阿里天池的新任务

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int MAXN = 1000005;
4  int w[MAXN], nextArr[MAXN];
5  char s[MAXN], pattern[MAXN];
6  int total = 0;
7
8  void computeLPS(const char *pat, int m, int *lps) {
9      int len = 0;
10     lps[0] = 0;
11     int i = 1;
12     while (i < m) {
13         if (pat[i] == pat[len]) {
14             len++;
15             lps[i] = len;
16             i++;
17         } else {
18             if (len != 0) {
19                 len = lps[len - 1];
20             } else {
21                 lps[i] = 0;
22                 i++;
23             }
24         }
25     }
26 }
27
28 void kmpSearch(const char *text, const char *pat) {
```

```

29     int n = strlen(text);
30     int m = strlen(pat);
31     computeLPS(pat, m, nextArr);
32     int i = 0, j = 0;
33     while (i < n) {
34         if (pat[j] == text[i]) {
35             i++;
36             j++;
37         }
38         if (j == m) {
39             total++;
40             j = nextArr[j - 1];
41         } else if (i < n && pat[j] != text[i]) {
42             if (j != 0) {
43                 j = nextArr[j - 1];
44             } else {
45                 i++;
46             }
47         }
48     }
49 }
50
51 int main() {
52     int n, a, b, l, r;
53     cin >> n >> a >> b >> l >> r;
54     cin >> pattern;
55     for (int i = 0; i < n; ++i) {
56         if (i == 0) {
57             w[i] = b;
58         } else {
59             w[i] = (w[i - 1] + a) % n;
60         }
61         if (w[i] >= l && w[i] <= r) {
62             s[i] = (w[i] % 2 == 0) ? 'A' : 'T';
63         } else {
64             s[i] = (w[i] % 2 == 0) ? 'G' : 'C';
65         }
66     }
67     kmpSearch(s, pattern);
68     cout << total;
69     return 0;
70 }

```

B 猴子打字

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

```
2  using namespace std;
3
4  const int ALPHABET_SIZE = 26;
5
6  struct Node {
7      Node *children[ALPHABET_SIZE];
8      Node *fail;
9      int cnt;
10     Node() {
11         for (int i = 0; i < ALPHABET_SIZE; i++) {
12             children[i] = nullptr;
13         }
14         fail = nullptr;
15         cnt = 0;
16     }
17 };
18
19 Node *root;
20
21 void insert(string word) {
22     Node *p = root;
23     for (char c : word) {
24         int idx = c - 'a';
25         if (p->children[idx] == nullptr) {
26             p->children[idx] = new Node();
27         }
28         p = p->children[idx];
29     }
30     p->cnt++;
31 }
32
33 void build() {
34     queue<Node*> q;
35     root->fail = root;
36     for (int i = 0; i < ALPHABET_SIZE; i++) {
37         if (root->children[i] != nullptr) {
38             root->children[i]->fail = root;
39             q.push(root->children[i]);
40         } else {
41             root->children[i] = root;
42         }
43     }
44
45     while (!q.empty()) {
46         Node *u = q.front();
47         q.pop();
```

```

48         for (int i = 0; i < ALPHABET_SIZE; i++) {
49             if (u->children[i] != nullptr) {
50                 Node *v = u->children[i];
51                 v->fail = u->fail->children[i];
52                 q.push(v);
53             } else {
54                 u->children[i] = u->fail->children[i];
55             }
56         }
57     }
58 }
59
60 long long query(string article) {
61     long long ans = 0;
62     Node *p = root;
63     for (char c : article) {
64         int idx = c - 'a';
65         p = p->children[idx];
66         Node *temp = p;
67         while (temp != root) {
68             ans += temp->cnt;
69             temp = temp->fail;
70         }
71     }
72     return ans;
73 }
74
75 int main() {
76     int n;
77     cin >> n;
78     root = new Node();
79     string word;
80     for (int i = 0; i < n; i++) {
81         cin >> word;
82         insert(word);
83     }
84     build();
85     string article;
86     cin >> article;
87     cout << query(article) << endl;
88     return 0;
89 }

```

C 糟糕的 Bug

```

1  #include <iostream>

```



```
2  #include <stdio>
3  #include <cstring>
4
5  using namespace std;
6
7  #define MAX_N 2333430
8  #define MAX_C 26
9
10 struct Trie {
11     int *ch[MAX_N];
12     int tot;
13     int cnt[MAX_N];
14
15     Trie() {
16         tot = 0;
17         memset(ch, 0, sizeof(ch));
18         memset(cnt, 0, sizeof(cnt));
19     }
20
21     void insert(const char *str) {
22         int p = 0;
23         for (int i = 0; str[i]; ++i) {
24             if (ch[p] == NULL) {
25                 ch[p] = new int[MAX_C];
26                 memset(ch[p], -1, sizeof(int) * MAX_C);
27             }
28             if (ch[p][str[i] - 'a'] == -1) {
29                 ch[p][str[i] - 'a'] = ++tot;
30             }
31             p = ch[p][str[i] - 'a'];
32         }
33         cnt[p]++;
34     }
35
36     bool find(const char *str) {
37         int p = 0;
38         for (int i = 0; str[i]; ++i) {
39             if (cnt[p] != 0) return true;
40             if (ch[p] == NULL) return false;
41             if (ch[p][str[i] - 'a'] == -1) return false;
42             p = ch[p][str[i] - 'a'];
43         }
44         return false;
45     }
46 };
47
```

```

48 char s[MAX_N][15];
49 Trie trie;
50
51 int main() {
52     int n;
53     scanf("%d", &n);
54     getchar();
55     bool ans = false;
56     for (int i = 0; i < n; ++i) {
57         scanf("%s", s[i]);
58         getchar();
59         trie.insert(s[i]);
60     }
61     for (int i = 0; i < n; ++i) {
62         if (trie.find(s[i])) {
63             ans = true;
64             break;
65         }
66     }
67     if (ans) {
68         puts("Bug!");
69     } else {
70         puts("Good Luck!");
71     }
72     return 0;
73 }

```

D 首尾相接

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  vector<int> getNext(const string& p) {
5      int m = p.size();
6      vector<int> next(m + 1, -1);
7      int i = 0, j = -1;
8      while (i < m) {
9          if (j == -1 || p[i] == p[j]) {
10             i++;
11             j++;
12             next[i] = j;
13         } else {
14             j = next[j];
15         }
16     }
17     return next;

```

```

18     }
19
20     int main() {
21         string s1, s2;
22         getline(cin, s1);
23         getline(cin, s2);
24
25         int n = s2.size();
26         int m = s1.size();
27         if (m == 0 || n == 0) {
28             cout << 0 << endl;
29             return 0;
30         }
31
32         vector<int> next = getNext(s1);
33         int j = 0, i = 0;
34         int overlap = 0;
35
36         while (i < n) {
37             if (j == m) {
38                 j = next[j];
39             }
40             if (j == -1 || s2[i] == s1[j]) {
41                 j++;
42                 i++;
43                 if (i == n) {
44                     overlap = j;
45                 }
46             } else {
47                 j = next[j];
48             }
49         }
50
51         if (overlap > 0) {
52             cout << s1.substr(0, overlap) << " " << overlap << endl;
53         } else {
54             cout << 0 << endl;
55         }
56
57         return 0;
58     }

```

E 旋转数字

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  int main() {
6      string s;
7      cin >> s;
8      int n = s.size();
9
10     vector<int> next(n, 0);
11     for (int i = 1; i < n; i++) {
12         int j = next[i-1];
13         while (j > 0 && s[i] != s[j]) {
14             j = next[j-1];
15         }
16         if (s[i] == s[j]) {
17             j++;
18         }
19         next[i] = j;
20     }
21
22     int T = n;
23     if (next[n-1] != 0 && n % (n - next[n-1]) == 0) {
24         T = n - next[n-1];
25     }
26
27     string ss = s + s;
28     int n2 = 2 * n;
29     vector<int> Z(n2, 0);
30     if (n2 > 0) {
31         Z[0] = n2;
32         int l = 0, r = 0;
33         for (int i = 1; i < n2; i++) {
34             if (i <= r) {
35                 Z[i] = min(r - i + 1, Z[i - l]);
36             }
37             while (i + Z[i] < n2 && ss[Z[i]] == ss[i + Z[i]]) {
38                 Z[i]++;
39             }
40             if (i + Z[i] - 1 > r) {
41                 l = i;
42                 r = i + Z[i] - 1;
43             }
44         }
45     }
```

```
45     }
46
47     int cnt_less = 0, cnt_equal = 0, cnt_greater = 0;
48     for (int k = 0; k < T; k++) {
49         int start = (n - k) % n;
50         if (start == 0) {
51             cnt_equal++;
52         } else {
53             if (Z[start] >= n) {
54                 cnt_equal++;
55             } else {
56                 if (ss[start + Z[start]] < ss[Z[start]]) {
57                     cnt_less++;
58                 } else {
59                     cnt_greater++;
60                 }
61             }
62         }
63     }
64
65     cout << cnt_less << " " << cnt_equal << " " << cnt_greater << endl;
66     return 0;
67 }
```
