ABC072 / ARC082 解説

sigma425

For International Readers: English editorial starts on page 3.

A: Sandglass2

$X > t$ なら $X - t$[g] 残っていて、$X \leq t$ なら 0[g] になっています。これは $max(X - t, 0)$ と書くことも出来ます。コードにすると次のようになります。

```
int main(){
        int X,t;
        cin>>X>>t;
        if(X>t) cout<<X-t<<endl;
        else cout<<0<<endl;
        return 0;
}
```

B: OddString

for 文をまわすなどして答えの文字列を求めると良いです。

```
int main(){
        string s;
        cin>>s;
        int N = s.size();
        string ans;
        for(int i=0;i<N;i+=2) ans += s[i];
        cout<<ans<<endl;
}
```

C: Together

X を先に選んでから数に対する操作をすることにします。X を決めると、この個数を最大にするには、

- すべての $X - 1$ に $+1$ する
- すべての X をそのままにする
- すべての $X + 1$ に -1 する

をすると他は関係ないので、($X - 1$ の個数)+(X の個数)+($X + 1$ の個数) となります。従って先に各数の個数を数えておき、この X を考えうる範囲で全て試すことで答えが求まります。

D: Derangement

$p_i \neq i$ となっている部分に o を, そうでない部分に x をつけることにします。N 箇所全て o になれば ok です。ox とならんでいる部分は、swap することで oo となります (xo も同様)。($\because$ もともと $p_i = x(x \neq i)$,$p_{i+1} = i+1$ だったとすると、$x \neq i+1$ でもあるから、$p_i = i+1$ も $p_{i+1} = x$ も o になる)

xx とならんでいる部分も、swap することで oo となります。($\because$ もともと $p_i = i(x \neq i)$,$p_{i+1} = i+1$ だったとすると、$p_i = i+1$ も $p_{i+1} = i$ も o になる)

従って、先頭から順番に見ていき、箇所 i に x があったら i と $i+1$($i = N$ なら $i-1$ と i) を swap する、とするのが最善になります。(i にある x を消すのに $i-1, i$ の swap と $i, i+1$ の swap どちらかは必要で、前から見ると $i-1$ はもう o なので $i+1$ を巻き込んだほうが得)

E: ConvexScore

$n - |S|$ というのは,「S の凸包 (境界含む) に含まれる与えられた点のうち S の点を除いた集合」(T_S とおく) の要素数と等しいです。なので、$2^{n-|S|}$ は T_S の部分集合の個数と等しくなります。

ここで、凸包の面積が正の部分集合 X を任意に取ってきます。そして X の凸包の頂点集合を S_X とします。すると、S を決めた時、$\{X | S = S_X\}$ と $\{S \cup t | t \in T_S\}$ は集合として等しいです。

このことから、凸包の面積が正の部分集合 X はちょうど一回 $S = S_X$ の時に一回分カウントされる、とみなすことが出来ます。

従って、答えは凸包の面積が正の部分集合の個数になります。これは 2^N- (共線な (一直線上に載せることの出来る) 頂点集合の数) となるので共線な頂点集合の数を計算すれば良いです。0 点集合は 1 個,1 点集合は N 個で固定なので、2 点以上で共線なものの個数を数えます。

これは、2 点以上乗りうる直線（高々 ${}_NC_2$ 本) をすべて試し、この上に k 点乗っていれば $2^k - k - 1$ を足すことで求まります。計算量は $O(N^3)$ です。

直線を正規化し set で管理するなどによって $O(N^2 logN)$ で解くことも出来ます。

F: Sandglass

t 秒後にパーツ A に入っている砂の量を、はじめパーツ A に入っている砂の量 x の関数とみて $f_t(x)$ とおきます。

すると、$f_t(x)$ は常に次のような形の関数になります。

$$f_t(x) = \begin{cases} a + c & (0 \leq x \leq a) \\ x + c & (a < x < b) \\ b + c & (b \leq x \leq X) \end{cases}$$

(ただし $a = b$ となり定数関数に潰れている場合もあります) これは、d を定数として、$f(x) + t$,$max(f(x), 0)$,$min(f(x), X)$ が上述の形で閉じていることからわかります。(時間の経過やパーツに入るのが 0 以上 X 以下であることから起こる $f_t(x)$ の変化は全て上の 3 つのどれかの合成として書けるので、この 3 つだけ考えれば良い)

よって、上の関数に含まれる a, b, c を保持して更新しながらその都度クエリに答えていけば良いです。

ABC072 / ARC082 Editorial

sigma425

A: Sandglass2

```
int main(){
	int X,t;
	cin>>X>>t;
	if(X>t) cout<<X-t<<endl;
	else cout<<0<<endl;
	return 0;
}
```

B: OddString

```
int main(){
	string s;
	cin>>s;
	int N = s.size();
	string ans;
	for(int i=0;i<N;i+=2) ans += s[i];
	cout<<ans<<endl;
}
```

C: Together

Let's choose X first. For a fixed X, the optimal choices are:

- Add 1 to all $X-1$.
- Do nothing for all X.
- Subtract 1 from all $X+1$.

The number of X after these operations is (the number of $X-1$ in the initial sequence) + (the number of X in the initial sequence) + (the number of $X+1$ in the initial sequence). We can compute the optimal value by trying all possible values for X.

D: Derangement

Define a sequence of 'o' and 'x' of length N as follows: if $p_i \neq i$, the i-th symbol is 'o', otherwise the i-th symbol is 'x'. Our objective is to change this sequence to 'ooo...ooo'.

- If there is a part "ox" (or "xo") in the sequence, we can change it to "oo" by swapping these two elements. ($\because$ If $p_i = x(x \neq i)$ and $p_{i+1} = i + 1$ in the initial sequence, after the swap, both $p_i = i + 1$ and $p_{i+1} = x$ will be 'o'.)
- If there is a part "xx" in the sequence, we can change it to "oo" by swapping these two elements. ($\because$ If $p_i = i(x \neq i)$ and $p_{i+1} = i + 1$ in the initial sequence, after the swap, both $p_i = i + 1$ and $p_{i+1} = i$ will be 'o'.)

Thus, we should check the sequence from left to right, and if we find an 'x' at the i-th position, we should swap i and $i + 1$ (unless $i = N$, in this case we should swap i and $i - 1$).

E: ConvexScore

For a set S that forms a convex polygon (from now on, we call it "convex set"), let T_S be the set of points in the convex hull of S, except for the vertices (points in S). Since $|T_S| = n - |S|$, $2^{n-|S|}$ corresponds to the number of subsets of T_S.

In the problem, for each convex set S, we are asked to count it $2^(n - S)$ times. Instead, for each pair of sets (S, U) such that S is a convex set and U is a subset of T_S, let's count $S \cup U$ once.

- When we know $S \cup U$, we can recover S: it must be the convex hull of $S \cup U$. Thus, this way each set is counted at most once.
- A set of points X is counted if and only if the convex hull of X has positive area: let S be the convex hull of X and U be $X \setminus S$.

Therefore, the answer is equal to the number of subsets of all N points whose convex hull has a positive area.

This can be computed by subtracting the number of colinear sets (sets whose points are on the same line) from 2^N. This can be done in $O(N^3)$ or $O(N^2 logN)$.

F: Sandglass

Let the amount of sand in bulb A at time t be a function $f_t(x)$, where x is the initial amount of sand in bulb A. We can prove that this function is always of the following form (for some constants a, b, c):

$$f_t(x) = \begin{cases} a + c & (0 \leq x \leq a) \\ x + c & (a < x < b) \\ b + c & (b \leq x \leq X) \end{cases}$$

When bulb A contains y grams of sand at some time, one second later, the amount of sand in bulb A is either $g_1(y) = max(y-1, 0)$ (in case A is above B) or $g_2(y) = max(y+1, X)$ (in case B is above A). Since the set of functions of the form above is closed under functions g_1 and g_2, f_t is always of this form.

Therefore, we can simulate the process while keeping the function f_t (i.e., parameters a, b, c) and answer queries.