2019 HDU 多校 5 出題者口胡題解

前言

非常感謝 dls、300iq、shik、蜥蜴、AA、台灣 2019 年 IOI team、wangyenjen 對於這場比賽的幫助，尤其是 dls，抱病在比賽前的一天幫忙驗題(>_<)，若沒有他們，這場比賽是辦不起來的，真的非常謝謝大家。

是說，經過驗題者反饋，似乎有些題直接在搜索引擎上就能找到答案了啊。。。出題者只好安慰自己：「這是 ICPC 的訓練賽，ICPC 是不能在網上搜答案的，相信大家為了達到最佳的訓練效果，也不會在網上搜答案的，就像以前的我一樣！」於是出題者就厚臉皮的依舊放上那些題目了！

再順帶一提，dls 在 8/3 和我反饋說太難了，所以就把還未完成的兩題都換成簽到題，我非常感謝 dls 讓我有更多的睡覺時間，請大家也一起感謝 dls。

最後也謝謝大家參與這場比賽。

1001 - fraction

「有些題目答案是分數，可是 Sample 卻只擺著 $a \times b^{-1} \pmod p$ 的值，還不給樣例解釋，這樣的 Sample 毫無參考價值呀！！！！」

大家是否曾發出像我一樣的怒吼呢？有些時候，我就會對可能的分數做一些假設，並寫個暴力去猜測真實的樣例答案，本題的情況就是一種常見的假設。暴力寫久了，就不免去想，到底有沒有比較優美的解呢？於是這題就誕生了！並且很開心的，我自以為找到了一個絕妙的做法。直到找了 dls 幫我驗題：

接受了 dls 的醍醐灌頂後，只好把自己的長長一篇題解給捨棄了，換成這個簡短的題解。

以下是 dls 提供的做法：

由於 $\frac{a}{b} \equiv x \pmod{p}$，多引入一個參數 y，即可假設 $a = bx - py$，於是我們可列出不等式 $0 < a = bx - py < b$ (根據題目要求)。 $0 < bx - py \Rightarrow \frac{p}{x} < \frac{b}{y}$，

$bx - py < b \Rightarrow \frac{b}{y} < \frac{p}{x-1}$。

其中 $\frac{p}{x}$ 和 $\frac{p}{x-1}$ 都是已知的值，我們要求最小的 b 滿足 $\frac{p}{x} < \frac{b}{y} < \frac{p}{x-1}$，這是個經典的輾轉相除法問題，沒見過的人可以參考 [2019年 Google Code Jam Round 2 - New Elements: Part 2 的題解]。

1002 - [three arrays]

這題是 N 年前的某場 Codeforces Div. 1 B 題 shik 看錯題的產物。

當初隱約覺得這題可以做，但直到最近，我才證出某個看起來很難分析時間複雜度的方法，時間複雜度應該是 $O(n \log^2 V)$，於是比賽前不到 24 小時，我再把 shik 抓來討論這題，shik 和我討論要卡掉哪些時間複雜度不正確的假解：

感覺測資要卡一下亂做

要什麼亂作方法呀?

對每個 ai 找最好的 bi，之後開個 min heap 每次 pop 最好的 pair 出來，發現 bi 已經被用掉就重找之後 push 回去

有多個最好的 bi 會 random 挑一個

情感上隨機測資應該不會跑太久 (?)

a_i全都一樣，b_i全部一樣就炸了?

所以才要隨機挑一個啊

說錯

b_i全部不一樣

同樣的 ai 不要重找就好

就 heap 裡同樣數值的 ai 只會有一個，pop 出來之後 cnt[a[i]]-->0 會再找一次 push 回去

那就只好...a_i全部<2^15,b_i全是2^15倍數?

聽起來不錯

喇賽做法二號：每次隨機一個 ai 開始找最近的 bj，之後找離 bj 最近的 ak 直到找到一對相親相愛的為止就把他們 pair 起來

這做法真強...

不太會卡...

說不定這是正解 0.0

怎麼說 @@

考慮trie

每次找到最近的兩個點lca

如果連兩次lca一樣

就相當於合併兩個子樹

解同樣的問題

下次lac深度勢必小1

(在合併後的子樹的深度加1)

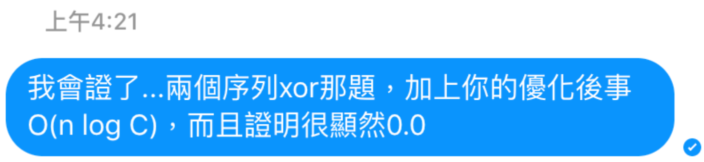
好像有點巧妙

滿有道理的 (?)

我在想... 說不定能做到只有一個log...

每次不要重新開始，找到相親相愛之後 pop 掉兩個，繼續找的話複雜度會變嗎

好難



嗯。。。這個故事告訴我們，出題的時候還是要多找別人討論。。。而且不要拖到最後一天 XDrz

回歸正題，參考作法如下。(方便起見假設相同數組裡的所有值都不一樣，若要處理重複出現的情形，只要紀錄一個值出現多少遍即可。)

我們稱 $a[i]$ 和 $b[j]$ 配對，當且僅當重排列後，這兩個數的下標一樣，且稱「配對值」為 $a[i]\ xor\ b[j]$。

由於我們假設同一個數組裡所有的數字都不同，所以對於每個 $a[i]$ 和 $b[j]$，在另一個數組裡都有唯一個「配對值」最小的數字。若把每個數字都當作圖論上的點，就把每個點連出一個有向邊，指向另一個數組「配對值」最小的點。

兩個最重要的觀察：

引理 1：若 $a[i]$ 在 b 中配對的值最小時是和 $b[j]$ 配對。而 $b[j]$ 在 a 中配對的值最小時也是和 $a[i]$ 配對，那麼在最終的排列中，$a[i]$ 和 b$[j]$ 配對一定是最好的。

證明：挺顯然的，若最終的排列 $a[i]$ 不是和 $b[j]$ 配對的話，交換成 $a[i]$ 和 $b[j]$ 配對一定更好。

我們稱這樣的一對 $a[i]$ 和 $b[j]$ 為相親相愛組合。

引理 2：不會有長度 > 2 的 cycle 出現。

證明：若點 x 指向 y ，點 y 又指向 z，這代表 x 和 y 的配對值嚴格大於 y 和 z 的配對值，所以若某個長度大於 2 的 cycle 通過從 x 指向 y 的邊，就能夠遞推得到 x 和 y 的配對值嚴格大於 x 和 y 的配對值，得到矛盾。

接著考慮以下算法：

有一個 stack，當 stack 為空但還存在未配對的點時，就任塞一個點進 stack。

否則，若 stack 頂端的點指向的點不在 stack 裡，就把該點加進去，否則指向的點一定是 stack 裡頂部第二個點，那麼這兩點就是一組相親相愛的組合，可以配對後移除。

這個算法的時間複雜度是 $O(N \log MAX\_V)$，MAX_V 是數組裡最大的數。

因為每次尋找一個點指向的點是哪個點可以使用 trie 做到 $O(\log MAX\_V)$ 的時間複雜度。而這件事我們只需做 $O(N)$ 次，因為每做一次時會發生下列兩種事情之一：1. 某個點被加入 stack。 2. 某兩個點被證實為相親相愛組合。顯然 1. 和 2. 都不會發生超過 N 次。

到此已不需要再說什麼了，感謝 shik，讚嘆 shik。

1003 - geometric problem

出題者小時候，每看到一個新概念，或看到一個令自己震懾的題，就喜歡模仿一下出個類似題，而這題是看到 sgu 312 後的產物。

把座標當作變數，此題可以列出許多線性方程式，但由於要求字典序最小，非唯一解時，字典序最小一定發生在 S 或 T 卡在邊界的情況。可以枚舉直線 AB、BN、NM、AM、MX、XY、NY 這幾條線是否為 S 和 T 座標限制的線性方程。找到所有合法的答案中字典序最小的。

1004 - equation

對於 $|a_i \cdot x + b_i|$ 來說，若 $x > \frac{-b_i}{a_i}$ 就會等於 $(a_i \cdot x + b_i)$ ，否則就會等於 $-(a_i \cdot x + b_i)$，當 x 從正無限大慢慢往負無限大移動時，會漸漸有一些式子變號，並整個 x 軸會被分為至多 $n + 1$ 個區間，每個區間內方程式的值都是線性變化。於是我們可以各個區間求解。

1005 - permutation 1

在還沒加新的簽到題之前的原簽到題。

雖然 N, K 其實都可以出更大，但還是把它弄的小小的。

這是為了安利 dreamoon 在知乎上寫的文章： 从枚举到 K 短路 - dreamoon的文章 - 知乎

爆搜有個好處：往往都可以方便地按照字典序直接搜索，不用想太多。

使用如下爆搜，時間複雜度是 $O(K \times N^2)$ 。

```
#include<cstdio>
#include<algorithm>
const int SIZE = 50;
int n, k;
int d[SIZE], u[SIZE];
// 此搜索函式在填值時只填相對的值，到最後一步才決定每個位置真正的值。
bool dfs(int id, int low, int hi){
    if(id == n) {
        k--;
        if(!k){
            for(int i = 0; i < n; i++) {
                if(i) printf(" ");
                printf("%d", d[i] - low + 1);
            }
            puts("");
            return 1;
        }
        return 0;
    }
    for(int i = hi - n + 1; i <= low + n - 1; i++){
        if(u[i]) continue;
        u[i] = 1;
        d[id] = i;
        if(dfs(id + 1, std::min(low,i), std::max(hi,i))) {
            u[i] = 0;
            return 1;
        }
        u[i] = 0;
```

```
    }
    return 0;
}
int main(){
    int T;
    scanf("%d", &T);
    while(T--) {
        scanf("%d%d", &n, &k);
        d[0] = n;
        u[n] = 1;
        dfs(1, n, n);
        u[n] = 0;
    }
    return 0;
}
```

1006 - string matching

簽到題之 1。

這是 N 年前出題者剛學會 Z 函數 (擴展 kmp) 時，興致沖沖出給學弟學妹們當練習題的題。

相信會 Z 函數的人這題一定是唾手可得，不會的人請自己去網上搜教程或抱隊友大腿。

但若你們想用用 kmp 解它，相信也是辦得到的。。。

1007 - permutation 2

簽到題之 2。

最初原本是要出總是 $p_1 = 1$ 且 $p_N = N$ ，也就是 Input 只給你一個正整數 N，這有簡單的遞歸式：$ans_x = ans_{x-1} + ans_{x-3}$，但 dls 嫌棄它，覺得無腦爆搜 + OEIS 或 BM 就能 AC 太無趣了，所以 dls 建議把題改成現在這樣。

但做法和原來也沒差多少 XD，以 $n = 12, x = 4, y = 9$ 為例，此序列必定長得如下：

$4, 2, 1, 3, 5, \ldots, 8, 10, 12, 11, 9$

相當於最初的題 $N = 4$ 的情形。

大致上就是這樣，還要根據 x 是否為 1， y 是否為 N 分成 4 種情形，請大家自行想像。

1008 - line symmetric

鑑於大家幾何普遍偏弱，於是挖出一個過去出給台大學弟妹練手的幾何給大家玩玩。

首先， $n \leq 4$ 時一定有解。

$5 \leq n$ 時，至少有一對相鄰或間隔一個點的點對是對稱的點，枚舉這 $O(n)$ 組點對，用 $O(n)$ 時間判斷是否可至多移一個點變成線對稱圖形。

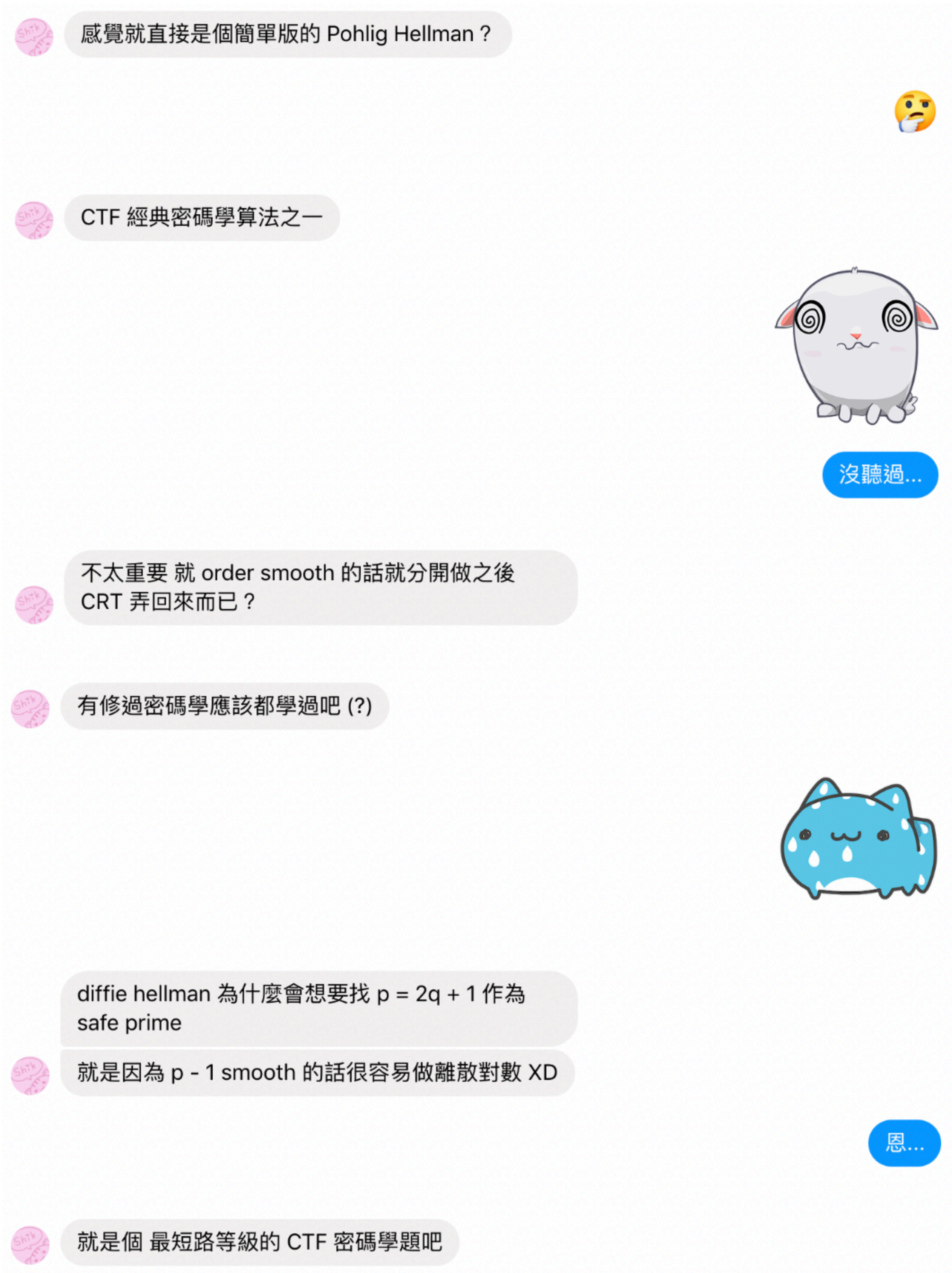
細節請大家自行小心。

1009 - discrete logarithm problem

N 年前，出題者在 Codechef 上看到一題，在 一個很多小質數相乘後再+1的質數取模下，對一個函數做多點求值的題目，覺得該題做法超厲害！於是就在思索，在該種特殊場景下，是否有其他問題也有很厲害的做法，於是就開發出這題。

但這題是個出題者自以為出了個好題，結果實際上在其他領域卻是個人盡皆知的 SB 題。。。會這樣說是因為，比賽前問了一下同時是 ICPC 和 CTF 的強者朋友，他告訴我這題之於 CTF，好比最短路之於 ICPC。。。

那就當作出題者在推廣 CTF 的最短路問題吧。。。有興趣者自行百度 Pohlig Hellman 算法。



1010 - [find hidden array]

令 $a_i = |b_i|$。

對於數字 a_i，他可能在原數組的可能位置會是一個區間，且區間右界是 $n+i-1$，而對於所有 a_i 的可能位置左界可以用 stack + binary search 依序得出。

對於不存在 a_i 裡的另外 n 個數字，可能位置也會是個區間，且右界是 $2n$，左界也可用 stack + binary search 得出。

先考慮一個比較簡單的問題：是否存在至少一種可能的排列。

這是個比較基礎的貪心問題，只需由編號小到大依序決定要放哪個數，每次選可以放的數當中，可以放的位置的右界最左邊的數即可。

神奇的事是這樣的：找任一組解可以用貪心，不僅這樣，想找字典序最小的解依舊可以用貪心！

貪心的算法如下：

> 由下標小至大依序決定隱藏數組的每個位置的值。
>
> 若存在某個值還沒被放入數組，且該值能擺的位置只剩下當前枚舉的數組位置，則放入該值。否則就選能放的值中最小的。

這個算法簡單來說就是幾乎不管擺了之後是否還有合法解，就只是為了達成字典序最小而貪心。所以若真的按照此算法決定完所有位置，那麼它肯定是字典序最小的，於是我們必須證明的事情是：每一時刻，都至少有一個數能放入。

再換個說法，

對於給定 Input，假設按照如上的貪心做法，數組第 i 個位置放入的數是 $target[i]$，而對於任意一組可能的隱藏數組(假定第 i 個位置的值是 $array_0[i]$) 我們有辦法依序地令 k 從 $1 \sim 2N$，都找到一組合法解，令第 k 組合法解的數組為 $array_k$，使得 $array_k$ 的前 k 個數都和數組 target 的前 k 個數一樣，那我們證明完畢了。因為這代表著我們擺上第 k 個該算法的值時，仍有解存在。

證明如下：

對於每一個 k，

若 $array_{k-1}[k] = target[k]$，直接令 $array_k = array_{k-1}$ 即可。否則，

令 $target[k]$ 在 $array_{k-1}$ 的位置為 x。

令 $array_{k-1}[k]$ 這個數能放的位置的右界位置為 y。

若 $x \leq y$，只需交換 $array_{k-1}$ 的第 k 和第 x 位置，即可得到 $array_k$。

否則，令 $array_{k-1}[y]$ 這個數能放的位置的右界位置是 y_2，y_2 必定大於 y (因為除了末位以外，每個位置最多是一一個數的可能位置右界)

若 $x \leq y_2$，我們有$k < y < x$把 $array_{k-1}$ 的第 k 個數換至第 y 個數，第 y 個數換至第 x 個數，第 x 個數再換回第 k 個數，就能得到 $array_k$。

否則，令 $array_{k-1}[y_2]$ 這個數能放的位置的右界位置是 y_3，y_3 必定大於 y_2 (因為除了末位以外，每個位置最多是一一個數的可能位置右界)

若 $x \leq y_3$，我們有$k < y < y_2 < x$把 $array_{k-1}$ 的第 k 個數換至第 y 個數，第 y 個數換至第 y_2 個數，第 y_2 個數再換至第 x 個數，最後地，第 x 個數再換回第 k 個數，就能得到 $array_k$。

...

...

...

依此類推，

由於 y_i 會越來越大，總有大於等於 x 的時候，所以一定能得到 $array_k$，於是我們已證畢。

抱歉由於出題者不太擅長寫文章，所以才把一個簡單的證明寫得這麼長，若有誰能把這段證明寫的簡短些，請務必提供給大家！

By the way, 這題原本是要出在百度之星的，但後來覺得...這題太莫名了！有勇氣用力猜的人就能 AC，於是就決定把這題搬到 regional 這一類的比賽， regional 訓練賽就是專門塞各種怪提的 XD。

結語

感謝 hdu 讓我有這個機會，把數年來縈繞在我心頭上的部分題目一口氣傾巢而出，加上牛客多校，讓我積存的題單瞬間少了十道左右，有種卸下好多塊小石頭的感覺，但題單裡仍有好多令人很糾結的題呀。。。嗚嗚嗚。