

1 2022mid1

1.1 Problem A. Task and Penalty

```

1 #include <iostream>
2 #include <algorithm>
3
4 using namespace std;
5
6 int a[1000000];
7
8 int main() {
9     int num;
10    cin >> num;
11    for (int i=0; i<num; i++) {
12        cin >> a[i];
13    }
14    sort(a, a + num);
15    int total = 0;
16    int ans = 0;
17    for (int i = 0; i < num; i++) {
18        total = total + a[i];
19        ans = ans + total;
20    }
21    cout << ans;
22 }

```

```

28 }
29
30 int main() {
31     cin >> n >> m;
32     grid.resize(n, vector<char>(m));
33     visited.resize(n, vector<bool>(m, false));
34
35     // 讀取地圖
36     for (int i = 0; i < n; i++) {
37         for (int j = 0; j < m; j++) {
38             cin >> grid[i][j];
39         }
40     }
41
42     int roomCount = 0;
43
44     // 遍歷整個地圖
45     for (int i = 0; i < n; i++) {
46         for (int j = 0; j < m; j++) {
47             // 如果找到未訪問的 `.` 表示找到一個新房間
48             if (grid[i][j] == '.' && !visited[i][j]) {
49                 dfs(i, j); // 執行 DFS
50                 roomCount++; // 房間計數加一
51             }
52         }
53     }
54
55     // 輸出房間數量
56     cout << roomCount << endl;
57
58     return 0;
59 }

```

1.2 Problem B. Counting Rooms

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int n, m;
6 vector<vector<char>> grid;
7 vector<vector<bool>> visited;
8
9 // 移動方向：上、下、左、右
10 int dx[] = {-1, 1, 0, 0};
11 int dy[] = {0, 0, -1, 1};
12
13 // 深度優先搜索 (DFS) 函數
14 void dfs(int x, int y) {
15     // 標記當前節點為已訪問
16     visited[x][y] = true;
17
18     // 遍歷四個方向
19     for (int i = 0; i < 4; i++) {
20         int nx = x + dx[i];
21         int ny = y + dy[i];
22
23         // 檢查邊界條件和是否可以訪問
24         if (nx >= 0 && nx < n && ny >= 0 && ny < m && !visited[nx][ny] && grid[nx][ny] == '.') {
25             dfs(nx, ny); // 繼續訪問相鄰的 `.` 格子
26         }
27     }
28 }

```

1.3 Problem C. Sum of Three Values

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 struct Number {
7     int value;
8     int index;
9 };
10
11 int main() {
12     int n;
13     long long x;
14     cin >> n >> x;
15
16     vector<Number> arr(n);
17     for (int i = 0; i < n; ++i) {
18         cin >> arr[i].value;
19         arr[i].index = i + 1; // 使用 1-based index
20     }
21
22     // 按值對數組進行排序

```

```

23 sort(arr.begin(), arr.end(), [](const
24     Number &a, const Number &b) {
25         return a.value < b.value;
26     });
27
28     // 嘗試固定第一個數並使用雙指針法查找另外兩個數
29     for (int i = 0; i < n - 2; ++i) {
30         long long target = x - arr[i].value;
31         int left = i + 1, right = n - 1;
32
33         // 使用雙指針尋找另外兩個數
34         while (left < right) {
35             long long sum = arr[left].value + arr[right].value;
36             if (sum == target) {
37                 // 找到結果，將索引排序後輸出
38                 vector<int> result = {arr[left].index, arr[right].index};
39                 sort(result.begin(), result.end()); // 將索引由小到大排序
40                 cout << result[0] << " " << result[1] << " " << result[2] << endl;
41                 return 0;
42             } else if (sum < target) {
43                 ++left; // 增加左指針
44             } else {
45                 --right; // 減少右指針
46             }
47         }
48
49         // 如果沒有找到任何解
50         cout << "IMPOSSIBLE" << endl;
51         return 0;
52 }

```

1.4 Problem D. LR insertion

```

1 #include <iostream>
2 #include <list>
3 #include <string>
4 using namespace std;
5
6 int main() {
7     int N;
8     string S;
9
10    // Read input
11    cin >> N >> S;
12
13    // Initialize list with 0
14    list<int> A;
15    A.push_back(0);
16
17    // Keep track of position of i-1
18    list<int>::iterator prev = A.begin();

```

```

19 // Process each character in the string
20 for (int i = 1; i <= N; i++) {
21     if (S[i-1] == 'L') {
22         // Insert to the left of previous number
23         prev = A.insert(prev, i);
24     } else {
25         // Insert to the right of previous number
26         // Need to move iterator one step forward before inserting
27         list<int>::iterator next = prev;
28         ++next;
29         prev = A.insert(next, i);
30     }
31 }
32
33 // Print the final sequence
34 bool first = true;
35 for (int num : A) {
36     if (!first) cout << " ";
37     cout << num;
38     first = false;
39 }
40 cout << endl;
41
42 return 0;
43 }
44 }

```

1.5 Problem E. Second Day in Ain-crads

```

1 #include <iostream>
2 #include <string>
3 #include <vector>
4 #include <cmath>
5 using namespace std;
6
7 // 用來生成所有可能的數字
8 void generateNumbers(const string &pattern, vector<int> &candidates) {
9     int numX = 0;
10    for (char ch : pattern) {
11        if (ch == 'X') numX++;
12    }
13
14    // 當沒有 X 時，直接將原值作為候選數
15    if (numX == 0) {
16        candidates.push_back(stoi(pattern));
17        return;
18    }
19
20    int len = pattern.size();
21    int maxReplace = pow(10, numX); // 用於生成所有組合
22    for (int i = 0; i < maxReplace; ++i) {
23        string numStr = pattern;
24        int temp = i;
25

```

```

26 // 替換掉 X
27 for (int j = len - 1; j >= 0; --j) {
28     if (numStr[j] == 'X') {
29         numStr[j] = '0' + (temp %
30             10);
31         temp /= 10;
32     }
33 }
34 // 去掉前導零的數
35 if (numStr[0] != '0' || numStr == "0") {
36     candidates.push_back(stoi(numStr));
37 }
38 }
39 }
40
41 int main() {
42     string S_A, OP, S_B, EQ, S_C;
43     cin >> S_A >> OP >> S_B >> EQ >> S_C;
44
45     vector<int> A_candidates, B_candidates,
46         C_candidates;
47
48     // 生成候選值
49     generateNumbers(S_A, A_candidates);
50     generateNumbers(S_B, B_candidates);
51     generateNumbers(S_C, C_candidates);
52
53     // 檢查每組候選值是否滿足等式
54     for (int A : A_candidates) {
55         for (int B : B_candidates) {
56             for (int C : C_candidates) {
57                 bool isValid = false;
58                 if (OP == "+" && A + B == C)
59                     isValid = true;
60                 else if (OP == "-" && A -
61                     B == C) {
62                     isValid = true;
63                 }
64
65                 if (isValid) {
66                     cout << A << " " << B <<
67                         " " << C << endl;
68                     return 0;
69                 }
70             }
71         }
72     }
73
74     // 若無解，輸出空解
75     cout << "IMPOSSIBLE" << endl;
76     return 0;
77 }

```

1.6 Problem F. Circular Nearest Smaller Values

```

2 #include <vector>
3 #include <stack>
4 using namespace std;
5
6 int main()
7 {
8     int n;
9     cin >> n;
10    vector<int> A(n), result(n);
11    for (int i = 0; i < n; ++i)
12    {
13        cin >> A[i];
14    }
15
16    // 將原陣列複製一份，形成 2n 長度的陣列
17    vector<int> extendedA(2 * n);
18    for (int i = 0; i < n; ++i)
19    {
20        extendedA[i] = extendedA[i + n] = A[i];
21    }
22
23    stack<int> st;
24
25    // 逆向遍歷複製後的 2n 長度的陣列
26    for (int i = 2 * n - 1; i >= 0; --i)
27    {
28        int index = i % n; // 獲取當前元素在
29        // 原陣列中的位置
30
31        // 移除棧中所有大於當前元素的索引
32        while (!st.empty() && extendedA[st.
33            top()] > extendedA[i])
34        {
35            st.pop();
36        }
37
38        // 設定結果為最近的較小或相等位置，
39        // 如果棧為空則表示無較小值
40        if (i < n)
41        {
42            // 只在第一次遍歷時填充結果
43            result[i] = st.empty() ? 0 : st.
44                top() % n + 1; // 加1轉為1-
45                based index
46        }
47
48        // 將當前索引壓入棧中
49        st.push(i);
50    }
51
52    // 輸出結果
53    for (int i = 0; i < n; ++i)
54    {
55        cout << result[i] << " ";
56    }
57    cout << endl;
58
59    return 0;
60 }

```

1.7 Problem G. Gluttony

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 typedef long long ll;
7
8 bool canAchieve(const vector<ll> &A, const
9     vector<ll> &F, ll M, ll K) {
10     ll neededTraining = 0;
11     int n = A.size();
12
13     for (int i = 0; i < n; ++i) {
14         ll maxA = M / F[i]; // 每個食物能夠
15         // 允許的最大消耗係數
16         if (A[i] > maxA) {
17             neededTraining += A[i] - maxA;
18             if (neededTraining > K) return
19                 false; // 超出訓練次數
20         }
21     }
22     return neededTraining <= K;
23 }
24
25 ll minScore(vector<ll> &A, vector<ll> &F, ll
26     K) {
27     sort(A.begin(), A.end());
28     sort(F.rbegin(), F.rend()); // F 降序排
29     // 列
30     ll left = 0, right = A.back() * F.front()
31     ();
32
33     while (left < right) {
34         ll mid = left + (right - left) / 2;
35         if (canAchieve(A, F, mid, K)) {
36             right = mid;
37         } else {
38             left = mid + 1;
39         }
40     }
41     return left;
42 }
43
44 int main() {
45     int n;
46     ll K;
47     cin >> n >> K;
48
49     vector<ll> A(n), F(n);
50     for (int i = 0; i < n; ++i) cin >> A[i];
51     for (int i = 0; i < n; ++i) cin >> F[i];
52
53     cout << minScore(A, F, K) << endl;
54     return 0;
55 }

```

1.8 Problem H. Sticks

```

2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 int n;
7 vector<int> sticks;
8 int total_length;
9
10 // 回溯法檢查能否成功組裝長度為 L 的木棍
11 bool canForm(int L, int current_sum, int
12     index, int used_count) {
13     if (used_count == n) return true; // 全
14     // 部段已使用
15     if (current_sum == L) return canForm(L,
16         0, 0, used_count); // 完成一根木棍，
17     // 重置
18
19     for (int i = index; i < n; ++i) {
20         if (sticks[i] == 0) continue; // 已
21         // 使用的段跳過
22
23         if (current_sum + sticks[i] <= L) {
24             int temp = sticks[i];
25             sticks[i] = 0; // 標記已使用
26
27             // 進行下一步回溯
28             if (canForm(L, current_sum +
29                 temp, i + 1, used_count + 1)
30                 ) return true;
31
32             sticks[i] = temp; // 還原狀態
33
34             if (current_sum == 0 ||
35                 current_sum + sticks[i] == L
36                 ) break; // 剪枝
37         }
38     }
39     return false;
40 }
41
42 int findMinimumLength() {
43     for (int L = 1; L <= total_length; ++L)
44     {
45         if (total_length % L == 0) { // 確保
46             // L 是 total_length 的因數
47             if (canForm(L, 0, 0, 0)) return
48                 L;
49         }
50     }
51     return total_length;
52 }
53
54 int main() {
55     while (cin >> n && n != 0) {
56         sticks.resize(n);
57         total_length = 0;
58
59         for (int i = 0; i < n; ++i) {
60             cin >> sticks[i];
61             total_length += sticks[i];
62         }
63
64         sort(sticks.rbegin(), sticks.rend())
65         ; // 降序排列便於剪枝
66     }
67 }

```

```

53     cout << findMinimumLength() << endl;
54 }
55
56 return 0;
57 }

```

2.2 Chinese Rings

2 w1

2.1 Apple Division

```

1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4
5 using namespace std;
6
7 int main() {
8     int n;
9     cin >> n;
10    vector<int> apples(n);
11
12    // 輸入蘋果重量
13    for (int i = 0; i < n; i++) {
14        cin >> apples[i];
15    }
16
17    long long totalWeight = 0;
18    for (int i = 0; i < n; i++) {
19        totalWeight += apples[i];
20    }
21
22    long long minDifference = totalWeight;
23    // 初始化差值為總重量，理論上最大值
24
25    // 枚舉所有可能的分組方式
26    for (int subset = 0; subset < (1 << n); subset++) {
27        long long group1Weight = 0;
28
29        // 計算當前子集 (分組) 的重量
30        for (int i = 0; i < n; i++) {
31            if (subset & (1 << i)) {
32                group1Weight += apples[i];
33            }
34        }
35
36        // 計算兩組重量差
37        long long group2Weight = totalWeight - group1Weight;
38        long long currentDifference = abs(group1Weight - group2Weight);
39
40        // 更新最小的重量差
41        minDifference = min(minDifference, currentDifference);
42    }
43
44    // 輸出最小的重量差
45    cout << minDifference << endl;

```

```

46     return 0;
47 }

```

2.2 Chinese Rings

```

1 #include <iostream>
2 #include <vector>
3
4 const int MAX_N = 20;
5 int n;
6 std::vector<bool> state(MAX_N);
7 int count_op = 0;
8
9 void move_out(int c) {
10     std::cout << "Move ring " << c + 1 << "
11         out" << std::endl; // Output "Move
12         ring n out"
13     state[c] = false; // Change state to
14         indicate it's out
15     count_op++;
16 }
17
18 void move_in(int c) {
19     std::cout << "Move ring " << c + 1 << "
20         in" << std::endl; // Output "Move
21         ring n in"
22     state[c] = true; // Change state to
23         indicate it's in
24     count_op++;
25 }
26
27 void solve(int n);
28 void rsolve(int n);
29
30 // 111 -> 000
31 void solve(int n) {
32     if (n == 1) {
33         move_out(0);
34     } else if (n == 2) {
35         move_out(1);
36         solve(1);
37     } else {
38         solve(n - 2);
39         move_out(n - 1);
40         rsolve(n - 2);
41         solve(n - 1);
42     }
43 }
44
45 // 000 -> 111
46 void rsolve(int n) {
47     if (n == 1) {
48         move_in(0);
49     } else if (n == 2) {
50         rsolve(1);
51         move_in(1);
52     } else {
53         rsolve(n - 1);
54         solve(n - 2);
55         move_in(n - 1);
56         rsolve(n - 2);
57     }
58 }

```

```

53 int main() {
54     // std::cout << "Enter the number of
55     rings: ";
56     std::cin >> n;
57     std::fill(state.begin(), state.begin() +
58         n, true); // Initialize state to
59         true (all rings in)
60     solve(n);
61     // std::cout << "Total moves: " <<
62         count_op << std::endl;
63     return 0;
64 }

```

2.3 Citizen attention offices

```

1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4 #include <algorithm>
5 #include <climits>
6
7 using namespace std;
8
9 int calculateDistance(pair<int, int> a, pair
10 <int, int> b) {
11     return abs(a.first - b.first) + abs(a.
12         second - b.second);
13 }
14
15 int main() {
16     int t;
17     cin >> t; // 測試案例數量
18
19     while (t--) {
20         int n;
21         cin >> n; // 非零人口的區域數量
22         vector<pair<pair<int, int>, int>>
23             areas(n); // 每個區域的座標及人
24             口
25
26         for (int i = 0; i < n; i++) {
27             int r, c, p;
28             cin >> r >> c >> p;
29             areas[i] = {{r, c}, p};
30         }
31
32         // 所有 25 個可能的位置
33         vector<pair<int, int>> allPositions;
34         for (int i = 0; i < 5; i++) {
35             for (int j = 0; j < 5; j++) {
36                 allPositions.push_back({i, j
37                     });
38             }
39         }
40
41         vector<int> bestOffices;
42         int minTotalDistance = INT_MAX;
43
44         // 枚舉所有選擇 5 個辦事處的位置
45         vector<int> comb(25);

```

```

41     fill(comb.begin(), comb.begin() + 5,
42         1); // 前 5 個 1 表示選擇這些位
43         置
44     do {
45         vector<int> currentOffices;
46         for (int i = 0; i < 25; i++) {
47             if (comb[i]) {
48                 currentOffices.push_back
49                     (i);
50             }
51         }
52         // 計算當前選擇下的總距離
53         int totalDistance = 0;
54         for (auto area : areas) {
55             int minDistance = INT_MAX;
56             for (int officeIdx :
57                 currentOffices) {
58                 pair<int, int> office =
59                     allPositions[
60                         officeIdx];
61                 int distance =
62                     calculateDistance(
63                         area.first, office);
64                 minDistance = min(
65                     minDistance,
66                     distance);
67             }
68             totalDistance += minDistance
69                 * area.second;
70         }
71         // 如果當前解更優，更新最優解
72         if (totalDistance <
73             minTotalDistance) {
74             minTotalDistance =
75                 totalDistance;
76             bestOffices = currentOffices
77                 ;
78         }
79     } while (prev_permutation(comb.begin()
80         (), comb.end()));
81
82     // 輸出結果，按遞增順序排列
83     sort(bestOffices.begin(),
84         bestOffices.end());
85     for (int i = 0; i < 5; i++) {
86         cout << bestOffices[i];
87         if (i < 4) {
88             cout << " ";
89         }
90     }
91     cout << endl;
92 }
93
94 return 0;
95 }

```

2.4 Combinations

```

1 #include <iostream>

```

```

2 #include <cmath>
3 #include <vector>
4
5 using namespace std;
6
7 void generateCombinations(vector<int>& A,
8   vector<int>& current, int start, int M)
9 {
10   // 如果當前組合長度達到 M，則輸出
11   if (current.size() == M) {
12     for (int i = 0; i < M; ++i) {
13       if (i > 0) cout << " ";
14       cout << current[i];
15     }
16     cout << endl;
17     return;
18   }
19   // 遍歷所有可能的選擇
20   for (int i = start; i < A.size(); ++i) {
21     current.push_back(A[i]); // 選擇當前元素
22     generateCombinations(A, current, i + 1, M); // 繼續選擇剩下的元素
23     current.pop_back(); // 回溯，取消選擇當前元素
24   }
25
26 int main() {
27   int n, m;
28   cin >> n >> m;
29   vector<int> input(n);
30   for (int i = 0; i < n; ++i) {
31     cin >> input[i];
32   }
33   vector<int> current;
34
35   generateCombinations(input, current, 0, m);
36 }

```

2.5 Creating Strings

```

1 #include <iostream>
2 #include <algorithm>
3 #include <set>
4 #include <string>
5
6 using namespace std;
7
8 int main() {
9   string s;
10  cin >> s;
11  sort(s.begin(), s.end());
12  set<string> permutations;
13  do {
14    permutations.insert(s);
15  } while (next_permutation(s.begin(), s.end()));
16  cout << permutations.size() << endl;
17  for (const string& perm : permutations) {

```

```

18   cout << perm << endl;
19 }
20 return 0;
21 }

```

2.6 Gray code

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 vector<string> geneGray(int n) {
6   if (n == 1) {
7     return {"0", "1"};
8   }
9   vector<string> prev = geneGray(n-1);
10  vector<string> graycode;
11
12  for (auto code:prev) {
13    graycode.push_back("0" + code);
14  }
15  for (int j=prev.size()-1; j>=0; j--) {
16    graycode.push_back("1" + prev[j]);
17  }
18  return graycode;
19 }
20
21 int main() {
22   int n;
23   cin >> n;
24   vector<string> ans;
25   ans = geneGray(n);
26   for (auto code:ans) {
27     cout << code << endl;
28   }
29 }
30
31 // #include <iostream>
32 // #include <vector>
33 // #include <string>
34 //
35 // using namespace std;
36 //
37 // vector<string> geneGrayCode(int n) {
38 //   if (n == 1) {
39 //     return {"0", "1"};
40 //   }
41 //   vector<string> prev = geneGrayCode(n-1);
42 //   vector<string> graycode;
43 //
44 //   for (auto code:prev) {
45 //     graycode.push_back("0" + code);
46 //   }
47 //   for (int i=prev.size()-1; i>=0; i--) {
48 //     graycode.push_back("1" + prev[i]);
49 //   }
50 //   return graycode;
51 // }
52 //
53 // int main() {
54 //   int n;
55 //   cin >> n;

```

```

56   // 11
57   // 10
58   // add "0"
59   // 000
60   // 001
61   // 011
62   // 010
63   // add "1" in reverse direction
64   // 110
65   // 111
66   // 101
67   // 100
68   // then it becomes...
69   // 000
70   // 001
71   // 011
72   // 010
73   // 110
74   // 111
75   // 101
76   // 100
77   // return graycode;
78   //
79   //
80   // int main() {
81   //   int n;
82   //   cin >> n;
83   //
84   //   vector<string> graycode = geneGrayCode(n);
85   //
86   //   for (auto code:graycode) {
87   //     cout << code << endl;
88   //   }
89   // }

```

2.7 Tower of Hanoi

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 void hanoi(int n, int from, int via, int to,
6   vector<pair<int, int>>& moves) {
7   if (n == 1) {
8     moves.push_back({from, to});
9     return;
10  }
11  hanoi(n-1, from, to, via, moves);
12  moves.push_back({from, to});
13  hanoi(n-1, via, from, to, moves);
14 }
15
16 int main() {
17   int n;
18   cin >> n;
19   vector<pair<int, int>> moves;
20   hanoi(n, 1, 2, 3, moves);
21   cout << moves.size() << endl;
22   for (auto ans:moves) {
23     cout << ans.first << " " << ans.second << endl;
24   }
25 }

```

```

24 }

```

3 w2

3.1 Chessboard and Queens

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int countWays = 0; // 记录可行方案的数量
6 vector<vector<char>> board(8, vector<char>(8)); // 棋盘
7
8 // 用于记录每列、主对角线、副对角线是否被占用
9 bool cols[8] = {false};
10 bool main_diag[15] = {false};
11 bool anti_diag[15] = {false};
12
13 void solve(int row) {
14   // 如果成功在第 8 行放置皇后，表示找到一种方案
15   if (row == 8) {
16     countWays++;
17     return;
18   }
19
20   for (int col = 0; col < 8; col++) {
21     // 如果当前格子是保留的或该列、对角线已被占用，则跳过
22     if (board[row][col] == '*' || cols[col] || main_diag[row + col + 7] || anti_diag[row + col])
23       continue;
24
25     // 标记该列、主对角线、副对角线为占用状态
26     cols[col] = main_diag[row + col + 7] = anti_diag[row + col] = true;
27
28     // 递归处理下一行
29     solve(row + 1);
30
31     // 回溯：恢复状态
32     cols[col] = main_diag[row + col + 7] = anti_diag[row + col] = false;
33   }
34 }
35
36 int main() {
37   // 输入棋盘
38   for (int i = 0; i < 8; i++) {
39     for (int j = 0; j < 8; j++) {
40       cin >> board[i][j];
41     }
42   }
43 }

```

```

44 // 从第 0 行开始放置皇后
45 solve(0);
46
47 // 输出可行方案的数量
48 cout << countWays << endl;
49
50 return 0;
51 }

```

3.2 Grid Path

```

1 #include <bits/stdc++.h>
2 #define int long long
3 #define double long double
4 #define pii pair<int, int>
5 #define N 200005
6 #define INF LONG_LONG_MAX
7 #define x first
8 #define y second
9 #define all(a) a.begin(),a.end()
10 #define IOS ios::sync_with_stdio(0),cin.tie
    (0)
11 using namespace std;
12 string str;
13 int dx[4] = {1,0,-1,0},dy[4] = {0,1,0,-1};
14 bool vis[9][9];
15 int ans = 0;
16
17 void solve(int x,int y,int s){
18     if(x < 1 || x > 7 || y < 1 || y > 7 ||
        vis[x][y])return;
19     if(x == 1 && y == 7 && s < 48)return;
20     if(vis[x-1][y] && vis[x+1][y] && !vis[x]
        ][y+1] && !vis[x][y-1])return;
21     if(!vis[x-1][y] && !vis[x+1][y] && vis[x]
        ][y+1] && vis[x][y-1])return;
22     if(s == 48){
23         ans++;
24         return;
25     }
26     vis[x][y] = 1;
27     int ans = 0;
28     if(str[s] == 'L')solve(x - 1, y, s + 1);
29     if(str[s] == 'R')solve(x + 1, y, s + 1);
30     if(str[s] == 'U')solve(x, y - 1, s + 1);
31     if(str[s] == 'D')solve(x, y + 1, s + 1);
32     if(str[s] == '?'){
33         for(int i = 0;i < 4;i++){
34             int nx = x + dx[i],ny = y + dy[i]
                ;
35             solve(nx,ny,s+1);
36         }
37     }
38     vis[x][y] = 0;
39 }
40
41 signed main(){
42     IOS;
43     cin>>str;
44     memset(vis,0,sizeof(vis));
45     for(int i=1;i<=7;i++){
46         vis[i][0] = 1;
47         vis[8][i] = 1;

```

```

48         vis[i][8] = 1;
49         vis[0][i] = 1;
50     }
51     solve(1,1,0);
52     cout<<ans<<"\n";
53 }

```

3.3 Hive

```

1 #include<bits/stdc++.h>
2 #define IOS ios::sync_with_stdio(0),cin.tie
    (0),cout.tie(0)
3 #define endl '\n'
4 #define pii pair<int,int>
5 #define F first
6 #define S second
7 using namespace std;
8 const int inf = 1e9+7;
9 int ans = inf;
10 int n,m;
11 int mp[105][105] = {},vis[105][105] = {},deg
    [105][105] = {};
12 int dx[6] = {-1,0,1,1,0,-1};
13 int dy[6] = {1,2,1,-1,-2,-1};
14 pii f(int x,int y){
15     return make_pair(x,(y-1)*2+(x&1)+(x
        %2==0?2:0));
16 }
17 int dijkstra(int sx,int sy,int ex,int ey){
18     int dis[105][105] = {},vis2[105][105] =
        {};
19     memset(dis,0x3f3f3f3f,sizeof(dis));
20     dis[sx][sy] = mp[sx][sy];
21     priority_queue<pair<int,pii>,vector<pair<
        int,pii>>,greater<pair<int,pii>>>pq;
22     pq.push({dis[sx][sy],{sx,sy}});
23     while(!pq.empty()){
24         auto [x,y] = pq.top().S;
25         pq.pop();
26         if(vis[x][y] or vis2[x][y])continue;
27         vis2[x][y] = 1;
28         for(int i = 0;i<6;i++){
29             int nx = x+dx[i],ny = y+dy[i];
30             if(nx<1 or ny<1 or nx>n or ny>2*m)
                continue;
31             if(vis[nx][ny] or vis2[nx][ny])
                continue;
32             if(dis[nx][ny]>dis[x][y]+mp[nx][ny]){
33                 dis[nx][ny] = dis[x][y]+mp[nx][ny];
34                 pq.push({dis[nx][ny],{nx,ny}});
35             }
36         }
37     }
38     return dis[ex][ey];
39 }
40 void dfs(int x,int y,int ex,int ey,int dis,
    int sx,int sy,int ex2,int ey2){
41     if(dis>ans)return;
42     if(x==ex and y==ey){
43         ans = min(ans,dis+dijkstra(sx,sy,ex2,ey2
            ));
44         return;
45     }

```

```

46     for(int i = 0;i<6;i++){
47         int nx = x+dx[i],ny = y+dy[i];
48         if(nx<1 or ny<1 or nx>n or ny>2*m)
            continue;
49         deg[nx][ny]++;
50     }
51     for(int i = 0;i<6;i++){
52         int nx = x+dx[i],ny = y+dy[i];
53         if(nx<1 or ny<1 or nx>n or ny>2*m)
            continue;
54         if(vis[nx][ny])continue;
55         if(deg[nx][ny]>=2)continue;
56         vis[nx][ny] = 1;
57         dfs(nx,ny,ex,ey,dis+mp[nx][ny],sx,sy,ex2
            ,ey2);
58         vis[nx][ny] = 0;
59     }
60     for(int i = 0;i<6;i++){
61         int nx = x+dx[i],ny = y+dy[i];
62         if(nx<1 or ny<1 or nx>n or ny>2*m)
            continue;
63         deg[nx][ny]--;
64     }
65 }
66 void solve(){
67     ans = inf;
68     int x[5] = {},y[5] = {};
69     cin>>n>>m;
70     for(int i = 1;i<=4;i++){
71         cin>>x[i];
72         cin>>y[i];
73         auto [nx,ny] = f(x[i],y[i]);
74         x[i] = nx,y[i] = ny;
75     }
76     for(int i = 1;i<=n;i++){
77         for(int j = 1;j<=m;j++){
78             auto [nx,ny] = f(i,j);
79             cin>>mp[nx][ny];
80         }
81     }
82     vis[x[1]][y[1]] = 1;
83     dfs(x[1],y[1],x[2],y[2],mp[x[1]][y[1]],x
        [3],y[3],x[4],y[4]);
84     if(ans==inf)cout<< -1 <<endl;
85     else cout<<ans<<endl;
86     vis[x[1]][y[1]] = 0;
87 }
88 int main(){
89     IOS;
90     int t;
91     cin>>t;
92     while(t--){
93         solve();
94     }
95 }

```

3.4 M Queen N Rocks

```

1 #include <iostream>
2 #include <vector>
3 #include <cstring>
4
5 using namespace std;

```

```

6 int N, M;
7 int board_size;
8 vector<vector<int>> board;
9
10 int place(int r, int n, int m);
11 int placeQ(int r, int c);
12 int placeR(int r, int c);
13
14 int main() {
15     int testcase;
16     cin >> testcase;
17
18     for (int i = 1; i <= testcase; i++) {
19         cin >> N >> M;
20         board_size = N + M;
21         board.assign(15, vector<int>(15, 0))
            ;
22
23         cout << place(0, 0, 0) << endl;
24     }
25     return 0;
26 }
27
28 int placeQ(int r, int c) {
29     for (int i = 1; i <= r; i++) {
30         if (c - i >= 0 && board[r - i][c - i
            ] != 0) {
31             return 0;
32         }
33         if (c + i <= board_size - 1 && board
            [r - i][c + i] != 0) {
34             return 0;
35         }
36         if (r - i >= 0 && board[r - i][c] !=
            0) {
37             return 0;
38         }
39     }
40     return 1;
41 }
42
43 int placeR(int r, int c) {
44     for (int i = 1; i <= r; i++) {
45         if (c - i >= 0 && board[r - i][c - i
            ] == 1) {
46             return 0;
47         }
48         if (c + i <= board_size - 1 && board
            [r - i][c + i] == 1) {
49             return 0;
50         }
51         if (r - i >= 0 && board[r - i][c] !=
            0) {
52             return 0;
53         }
54     }
55     return 1;
56 }
57
58 int place(int r, int n, int m) {
59     if (r == board_size) return 1;
60     int cnt = 0;
61     for (int i = 0; i < board_size; i++) {
62         if (n < N && placeQ(r, i)) {
63             board[r][i] = 1;

```



```

65         cnt += place(r + 1, n + 1, m);
66         board[r][i] = 0;
67     }
68     if (m < M && placeR(r, i)) {
69         board[r][i] = -1;
70         cnt += place(r + 1, n, m + 1);
71         board[r][i] = 0;
72     }
73 }
74 return cnt;
75 }

```

3.5 Square

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  using namespace std;
6
7  // Function to check if we can form a square
8  // using the sticks
9  bool canFormSquare(vector<int>& sticks,
10 vector<int>& sides, int index, int
11 sidelength) {
12     // If we have assigned all sticks
13     if (index == sticks.size()) {
14         // Check if all four sides are of
15         // equal length
16         return sides[0] == sidelength &&
17             sides[1] == sidelength &&
18             sides[2] == sidelength &&
19             sides[3] == sidelength;
20     }
21
22     // Try to place the current stick in
23     // each side
24     for (int i = 0; i < 4; i++) {
25         if (sides[i] + sticks[index] <=
26             sidelength) {
27             sides[i] += sticks[index]; //
28             Place the stick
29             if (canFormSquare(sticks, sides,
30                 index + 1, sidelength)) {
31                 return true;
32             }
33             sides[i] -= sticks[index]; //
34             Backtrack
35         }
36     }
37
38     return false;
39 }

```

```

30 int main() {
31     int N; // Number of test cases
32     cin >> N;
33
34     while (N--) {
35         int M; // Number of sticks
36         cin >> M;
37
38         vector<int> sticks(M);

```

```

39     int totalLength = 0;
40
41     // Read the stick lengths and
42     // calculate the total length
43     for (int i = 0; i < M; i++) {
44         cin >> sticks[i];
45         totalLength += sticks[i];
46     }
47
48     // If the total length is not
49     // divisible by 4, we cannot form a
50     // square
51     if (totalLength % 4 != 0) {
52         cout << "no" << endl;
53         continue;
54     }
55
56     int sideLength = totalLength / 4; //
57     // Length of each side of the
58     // square
59     vector<int> sides(4, 0); //
60     // Initialize the 4 sides of the
61     // square
62
63     // Sort the sticks in descending
64     // order for better performance
65     sort(sticks.rbegin(), sticks.rend())
66     ;
67
68     // If the largest stick is greater
69     // than the side length, it's
70     // impossible
71     if (sticks[0] > sideLength) {
72         cout << "no" << endl;
73         continue;
74     }
75
76     // Use backtracking to check if we
77     // can form a square
78     if (canFormSquare(sticks, sides, 0,
79         sideLength)) {
80         cout << "yes" << endl;
81     } else {
82         cout << "no" << endl;
83     }
84
85     return 0;
86 }

```

3.6 Sudoku

```

1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <algorithm>
5
6  using namespace std;
7
8  const int N = 9;
9  vector<pair<int, int>> empty_cells; // To
10 store the positions of all empty cells

```

```

11 /* A utility function to print grid as a
12    single line */
13 void printGrid(int grid[N][N]) {
14     for (int i = 0; i < N; i++)
15         for (int j = 0; j < N; j++)
16             cout << grid[i][j];
17     cout << endl;
18 }
19
20 /* Checks whether it will be legal to assign
21    num to the given row, col */
22 bool isSafe(int grid[N][N], int row, int col
23 , int num) {
24     for (int x = 0; x < N; x++)
25         if (grid[row][x] == num || grid[x][
26             col] == num)
27             return false;
28
29     int startRow = row - row % 3, startCol =
30     col - col % 3;
31     for (int i = 0; i < 3; i++)
32         for (int j = 0; j < 3; j++)
33             if (grid[i + startRow][j +
34                 startCol] == num)
35                 return false;
36
37     return true;
38 }
39
40 /* Checks if the initial Sudoku grid is
41    valid */
42 bool isValidSudoku(int grid[N][N]) {
43     for (int i = 0; i < N; i++) {
44         vector<bool> rowCheck(N + 1, false);
45         vector<bool> colCheck(N + 1, false);
46         for (int j = 0; j < N; j++) {
47             if (grid[i][j] != 0) {
48                 if (rowCheck[grid[i][j]])
49                     return false;
50                 rowCheck[grid[i][j]] = true;
51             }
52             if (grid[j][i] != 0) {
53                 if (colCheck[grid[j][i]])
54                     return false;
55                 colCheck[grid[j][i]] = true;
56             }
57         }
58     }
59
60     for (int row = 0; row < N; row += 3) {
61         for (int col = 0; col < N; col += 3) {
62             vector<bool> boxCheck(N + 1,
63                 false);
64             for (int i = 0; i < 3; i++)
65                 for (int j = 0; j < 3; j++)
66                     int num = grid[row + i][
67                         col + j];
68                     if (num != 0) {
69                         if (boxCheck[num])
70                             return false;
71                         boxCheck[num] = true
72                     }
73         }
74     }
75
76     return true;
77 }

```

```

62     }
63 }
64 return true;
65 }
66
67 /* Recursive function to solve the Sudoku
68    grid */
69 bool solveSudoku(int grid[N][N], int index =
70     0) {
71     if (index == empty_cells.size()) return
72     true;
73
74     int row = empty_cells[index].first;
75     int col = empty_cells[index].second;
76
77     // Attempt to place numbers in ascending
78     // order for Lexicographically
79     // smallest solution
80     for (int num = 1; num <= N; num++) {
81         if (isSafe(grid, row, col, num)) {
82             grid[row][col] = num;
83             if (solveSudoku(grid, index + 1))
84                 return true;
85             grid[row][col] = 0; // Backtrack
86             if not successful
87         }
88     }
89     return false;
90 }
91
92 /* Converts a single string line to a 9x9
93    Sudoku grid */
94 void parseGrid(const string& line, int grid[
95     N][N]) {
96     empty_cells.clear(); // Reset empty
97     // cells for each new grid
98
99     for (int i = 0; i < N * N; i++) {
100         int row = i / N, col = i % N;
101         char ch = line[i];
102         grid[row][col] = (ch == '.') ? 0 :
103         ch - '0';
104
105         // Store positions of empty cells
106         // for backtracking
107         if (grid[row][col] == 0)
108             empty_cells.emplace_back(row,
109                 col);
110     }
111
112     int main() {
113         string line;
114         while (cin >> line && line != "end") {
115             int grid[N][N];
116             parseGrid(line, grid);
117
118             // 檢查初始數獨是否合法
119             if (!isValidSudoku(grid)) {
120                 cout << "No solution." << endl;
121                 continue;
122             }
123
124             if (solveSudoku(grid))

```

```

114     printGrid(grid);
115     else
116         cout << "No solution." << endl;
117 }
118
119 return 0;
120 }

```

3.7 Sum It Up

```

1 #include <iostream>
2 #include <vector>
3 #include <set>
4 #include <algorithm>
5
6 using namespace std;
7
8 int t, n;
9 vector<int> nums;
10 set<vector<int>> resultSet;
11
12 void findSums(int target, int index, vector<
13     int>& current) {
14     if (target == 0) {
15         resultSet.insert(current); // Add
16         // the current sum to the result
17         // set
18         return;
19     }
20     for (int i = index; i < n; i++) {
21         if (i > index && nums[i] == nums[i -
22             1]) continue; // Skip
23         // duplicates
24         if (nums[i] <= target) {
25             current.push_back(nums[i]);
26             findSums(target - nums[i], i +
27                 1, current); // Recursively
28             // find sums
29             current.pop_back(); // Backtrack
30         }
31     }
32 }
33
34 int main() {
35     while (true) {
36         // Read input
37         cin >> t >> n;
38         if (n == 0) break; // Exit condition
39
40         nums.resize(n);
41         for (int i = 0; i < n; i++) {
42             cin >> nums[i];
43         }
44
45         resultSet.clear(); // Reset result
46         // set for each test case
47         vector<int> current;
48
49         findSums(t, 0, current);
50
51         // Output results
52         cout << "Sums of " << t << ":\n";

```

```

46     if (resultSet.empty()) {
47         cout << "NONE\n";
48     } else {
49         for (auto it = resultSet.rbegin
50             (); it != resultSet.rend();
51             ++it) {
52             for (size_t i = 0; i < it->
53                 size(); i++) {
54                 if (i > 0) cout << "+";
55                 cout << (*it)[i];
56             }
57             cout << endl;
58         }
59     }
60 }

```

4 w3

4.1 Array Arrangement

```

1 #include <iostream>
2 #include <list>
3 #include <unordered_map>
4
5 using namespace std;
6
7 int main() {
8     int t; // Number of test cases
9     cin >> t;
10
11     while (t-- > 0) {
12         int n, q; // Length of sequence and
13         // number of queries
14         cin >> n >> q;
15
16         list<int> sequence; // To store the
17         // current sequence
18         unordered_map<int, list<int>>::
19             iterator positions; // To store
20             // positions of elements
21
22         // Initialize the sequence from 1 to
23         // n
24         for (int i = 1; i <= n; ++i) {
25             sequence.push_back(i);
26             positions[i] = --sequence.end();
27             // Store iterator pointing
28             // to the element
29         }
30
31         // Process each query
32         for (int i = 0; i < q; ++i) {
33             char type;
34             int x;
35             cin >> type >> x;
36
37             // Remove x from its current
38             // position

```

```

31     if (positions.find(x) !=
32         positions.end()) {
33         sequence.erase(positions[x]);
34     }
35
36     // Move x to the head or the
37     // tail
38     if (type == 'H') {
39         // Move x to the head
40         sequence.push_front(x);
41         positions[x] = sequence.
42             begin();
43     } else if (type == 'T') {
44         // Move x to the tail
45         sequence.push_back(x);
46         positions[x] = --sequence.
47             end();
48     }
49
50     // Print the final sequence
51     for (int i : sequence) {
52         cout << i << " ";
53     }
54     cout << endl;
55 }

```

4.2 Banana milk lover

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4
5 using namespace std;
6
7 struct Group {
8     int index;
9     int people;
10     long long total_banana_milk;
11     int max_contribution;
12     vector<int> contributions;
13
14     bool operator<(const Group &other) const
15     {
16         if (total_banana_milk != other.
17             total_banana_milk) {
18             return total_banana_milk > other
19                 .total_banana_milk; //
20                 // Descending by total banana
21                 // milk
22         }
23         if (max_contribution != other.
24             max_contribution) {
25             return max_contribution > other.
26                 max_contribution; //
27                 // Descending by max
28                 // contribution
29         }
30         if (people != other.people) {

```

```

22         return people > other.people; //
23         // Descending by number of
24         // people
25     }
26     return index < other.index; //
27     // Ascending by input order
28 }
29
30 int main() {
31     int T;
32     cin >> T;
33     while (T-- > 0) {
34         int N;
35         cin >> N;
36         vector<Group> groups;
37
38         for (int i = 0; i < N; ++i) {
39             int K;
40             cin >> K;
41             Group group;
42             group.index = i;
43             group.people = K;
44             group.total_banana_milk = 0;
45             group.max_contribution = 0;
46             group.contributions.resize(K);
47
48             for (int j = 0; j < K; ++j) {
49                 cin >> group.contributions[j];
50             }
51             group.total_banana_milk +=
52                 group.contributions[j];
53             group.max_contribution = max
54                 (group.max_contribution,
55                 group.contributions[j]);
56         }
57
58         sort(groups.begin(), groups.end());
59
60         for (const auto &group : groups) {
61             for (int contribution : group.
62                 contributions) {
63                 cout << contribution << " ";
64             }
65             cout << endl;
66         }
67     }
68     return 0;
69 }

```

4.3 Broken Keyboard (a.k.a. Beiju Text)

```

1 #include <iostream>
2 #include <list>
3 #include <string>
4

```

```

5 int main() {
6     std::string line;
7     while (std::getline(std::cin, line)) {
8         std::list<char> text;
9         auto it = text.begin();
10        for (char ch : line) {
11            if (ch == '[') {
12                it = text.begin();
13            } else if (ch == ']') {
14                it = text.end();
15            } else {
16                text.insert(it, ch);
17            }
18        }
19        for (char ch : text) {
20            std::cout << ch;
21        }
22        std::cout << std::endl;
23    }
24    return 0;
25 }

```

4.4 Hunting and Distributing Meals

```

1 #include <iostream>
2 #include <vector>
3 #include <string>
4 #include <algorithm>
5 #include <map>
6
7 using namespace std;
8
9 // 定義貓的結構體
10 struct Cat {
11     string name;
12     string position;
13     int age;
14 };
15
16 // 定義地位優先級映射
17 map<string, int> priority = {
18     {"elder", 1}, {"nursy", 2}, {"kit", 3},
19     {"warrior", 4}, {"apprentice", 5}, {"medicent", 6},
20     {"deputy", 7}, {"Leader", 8}
21 };
22
23 // 排序函數
24 bool compareCats(const Cat& a, const Cat& b) {
25     // 根據地位優先級排序
26     if (priority[a.position] != priority[b.position])
27         return priority[a.position] < priority[b.position];
28
29     // 同一地位時，按年齡排序
30     if (a.position == "apprentice") {
31         // 對於 apprentice，年齡小的優先
32         if (a.age != b.age)
33             return a.age < b.age;
34     } else {

```

```

35         // 對其他地位，年齡大的優先
36         if (a.age != b.age)
37             return a.age > b.age;
38     }
39
40     // 若地位和年齡相同，按名稱的字典序排序
41     return a.name < b.name;
42 }
43
44 int main() {
45     int N, M;
46     // 讀取輸入，直到 EOF
47     while (cin >> N >> M) {
48         // 邊界檢查
49         if (N <= 0 || M <= 0) {
50             cout << "No valid data." << endl;
51             continue;
52         }
53
54         vector<Cat> cats(N);
55
56         // 讀取每只貓的信息
57         for (int i = 0; i < N; ++i) {
58             cin >> cats[i].name >> cats[i].position >> cats[i].age;
59             // 檢查地位是否在優先級映射中
60             if (priority.find(cats[i].position) == priority.end()) {
61                 cerr << "Invalid position: "
62                     << cats[i].position << endl;
63                 return 1;
64             }
65
66             // 排序
67             sort(cats.begin(), cats.end(), compareCats);
68
69             // 輸出前 M 個貓的名字，避免 M 大於 N 的情況
70             for (int i = 0; i < min(M, N); ++i) {
71                 cout << cats[i].name << endl;
72             }
73         }
74
75         return 0;
76 }

```

4.5 Problem H. Yet Another Alice and Bob

```

1 #include <iostream>
2 #include <vector>
3 #include <set>
4 #include <tuple>
5 using namespace std;
6

```

```

7 struct Position {
8     long long x, y;
9     bool operator<(const Position& other) const {
10         return tie(x, y) < tie(other.x, other.y);
11     }
12     bool operator==(const Position& other) const {
13         return x == other.x && y == other.y;
14     }
15 };
16
17 int main() {
18     long long k;
19     int n, m;
20     cin >> k >> n >> m;
21
22     vector<long long> aliceTurns(n), bobTurns(m);
23     for (int i = 0; i < n; ++i) cin >> aliceTurns[i];
24     for (int i = 0; i < m; ++i) cin >> bobTurns[i];
25
26     set<Position> redCells, blueCells, purpleCells;
27     Position alice = {1, 1}, bob = {1, 1};
28     bool aliceRight = true, bobDown = true;
29     int aliceTurnIdx = 0, bobTurnIdx = 0;
30
31     // Process each second
32     for (long long t = 1; t <= k; ++t) {
33         // Check if Alice turns
34         if (aliceTurnIdx < n && t == aliceTurns[aliceTurnIdx]) {
35             aliceRight = !aliceRight;
36             aliceTurnIdx++;
37         }
38
39         // Check if Bob turns
40         if (bobTurnIdx < m && t == bobTurns[bobTurnIdx]) {
41             bobDown = !bobDown;
42             bobTurnIdx++;
43         }
44
45         // Move Alice and Bob
46         if (aliceRight) {
47             alice.y++;
48         } else {
49             alice.x++;
50         }
51
52         if (bobDown) {
53             bob.x++;
54         } else {
55             bob.y++;
56         }
57
58         // Skip (1,1) as it remains white
59         if (alice.x == 1 && alice.y == 1) continue;
60         if (bob.x == 1 && bob.y == 1) continue;
61

```

```

62         // Update cell colors
63         if (alice == bob) {
64             // Remove cell from red and blue sets if it exists
65             redCells.erase(alice);
66             blueCells.erase(alice);
67             purpleCells.insert(alice);
68         } else {
69             // Update Alice's position
70             if (!(alice.x == 1 && alice.y == 1)) {
71                 if (!purpleCells.count(alice)) {
72                     blueCells.erase(alice);
73                     redCells.insert(alice);
74                 }
75             }
76
77             // Update Bob's position
78             if (!(bob.x == 1 && bob.y == 1)) {
79                 if (!purpleCells.count(bob)) {
80                     redCells.erase(bob);
81                     blueCells.insert(bob);
82                 }
83             }
84         }
85     }
86
87     // Output the counts of red, blue, and purple cells
88     cout << redCells.size() << " " << blueCells.size() << " " << purpleCells.size() << endl;
89
90     return 0;
91 }
92
93 // 兩個人每秒走一格，過程中只會往右和往下走，問
94 // 秒之後他們在幾個格子相遇。
95 // 觀察：畫出兩個人的路徑，可以觀察到兩路徑相交的點就是他們相遇的點。
96
97 // 最 naive 的作法就是直接一秒一秒模擬兩人走路的过程，但是
98 // 秒 ≤ 10^18
99 // 這麼做會 TLE，所以我們需要一些優化。
100
101 // 想法一：不要每次只有一步。模擬時，一直走直到有人轉彎，此時檢查一下重疊狀況。
102
103 // 想法二：由於只會往右和往下移動，因此可以完全忽略自己左方和上方的路徑。因此模擬時，我們可以每次讓比較左邊的人移動。這樣一來，我們只需要紀錄每個人最後移動的橫線和直線。
104
105 // Time complexity:
106 // ( + + )
107

```



```
109 // 另外的解法：使用掃描線，配上一些資料結
    | 構，可以作到  $O((n + m) \log(n + m))$ 
```

4.6 Queuing Problem

```
1 // #include <iostream>
2 // #include <vector>
3 // #include <algorithm>
4
5 // using namespace std;
6
7 // int where[1000005];
8 // vector<int> queue[1000005];
9
10 // int main() {
11 //     int n,m;
12 //     cin >> n >> m;
13 //     for (int i=1;i<=n;i++) {
14 //         where[i] = i;
15 //         queue[i].push_back(i);
16 //         //queue[i].push_back(3);
17 //     }
18 //     int ins,a,b;
19 //     for (int i=1;i<=m;i++) {
20 //         // for (int j=0;j<queue[i].size()
21 //         ;j++) {
22 //             cout << queue[i][j];
23 //             // }
24 //             // cout << endl;
25 //             cin >> ins >> a >> b;
26 //             if (a == b) continue;
27 //             if (ins == 0) {
28 //                 for (int j=0;j<queue[where[a
29 //                 ]].size();j++) {
30 //                     if (queue[where[a]][j] ==
31 //                     a) {
32 //                         queue[where[a]].erase
33 //                         (queue[where[a]].begin()+j);
34 //                         for (int k=0;k<queue[
35 //                         where[b]].size();k++) {
36 //                             if (queue[where[b
37 //                             ]][k] == b) {
38 //                                 queue[b].
39 //                                 insert(queue[b].begin()+k+1, a);
40 //                                 }
41 //                                 where[a] = where[b];
42 //                                 break;
43 //                             }
44 //                         }
45 //                     }
46 //                     else if (ins == 1) {
47 //                         queue[b].insert(queue[b].end
48 //                         (), queue[a].begin(), queue[a].end());
49 //                         for (int j=0;j<queue[a].size
50 //                         ());j++) {
51 //                             where[queue[a][j]] = b;
52 //                         }
53 //                         queue[a].clear();
54 //                     }
55 //                 }
56 //                 for (int i=1;i<=n;i++) {
57 //                     cout << "#" << i << ":";
```

```
50 //         for(int j=0;j<queue[i].size();j
51 //         ++){
52 //             cout << " " << queue[i][j];
53 //             // }
54 //             cout << "\n";
55 //         }
56 //     }
57 // #include <iostream>
58 // #include <vector>
59 // #include <list>
60 // #include <unordered_map>
61 // using namespace std;
62
63 // class QueueSystem {
64 // private:
65 //     vector<list<int>> queues; // 每個隊列使
66 //     用 list 儲存
67 //     unordered_map<int, pair<int, list<int>::
68 //     iterator>> personPosition; // 記錄每
69 //     個人的當前位置和所屬隊列
70 // public:
71 //     QueueSystem(int n) {
72 //         queues.resize(n + 1); // 1-based
73 //         indexing
74 //
75 //         // 初始化：每個人都在自己的隊列中
76 //         for (int i = 1; i <= n; i++) {
77 //             queues[i].push_back(i);
78 //             personPosition[i] = {i, queues[i
79 //             ].begin()}; // 記錄位置
80 //         }
81 //
82 //         // 指令 0：將人 a 移動到人 b 後方
83 //         void movePerson(int a, int b) {
84 //             // 找到 a 的當前隊列並將其移除
85 //             auto [fromQueue, posA] =
86 //             personPosition[a];
87 //             queues[fromQueue].erase(posA);
88 //
89 //             // 找到 b 的當前隊列，並將 a 插入到
90 //             b 的後面
91 //             int toQueue = personPosition[b].
92 //             first;
93 //             auto posB = personPosition[b].second
94 //             ;
95 //             auto it = queues[toQueue].insert(
96 //             next(posB), a);
97 //
98 //             // 更新 a 的位置
99 //             personPosition[a] = {toQueue, it};
100 //         }
101 //
102 //         // 指令 1：將隊列 a 合併到隊列 b
103 //         void moveQueue(int a, int b) {
104 //             if (a == b || queues[a].empty())
105 //                 return;
106 //
107 //             // 將隊列 a 的所有人移動到隊列 b 的
108 //             末尾
109 //             auto &queueA = queues[a];
110 //             auto &queueB = queues[b];
```

```
100 // 將 a 的所有元素移動到 b 並更新每
101 // 個人的位置
102 // queueB.splice(queueB.end(), queueA);
103 // for (auto it = queueB.begin(); it !=
104 //     queueB.end(); ++it) {
105 //     personPosition[*it] = {b, it};
106 // }
107 // }
108 //
109 // 輸出當前所有隊列的狀態
110 // void printQueues() {
111 //     for (int i = 1; i < queues.size(); i
112 //     ++){
113 //         cout << "#" << i << ":";
114 //         for (int person : queues[i]) {
115 //             cout << " " << person;
116 //         }
117 //         cout << endl;
118 //     }
119 // }
120 //
121 // int main() {
122 //     int n, m;
123 //     cin >> n >> m;
124 //
125 //     QueueSystem system(n);
126 //
127 //     // 處理 m 條指令
128 //     for (int i = 0; i < m; i++) {
129 //         int t, a, b;
130 //         cin >> t >> a >> b;
131 //
132 //         if (t == 0) {
133 //             system.movePerson(a, b);
134 //         } else {
135 //             system.moveQueue(a, b);
136 //         }
137 //     }
138 //
139 //     system.printQueues();
140 //     return 0;
141 // }
```

4.7 Tasks and Deadlines

```
1 // #include <iostream>
2 // #include <vector>
3 // #include <algorithm>
4
5 // using namespace std;
6
7 // int main() {
8 //     int n;
9 //     cin >> n;
10
11 //     vector<pair<int,int>> tasks(n);
12
13 //     for (int i=0;i<n;i++) {
14 //         cin >> tasks[i].first >> tasks[i].
15 //         second ;
```

```
15 //     }
16 //
17 //     sort(tasks.begin(),tasks.end());
18 //
19 //     long long currentTime = 0;
20 //     long long reward = 0;
21 //
22 //     for (int i=0;i<n;i++) {
23 //         currentTime += tasks[i].first;
24 //         reward += (tasks[i].second -
25 //         currentTime);
26 //     }
27 //
28 //     cout << reward << endl;
29 // }
30 //
31 // #include <iostream>
32 // #include <vector>
33 // #include <algorithm>
34 // using namespace std;
35 //
36 // int main() {
37 //     int n;
38 //     cin >> n;
39 //
40 //     vector<pair<int, int>> tasks(n); //
41 //     存储任务的持续时间和截止时间
42 //
43 //     // 读取输入
44 //     for (int i = 0; i < n; i++) {
45 //         cin >> tasks[i].first >> tasks[i
46 //         ].second; // first 是持续时间 a，second
47 //         是截止时间 d
48 //     }
49 //
50 //     // 根据任务的持续时间 a 排序
51 //     sort(tasks.begin(), tasks.end());
52 //
53 //     Long Long currentTime = 0; // 当前完
54 //     成时间
55 //     Long Long totalReward = 0; // 总奖励
56 //
57 //     // 处理每个任务
58 //     for (int i = 0; i < n; i++) {
59 //         currentTime += tasks[i].first; //
60 //         任务完成时间 = 前面的完成时间 + 当前任
61 //         务的持续时间
62 //         totalReward += tasks[i].second -
63 //         currentTime; // 奖励 = d - f (d 是截止时
64 //         间，f 是完成时间)
65 //     }
66 //
67 //     cout << totalReward << endl;
68 //
69 //     return 0;
70 // }
```

5 w4

5.1 2022 Competitive Programming Training (I) Midterm Exam Problem A Task and Penalty

```
1 #include <iostream>
2 #include <string>
3 #include <algorithm>
4 #include <vector>
5
6 using namespace std;
7 vector<int> a(1000005);
8
9 int main() {
10     int n;
11     cin >> n;
12     for (int i=0; i<n; i++) {
13         int tmp;
14         cin >> tmp;
15         a.at(tmp);
16     }
17     sort(a.begin(), a.end());
18     int penalty = 0;
19     int time = 0;
20     for (int i=0; i<n; i++) {
21         time += a[i];
22         penalty += time;
23     }
24     cout << penalty << endl;
25 }
```

5.2 Array Division

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 // Function to check if we can divide the
7 // array into k subarrays with a maximum
8 // sum of 'maxSum'
9 bool canDivide(const vector<int>& nums, int
10 n, int k, long long maxSum) {
11     int subarrayCount = 1;
12     long long currentSum = 0;
13
14     for (int num : nums) {
15         if (currentSum + num > maxSum) {
16             // Start a new subarray
17             subarrayCount++;
18             currentSum = num;
19             if (subarrayCount > k) {
20                 return false; // More
21                 subarrays than allowed
22             }
23         } else {
24             currentSum += num;
25         }
26     }
27 }
```

```
22     }
23
24     return true;
25 }
26
27 // Function to find the minimum possible
28 // maximum subarray sum
29 long long arrayDivision(const vector<int>&
30 nums, int n, int k) {
31     long long left = *max_element(nums.begin
32 (), nums.end()); // Max element in
33     the array
34     long long right = 0;
35     for (int num : nums) {
36         right += num; // Sum of all
37         elements
38     }
39
40     while (left < right) {
41         long long mid = left + (right - left
42 ) / 2;
43
44         if (canDivide(nums, n, k, mid)) {
45             right = mid; // Try to find a
46             smaller maximum subarray sum
47         } else {
48             left = mid + 1; // Increase the
49             maximum sum
50         }
51     }
52
53     return left; // This is the minimized
54     maximum subarray sum
55 }
56
57 int main() {
58     int n, k;
59     cin >> n >> k;
60
61     vector<int> nums(n);
62     for (int i = 0; i < n; ++i) {
63         cin >> nums[i];
64     }
65
66     // Output the minimized maximum subarray
67     sum
68     cout << arrayDivision(nums, n, k) <<
69     endl;
70
71     return 0;
72 }
```

5.3 C. Sum of Three Values

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4
5 using namespace std;
6
7 struct Element {
8     int value;
9     int index;
```

```
10 };
11
12 int main() {
13     int n, x;
14     cin >> n >> x; // 讀取數組大小和目標和
15
16     vector<Element> arr(n); // 存儲數組的值
17     和它們的索引
18     for (int i = 0; i < n; ++i) {
19         cin >> arr[i].value;
20         arr[i].index = i + 1; // 原始位置的
21         索引 · 從 1 開始
22     }
23
24     // 將數組按值進行排序
25     sort(arr.begin(), arr.end(), [](const
26     Element &a, const Element &b) {
27         return a.value < b.value;
28     });
29
30     // 開始搜索三元組
31     for (int i = 0; i < n - 2; ++i) {
32         int target = x - arr[i].value; // 固
33         定一個值 · 剩餘部分需要找到兩個數
34         和為 target
35         int left = i + 1, right = n - 1;
36
37         while (left < right) {
38             int sum = arr[left].value + arr[
39             right].value;
40
41             if (sum == target) {
42                 // 找到解決方案 · 輸出對應的
43                 索引
44                 cout << arr[i].index << " "
45                 << arr[left].index << "
46                 " << arr[right].index <<
47                 endl;
48                 return 0;
49             } else if (sum < target) {
50                 ++left; // 如果當前和小於目
51                 標 · 增加 Left 指針
52             } else {
53                 --right; // 如果當前和大於目
54                 標 · 減少 right 指針
55             }
56         }
57
58         // 如果找不到解決方案 · 輸出 "IMPOSSIBLE"
59         cout << "IMPOSSIBLE" << endl;
60
61         return 0;
62     }
63 }
```

5.4 Cart Racing

```
1 #include <iostream>
2 #include <algorithm>
```

```
3 using namespace std;
4
5 int main() {
6     long long S, T, A, B;
7     cin >> S >> T >> A >> B;
8
9     long long C, D, M;
10    cin >> M >> C >> D;
11
12    // Check if normal speed alone is enough
13    // to complete the track within time
14    if (S <= A * T) {
15        cout << "Yes" << endl;
16        cout << (S + A - 1) / A << endl; //
17        Minimum time needed at normal
18        speed
19        return 0;
20    }
21
22    // Initialize variables for simulation
23    long long time = 0;
24    long long distance_covered = 0;
25    long long gas = M;
26
27    while (time < T) {
28        // Calculate max distance with
29        // current gas using high speed
30        long long high_speed_time = min(gas
31        / C, T - time); // Time we can
32        go at high speed with available
33        gas
34        distance_covered += high_speed_time
35        * B; // Distance covered at
36        high speed
37        gas -= high_speed_time * C; // Gas
38        consumed at high speed
39        time += high_speed_time;
40
41        // Check if we've completed the race
42        if (distance_covered >= S) {
43            cout << "Yes" << endl;
44            cout << time << endl;
45            return 0;
46        }
47
48        // Check if remaining time is enough
49        // to cover the remaining distance
50        // at normal speed
51        long long remaining_distance = S -
52        distance_covered;
53        if (remaining_distance <= (T - time)
54        * A) {
55            cout << "Yes" << endl;
56            cout << time + (
57            remaining_distance + A - 1)
58            / A << endl;
59            return 0;
60        }
61
62        // Refill gas if time allows
63        if (gas < C) {
64            long long refill_time = (C - gas
65            + D - 1) / D;
66            if (time + refill_time >= T) {
67                return 0;
68            }
69        }
70    }
```

```

51         break; // No time left to
           refill and use high
           speed
52     }
53     gas += refill_time * D;
54     time += refill_time;
55 }
56 }
57
58 // If we exit the loop without finishing
   , output "No" and the max reachable
   distance
59 distance_covered += (T - time) * A;
60 cout << "No" << endl;
61 cout << distance_covered << endl;
62
63 return 0;
64 }

```

5.6 Points to Cost

5.5 Final Day

```

1 // only need to consider the best case
2 // if want to become rank k => then
   consider add "300" can be higher than
   score[k]
3 // if want to lower the rank => minus 300
   can be lower than rank k (in the
   becoming rank k situation => )
4
5 #include <bits/stdc++.h>
6
7 using namespace std;
8 int n,k,num;
9 int p[100005];
10 int score[100005];
11 // bool ans[100005];
12
13 int main()
14 {
15     ios::sync_with_stdio(0);
16     cin.tie(0);
17     cin >> n >> k;
18     for(int i = 0; i < n; i++)
19     {
20         int total = 0;
21         for(int h = 0; h < 3; h++)
22         {
23             cin >> num;
24             total += num;
25         }
26         p[i] = total;
27         score[i] = total; // score rank
28     }
29     sort(score, score + n, greater<int>()); //
       big to small
30
31     for(int i = 0; i < n; i++)
32     {
33         // if(i+1 <= k)
34         // {
35         //     if(p[i]-300 <= score[k-1])
36         //         cout << "Yes\n";
37         //     else cout << "No\n";

```

```

37     // }
38     // else // need add score to be rank
       k
39     // {
40     //     if(p[i]+300 >= score[k-1])
41     //         cout << "Yes\n";
42     //     else cout << "No\n";
43     // }
44
45 // can go in the top k rank?
46 if(p[i] + 300 >= score[k-1]) cout <<
   "Yes\n";
47 else cout << "No\n";
48 }
49 return 0;

```

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4
5 const double eps = 1e-9;
6 const ll mxn = 200005;
7 ll x[mxn];
8 ll y[mxn];
9 ll n, c;
10
11 double f(double p) {
12     double cost = 0;
13     for (ll i = 0; i < n; i++) cost += (x[i]
       - p) * (x[i] - p);
14     return cost;
15 }
16
17 int main() {
18     (void)!scanf("%lld%lld", &n, &c);
19     for (ll i = 0; i < n; i++) (void)!scanf(
       "%lld%lld", &x[i], &y[i]);
20     ll ycost = 0;
21     for (ll i = 0; i < n; i++) ycost += (y[i]
       - c) * (y[i] - c);
22
23     double l = *min_element(x, x + n), r = *
       max_element(x, x + n);
24     while (r - l > eps) {
25         double lm = l + (r - l) / 2;
26         double rm = l + (r - l) / 2;
27         if (f(lm) < f(rm)) r = rm;
28         else l = lm;
29     }
30     printf("%.15lf\n", f(l + (r - l) / 2) +
       ycost);
31 }

```

5.7 Subarray Sums

```

1 // #include <iostream>
2 // #include <unordered_map>

```

```

3 // #include <vector>
4
5 // using namespace std;
6
7 // int main() {
8 //     int n;
9 //     long long x;
10 //     cin >> n >> x;
11
12 //     vector<long long> a(n);
13 //     for (int i = 0; i < n; ++i) {
14 //         cin >> a[i];
15 //     }
16
17 //     unordered_map<long long, vector<int>>
       prefix_sums; // 用來記錄 prefix_sum 和
       對應的索引
18 //     prefix_sums[0].push_back(-1); // 處理
       從開頭到某一點的子數組情況，-1 表示從開
       頭開始
19
20 //     long long current_sum = 0;
21 //     long long count = 0;
22
23 //     for (int i = 0; i < n; ++i) {
24 //         current_sum += a[i];
25
26 //         // 檢查是否存在 current_sum - x
27 //         if (prefix_sums.find(current_sum
       - x) != prefix_sums.end()) {
28 //             // 找到符合條件的子數組，從每
       個滿足條件的 prefix_sum 出現的起始點開始
29 //             for (int start : prefix_sums[
       current_sum - x]) {
30 //                 // cout << "Subarray from
       index " << (start + 1) << " to " << i
       << " sums to " << x << endl;
31 //             }
32 //             count += prefix_sums[
       current_sum - x].size(); // 增加符合條件
       的子數組數量
33 //         }
34
35 //         // 更新 prefix_sums
36 //         prefix_sums[current_sum].
       push_back(i);
37 //     }
38 //     // << "Total subarrays: "
39 //     cout << count << endl;
40 //     return 0;
41 // }
42 #include <iostream>
43 #include <vector>
44 #include <string>
45
46 using namespace std;
47
48 vector<int> a(1000005);
49 vector<long long> prefix(1000005);
50
51 int main() {
52     int n;
53     long long sum;
54     cin >> n >> sum;

```

```

55     prefix[0] = 0;
56     for (int i=1; i<=n; i++) {
57         cin >> prefix[i];
58         prefix[i] = prefix[i] + prefix[i-1];
59         //cout << prefix[i] << " ";
60     }
61     int cnt = 0;
62     for (int i=0; i<=n; i++) {
63         for (int j=i+1; j<=n; j++) {
64             if (prefix[j] - prefix[i] == sum
       ) {
65                 cnt ++ ;
66             }
67         }
68     }
69     cout << cnt << endl;
70 }

```

5.8 Tian Ji horse racing

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 typedef long long ll;
7 const ll INF = 1e18;
8
9 struct Horse {
10     ll base_speed;
11     ll growth;
12
13     // Calculate speed after M days
14     ll speed_at_day(ll M) const {
15         return base_speed + growth * M;
16     }
17 };
18
19 // Check if it's possible to win K races
   after M days of training
20 bool can_win_k_races(const vector<Horse>&
   my_horses, const vector<ll>&
   opponent_horses,
21                     int K, ll M) {
22     int N = my_horses.size();
23
24     // Calculate my horses' speeds after M
   days
25     vector<ll> my_speeds(N);
26     for (int i = 0; i < N; i++) {
27         my_speeds[i] = my_horses[i].
           speed_at_day(M);
28     }
29
30     // Sort both sides' speeds in ascending
   order
31     vector<ll> opp_speeds = opponent_horses;
32     sort(my_speeds.begin(), my_speeds.end())
       ;
33     sort(opp_speeds.begin(), opp_speeds.end
       ());
34
35     // Use two pointers to count wins

```

```

36 int i = 0, j = 0, wins = 0;
37
38 while (i < N && j < N) {
39     if (my_speeds[i] > opp_speeds[j]) {
40         // If my horse can beat opponent
41         // 's horse, count as a win
42         wins++;
43         j++; // Move to next opponent's
44             horse
45     }
46     i++; // Move to next of my horses
47
48     if (wins >= K) return true; //
49         Already achieved K wins
50 }
51
52 ll solve_test_case() {
53     int N, K;
54     cin >> N >> K;
55
56     vector<Horse> my_horses(N);
57     for (int i = 0; i < N; i++) {
58         cin >> my_horses[i].base_speed >>
59             my_horses[i].growth;
60     }
61
62     vector<ll> opponent_horses(N);
63     for (int i = 0; i < N; i++) {
64         cin >> opponent_horses[i];
65     }
66
67     // Binary search over days
68     ll left = 0, right = 1e9;
69     ll result = -1;
70
71     while (left <= right) {
72         ll mid = left + (right - left) / 2;
73
74         if (can_win_k_races(my_horses,
75             opponent_horses, K, mid)) {
76             result = mid;
77             right = mid - 1; // Try to find
78                 a smaller number of days
79         } else {
80             left = mid + 1;
81         }
82     }
83
84     return result;
85 }
86
87 int main() {
88     ios_base::sync_with_stdio(false);
89     cin.tie(nullptr);
90
91     int T;
92     cin >> T;
93
94     while (T--) {
95         cout << solve_test_case() << "\n";
96     }
97
98     return 0;

```

6 w5

6.1 Circular Sliding Window Maximum

```

1 #include <iostream>
2 #include <vector>
3 #include <deque>
4
5 using namespace std;
6
7 vector<int> circularSlidingWindowMaximum(
8     const vector<int>& arr, int N, int K) {
9     vector<int> result(N); // 儲存結果
10    deque<int> dq;
11
12    // 處理第一圈的窗口最大值
13    for (int i = 0; i < N + K - 1; i++) {
14        int idx = i % N;
15
16        // 移除不在窗口範圍內的元素
17        if (!dq.empty() && dq.front() <= i -
18            K) {
19            dq.pop_front();
20        }
21
22        // 移除隊列中小於當前元素的所有元素
23        while (!dq.empty() && arr[dq.back()]
24            % N] <= arr[idx]) {
25            dq.pop_back();
26        }
27
28        // 將當前索引加入隊列
29        dq.push_back(i);
30
31        // 記錄窗口最大值，從第 K - 1 個元素
32        // 開始
33        if (i >= K - 1 && i - K + 1 < N) {
34            result[i - K + 1] = arr[dq.front()
35                % N];
36        }
37    }
38
39    return result;
40 }
41
42 int main() {
43     int N, K;
44     cin >> N >> K;
45
46     vector<int> arr(N);
47     for (int i = 0; i < N; i++) {
48         cin >> arr[i];
49     }
50
51     vector<int> result =
52         circularSlidingWindowMaximum(arr, N,

```

```

53     K);
54
55     for (int maxVal : result) {
56         cout << maxVal << " ";
57     }
58     cout << endl;
59
60     return 0;
61 }
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95

```

6.2 Find the Medians

```

1 #include <iostream>
2 #include <vector>
3 #include <queue>
4 #include <functional>
5
6 using namespace std;
7
8 vector<int> find_medians(const vector<int> &
9     arr)
10 {
11     int n = arr.size();
12     vector<int> medians;
13
14     // Max-heap to store the smaller half of
15     // numbers
16     priority_queue<int> max_heap;
17
18     // Min-heap to store the larger half of
19     // numbers
20     priority_queue<int, vector<int>, greater
21         <int>> min_heap;
22
23     for (int i = 0; i < n; ++i)
24     {
25         int num = arr[i];
26
27         // Insert the number into the
28         // appropriate heap
29         if (max_heap.empty() || num <=
30             max_heap.top())
31         {
32             max_heap.push(num);
33         }
34         else
35         {
36             min_heap.push(num);
37         }
38
39         // Balance the heaps: max_heap can
40         // only have at most one more
41         // element than min_heap
42         if (max_heap.size() > min_heap.size()
43             + 1)
44         {
45             min_heap.push(max_heap.top());
46             max_heap.pop();
47         }
48         else if (min_heap.size() > max_heap.
49             size() + 1)
50         {
51             max_heap.push(min_heap.top());
52             min_heap.pop();
53         }
54     }
55 }

```

```

43
44     // Current median is the root of
45     // max_heap
46     medians.push_back(max_heap.top());
47 }
48
49 return medians;
50 }
51
52 int main()
53 {
54     int n;
55     cin >> n;
56     vector<int> arr(n);
57     for (int i = 0; i < n; ++i)
58     {
59         cin >> arr[i];
60     }
61
62     // Get medians for each prefix
63     vector<int> medians = find_medians(arr);
64
65     // Print all medians in one line
66     for (int median : medians)
67     {
68         cout << median << " ";
69     }
70     cout << endl;
71
72     return 0;
73 }

```

6.3 Nearest Smaller Values

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 int a[200005];
4 int main() {
5     int n; cin >> n;
6     stack<int> stk;
7     for (int i = 0; i < n; i++) {
8         cin >> a[i];
9         while (!stk.empty() && a[stk.top()]
10             >= a[i]) stk.pop();
11         if (stk.empty()) cout << 0;
12         else cout << stk.top() + 1;
13         cout << " \n"[i + 1 == n];
14         stk.push(i);
15     }
16 }

```

6.4 Nice Boat!

```

1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 // Function to check if a vector is safe
6 bool isSafe(const vector<int>& arr, int
7     start, int len) {

```

6.5 Paint the sticks

```

7  if (len == 0) return false;
8
9  // Check prefix sums
10 long long prefixSum = 0; // Using long
    long to prevent overflow
11 for (int i = start; i < start + len; i
    ++){
12     prefixSum += arr[i];
13     if (prefixSum < 0) return false;
14 }
15
16 // Check postfix sums
17 long long postfixSum = 0;
18 for (int i = start + len - 1; i >= start
    ; i--){
19     postfixSum += arr[i];
20     if (postfixSum < 0) return false;
21 }
22
23 return true;
24 }
25
26 // Function to find the Longest safe
    subarray
27 int findLongestSafeSubarray(const vector<int
    >& arr) {
28     int n = arr.size();
29     int maxLen = 0;
30
31     // Try all possible subarrays
32     for (int i = 0; i < n; i++) {
33         for (int len = 1; len <= n - i; len
            ++){
34             // Check if current subarray is
                safe
35             if (isSafe(arr, i, len)) {
36                 maxLen = max(maxLen, len);
37             }
38         }
39     }
40
41     return maxLen;
42 }
43
44 int main() {
45     ios_base::sync_with_stdio(false);
46     cin.tie(nullptr);
47
48     int T;
49     cin >> T;
50
51     while (T--) {
52         int N;
53         cin >> N;
54
55         vector<int> arr(N);
56         for (int i = 0; i < N; i++) {
57             cin >> arr[i];
58         }
59
60         cout << findLongestSafeSubarray(arr)
            << '\n';
61     }
62
63     return 0;
64 }

```

```

1 #include <iostream>
2 #include <vector>
3 #include <stack>
4 using namespace std;
5
6 long long maxRectangle(vector<int> h)
7 {
8     h.emplace_back(0); // Add a sentinel
        value to handle remaining heights
9     stack<pair<int, int>> STK;
10    long long ans = 0;
11    for (int i = 0; i < (int)h.size(); ++i)
12    {
13        int corner = i;
14        while (!STK.empty() && STK.top().
            first >= h[i])
15        {
16            corner = STK.top().second;
17            ans = max(ans, 1LL * (i - corner
                ) * STK.top().first);
18            STK.pop();
19        }
20        STK.emplace(h[i], corner);
21    }
22    return ans;
23 }
24
25 int main()
26 {
27     int n;
28     while (cin >> n)
29     {
30         vector<int> heights(n);
31         for (int i = 0; i < n; ++i)
32         {
33             cin >> heights[i];
34         }
35         cout << maxRectangle(heights) <<
            endl;
36     }
37     return 0;
38 }

```

6.6 Sorting Queries

```

1 #include <iostream>
2 #include <queue>
3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(nullptr);
8
9     int Q;
10    cin >> Q;
11
12    queue<int> regular_queue; //
        For elements in original order

```

```

13 priority_queue<int, vector<int>, greater
    <int>> sorted_queue; // For sorted
        elements
14
15 while (Q--) {
16     int type;
17     cin >> type;
18
19     if (type == 1) {
20         int x;
21         cin >> x;
22         regular_queue.push(x);
23     }
24     else if (type == 2) {
25         // If we have sorted elements,
            take from there first
26         if (!sorted_queue.empty()) {
27             cout << sorted_queue.top()
                << "\n";
28             sorted_queue.pop();
29         }
30         // Otherwise take from regular
            queue
31         else {
32             cout << regular_queue.front()
                << "\n";
33             regular_queue.pop();
34         }
35     }
36     else { // type == 3
37         // Move all elements from
            regular queue to sorted
            queue
38         while (!regular_queue.empty()) {
39             sorted_queue.push(
                regular_queue.front());
40             regular_queue.pop();
41         }
42     }
43 }
44
45 return 0;
46 }

```

6.7 SString

```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     ios_base::sync_with_stdio(false);
7     cin.tie(nullptr);
8
9     string X;
10    cin >> X;
11
12    // Count S and T separately
13    int countS = 0;
14    for (char c : X) {
15        if (c == 'S') countS++;
16    }
17    int countT = X.length() - countS;

```

```

18 // Keep track of unpaired S's we've seen
19 int unpairedS = 0;
20 int removedPairs = 0;
21
22 // Process string from left to right
23 for (char c : X) {
24     if (c == 'S') {
25         unpairedS++;
26     } else { // c == 'T'
27         if (unpairedS > 0) {
28             // We can form an ST pair
29             unpairedS--;
30             removedPairs++;
31         }
32     }
33 }
34
35 // Calculate final length
36 // Each removed pair reduces length by 2
37 int finalLength = X.length() - (
    removedPairs * 2);
38
39 cout << finalLength << endl;
40
41 return 0;
42 }
43 }

```

7 w6

7.1 Counting Rooms

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const int dx[] = {-1, 0, 1, 0};
4 const int dy[] = {0, 1, 0, -1};
5
6 int main() {
7     int n, m; cin >> n >> m;
8     vector<vector<char>> a(n + 2, vector<
        char>(m + 2, '#')));
9     for (int i = 1; i <= n; i++)
10     for (int j = 1; j <= m; j++)
11         cin >> a[i][j];
12
13     function<void(int, int)> dfs = [&](int x
        , int y) {
14         a[x][y] = '#';
15         for (int k = 0; k < 4; k++) {
16             int nx = x + dx[k], ny = y + dy[
                k];
17             if (a[nx][ny] == '.') dfs(nx, ny
                );
18         }
19     };
20
21     int ans = 0;
22     for (int i = 1; i <= n; i++)
23     for (int j = 1; j <= m; j++)
24         if (a[i][j] == '.') ans++, dfs(i
            , j);
25
26     cout << ans << '\n';
27 }

```


7.2 Grid Maze

```

1 #include <iostream>
2 #include <vector>
3 #include <queue>
4 #include <string>
5 #include <algorithm> // 添加此行以包含
   reverse 函數
6
7 using namespace std;
8
9 struct Position {
10     int x, y;
11 };
12
13 // 四個移動方向和對應的方向字符
14 const vector<Position> directions = {{-1,
15     0}, {1, 0}, {0, -1}, {0, 1}};
16 const string dirChars = "UDLR";
17
18 bool isValid(int x, int y, int n, int m,
19     const vector<vector<char>>& maze) {
20     return x >= 0 && x < n && y >= 0 && y <
21     m && maze[x][y] != '#';
22 }
23
24 int main() {
25     int n, m;
26     cin >> n >> m;
27
28     vector<vector<char>> maze(n, vector<char>
29     >(m));
30     Position start, end;
31
32     // 讀取迷宮並找到起點和終點
33     for (int i = 0; i < n; i++) {
34         for (int j = 0; j < m; j++) {
35             cin >> maze[i][j];
36             if (maze[i][j] == 'A') start = {
37                 i, j};
38             if (maze[i][j] == 'B') end = {i,
39                 j};
40         }
41     }
42
43     // BFS 初始化
44     queue<Position> q;
45     q.push(start);
46     vector<vector<int>> steps(n, vector<int>
47     >(m, -1));
48     vector<vector<int>> fromDirection(n,
49     vector<int>(m, -1));
50     steps[start.x][start.y] = 0;
51
52     bool found = false;
53
54     // BFS 遍歷
55     while (!q.empty() && !found) {
56         Position pos = q.front();
57         q.pop();
58
59         for (int d = 0; d < 4; d++) {
60             int nx = pos.x + directions[d].x
61             ;

```

```

62         int ny = pos.y + directions[d].y
63         ;
64         if (isValid(nx, ny, n, m, maze)
65             && steps[nx][ny] == -1) {
66             steps[nx][ny] = steps[pos.x
67             ][pos.y] + 1;
68             fromDirection[nx][ny] = d;
69             q.push({nx, ny});
70
71             // 若找到終點・結束搜尋
72             if (nx == end.x && ny == end
73             .y) {
74                 found = true;
75                 break;
76             }
77         }
78     }
79
80     // 檢查結果
81     if (steps[end.x][end.y] != -1) {
82         cout << "YES\n";
83         cout << steps[end.x][end.y] << '\n';
84
85         // 回溯路徑
86         string path;
87         Position pos = end;
88         while (pos.x != start.x || pos.y !=
89             start.y) {
90             int d = fromDirection[pos.x][pos
91             .y];
92             path += dirChars[d];
93             pos.x -= directions[d].x;
94             pos.y -= directions[d].y;
95         }
96         reverse(path.begin(), path.end());
97         cout << path << '\n';
98     } else {
99         cout << "NO\n";
100     }
101     return 0;
102 }

```

7.3 Monsters

```

1 #include <algorithm>
2 #include <climits>
3 #include <cstring>
4 #include <iostream>
5 #include <queue>
6 #include <vector>
7 #define pii pair<int, int>
8 #define mn 1005
9 using namespace std;
10
11 int N, M;
12 queue<pii> q;
13 int paths[mn][mn];
14 pii from[mn][mn];

```

```

15 int oo = INT_MAX;
16 pii A;
17 string ans;
18 bool possible = false;
19
20 void retrace(pii node) { // retrace from
   final node, adding direction from
21     // previous node to
22     // a string. This
23     // string will be
24     // backwards but
25     // will be reversed
26     // before output.
27     pii origin = from[node.first][node.second
28     ];
29     if (origin == pii(0, 0)) return;
30     if (origin.first == node.first + 1) ans.
31     push_back('U');
32     if (origin.first == node.first - 1) ans.
33     push_back('D');
34     if (origin.second == node.second + 1) ans.
35     push_back('L');
36     if (origin.second == node.second - 1) ans.
37     push_back('R');
38     retrace(origin);
39 }
40
41 void check(pii origin,
42     pii dest) { // check if the
43     // considered destination may be
44     // traveled to
45     int pl = paths[origin.first][origin.second
46     ];
47     if (pl + 1 < paths[dest.first][dest.second
48     ]) {
49         paths[dest.first][dest.second] = pl + 1;
50         q.push(dest);
51         from[dest.first][dest.second] = origin;
52     }
53 }
54
55 bool mora = false; // false if bfs for
   monsters, true if bfs for A
56
57 void bfs() {
58     while (!q.empty()) {
59         pii loc = q.front(), next;
60         q.pop();
61         next = loc;
62         next.first++;
63         check(loc, next); // go through
64         // adjacent locations
65         next = loc;
66         next.first--;
67         check(loc, next);
68         next = loc;
69         next.second++;
70         check(loc, next);
71         next = loc;
72         next.second--;
73         check(loc, next);
74         if (mora &&
75             (loc.first == 1 || loc.second == 1
76             || loc.first == N || loc.second
77             == M)) {
78             cout << "YES" << endl;
79             cout << paths[loc.first][loc.second]
80             << endl;
81             retrace(loc);
82         }
83     }
84 }

```

```

62     possible = true;
63     return;
64 }
65 }
66 }
67
68 int main() {
69     cin >> N >> M;
70     for (int i = 1; i <= N; i++) {
71         string s;
72         cin >> s;
73         for (int j = 1; j <= M; j++) {
74             paths[i][j] = oo;
75             if (s[j - 1] == '#') paths[i][j] = 0;
76             if (s[j - 1] == 'M') {
77                 q.push(pii(i, j));
78                 paths[i][j] = 0;
79             }
80             if (s[j - 1] == 'A') {
81                 A.first = i;
82                 A.second = j;
83             }
84         }
85     }
86     bfs(); //
87     // monster bfs
88     mora = true; //
89     // change next bfs to A bfs
90     from[A.first][A.second] = pii(0, 0); //
91     // give the retrace a terminating
92     // location
93     paths[A.first][A.second] = 0;
94     q.push(A); // get ready for next bfs
95     bfs(); // bfs with A
96     if (possible) {
97         reverse(ans.begin(), ans.end());
98         cout << ans << endl;
99     } else cout << "NO" << endl;
100 }

```

7.4 Problem A - Rush Hour Puzzle

```

1 #include <bits/stdc++.h>
2 #define ll long long
3 #define maxn 2010
4 #define mod 998244353
5 using namespace std;
6 struct cv {
7     int mp[6][6];
8     int mov;
9     friend bool operator<(cv p, cv q) {
10         for (int i = 0; i < 6; i++) {
11             for (int j = 0; j < 6; j++) {
12                 if (p.mp[i][j] != q.mp[i][j]) return
13                 p.mp[i][j] < q.mp[i][j];
14             }
15         }
16         return p.mp[0][0] < q.mp[0][0];
17     }
18 }dd, hf;
19 int to[4][2] = {0, 1, 0, -1, 1, 0, -1, 0};
20 int main() {
21     for (int i = 0; i < 6; i++) {
22         for (int j = 0; j < 6; j++) {

```

7.5 Rubik 2³

```

22     scanf("%d", &dd.mp[i][j]);
23 }
24 }
25 dd.mov = 0;
26 queue<cv>q;
27 q.push(dd);
28 set<cv>st;
29 st.insert(dd);
30 while (!q.empty()) {
31     dd = q.front();
32     q.pop();
33     if (dd.mp[2][5] == 1) {
34         int cnt = 1;
35         for (int i = 4; i >= 0; i--) {
36             if (dd.mp[2][i] != 1) break;
37             cnt++;
38         }
39         int ans = (dd.mov + cnt);
40         if (ans > 10) ans = -1;
41         printf("%d\n", ans);
42         return 0;
43     }
44     if (dd.mov == 10) break;
45     for (int i = 0; i < 6; i++) {
46         for (int j = 0; j < 6; j++) {
47             if (dd.mp[i][j] == 0) {
48                 for (int k = 0; k < 4; k++) {
49                     int x = i + to[k][0], y = j + to
50                         [k][1];
51                     if (x < 0 || x >= 6 || y < 0 ||
52                         y >= 6 || dd.mp[x][y] == 0)
53                         continue;
54                     int xx = x + to[k][0], yy = y +
55                         to[k][1];
56                     if (xx < 0 || xx >= 6 || yy < 0
57                         || yy >= 6 || dd.mp[xx][yy] !=
58                         dd.mp[xx][yy]) {
59                         continue;
60                     }
61                     while (xx >= 0 && xx < 6 && yy
62                         >= 0 && yy < 6 && dd.mp[xx][y
63                         ] == dd.mp[xx][yy]) {
64                         xx = xx + to[k][0], yy = yy +
65                         to[k][1];
66                     }
67                     xx = xx - to[k][0], yy = yy - to
68                     [k][1];
69                     hf = dd;
70                     hf.mov++;
71                     swap(hf.mp[i][j], hf.mp[xx][yy])
72                     if (!st.count(hf)) {
73                         st.insert(hf);
74                         q.push(hf);
75                     }
76                 }
77             }
78         }
79     }
80     printf("-1\n");
81     return 0;
82 }

```

```

1 // -----
2 // SLPC2009 - Rubik 2^3 solution, verifier,
3 // and generator all-in-one :)
4 // Run "rubik2 < (input)" to solve, "rubik2
5 // -g" to generate tests, or
6 // "rubik2 (output) < (input)" to verify
7 // output against input
8 // -----
9
10 #include <iostream>
11 #include <fstream>
12 #include <string>
13 #include <vector>
14 #include <set>
15 #include <ctime>
16
17 using namespace std;
18
19 int g_wrap[][8] = {
20     { -1, -1, 0, 1, -1, -1, -1, -1 },
21     { -1, -1, 3, 2, -1, -1, -1, -1 },
22     { 4, 5, 8, 9, 12, 13, 16, 17 },
23     { 7, 6, 11, 10, 15, 14, 19, 18 },
24     { -1, -1, 20, 21, -1, -1, -1, -1 },
25     { -1, -1, 23, 22, -1, -1, -1, -1 }
26 };
27
28 int g_turn[3][24] = {
29     { 0, 19, 16, 3, 4, 5, 6, 7, 8, 1, 2, 11,
30       13, 14, 15, 12, 22, 17, 18, 21, 20,
31       9, 10, 23 },
32     { 1, 2, 3, 0, 16, 17, 6, 7, 4, 5, 10,
33       11, 8, 9, 14, 15, 12, 13, 18, 19,
34       20, 21, 22, 23 },
35     { 0, 1, 15, 12, 4, 2, 3, 7, 9, 10, 11,
36       8, 21, 13, 14, 20, 16, 17, 18, 19,
37       5, 6, 22, 23 }
38 };
39
40 const string g_moves[] = { "X", "Y", "Z", "
41 XXX", "YYY", "ZZZ" };
42 const string g_base[2] = {
43     "WWWB BBBB OOOO GGGG RRRR RYYY",
44     "RRRR GGGG BBBB YYYYY WWWO OOOO"
45 };
46 const string g_dots = "
47 .....";
48
49 const int g_mode = 3; // can also do a
50 branch factor 6 search, but it's slower
51
52 set<string> g_seen;
53
54 // function to convert an internal
55 representation into a "wrapper map"

```

```

46 void output(const string &s, ostream &stream
47 )
48 {
49     for (int i = 0; i < 6; ++i) {
50         for (int j = 0; j < 8; ++j) {
51             if (g_wrap[i][j] >= 0) stream <<
52                 s[g_wrap[i][j]];
53             else stream << '.';
54         }
55         stream << endl;
56     }
57 }
58
59 string transform(int turn, const string &s)
60 {
61     // apply the permutation on the tiles
62     // from s to form new ordering t
63     string t = g_dots;
64     if (turn < 3)
65         for (int i = 0; i < 24; ++i) t[i] =
66             s[g_turn[turn][i]];
67     else
68         for (int i = 0; i < 24; ++i) t[
69             g_turn[turn-3][i]] = s[i];
70     return t;
71 }
72
73 bool solved(const string &s)
74 {
75     // check that each of the six faces have
76     // all matching colours
77     for (int i = 0; i < 6; ++i) {
78         int j = i*4;
79         char colour = s[j];
80         for (++j; j < (i+1)*4; ++j)
81             if (s[j] != colour) return false;
82     }
83     return true;
84 }
85
86 string scramble(string s, int steps = 42)
87 {
88     for (int i = 0; i < steps; ++i)
89         s = transform(rand()%6, s);
90     return s;
91 }
92
93 bool bdfs(int depth, int limit, const string
94 &s, string moves)
95 {
96     // cache the configuration so we don't
97     // search it more than once
98     if (g_seen.find(s) != g_seen.end())
99         return false;
100     g_seen.insert(s);
101
102     // hooray!
103     if (solved(s)) {
104         cout << moves << endl;
105         return true;
106     }
107
108     // recurse, transforming the cube and
109     // deeping the level
110     if (depth < limit) {

```

```

101     int m = rand() % g_mode;
102     for (int i = 0; i < g_mode; ++i, m =
103         (m+1)%g_mode)
104         if (bdfs(depth+1, limit,
105             transform(m, s), moves+
106             g_moves[m]))
107             return true;
108     }
109     return false;
110 }
111
112 string readconfig(istream &in)
113 {
114     // read in the map and create an initial
115     // configuration from it
116     string input, config = g_dots;
117     for (int i = 0; i < 6; ++i) {
118         in >> input;
119         for (int j = 0; j < 8; ++j)
120             if (g_wrap[i][j] >= 0) config[
121                 g_wrap[i][j]] = input[j];
122     }
123     return config;
124 }
125
126 int main()
127 {
128     istream &in = cin;
129     srand(time(0));
130
131     for (;;)
132     {
133         // read and check for sentinel
134         string initial = readconfig(in);
135         if (initial == g_dots) break;
136
137         // perform a bounded depth-first
138         // search until we're done
139         for (int d = 1; ; ++d) {
140             g_seen.clear();
141             if (bdfs(0, d, initial, ""))
142                 break;
143             //cout << "depth " << d << " -
144             // configurations searched: "
145             // << g_seen.size() << endl;
146         }
147     }
148     return 0;
149 }

```

7.6 Swap Game

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 unordered_set<string> visited;
5
6 // Defined globally to be used in
7 // process_swap
8 queue<pair<string, int>> q;
9 int moves;

```

```

9 string curboard;
10
11 // Processing swapping the numbers in x, y
   positions
12 void process_swap(int x, int y) {
13     swap(curboard[x], curboard[y]);
14     // Check whether already visited this
       potential board
15     if (visited.find(curboard) == visited.end()
       ) {
16         q.push({curboard, moves + 1});
17         visited.insert(curboard);
18     }
19     // Restore to original board
20     swap(curboard[x], curboard[y]);
21 }
22
23 int main() {
24     string inp;
25     // Rewriting the input as a string
26     for (int i = 0; i < 9; i++) {
27         int a;
28         cin >> a;
29         inp += to_string(a - 1);
30     }
31
32     q.push({inp, 0});
33     while (!q.empty()) {
34         tie(curboard, moves) = q.front();
35         q.pop();
36         if (curboard == "012345678") {
37             cout << moves << endl;
38             return 0;
39         }
40
41         // Horizontal swaps
42         for (int i = 0; i < 9; i += 3) {
43             process_swap(i, i + 1);
44             process_swap(i + 1, i + 2);
45         }
46         // Vertical swaps
47         for (int i = 0; i < 3; i++) {
48             process_swap(i, i + 3);
49             process_swap(i + 3, i + 6);
50         }
51     }
52 }

```

7.7 Water Jug Puzzle

```

1 #include <iostream>
2 #include <vector>
3 #include <queue>
4 #include <set>
5 #include <tuple>
6 using namespace std;
7
8 // 狀態表示：每個水壺的水量
9 typedef vector<int> State;
10
11 // 檢查是否已經達到目標水量
12 bool reachedTarget(const State& state, int Q
   ) {

```

```

13     for (int water : state) {
14         if (water == Q) return true;
15     }
16     return false;
17 }
18
19 // 獲取所有可能的下一步狀態
20 vector<State> getNextStates(const State&
   current, const vector<int>& capacities)
21 {
22     vector<State> nextStates;
23     int n = capacities.size();
24
25     // 每個水壺的裝滿和清空操作
26     for (int i = 0; i < n; i++) {
27         // 裝滿水壺 i
28         State filled = current;
29         filled[i] = capacities[i];
30         nextStates.push_back(filled);
31
32         // 清空水壺 i
33         State emptied = current;
34         emptied[i] = 0;
35         nextStates.push_back(emptied);
36     }
37
38     // 倒水操作
39     for (int i = 0; i < n; i++) {
40         for (int j = 0; j < n; j++) {
41             if (i != j && current[i] > 0 &&
               current[j] < capacities[j])
42             {
43                 State transferred = current;
44                 int transferAmount = min(
                   current[i], capacities[j]
                   ) - current[j];
45                 transferred[i] -=
                   transferAmount;
46                 transferred[j] +=
                   transferAmount;
47                 nextStates.push_back(
                   transferred);
48             }
49         }
50     }
51
52     return nextStates;
53 }

```

```

54 // 主函數：計算達到目標水量的最小步數
55 int minStepsToReachTarget(const vector<int>&
   capacities, int Q) {
56     int n = capacities.size();
57     State initial(n, 0); // 初始狀態：所有水
       壺都是空的
58     queue<pair<State, int>> q; // 狀態隊列，
       儲存狀態和步數
59     set<State> visited; // 訪問過的狀態
60
61     q.push({initial, 0});
62     visited.insert(initial);
63
64     while (!q.empty()) {
65         State current = q.front().first;

```

```

65         int steps = q.front().second;
66         q.pop();
67
68         // 檢查是否已經達到目標水量
69         if (reachedTarget(current, Q)) {
70             return steps;
71         }
72
73         // 產生下一步的所有可能狀態
74         for (const State& next :
           getNextStates(current,
               capacities)) {
75             if (visited.find(next) ==
               visited.end()) {
76                 visited.insert(next);
77                 q.push({next, steps + 1});
78             }
79         }
80
81         // 如果無法達到目標水量，返回 -1
82         return -1;
83     }
84 }
85
86 int main() {
87     int N, Q;
88     cin >> N;
89     vector<int> capacities(N);
90     for (int i = 0; i < N; i++) {
91         cin >> capacities[i];
92     }
93     cin >> Q;
94
95     int result = minStepsToReachTarget(
       capacities, Q);
96     cout << result << endl;
97
98     return 0;
99 }

```

8 w7

8.1 Binomial Coefficients

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const int N = 1e6 + 5;
5 const int mod = 1e9 + 7;
6 ll f[N];
7 ll fp(ll a, ll b) {
8     ll ret = 1;
9     for (; b >= 1; a = a * a % mod)
10         if (b & 1) ret = ret * a % mod;
11     return ret;
12 }
13
14 ll inv(ll x) {return fp(x, mod - 2);}
15
16 int main() {

```

```

17     f[0] = 1;
18     for (int i = 1; i <= 1e6; i++) f[i] = f[
       i - 1] * i % mod;
19     int q; cin >> q;
20     while (q--) {
21         int a, b; cin >> a >> b;
22         cout << f[a] * inv(f[b]) * f[a - b] %
           mod % mod << '\n';
23     }
24 }

```

8.2 Common Divisors

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 const int MAX_VAL = 1000000;
7
8 int main() {
9     int n;
10    cin >> n;
11    vector<int> arr(n);
12    vector<int> freq(MAX_VAL + 1, 0);
13
14    // 讀入數列並計算每個數的出現次數
15    for (int i = 0; i < n; ++i) {
16        cin >> arr[i];
17        freq[arr[i]]++;
18    }
19
20    // 計算每個因數的倍數的出現次數
21    for (int i = 1; i <= MAX_VAL; ++i) {
22        for (int j = i + i; j <= MAX_VAL; j
           += i) {
23            freq[i] += freq[j];
24        }
25    }
26
27    // 找到最大的能被至少兩個數整除的因數
28    for (int i = MAX_VAL; i >= 1; --i) {
29        if (freq[i] > 1) {
30            cout << i << endl;
31            return 0;
32        }
33    }
34
35    return 0;
36 }

```

8.3 Counting Coprime Pairs

```

1 /*
2 Problem Name: Counting Coprime Pairs
3 Problem Link: https://cses.fi/problemset/
   task/2417
4 Author: Sachin Srivastava (mrsac7)
5 */

```

8.4 Exponentiation II

```

6 #include <bits/stdc++.h>
7 using namespace std;
8
9 #define int long long
10 #define endl '\n'
11
12 const int mxN = 1e6+6;
13 int spf[mxN];
14
15 void sieve() {
16     spf[0] = 1;
17     for (int i = 1; i < mxN; i++)
18         spf[i] = i;
19     for (int i = 2; i*i < mxN; i++) {
20         if (spf[i] == i) {
21             for (int j = i*i; j < mxN; j += i) {
22                 if (spf[j] == j)
23                     spf[j] = i;
24             }
25         }
26     }
27 }
28
29 int cnt[mxN];
30
31 signed main(){
32     ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
33     #ifdef LOCAL
34     freopen("input.txt", "r", stdin);
35     freopen("output.txt", "w", stdout);
36     #endif
37
38     int n; cin >> n;
39     int ans = 0;
40     sieve();
41     for (int i = 0; i < n; i++) {
42         int x; cin >> x;
43         vector<int> v;
44         while (x > 1) {
45             int y = spf[x];
46             v.push_back(y);
47             while (x % y == 0)
48                 x /= y;
49         }
50         int k = v.size();
51         for (int s = 1; s < (1<<k); s++) {
52             int p = 1;
53             for (int j = 0; j < k; j++) {
54                 if (s>>j&1) {
55                     p *= v[j];
56                 }
57             }
58             int sgn = -1;
59             if (__builtin_popcount(s)&1) sgn = 1;
60             ans += sgn*cnt[p];
61             cnt[p]++;
62         }
63     }
64     cout<<n*(n-1)/2 - ans;
65 }

```

```

1 #include <iostream>
2 using namespace std;
3 using ll = long long;
4 const int MOD = 1000000007;
5
6 // 快速冪計算  $a^b \% mod$ 
7 ll mod_exp(ll a, ll b, ll mod) {
8     ll result = 1;
9     while (b > 0) {
10         if (b % 2 == 1) {
11             result = (result * a) % mod;
12         }
13         a = (a * a) % mod;
14         b /= 2;
15     }
16     return result;
17 }
18
19 // 主函數
20 int main() {
21     int n;
22     cin >> n;
23     while (n--) {
24         ll a, b, c;
25         cin >> a >> b >> c;
26         // 計算  $b^c \% (MOD-1)$  · 因為 MOD 是質數
27         ll exp = mod_exp(b, c, MOD - 1);
28         // 計算  $a^{exp} \% MOD$ 
29         ll answer = mod_exp(a, exp, MOD);
30         cout << answer << endl;
31     }
32     return 0;
33 }

```

8.5 Problem D. Candies

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef tuple<ll, ll, ll> tlll;
5
6 // 擴展歐幾里得算法 · 用於求解  $ax + by = gcd(a, b)$ 
7 tlll extgcd(ll a, ll b) {
8     if (b == 0) return {a, 1, 0};
9     auto [g, x, y] = extgcd(b, a % b);
10    return {g, y, x - a / b * y};
11 }
12
13 int main () {
14     ll t;
15     cin >> t;
16     while (t--) {
17         ll n, m, x, y, vx, vy;
18         cin >> n >> m >> x >> y >> vx >> vy;
19
20         // 當 vx 為 0 時 · 只在 x 為 0 或 n
21         // 時可能進口袋

```

```

80 }
22 if (x == 0 || x == n)
23     cout << x << " " << (vy == 1 ? m : 0) << endl;
24
25 else
26     cout << "-1" << endl;
27     continue;
28 }
29
30 // 當 vy 為 0 時 · 只在 y 為 0 或 m
31 // 時可能進口袋
32 if (vy == 0) {
33     if (y == 0 || y == m)
34         cout << (vx == 1 ? n : 0) << " " << y << endl;
35     else
36         cout << "-1" << endl;
37     continue;
38 }
39
40 // 將速度方向為負的初始坐標映射到正方向
41 bool ix = false, iy = false;
42 if (vx == -1) {
43     ix = true;
44     x = n - x;
45 }
46 if (vy == -1) {
47     iy = true;
48     y = m - y;
49 }
50
51 // 使用擴展歐幾里得算法檢查是否存在解
52 auto [g, a, b] = extgcd(n, m);
53 if ((x - y) % g != 0) {
54     cout << "-1" << endl;
55     continue;
56 }
57
58 // 轉換到可行解
59 ll n2 = n / g;
60 ll m2 = m / g;
61 ll k = (x - y) / g;
62 a *= k;
63 b *= k;
64
65 // 根據反射確定最終結果
66 if (a == 0) {
67     a = m2;
68     b = -(-n2);
69 } else {
70     a = (a % m2 + m2) % m2;
71     b = -((x - y) - a * n) / m;
72 }
73
74 ll ansx = (a & 1) ? n : 0;
75 ll ansy = (b & 1) ? m : 0;
76 if (ix) ansx = n - ansx;
77 if (iy) ansy = m - ansy;
78
79 cout << ansx << " " << ansy << endl;
80 }
81 return 0;

```

8.6 Sum of Divisors

```

1 #include <iostream>
2
3 using std::cout;
4 using std::endl;
5
6 const int MOD = 1e9 + 7;
7 const int TWO_MOD_INV = 500000004;
8
9 /** @return The sum of all numbers in [start, end] mod MOD. */
10 long long total_sum(long long start, long long end) {
11     return (((end - start + 1) % MOD) * ((start + end) % MOD) % MOD) * TWO_MOD_INV % MOD;
12 }
13
14 int main() {
15     long long n;
16     std::cin >> n;
17
18     long long total = 0;
19     long long at = 1;
20     while (at <= n) {
21         long long add_amt = n / at; // Our divisor to process
22         // The largest number that still has the same value of q
23         long long last_same = n / add_amt;
24
25         total = (total + add_amt * total_sum(at, last_same)) % MOD;
26         at = last_same + 1;
27     }
28
29     cout << total << endl;
30 }

```

ACM ICPC Team Reference - Angry Crow Takes Flight!

Contents

<div>ACM ICPC Team Reference - Angry Crow Takes Flight!</div>																			
Contents																			
1	2022mid1	1	2	w1	3	4	5	w4	10	7	w6	13							
1.1	Problem A. Task and Penalty	1	1.7	Problem G. Gluttony	2	2.1	Apple Division	3	4.3	Broken Keyboard (a.k.a. Beiju Text)	7	6.2	Find the Medians	12					
1.2	Problem B. Counting Rooms	1	1.8	Problem H. Sticks	2	2.2	Chinese Rings	3	4.4	Hunting and Distributing Meals	8	6.3	Nearest Smaller Values	12					
1.3	Problem C. Sum of Three Values	1				2.3	Citizen attention offices . . .	3	4.5	Problem H. Yet Another Alice and Bob	8	6.4	Nice Boat!	12					
1.4	Problem D. LR insertion . . .	1				2.4	Combinations	3	4.6	Queuing Problem	9	6.5	Paint the sticks	13					
1.5	Problem E. Second Day in Aincrad	1				2.5	Creating Strings	4	4.7	Tasks and Deadlines	9	6.6	Sorting Queries	13					
1.6	Problem F. Circular Nearest Smaller Values	2	3	w2	4	2.6	Gray code	4				6.7	STring	13					
						2.7	Tower of Hanoi	4	5.1	2022 Competitive Programming Training (I) Midterm Exam ProblemA Task and Penalty	10	7.1	Counting Rooms	13					
						3.1	Chessboard and Queens . . .	4	5.2	Array Division	10	7.2	Grid Maze	14					
						3.2	Grid Path	5	5.3	C. Sum of Three Values . . .	10	7.3	Monsters	14					
						3.3	Hive	5	5.4	Cart Racing	10	7.4	Problem A - Rush Hour Puzzle	14					
						3.4	M Queen N Rocks	5	5.5	Final Day	11	7.5	Rubik 2 ³	15					
						3.5	Square	6	5.6	Points to Cost	11	7.6	Swap Game	15					
						3.6	Sudoku	6	5.7	Subarray Sums	11	7.7	Water Jug Puzzle	16					
						3.7	Sum It Up	7	5.8	Tian Ji horse racing	11								
			4	w3	7				6	w5	12	8	w7	16					
						4.1	Array Arrangement	7	6.1	Circular Sliding Window Maximum	12	8.1	Binomial Coefficients	16					
						4.2	Banana milk lover	7				8.2	Common Divisors	16					
												8.3	Counting Coprime Pairs . . .	16					
												8.4	Exponentiation II	17					
												8.5	Problem D. Candies	17					
												8.6	Sum of Divisors	17					

ACM ICPC Judge Test - Angry Crow Takes Flight!

C++ Resource Test

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 namespace system_test {
5
6 const size_t KB = 1024;
7 const size_t MB = KB * 1024;
8 const size_t GB = MB * 1024;
```

```
9 size_t block_size, bound;
10 void stack_size_dfs(size_t depth = 1) {
11     if (depth >= bound)
12         return;
13     int8_t ptr[block_size]; // 若無法編譯將
14                             // block_size 改成常數
15     memset(ptr, 'a', block_size);
16     cout << depth << endl;
17     stack_size_dfs(depth + 1);
18 }
19
20 void stack_size_and_runtime_error(size_t
21     block_size, size_t bound = 1024) {
22     system_test::block_size = block_size;
23     system_test::bound = bound;
24     stack_size_dfs();
25 }
26
27 double speed(int iter_num) {
28     const int block_size = 1024;
29     volatile int A[block_size];
30     auto begin = chrono::high_resolution_clock
31         ::now();
32     while (iter_num--)
33         for (int j = 0; j < block_size; ++j)
34             A[j] += j;
35     auto end = chrono::high_resolution_clock::
36         now();
```

```
37 chrono::duration<double> diff = end -
38     begin;
39     return diff.count();
40 }
41
42 void runtime_error_1() {
43     // Segmentation fault
44     int *ptr = nullptr;
45     *(ptr + 7122) = 7122;
46 }
47
48 void runtime_error_2() {
49     // Segmentation fault
50     int *ptr = (int *)memset;
51     *ptr = 7122;
52 }
53
54 void runtime_error_3() {
55     // munmap_chunk(): invalid pointer
56     int *ptr = (int *)memset;
57     delete ptr;
58 }
59
60 void runtime_error_4() {
61     // free(): invalid pointer
62     int *ptr = new int[7122];
63     ptr += 1;
64     delete[] ptr;
65 }
```

```
62
63 void runtime_error_5() {
64     // maybe illegal instruction
65     int a = 7122, b = 0;
66     cout << (a / b) << endl;
67 }
68
69 void runtime_error_6() {
70     // floating point exception
71     volatile int a = 7122, b = 0;
72     cout << (a / b) << endl;
73 }
74
75 void runtime_error_7() {
76     // call to abort.
77     assert(false);
78 }
79
80 } // namespace system_test
81
82 #include <sys/resource.h>
83 void print_stack_limit() { // only work in
84     Linux
85     struct rlimit l;
86     getrlimit(RLIMIT_STACK, &l);
87     cout << "stack_size = " << l.rlim_cur << "
88         byte" << endl;
89 }
```