ABC 157 解説

writer: kort0n

2020 年 3 月 1 日

For International Readers: English editorial starts on page 8.

A: Duplex Printng

答えは floor $\left(\frac{N}{2}\right)$ です (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出来ます。
以下は C++ での実装例です。

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

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。

```
#include <bits/stdc++.h>
using namespace std;
int N, A[3][3], b[10];
bool appear[3][3];
int main() {
    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 3; j++) cin >> A[i][j];
    }
    cin >> N;
    for(int i = 0; i < N; i++) cin >> b[i];
    for(int i = 0; i < N; i++) {
        for(int j = 0; j < N; j++) {
            appear[i][j] = false;
            for(int k = 0; k < N; k++) {
                if(A[i][j] == b[k]) appear[i][j] = true;
            }
        }
    }
    string ANS = "No";
    for(int i = 0; i < 3; i++) {
        if(appear[i][0] and appear[i][1] and appear[i][2]) ANS = "Yes";
    }
    for(int i = 0; i < 3; i++) {
        if(appear[0][i] and appear[1][i] and appear[2][i]) ANS = "Yes";
    }
    if(appear[0][0] and appear[1][1] and appear[2][2]) ANS = "Yes";
    if(appear[0][2] and appear[1][1] and appear[2][0]) ANS = "Yes";
    cout << ANS << endl;
    return 0;
}
```

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 $O\left(10^N M\right)$ 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 $O\left(N + M\right)$ 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 枚の円板がある。i 枚目の円板の中心は (x_i, y_i) であり、半径は $\frac{T}{c_i}$ である。K 枚以上の円板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 枚の円板がある。i 枚目の円板の中心は (x_i, y_i) であり、半径は $\frac{T}{c_i}$ である。これらの N 枚の円板から K 枚の円板を選ぶ選び方であって、K 枚の円板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 枚の円板の積集合 X が非空であったとき、その集合は 1 枚の円板と一致するか、或いは互いに包含関係に無い複数枚の円板の積集合と一致します。
前者の場合はその円板の中心の座標が X に含まれ、後者の場合はある 2 枚の円板が存在し、それらの境界を成す円の交点が 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、その点を含む円板が 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 $O\left(N^3\right)$ 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 $\frac{T}{c_i}$ の円板及び肉 j を中心とする半径 $\frac{T}{c_j}$ の円板に含まれ、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 枚以上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 $O\left(N^4 \log N\right)$ であり、選択アルゴリズムを用いれば $O\left(N^4\right)$ です。

A: Duplex Printng

The answer is floor $\left(\frac{N}{2}\right)$ (where floor (x) denotes the maximum integer that is not less than x)。

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出来ます。

For example in C++, when N is a variable of type int, this can be calculated by $(N+1)/2$.

The following is a sample code in C++.

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

B: Bingo

Follow the problem statement's instruction and manage if each square is marked or not, and check if there is a row that forms a bingo.
When managing the bingo card, two-dimensional array is useful.
The following is a sample code in C++.

```
#include <bits/stdc++.h>
using namespace std;
int N, A[3][3], b[10];
bool appear[3][3];
int main() {
    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 3; j++) cin >> A[i][j];
    }
    cin >> N;
    for(int i = 0; i < N; i++) cin >> b[i];
    for(int i = 0; i < N; i++) {
        for(int j = 0; j < N; j++) {
            appear[i][j] = false;
            for(int k = 0; k < N; k++) {
                if(A[i][j] == b[k]) appear[i][j] = true;
            }
        }
    }
    string ANS = "No";
    for(int i = 0; i < 3; i++) {
        if(appear[i][0] and appear[i][1] and appear[i][2]) ANS = "Yes";
    }
    for(int i = 0; i < 3; i++) {
        if(appear[0][i] and appear[1][i] and appear[2][i]) ANS = "Yes";
    }
    if(appear[0][0] and appear[1][1] and appear[2][2]) ANS = "Yes";
    if(appear[0][2] and appear[1][1] and appear[2][0]) ANS = "Yes";
    cout << ANS << endl;
    return 0;
}
```

C: Guess the Number

Solution 1

Check for all the integers more than or equal to 0, and less than 10^N. When checking the conditions, temporal conversion from integer to string will make the implementation easier. The total time complexity is $O\left(10^N M\right)$.

Solution 2

For each of M conditions, narrow down the candidates while checking for contradictions, and at last output the minimum of them.
Note that the 1-st digit should not be 0 **only if** $N \geq 2$.
The total time complexity is $O\left(N + M\right)$.

D: Friend Suggestions

Consider a graph, whose vertices correspond the users and whose edges correspond to the friendship. Then the answer for user i is

(The size of the connected component of vertex i)
$-$(The number of j, such that vertex i and vertex j belong to the same connected component, and user i and user j is in a friendship or in a blockship)
-1.

(The last 1 is the user him/herself.)
By finding the connected component to which each vertex belongs, and finding the size of each connected component, the value mentioned above for each user can be calculated fast.
This can be achieved by using data structures like Union Find, or performing a DFS on the graph. The time complexity is, $O(N + M + K)$ in case of DFS for example.

E: Simple String Queries

Consider managing the set of positions such that each alphabets (a, b, ..., z) appears in the string.

For example, by using a set (self-balancing binary search tree), a query of type 1 can be processed in a $O(\log N)$ time using erase function and insert function, and a query of type 2 can be processed in a $O(\log N)$ time by find the position of the first appearance after l_q-th letter, and then comparing it with r_q, using lower_bound function. Therefore, the overall queries can be processed in a total of $O((\text{Alphabet Size})\, Q \log N)$ time.

Otherwise, the query can be processed fast by managing the number of appearances of each alphabet in each segment, or the set of alphabets that appear in each segment, using Segment Tree or square-root splitting.

F: Yakiniku Optimization Problem

Solution 1

By considering a decision problem of "does there exist a place to put the heat source such that K or more pieces of meat is ready at time T?", then this problem has monotonicity, so the answer can be found in a binary search.

A mathematical formalization of this decision problem is as follows:

"You are given N disks. The center of i-th disk is (x_i, y_i) and its radius is $\frac{T}{c_i}$. Determine whether there exists a point such that is included in K or more disks."

This problem can be reworded as follows.

"You are given N disks. The center of i-th disk is (x_i, y_i) and its radius is $\frac{T}{c_i}$. Determine if there exists a set of K disks out of those N disks, such that the union set of those K disks is non-empty."

Here, if a union of K disks X is non-empty, then it corresponds to a disk, or a union of some disjoint multiple disks.

In the former case, the center coordinate of the disk is included in X, and in the latter case, there exist two disks such that the intersection of their boundary circles is included in X.

Hence, by checking if a point is included in K or more disks for all the N centers and all the intersections of pairs of circles, the aforementioned decision problem can be solved in a total of $O\left(N^3\right)$ time.

Therefore, denoting by *eps* the maximum permissible error, the original problem can be solved in a total of $O\left(\frac{c_{\max}(|x_{\max}|+|y_{\max}|)}{eps}N^3\right)$ time.

Solution 2

If $K = 1$, then obviously the answer is 0. Hereinafter assume that $K \geq 2$. Under $K \geq 2$, the answer is obviously more than 0.

Hereinafter we define a "distance" between point $P\left(X, Y\right)$ and meat i by $c_i\sqrt{\left(X - x_i\right)^2 + \left(Y - y_i\right)^2}$.

The set of points such that the distances from meat i and from meat j are equal is a line if $c_i = c_j$, and a circle if $c_i \neq c_j$.

Under $K \geq 2$, the optimal position of heat source is a point such that the distances from distinct two pieces of meat are the equal and its value is minimum, or a point such that the distances from distinct three pieces of meat are the equal.

We will show this by contradiction. Suppose that a point P such that does not satisfy the aforementioned conditions is the optimal heat source position. Let T be the answer in such

case. We will call a piece of meet that becomes ready at time T. By assumption, there exist at most two pieces of meat is tight.

If 0 pieces of meat are tight, by placing a heat source at P, K or more pieces of meat is ready even before time T.

If 1 piece of meat is tight, at the point a little nearer to the meat than point P, K or more pieces of meat is ready at the time a little before time T.

If 2 pieces of meat are tight, denoting those meat by i and j, a set of points, such that is included in a disk whose center is meat i and whose radius is $\frac{T}{c_i}$, is included in a disk whose center is meat j and whose radius is $\frac{T}{c_j}$, and the distances from meat i and from meat j is equal, is not a singleton, but a segment or a curve of finite length, by assumption. By placing a heat source at the interior point of this curve, K or more pieces of meat is ready at the time a little before time T.

Therefore, the candidates of optimal heat source positions are narrowed down into such points that satisfy the aforementioned conditions.

Hence, by iterating all the points mentioned above, finding the time needed for each meet to be ready when the heat source is placed at each point, and by finding the time needed until the K-th meat is ready, the solution can be found.

The time complexity is $O\left(N^4 \log N\right)$ if a sort is performed when finding the time needed until the K-th meat is ready, and is $O\left(N^4\right)$ if a selection algorithm is used.