ARC 076 解説

DEGwer

2017/06/24

For International Readers: English editorial starts on page 4.

A: Expired?

$A \geq B, A < B \leq A + X, B > A + X$ の 3 通りに場合分けをする問題です。C++ での実装例を以下に示します。

```
#include<stdio.h>
int main()
{
        int x, a, b;
        scanf("%d%d%d", &x, &a, &b);
        if (-a + b <= 0)printf("delicious\n");
        else if (-a + b <= x)printf("safe\n");
        else printf("dangerous\n");
}
```

B: Trained?

どのボタンが光るかを順にシミュレートしていき、もしボタン 2 が光ったなら、それまでにボタンを押した回数を出力すればよいです。

では、ボタン 2 が永遠に光らない場合はどうでしょうか。コンピュータは有限回の操作しかできないため、「いつまでたってもボタン 2 は光らない」ということを直接シミュレーションで確かめることはできません。

N 回ボタンを押してもボタン 2 が光らなかったとします。このとき、その間に $N+1$ 回いずれかのボタンが光るので、そのうちのある 2 つは同じボタンです。同じボタンが光っている状態から光っているボタンを押すことを繰り返しても、同じ順にボタンが光るだけなので、この場合ボタン 2 は永遠に光ることはありません。

よって、N 回までボタンを押してみて、もし途中でボタン 2 が光ればそれまでにボタンを押した回数を出力し、そうでなければ -1 を出力すれば、この問題を解くことができます。

以下に C++ による実装例を示します。

```
#include<stdio.h>
#include<vector>
#include<algorithm>
using namespace std;
int main()
{
        int n;
        scanf("%d", &n);
        vector<int>v;
        for (int i = 0; i < n; i++)
        {
                int z;
                scanf("%d", &z);
                z--;
                v.push_back(z);
        }
        int now = 0, c = 0;
        for (;;)
        {
                if (now == 1)
                {
                        printf("%d\n", c);
                        break;
                }
                if (c >= n)
                {
                        printf("-1\n");
                        break;
                }
                c++;
                now = v[now];
        }
}
```

C: Reconciled?

N と M の差 (の絶対値) が 2 以上のとき、条件を満たすように並べることはできません。

N と M の差が 1 のとき、$N+1=M$ とすれば、条件を満たすためには猿犬猿... 犬猿 の順に並べることになります。このような並べ方は、犬 N 匹を並べる場合の数と、猿 M 匹を並べる場合の数の積なので、$N!M!$ 通りです。$M+1=N$ の場合も同様です。

$N=M$ のとき、猿犬猿... 猿犬と犬猿犬... 犬猿の 2 種類の並べ方があります。上記の場合と同じように考えれば、それぞれ $N!M!$ 通りの並べ方があることがわかるので、全体の場合の数は $2N!M!$ 通りです。

10^9+7 で割ったあまりを求めるのを忘れないようにしてください。

D: Built?

最小全域木を求める問題です。素直にグラフを構築し、最小全域木を求めた場合、辺数が $O(N^2)$ 本となり、間に合いません。

さて、座標 (a,b) の点と座標 (c,d) の点を結ぶコストは $min(|a-c|,|b-d|)$ ですが、最小全域木を求める上では、(多重辺はコストが最小のもののみ考えればいいので) コスト $|a-c|$ の辺とコスト $|b-d|$ の辺がそれぞれ存在すると考えても構いません。

さて、x 座標と点の番号の組がそれぞれ $(s,i)<(t,j)<(u,k)$ (ただし、不等号は辞書式順序で入れる) である 3 点を考えたときに、もし 1 番目の点と 3 番目の点をコスト $u-s$ の辺で結ぶ辺を全域木に使うならば、その代わりに $1,2$ 番目の点を結ぶコスト $u-t$ の辺と $2,3$ 番目の点を結ぶコスト $t-s$ の辺を使うことにしても、全域木のコストは大きくなりません。よって、$1,3$ 番目の点を直接結ぶコスト $u-t$ の辺は、考えなくていいことが分かります。(点の番号を一緒にして並べているのは、同じ x 座標の点が出現した場合に順序をつけるための便宜的な措置です)

以上より、点たちを x 座標でソートした列において隣接する 2 点と、y 座標でソートした列において隣接する 2 点の間にそれぞれ辺を張ったグラフの最小全域木を求めればよいです。辺の本数は $O(N)$ 本に減ったため、全体で $O(NlogN)$ 時間でこの問題を解くことができます。

E: Connected?

まず、少なくとも片方が長方形の周上以外に書かれている整数については、考慮しなくてよいことを示します。

両方が長方形の周上に書かれている整数たちをうまく結べたとします。片方が周上に書かれている整数についてはその周上の位置に、そうでない整数については適当な長方形内部に、その整数を 2 つ書き、すでに書かれている曲線たちを「押し出す」ように連続的に変形させることで、新しく書いた整数たちを所望の位置に移動することができるため、示されました。

さて、両方が長方形の周上に書かれている整数たちだけを考えたとき、条件を満たすように整数たちを結べるかどうかは、どのように判定すればいいでしょうか？もし整数 i,j が、i,j,i,j の順に周上に現れたなら、このようなことは不可能です。逆に、そのような i,j が存在しなければ、可能であることもわかります。

これは、周上の適当な位置から順に、時計回りに周上の点たちを stack などを用いて管理していけば、$O(N)$ 時間 (ただし、最初のソートに $O(NlogN)$ 時間) で判定することができます。

F: Exhausted?

高橋君の集合 X に対し、その集合内のいずれかの高橋君が座ることのできる椅子の集合を $\Gamma(X)$ と書くことにします。

Hall の定理より、追加すべき椅子の個数の最小値は、全ての高橋君の部分集合 X に対する、$|X| - \Gamma(X)$ の最大値と一致します。

さて、$\Gamma(X)$ としてありうるものは、高橋君のこだわりの条件より、「ある $s, t(s+1<t)$ に対し、座標 s 以下または座標 t 以上にある椅子すべて」もしくは「椅子すべて」という形をしています。$\Gamma(X)$ を固定したとき、先の X としてはサイズの一番大きいもののみを考えればよいです。

$\Gamma(X)$ が全体でない場合、全ての $s+1<t$ に対する、座標 s 以下または座標 t 以上にある椅子の個数から、$L_i \leq s$ かつ $t \leq R_i$ なる i の個数を引いたものの最大値を求めればいいです。$\Gamma(X)$ が全体の場合、X として高橋君全員からなる集合を取れます。以上より、求める答えは、$s+1<t$ に対して求めた最大値と、$N-M$ の大きい方となります。

さて、$s+1<t$ に対しては、どのように最大値を求めればいいでしょうか？これは、高橋君たちを L_i の昇順にソートして順に見ていきながら、区間加算と全体 min のクエリの処理できる segment tree を用いることで、$O(NlogN)$ で求めることができます。

ARC 076 Editorial

DEGwer

2017/06/24

A: Expired?

There are three cases: $A \geq B, A < B \leq A + X, B > A + X$.

```
#include<stdio.h>
int main()
{
        int x, a, b;
        scanf("%d%d%d", &x, &a, &b);
        if (-a + b <= 0)printf("delicious\n");
        else if (-a + b <= x)printf("safe\n");
        else printf("dangerous\n");
}
```

B: Trained?

We simulate the process. If 2 is lighten up during the simulation, we get the answer. Otherwise, if 2 is not lighten up in the first N steps, we can prove that this button will never be lighten up, thus the answer is -1.

```
#include<stdio.h>
#include<vector>
#include<algorithm>
using namespace std;
int main()
{
        int n;
        scanf("%d", &n);
        vector<int>v;
        for (int i = 0; i < n; i++)
        {
                int z;
                scanf("%d", &z);
                z--;
```

```
            v.push_back(z);
        }
        int now = 0, c = 0;
        for (;;)
        {
            if (now == 1)
            {
                printf("%d\n", c);
                break;
            }
            if (c >= n)
            {
                printf("-1\n");
                break;
            }
            c++;
            now = v[now];
        }
}
```

C: Reconciled?

If the absolute difference of N and M is at least 2, it is impossible to satisfy the conditions.

If $N + 1 = M$, the animals must be arranged "MDMDM...DM" (here M stands for monkeys and D stands for dogs). There are $N!M!$ ways to do this. The case where $M + 1 = N$ is similar.

If $N = M$, both "MDMD...MD" and "DMDM...DM" are possible, so the answer is $2N!M!$. Don't forget to compute the answer modulo $10^9 + 7$.

D: Built?

We are asked to compute the minimum spanning tree.

Between two points (a, b) and (c, d), instead of adding an edge of cost $min(|a - c|, |b - d|)$, we add two edges: one edge with cost $|a - c|$ and one edge with cost $|b - d|$.

Suppose that there are three points p, q, r, and their x-coordinates satisfy $x_p < x_q < x_r$. Then, the edge between p and r with cost $x_r - x_p$ never appear in the MST (it is better to use an edge between p and q with cost $x_q - x_p$ and an edge between q and r with cost $x_r - x_q$).

Thus, we only need to consider the following $2(N - 1)$ edges:

- We sort the point by their x-coordinates, and for each adjacent pair of points add an edge between them (the cost is the difference of their x-coordinates).
- We sort the point by their y-coordinates, and for each adjacent pair of points add an edge between them (the cost is the difference of their y-coordinates).

We compute the MST of these edges. This can be done in $O(NlogN)$.

E: Connected?

We call an integer "important", if both occurrences of this integer is on the boundary of the rectangle.

Suppose that there are two important integers i and j, and they appear in the boundary in the order i, j, i, j. In this case, it is clear that we can't connect both integers without crossing, thus the answer is "No".

Otherwise, we can connect all integers in the following order:

- First, we connect non-important integers one by one.
- Then, we connect important integers one by one.

This way, we never get stuck.

How can we check if there exists such a pair (i, j) of important integers? Start from an arbitrary point on the boundary and check all important points on the boundary one by one in clockwise order (ignore non-important integers on the boundary). Initially we have an empty stack, and whenever we find an important integer x on the boundary, do the following:

- If the top element of the stack is also x, pop it.
- Otherwise, push x into the stack.

If the stack becomes empty after we check all points, the answer is "Yes". Otherwise the answer is "No". This algorithm works in $O(NlogN)$.

F: Exausted?

Let X be a set of people. Define $\Gamma(X)$ as the set of all chairs that can be used by at least one person in X. By Hall's Theorem, the answer of this task is the maximum of $|X| - \Gamma(X)$ (here X moves among all subsets of Takahashis).

$\Gamma(X)$ is one of the following forms:

- There exists some $s, t (s + 1 < t)$ and $\Gamma(X)$ is the set of all chairs whose coordinates are at most s or at least t.
- $\Gamma(X)$ is the set of all chairs.

For a fixed $\Gamma(X)$, we only need to consider X with the maximum cardinality.

If $\Gamma(X)$ is the set of all chairs, we can assume that X is the entire set. In this case, $|X| - \Gamma(X) = N - M$.

If $\Gamma(X)$ is the set of all chairs whose coordinates are at most s or at least t, the cardinality of X is the number of i such that $L_i \leq s$ and $t \leq R_i$. How can we compute the maximum of this value? For each i, we plot a point (L_i, R_i) on a plane, and we do a sweepline algorithm with a segment tree. The segment tree supports two queries: "add a given constant to a given range" and "find the maximum value among a given range". This works in $O(NlogN)$.