Tenka1 Programmer Contest 2017 解説

DEGwer

2017/09/30

For International Readers: English editorial starts on page 4.

A: Accepted...?

与えられた文字列に、1 がいくつあるかを数えればよいです。

```
#include<stdio.h>
int main()
{
        char s[10];
        scanf("%s", s);
        int r = 0;
        for (int i = 0; i < 6; i++)r += s[i] - '0';
        printf("%d\n", r);
}
```

B: Different Distribution

与えられた順位のうち最大のものを R とし、その人の得点を S とします。R 位より低い順位の人の得点は S より小さな相異なる非負整数値を取るので、R 位より低い順位の人は最大でも S 人しか存在しません。また、与えられる入力に対しては条件を満たす得点分布が存在することが保障されているので、求める答えは $R+S$ となります。

```
#include<stdio.h>
int main()
{
        int n;
        scanf("%d", &n);
        int maxi = 0, ans;
        for (int i = 0; i < n; i++)
        {
                int za, zb;
                scanf("%d%d", &za, &zb);
                if (maxi < za)
                {
                        maxi = za;
                        ans = za + zb;
                }
        }
        printf("%d\n", ans);
}
```

C: 4/N

N が与えられるので、$4/N = 1/h + 1/n + 1/w$ を満たす正整数 h, n, w を求める問題です。
上記式を満たす h, n, w が複数ある場合は、どれを出力しても良いです。
制約で $h, n, w \leq 3500$ となる解が存在することが保証されています。
h と n を総当りし、w は下記のように式変形して計算すると一意に定まります。
$w = (N * h * n)/(4 * h * n - N * n - N * h)$
w が正整数になれば、その h, n, w が答えです。
h, n, w を総当りすると、3500^3 回程度のループになり、制限時間に間に合いませんが、
h, n の総当りは、3500^2 回程度のループになるので、制限時間内に回答することができます。

D: IntegerotS

以下、X で立っていないビットが立っていないような整数を、X に含まれる整数と呼ぶことにします。

K 以下の非負整数 X の 2 進表記は、K 自身か、あるいはある非負整数 r が存在して、下から r 桁以外は K と等しく、K の下から r 桁目が 1 で、X の下から r 桁目が 0 となります。すなわ

ち、与えられた整数の集合の部分集合であって、bitwise or が K 以下となるものの bitwise or の値は、K に含まれるか、またはある K の下から r 桁目が 1 となるような非負整数 r に対し、「K の下から r 桁以外を保持し、r 桁目を 0 にし、それ以外を 1 で埋めたもの K_r」に含まれます。

与えられた整数の部分集合であって、bitwise or が 特定の整数 X に含まれるようなものに対する価値の総和の最大値は、与えられた整数であって X に含まれるようなものの価値の総和に等しく、これは $O(N)$ 時間で求めることができます。

答えを求めるためには上述の X の候補として $K, K_0, \ldots, K_r, \ldots$ を全て試せばよく、r の候補は $O(log(max(A_i)))$ 個なので、全体で $O(Nlog(max(A_i)))$ 時間でこの問題を解くことができます。

E: CARtesian Coodinate

まず、左向きに x 座標を、上向きに y 座標を取ります。コスト関数はマンハッタン距離の和なので、求める座標は x 座標と y 座標について独立に考えることができます。

以下、答えの x 座標を求めることを考えます。y 座標に関しても同様です。

$\frac{N(N-1)}{2}$ 個の交点の x 座標を順に並べたとき、全体で $\lceil \frac{N(N-1)}{4} \rceil$ 番目にくるものが求める x 座標です。これは、左右に微小距離進んだときに、各点からの距離の合計がどう変化するかを考えることで証明することができます。

さて、この値はどう求めればよいでしょうか？ 交点をすべて列挙していては、$O(N^2)$ 時間かかって間に合わないため、別のアプローチを考えることにします。

二分探索をすることを考えると、「x 座標 s 以下に交点がいくつあるか」が高速に求められれば良いことが分かります。あらかじめ与えられた直線たちを傾き順にソートして、順に $L_1, \ldots L_N$ とおけば、x 座標が十分小さい領域では、x 座標一定の直線と L_i との交点 P_i たちの y 座標は、y 座標の小さい順に $P_1, \ldots P_N$ と並びます。

以下、x 座標一定の直線と L_i との交点である P_i を、x 座標一定の直線が右へと動くのに対応して動く点であるとします。

座標平面を右方向に走査していき、ある 2 直線の交点が現れたとします。この交点を過ぎると、P_i たちを y 座標の小さい順に並べた列において、隣り合う 2 要素が入れ替わることになります。これは、バブルソートにおける交換操作にほかなりません。すなわち、P_i たちを y 座標の小さい順に並べた列の添え字の反転数が、今見ている走査線より左側にある交点の個数と等しくなります。

反転数は BIT などを用いて $O(NlogN)$ 時間で求めることができるので、二分探索によってこの問題は $O(NlogN(log(maxC_i) + log(\text{許容誤差}^{-1}))$ で解くことができます。

F: ModularPowerEquation!!

クエリひとつに答えることを考え、以降 A_i を A と、M_i を M と、K_i を K と書くことにします。

$A = 0, 1$ のときは簡単なので、$A \geq 2$ とします。

この問題は、整数 K に対して A^K を返す操作の $modM$ での不動点を求める問題です。天下り的ですが、$A^{A^{A^{\cdots}}}$ (ただし、A の個数は十分大きいものとする) が、この方程式の解となることが証明できます。(すなわち、この操作を十分な回数繰り返すと不動点に収束します。)

これを証明し、さらに再帰的に指数の肩に乗せる A の必要個数を上から抑えます。$\phi(N)$ で N 以下で N と互いに素な正の整数の個数を表すことにします。

A と互いに素な正の整数 N に対し、Euler の定理より、$A^{\phi(N)} \equiv 1(modN)$ です。

A と N が互いに素でない場合も、A, N の素因数分解の $gcd(A, N)$ に含まれない素因数のみを取り出してきて掛け合わせてできる数をそれぞれ A', N' とすれば、$A'^{\phi(N')} \equiv 1(modN')$ となります。正の整数 S を $S\phi(N') \geq log_2 N$ となるようにとれば、$(\frac{A}{A'})^{S\phi(N')} \equiv 0(mod\frac{N}{N'})$ となるので、中国剰余定理より、$log_2 N$ 以上の T に対し、$A^T \equiv A^{T+\phi(N')}(modN)$ となることが分かります。$\phi(N')$ は $\phi(N)$ の約数なので、結局 $A^T \equiv A^{T+\phi(N)}(modN)$ となります。

以上より、もし K が十分大きければ、$A^K modM$ は K の $mod\phi(M)$ での値のみに依存することが分かりました。

$mod1$ では任意の K が $A^K \equiv K$ を満たすので、$M = 1$ の場合は簡単に解を求めることができます。$mod\phi(M)$ のときの解がひとつ求まっているとし、その解を K とします。このとき、$A^K \equiv K(mod\phi(M))$ なので、K が十分大きくとられていれば $A^{A^K} \equiv A^K(modM)$ となり、A^K が $modM$ での解となります。すなわち、$modM$ の場合を解くために $mod\phi(M)$ の場合を解き……と再帰的に繰り返していけばよいことが分かります。$mod1$ での解 K を十分大きくとっておけば、K が小さくなることはないので、$A^{A^{A^{A^{\cdots}}}}$ がもとの方程式の解となることが示されました。また、ϕ 関数は $O(logN)$ 回かけることで 1 となる (奇数にかけると偶数となり、偶数にかけると半分以下になるため) ので、指数の肩の A の個数も上から抑えることができました。

さて、この値は非常に大きいので、小さい解を求める必要があります。K が十分大きいとき、$A^K modM$ の値は K の $mod\phi(M)$ での値のみに依存するので、上側から再帰的に $A^{A^{A^{A^{\cdots}}}}$ を求めていく際に、その都度適切な法で mod を取っておけばよいことが分かります。最終的には $A^K \equiv K$ を満たす K の $modM$ での値と $mod\phi(M)$ での値が求まるので、拡張 Euclid の互除法などを用いて条件を満たす解を構成することができます。

mod をとることによって $mod\phi(M)$ での値の小さな解 K が得られた場合、A の肩に乗せるときに、$gcd(A, M)$ に含まれるある素因数の個数が M に含まれるその素因数の個数より小さくなってしまい、正しく $modM$ での A^K の値を求めることができない場合や、最終的に構成した解が $mod(lcm(M, \phi(M)))$ で小さすぎる場合に注意してください。どちらも、K に十分大きな適切な値を足すことで対処できます。

時間計算量は、$\phi(M)$ を求める部分がボトルネックで、クエリあたり $O(\sqrt{M})$ となります。2 段階再帰が進むことで M の値は半分以下になるので、再帰の各段階で毎回 $\phi(M)$ を $O(\sqrt{M})$ 時間かけて求めても、時間計算量は変わらず $O(\sqrt{M})$ です。

Tenka1 Programmer Contest 2017 Editorial

DEGwer

2017/09/30

A: Accepted...?

```
#include<stdio.h>
int main()
{
        char s[10];
        scanf("%s", s);
        int r = 0;
        for (int i = 0; i < 6; i++)r += s[i] - '0';
        printf("%d\n", r);
}
```

B: Different Distribution

```
#include<stdio.h>
int main()
{
        int n;
        scanf("%d", &n);
        int maxi = 0, ans;
        for (int i = 0; i < n; i++)
        {
                int za, zb;
                scanf("%d%d", &za, &zb);
                if (maxi < za)
                {
                        maxi = za;
                        ans = za + zb;
                }
        }
        printf("%d\n", ans);
}
```

C: 4/N

You are given an integer N and you want to find positive integers h, n, w such that $4/N = 1/h + 1/n + 1/w$.

If there are multiple such h, n, w, you can output any.

The constraints guarantee that there exists a solution such that $h, n, w \leq 3500$.

Brute force h and n, and w can be uniquely determined as follows:

$w = (N * h * n)/(4 * h * n - N * n - N * h)$

If w becomes a positive integer, this (h, n, w) is a solution.

If we brute force h, n, w, the number of iterations becomes 3500^3 and we will get TLE.

However, if we brute force only h, n, the number of iterations becomes 3500^2 and the solution works in time.

D: IntegerotS

We say an integer A contains an integer B, if for each k such that A doesn't contain the k-th bit, B also doesn't contain the k-th bit.

Suppose that X is an integer such that $X < K$. Then, there exists an integer r such that:

- The binary representations of X and K are the same except for the last (least significant) r digits.
- The r-th bit of X is 0 and the r-th bit of K is 1.

Therefore, all integers that are less than or equal to K are contained in one of $K, K_0, \ldots, K_r, \ldots$. Here, K_r is defined only when the r-th bit of K is 1, and K_r is obtained by changing the r-th bit of K to 0 and lower bits of K to 1.

For a fixed integer X, it is easy to compute the maximum sum of utilities when the XOR must be contained in X in $O(N)$.

Since we only need to try $K, K_0, \ldots, K_r, \ldots$ as candidates of X (there are $O(log(max(A_i)))$ candidates), this solution works in $O(Nlog(max(A_i)))$ time.

E: CARtesian Coordinate

Since the objective function is the sum of Manhattan Distances, we can handle x-coordinates and y-coordinates independently. In this editorial, we compute the x-coordinate of the exhibition. (We can compute the y-coordinate similarly.)

The answer is the median of all x-coordinates of $\frac{N(N-1)}{2}$ intersections. How can we compute this?

By binary search, it is sufficient to answer queries of the form "How many intersection are there in the region $x \leq s$"? Let's sort the given lines by their slopes, and call them $L_1, \ldots L_N$ in order.

Consider a vertical line $x = c$, and let P_i be the intersection between the vertical line and the line L_i. We move the vertical line from left to right, and the points P_i also move correspondingly.

When c is small, the points P_i are arranged in the order $P_1, \ldots P_N$ from bottom to top. While the vertical line moves, the order of two points in $P_1, \ldots P_N$ changes when it hits an intersection. Therefore, the number of intersection in the region $x \leq s$ is equal to the inversion number of points $P_1, \ldots P_N$ when $x = s$.

Since the inversion number can be computed in $O(NlogN)$ using binary indexed tree, this solution works in $O(NlogN(log(maxC_i) + log(precision^{-1}))$ with binary search.

F: ModularPowerEquation!!

By Euler's theorem, when k is sufficiently large, the value of $A^k \% M$ only depends on $k \% \phi(M)$.

Suppose that we know an integer y that satisfies the following:

- $A^y \equiv y \pmod{gcd(M, \phi(M))}$
- $y \geq 100$

Then, we can find an integer x that satisfies the following using extended gcd:

- $x \equiv A^y \pmod{M}$
- $x \equiv y \pmod{\phi(M)}$
- $x \geq 100$

It is easy too see that this x satisfies $A^x \equiv x \pmod{M}$.

Therefore, in order to answer the query (A, M), it is sufficient to solve the query $(A, gcd(M, \phi(M)))$. We can repeat this process recursively and the recursion terminates in $O(logM)$ steps (because in two steps M is always halved). Each step takes $O(\sqrt{M})$ time (for computing $\phi(M)$), and in total this solution also works in $O(\sqrt{M})$ time.