

ABC 157 解説

writer: kort0n

2020 年 3 月 1 日

For International Readers: English editorial will be published in a few days.

A: Duplex Printng

答えは $\text{floor}(\frac{N}{2})$ です ($\text{floor}(x)$ は x より小さくない最大の整数を表す)。
例えば、C++ では N を `int` 型の変数とすると、これは $(N + 1) / 2$ として求めることができます。
以下は C++ での実装例です。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int main() {
4     int N;
5     cin >> N;
6     int ans = (N + 1) / 2;
7     cout << ans << endl;
8     return 0;
9 }
```

B: Bingo

問題文の指示通りに各マスにマークが付いているか否かを管理し、ビンゴが成立している列が存在するかを確かめます。

ビンゴカードの管理には二次元配列が便利です。

以下は C++ での実装例です。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int N, A[3][3], b[10];
4 bool appear[3][3];
5 int main() {
6     for(int i = 0; i < 3; i++) {
7         for(int j = 0; j < 3; j++) cin >> A[i][j];
8     }
9     cin >> N;
10    for(int i = 0; i < N; i++) cin >> b[i];
11    for(int i = 0; i < N; i++) {
12        for(int j = 0; j < 3; j++) {
13            appear[i][j] = false;
14            for(int k = 0; k < N; k++) {
15                if(A[i][j] == b[k]) appear[i][j] = true;
16            }
17        }
18    }
19    string ANS = "No";
20    for(int i = 0; i < 3; i++) {
21        if(appear[i][0] and appear[i][1] and appear[i][2]) ANS = "Yes";
22    }
23    for(int i = 0; i < 3; i++) {
24        if(appear[0][i] and appear[1][i] and appear[2][i]) ANS = "Yes";
25    }
26    if(appear[0][0] and appear[1][1] and appear[2][2]) ANS = "Yes";
27    if(appear[0][2] and appear[1][1] and appear[2][0]) ANS = "Yes";
28    cout << ANS << endl;
29    return 0;
30 }
```

C: Guess the Number

解法 1

0 以上 10^N 未満の整数を全て調べます。条件の成立チェックを行う上では、整数を一旦文字列に変換すると楽です。

時間計算量は $O(10^N M)$ です。

解法 2

M 個の条件について、矛盾が存在しないかを前から順に確かめつつ解の候補を絞り、最後にそれらの最小値を出力します。

ただし、 $N \geq 2$ の場合に限り左から 1 桁目は 0 が許されないことに注意してください。

時間計算量は $O(N + M)$ です。

D: Friend Suggestions

人を頂点、友達関係を辺とした無向グラフを考えると、人 i に対する答えは、

(頂点 i の連結成分のサイズ)

−(頂点 i と頂点 j が同じ連結成分に含まれて、人 i と人 j が友達関係もしくはブロック関係にあるような j の数)

−1

です (最後の 1 は自分自身)。

各頂点が属する連結成分を求め、各連結成分のサイズを求めておけば、各人ごとの上記の値を高速に計算することが出来ます。

これは、Union Find のようなデータ構造を用いるか、もしくはグラフ上で DFS を行うこと等により実現出来ます。

時間計算量は、例えば DFS の場合は $O(N + M + K)$ です。

E: Simple String Queries

各文字 (a, b, ..., z) 毎に文字列中に現れる場所の集合を管理することを考えます。

例えば、set(平衡二分探索木) を使えば、type 1 のクエリは erase 関数と insert 関数を用いて $O(\log N)$ で処理可能であり、type 2 のクエリは各文字ごとに lower_bound 関数で l_q 文字目以降初めて存在する場所を求め、その値と r_q を比較することにより、 $O(\log N)$ で処理可能です。以上より、全体の時間計算量 $O((\text{Alphabet Size}) Q \log N)$ で処理可能です。

他にも、Segment Tree や平方分割により各文字が各区間に表れる回数を管理したり、各区間に表れる文字の集合を管理したりすることによって、クエリを高速に処理することも可能です。

F: Yakiniku Optimization Problem

解法 1

「時刻 T の時点で K 枚以上の肉が焼けるような熱源の配置場所は存在するか？」という判定問題を考えると、この判定問題には単調性がありますから、答えを二分探索で求めることが出来ます。

この判定問題を数学的に定式化すると、以下の通りになります。

「 N 枚の円盤がある。 i 枚目の円盤の中心は (x_i, y_i) であり、半径は $\frac{T}{c_i}$ である。 K 枚以上の円盤に含まれるような点が存在するか判定せよ。」

この問題は、以下のように言い換えることが出来ます。

「 N 枚の円盤がある。 i 枚目の円盤の中心は (x_i, y_i) であり、半径は $\frac{T}{c_i}$ である。これらの N 枚の円盤から K 枚の円盤を選ぶ選び方であって、 K 枚の円盤の積集合が非空となるような選び方があるか判定せよ。」

ここで、 K 枚の円盤の積集合 X が非空であったとき、その集合は 1 枚の円盤と一致するか、或いは互いに包含関係に無い複数枚の円盤の積集合と一致します。

前者の場合はその円盤の中心の座標が X に含まれ、後者の場合はある 2 枚の円盤が存在し、それらの境界を成す円の交点 X に含まれます。

これより、 N 個の円の中心及び N 個の円のうち 2 個の円の交点全てについて、その点を含む円盤が K 枚以上存在するかを調べることにより、前述の判定問題を $O(N^3)$ で解くことが出来ます。

以上より、許容誤差を eps として、元の問題を $O\left(\frac{c_{\max}(|x_{\max}|+|y_{\max}|)}{eps}N^3\right)$ で解くことが出来ます。

解法 2

$K = 1$ のときは答えは明らかに 0 です。以下では $K \geq 2$ とします。 $K \geq 2$ の元、答えは明らかに 0 より大きくなります。

以下では点 $P(X, Y)$ と肉 i の”距離”を、 $c_i \sqrt{(X - x_i)^2 + (Y - y_i)^2}$ で定めます。

肉 i と肉 j からの距離が等しい点の集合は、 $c_i = c_j$ のときは直線であり、 $c_i \neq c_j$ のときは円です。

$K \geq 2$ のもと、最適な熱源位置は、異なる 2 枚の肉からの距離が等しく、かつその値が最小となる点か、或いは異なる 3 枚の肉からの距離が等しい点です。

これを背理法により示します。前述の条件を満たさない点 P が最適な熱源配置であると仮定します。

このときの答えを T と置きます。丁度時刻 T に焼ける肉を tight な肉と呼びます。仮定より、tight な肉は高々 2 枚です。

tight な肉が 0 枚であるとき、熱源を P に置くと、時刻 T より僅かに前でも K 枚以上の肉が焼けています。

tight な肉が 1 枚であるとき、点 P からその肉の方向へ僅かに進んだ点では、時刻 T より僅かに前でも K 枚以上の肉が焼けています。

tight な肉が 2 枚であるとき、それらを肉 i 、肉 j と置くと、肉 i を中心とする半径 $\frac{T}{c_i}$ の円盤及び肉 j を中心とする半径 $\frac{T}{c_j}$ の円盤に含まれ、更に肉 i からの距離と肉 j からの距離が等しい点の集合は、仮定より一点集合ではなく、有限の長さを持つ線分或いは曲線となります。この線の内部の点に熱源を配置すると、時刻 T より僅かに前でも K 枚の肉が焼けています。

以上より、最適な熱源配置の候補は、前述の条件を満たす点に限られます。

これより、前述の点を全列挙して各点に熱源を配置した際に各肉が焼けるまでの時間を求め、 K 枚目の肉が焼けるまでに掛かる時間を求めることにより、解を得ることが出来ます。

時間計算量は、 K 枚目の肉が焼けるまでに掛かる時間を求める際にソートを行えば $O(N^4 \log N)$ であり、選択アルゴリズムを用いれば $O(N^4)$ です。