ARC070 / ABC056 解説

sigma425

For International Readers: English editorial starts on page 5.

A: HonestOrDishonest

$a =' H', b =' H'$ の場合は $'H'$ を、$a =' H', b =' D'$ の場合は $'D'$ を、$a =' D', b =' H'$ の場合は $'D'$ を、$a =' D', b =' D'$ の場合は $'H'$ を出力すればよいです。コードにすると次のようになります。

```
int main(){
        char a,b;
        cin >> a >> b;
        if(a=='H' && b=='H') cout << 'H';
        if(a=='H' && b=='D') cout << 'D';
        if(a=='D' && b=='H') cout << 'D';
        if(a=='D' && b=='D') cout << 'H';
        return 0;
}
```

B: NarrowRectanglesEasy

もし既に連結なら答えは 0 です。これは $|a-b| \leq W$ かどうかで判定できます。そうでない場合は、答えは $|a-b| - W$ になります。

C: Go Home

もし $1+2+...+t < X$ なら時刻 t には座標 X にはたどり着けません。逆に、$1+2+...+t \geq X$ なら時刻 t に座標 X にたどり着けることが示せます。実際、$\{1, 2, .., t\}$ の部分集合で和が X になるものが取れて、そうすると部分集合に選んだ長さのときだけ右にジャンプし、それ以外のときはジャンプしないという戦略によって座標 X にたどり着けます。よってあとはこのような最小の t を求めれば良いです。これはおおよそ $\frac{-1+\sqrt{8X+1}}{2}$ になるので、この関数によって計算すると $O(1)$ で求まります。そこまでしなくても、答えが $O(\sqrt{X})$ になることから、t を前から順番に試して $\frac{t(t+1)}{2} \geq X$ か毎回判定するだけでも, 時間計算量 $O(\sqrt{X})$ で答えを求めることが出来ます。

D: No Need

まず、「カード i が不必要」という条件は次のように言い換えられます: 「カード i を使わずに、総和が $K - a_i$ 以上 K 未満な集合が取れる」.

左 ⇒ 右 ・・・ 不必要ならその証拠となる i を含むよい集合があって、これから i を除いたものが言い換え後の条件を満たす証拠になっています。

右 ⇒ 左 ・・・ 上とは逆に、この条件を満たす集合に i を加えたものが元の条件を満たす証拠となっています。

よって、各カード i ごとに、そのカードを除いた後つぎのような DP をします。

「$dp[x][y] = x$ 枚目まで使ったときに総和を y にできるか?」

y としては K 未満しか考えなくて良いため、この DP の計算量は $O(NK)$ になります。これを N 枚のカードに対してやるので、全体の計算量は $O(N^2K)$ となり、部分点を取ることが出来ます。

満点解法を取るにはいくつか方法があります。ひとつは、カード i が不必要な時、$a_i \geq a_j$ を満たすカード j も不必要である (つまり、単調性が成り立つ) ことを利用する方法で、どのカードまで不必要かを二分探索することで、上述の dp の回数を $O(logN)$ 回に減らせるので、全体で $O(NKlogN)$ になります。

他の方法として、dp を, カード 1,2,..,N の順に使う方と N,$N-1$,...,1 の順に使う方の両方を途中経過を残して計算しておいて、その結果を利用することで各 i について $O(K)$ で判定する方法があります。実際、カード i について判定するには、1,2,...,$i-1$ を使って出来る値の集合と、N,$N-1$,..,$i+1$ を使ってできる値の集合がわかっていれば良いです。これがわかっているので、あとは累積和を使ったりすると $O(K)$ で判定できて、全体の計算量は $O(NK)$ になります。

また、bitset 等を使うことで DP を高速化すると計算量は変わらなくても少し遅い解法でも通るかもしれません。

E: NarrowRectangles

まず部分点を解くには、次のような dp をすればよいです。

$dp[i][x] =$ 上から i 個までの長方形を動かしていて、i 個めの長方形の横座標を x に動かした時点でのそこまでのコストの最小値

上から順番に動かし方を決めていくと、そこまでが連結なら、そこから下を連結に出来るかどうかは直前の長方形の一にしか関係がないためこのような dp が出来ます。

計算すると.

$$dp[i][x] = |x - l_i| + \min_{x-(R_{i-1}-L_{i-1}) \leq x' \leq x+(R_i-L_i)} dp[i-1][x'] \tag{1}$$

となる (min についている条件は $i-1$ と i が連結になる条件です) ので、これを愚直に計算すると、座標幅を X とおくと、$O(NX^2)$ になり、部分点を取ることが出来ます。

満点を取るには、$dp[i][]$ がどのような形になっているか考える必要があります。$dp[i-1]$ から $dp[i]$ を計算するには、

1. x 軸方向に平行移動する
2. ある一定長さの区間の min をとる
3. $|x - l_i|$ を足す

が出来る必要があります。この操作をしていくとどうなるか考えると、$dp[i]$ では、「十分に左では傾きが $-i$ で、そこから単調に傾き (常に整数) が大きくなっていき、十分に右では傾きが i になる」という形になっていることがわかります。従って、$dp[i][x]$ をすべて持つ代わりに、「傾きが変わる点」の x 座標をすべて持つことにします。

重要な事実として、このような形に対して「ある一定長さの区間の min をとる」をすると、「傾き 0 の部分 (つまり底の部分) がその一定長さ延びる」という結果になることがわかります。なのでこれは底の部分の左右にわけて両者を平行移動するとみなせます。従って、傾き 0 の区間より左で傾きが変わる点, 右で傾きが変わる点をそれぞれ set で持って、平行移動は set 全体に対して足された値として管理すると操作 1 と 2 に関しては処理できます.

これを持てば操作 3 も簡単に処理できて、「ある点で傾きが 2 回変わることになる」という捉え方をすると、その点が傾き 0 の区間の中にあるか、左にあるか、右にあるかで場合分けして、set に点を追加して、必要に応じて片方の set の中身をひとつもう片方の set に移し替えることで実現できます。

最終的な答えは、傾き 0 の部分の高さを常に保持することで求めることが出来ます。計算量は $O(NlogN)$ です。

F: NarrowRectangles

まず、$A \leq B$ のときは $Impossible$ です。これは、不親切な人が次の戦略を取ると何度質問をしても絶対に特定できないことからわかります:「不親切な人 B 人のうち A 人をあらかじめ選んでおいて、不親切な人はその A 人が正直者であるかのように質問に答える」

それ以外の場合は常に可能なことを構成的に示します。

まず頂点が 0 から $N-1$ の N 個あるグラフを持ち、質問の結果に応じて次のように辺を張っていきます。a が b のことを正直者だと言ったときには a から b に'Y' のラベルの付いた辺を張ります。($a \xrightarrow{Y}$b とかく) 同様に不親切な人だと言ったときには'N' のラベルの付いた辺を張ります。($a \xrightarrow{N}$b とかく)

まず N 回使って正直者を一人特定します。$a_B \xrightarrow{Y} a_{B-1} \xrightarrow{Y} ... \xrightarrow{Y} a_0$ という部分グラフがあるとします (a_i はそれぞれ異なるとする)。この時、a_0 は正直者です。($\because$ ある a_i が正直者だったとすると、推移的に a_0 も正直者になります。a_0 から a_B まで全員不親切な人だとすると、不親切な人が $B+1$ 人いることになり矛盾します。)

よって基本的には $\xrightarrow{Y}$ でつながった path を伸ばしていくことを目指します。

$a \xrightarrow{N} b$ となった場合は、この二人を一旦 N 人から取り除きます。a と b の少なくとも一方は不親切な人であることがわかるので、こうしたとしても $A > B$ という条件は成り立ったままです。

このことから次のようなアルゴリズムが考えられます。

```
bool ask(int a,int b){
        cout<<"? "<<a<<" "<<b<<endl;
        char c;
        cin>>c;
        return c=='Y';
}

int need = B+1;
vector<int> path;
for(int i=0; i<N; i++){
        if(path.empty()){
                path.push_back(i);
                if(path.size()>=need) break;
                continue;
        }
        bool b = ask(i,path.back());
        if(b){
```

```
                path.push_back(i);
                if(path.size()>=need) break;
        }
        else{
                path.pop_back();
                need--;
        }
}
int honest = path.front();
```

このアルゴリズムは N 回の質問で正直者を一人特定できます。あとはその人に他の人が正直者かを聞くことで全体で $2N$ 回で全ての正直者を特定できます。

ARC070 / ABC056 Editorial

sigma425

A: HonestOrDishonest

```
int main(){
        char a,b;
        cin >> a >> b;
        if(a=='H' && b=='H') cout << 'H';
        if(a=='H' && b=='D') cout << 'D';
        if(a=='D' && b=='H') cout << 'D';
        if(a=='D' && b=='D') cout << 'H';
        return 0;
}
```

B: NarrowRectanglesEasy

If $|a-b| \leq W$, the answer is 0. Otherwise the answer is $|a-b|-W$.

C: Go Home

The answer is the minimum t such that $1+2+...+t \geq X$.

D: No need

The card a_i is unnecessary if there is a subset of the other $N-1$ cards whose sum is in the interval $[K-a_i, K)$. Thus, the straightforward DP is $O(N^3)$.

There are two ways to improve it:

- $O(N^2 logN)$: It turns out that if $a_p < a_q$ and a_q is unnecessary, a_p is also unnecessary. Do binary search using this fact.
- $O(N^2)$: Do DP twice: for prefixes and for suffixes.

Also, it can be 64 times faster if you use bitset.

E: NarrowRectangles

Define $dp[i][x]$: the minimum cost to move the first i rectangles such that the last (the i-th) rectangle's leftmost coordinate is x. This will lead to a solution for partial score.

Now, see $dp[i]$ as a function: it returns $dp[i][x]$ for given x. It turns out that this function is a polyline consisting of $2i+3$ sections, and the slopes of the sections are $-i-1, -i, \ldots, i, i+1$ from left to right. Thus, this polyline can be represented using $2i+3$ integers $l_0, l_1, \ldots, l_i$, $r_0, r_1, \ldots, r_i$, and c.

- In the interval $(-\infty, l_i]$, the slope of the polyline is $-i-1$.
- In the interval $[l_i, l_{i-1}]$, the slope of the polyline is $-i$.
- $\cdots$
- In the interval $[l_1, l_0]$, the slope of the polyline is -1.
- In the interval $[l_0, r_0]$, the polyline is constant, and the value is c.
- In the interval $[r_0, r_1]$, the slope of the polyline is 1.
- $\cdots$
- In the interval $[r_{i-1}, r_i]$, the slope of the polyline is i.
- In the interval $[r_i, \infty)$, the slope of the polyline is $i+1$.

Now we compute the polylines in the order $dp[0], dp[1], \ldots, dp[N]$. We should keep two sets (or priority queues) representing $\{l_0, \ldots, l_i\}$ and $\{r_0, \ldots, r_i\}$ meanwhile, and the solution works in $O(NlogN)$.

F: NarrowRectangles

When $A \leq B$, the solution is "impossible" because A of unkind people can behave like honest people (call each other "honest" and call everyone else "unkind"). Otherwise, we can determine honest people in the following way:

First, notice that when a person p says "q is unkind", p and q can't be honest at the same time. Thus, even if we ignore both p and q, more than half of the remaining people is honest. Whenever we get the response "unkind", we ignore the two people.

Create an empty stack and try to push N people to this stack one by one. When we push a person p in to the stack, we do the following. Let q be the person that is currently at the top of the stack. Ask q about person p, and if we get the answer "unkind", we pop q from the stack and ignore p and q. Otherwise, push p into the stack.

After we try to push N people, what happens in the stack? Let $a_0, \ldots, a_{k-1}$ be the people in the stack from bottom to top. From the construction of the stack, for each i, a_i says "a_{i+1} is honest". Also, at least one of $a_0, \ldots, a_{k-1}$ is honest. Thus, we are sure that a_{k-1} is honest.

Now we can ask additional N queries to this honest person and get the answer.