

A 关于时间复杂度

问 4 段代码是否会超时。

解题思路

- 代码 A 可以以 `++k` 作为时间复杂度的判断依据，约为 $O(\sum_{i=1}^n i) = O(n^2)$ ，会超时。
- 代码 B 中 i 与 j 独立， i 的范围为 $O(n)$ ， j 的范围为 $O(\sqrt{n})$ ，根据乘法原理，总时间复杂度为 $O(n\sqrt{n})$ ，不会超时。
- 代码 C 中 `++k` 的运行次数为 $O(\frac{n}{1} + \frac{n}{2} + \cdots + \frac{n}{n}) = O(n \sum_{i=1}^n \frac{1}{i}) = O(n \log n)$ ，不会超时。

$(O(\frac{n}{1} + \frac{n}{2} + \cdots + \frac{n}{n}) = O(n \sum_{i=1}^n \frac{1}{i}) = O(n \log n))$ 是算法常用复杂度，需要熟悉

代码 D 中 j 的改变次数与 i 有关，过程中则若 j 为偶数则 $\div 2$ ，若 j 为奇数则 -1 变为偶数，其中 j 作为偶数 $\div 2$ 的次数为 $O(\log_2 i)$ ，作为奇数 -1 的次数小于 $O(\log_2 i)$ ，故 j 的改变次数为 $O(\log_2 i)$ 。

代码 D 的总时间复杂度为 $O(\log_2 1 + \log_2 2 + \cdots + \log_2 n) \leq O(n \log_2 n)$ ，不会超时。

```
print('BCD')
```

B 老实点 每次都有你

求存在哪些 k 使得 k 为所有 a_i 的因数。

解题思路

首先用 $O(n \log n)$ 的复杂度求出所有 a_i 的最大公因数 gcd 。

随后我们设 $gcd = a * b$ ，其中 $a \leq b$ ，则 a 的取值范围为 \sqrt{gcd} 。用 $O(\sqrt{a})$ 的复杂度枚举所有的 a ，然后同样推出所有的 b ，即可得到所有因数。

```
import math
import functools

T = int(input())
for _ in range(T):
    n = int(input())
    a = list(map(int, input().split())) # list of integers
    gcd = functools.reduce(math.gcd, a) # gcd of all elements
    sqrt = math.ceil(math.sqrt(gcd)) # square root of gcd
    cd = [x for x in range(1, sqrt) if gcd % x == 0]
    cd.extend([gcd // x for x in cd])
    if sqrt * sqrt == gcd: cd.append(sqrt)
    cd.sort() # divisors of gcd
    print(" ".join(list(map(str, cd))))
```

C 木桶效应

找编号最小的点所在连通块，弹出其编号最大的点，求弹出顺序。

解题思路

搜索解法： $O(n)$

从小到大枚举编号最小的点，若这点没有被弹出，则通过搜索得到其连通块。

连通块排序后从大到小输出即可。

并查集解法： $O(n \log n)$

通过并查集连通所有的点，连接过程中优先以编号最小的点为根，则最后能保证所有点的祖先结点为连通块中编号最小的点。

按（祖先结点编号最小，自己编号最大）这样的双关键字排序，即可得到最后的弹出顺序。

```
#include<cstdio>
#include<vector>
#include<algorithm>
using namespace std;
constexpr int MN=1e5+5;
int T,n,m,fa[MN];
vector<pair<int,int>> vec;

int getfa(int x) {
    return fa[x]==x?x:fa[x]=getfa(fa[x]);
}

int main() {
    scanf("%d",&T);
    for (; T; --T) {
        vec.clear();
        scanf("%d%d",&n,&m);
        for (int i=1; i<=n; ++i) fa[i]=i;
        for (int x,y; m; --m) {
            scanf("%d%d",&x,&y);
            x=getfa(x),y=getfa(y);
            x<y?fa[y]=x:fa[x]=y;
        }
        for (int i=1; i<=n; ++i)
            vec.push_back(make_pair(getfa(i),-i));
        sort(vec.begin(),vec.end());
        for (auto i:vec)
            printf("%d ",-i.second);
        puts("");
    }
}
```

D 究极手

(快点到 5.12 玩《塞尔达传说：王国之泪》)

给定 n 个点，给出这些点的度，问是否能构成一棵树（即无向无环连通图）。

解题思路

一棵树只允许有 $n - 1$ 条边，一条边需要 2 个度，故所有点的度之和必须为 $2n - 2$ 。同时由于所有点必须要被连通起来，所以不允许存在度为 0 的点。

当 $n = 1$ 时，没有其它点需要连通，故不再需要添加新的边。

当 $n = 2$ 时，其度必然为 1, 1。用一条边将两个点连起来即可。

当 $n > 2$ 时，由于没有度为 0 的点，根据鸽巢原理， $2n - 2 < 2n$ 故必然存在度为 1 的点； $2n - 2 > n$ 故必然也存在度大于 1 的点。

用一条边连接这样两个点，则会使一个点度数被填满，则此时点数 $n \rightarrow n - 1$ ，度数 $2n - 2 \rightarrow 2(n - 1) - 2$ ，仍满足规定结构，归纳得到所有的边。

具体实现

用两个栈，一个存度数为 1 的点，一个存度数大于 1 的点，每次各取一个点连接即可，直到最后只剩两个度数为 1 的点相互连接。

E Password

求满足 $\sqrt{5 * x^2 + 4}$ 为整数的最大的小于 n 的整数。

解题思路

打表盲猜写法

打表：0, 1, 3, 8, ...

然后找规律，发现是 [0], 1, [1], 2, [3], 5, [8], ...。

原来是斐波拉契数列的奇数位。

斐波拉契数列增长很快，所以不用担心数量会很多。

顺带一提，满足 $\sqrt{5 * x^2 - 4}$ 为整数的是斐波拉契数列的偶数位。

盲猜线性写法

盲猜是线性方程。

$$\begin{pmatrix} ans_{n-1} \\ ans_n \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} ans_n \\ ans_{n+1} \end{pmatrix}$$

然后算出 $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ 。

由于斐波拉契数列也是线性方程组，所以这做法得到的解是正确的。

正规证明 from hitass_SY

$(n^2 - nm - m^2)^2 = 1, n, m \in \mathbb{Z}$ 有解当且仅当

$n = \frac{m^2 \pm \sqrt{5m^2 + 4}}{2}, n, m \in \mathbb{Z}$ 有解

即 $\sqrt{5m^2 + 4}$ 为整数，不妨设 $m \geq 0$ ，则 $n = -1$ 或 $n \geq 0$

充分性：

$$F_0 = 0, F_1 = 1$$

$$0^2 - 0 * 1 - 1^2 = 1$$

$$\text{若 } F_i^2 - F_i F_{i-1} - F_{i-1}^2 = 1$$

则

$$\begin{aligned} F_{i+1} F_i - F_{i+1} F_i - F_i^2 &= (F_i + F_{i-1})^2 - (F_i + F_{i-1}) F_i - F_i^2 \\ &= -F_i^2 + F_i F_{i-1} + F_{i-1}^2 \\ &= -1 \end{aligned}$$

$$\begin{aligned}
F_{i+k}F_{i+k-1} - F_{i+k}F_{i+k-1} - F_{i+k-1}^2 &= -(F_{i+k-1}F_{i+k-2} - F_{i+k-1}F_{i+k-2} - F_{i+k-2}^2) \\
&= \dots \\
&= 1 * (-1)^k
\end{aligned}$$

必要性：

若 $nm < 0, |n| > |m|$ 则 $n^2 - m^2 \geq 1, -nm \geq 1$, 即 $(n^2 - nm - m^2)^2 \geq 4$

在具有斐波那契递推性质的数列 $\{A_i\} (i \in \mathbb{Z}, A_i \in \mathbb{Z})$ 中, A_i 不可能全为正, 不可能全为负。若全为正, 则 $A_i < A_{i+1}$, 即 $A_i \leq A_i - 1$, 即 $A_i \leq A_0 + i$, 则 $\lim_{i \rightarrow -\infty} A_i < A_0 + \lim_{i \rightarrow -\infty} i = -\infty < 0$, 与假设矛盾。同理可证 A_i 不可全为负。

若存在连续两项为正, 不妨设 $A_i > 0, A_{i+1} > 0, A_{i-1} \leq 0$ 。若 $A_{i-1} = 0, A_i = 1$, 则为斐波那契数列本身, 若 $A_{i-1} = 0, A_i > 1$, 则 $(A_i^2 - A_i A_{i-1} - A_{i-1}^2)^2 = A_i^4 > 1$

同理可证存在连续两项为负, 要么为斐波那契数列本身逐项取相反数得到的数列, 要么 $(A_i^2 - A_i A_{i-1} - A_{i-1}^2)^2 > 1$

若数列两项正负交替, 即 $\forall i (A_i A_{i+1} \leq 0)$, 易知 $|A_i| > |A_{i+1}|$, $|A_{i+1}| \leq |A_i| - 1$, $|A_i| \leq |A_0| - i$, $\lim_{i \rightarrow +\infty} |A_i| < |A_0| + \lim_{i \rightarrow -\infty} i = -\infty < 0$, 绝对值小于0, 矛盾。

综上, 具有斐波那契的性质的数列中只有斐波那契数列本身的相邻两项可以满足 $(n^2 - nm - m^2)^2 = 1, n, m \in \mathbb{Z}$

```

#include<bits/stdc++.h>
using namespace std;
vector<unsigned long long> fib;
constexpr unsigned long long MN=1e18;
unsigned long long n;
int T;
int main() {
    unsigned long long s=0,ls=1;
    while (s<=MN) {
        fib.push_back(s);
        s+=ls;
        ls=s-ls;
        s+=ls;
        ls=s-ls;
    }
    scanf("%d",&T);
    for (;T-->0) {
        scanf("%llu",&n);
        printf("%llu\n",
            *(upper_bound(fib.begin(),fib.end(),n)-1));
    }
}

```

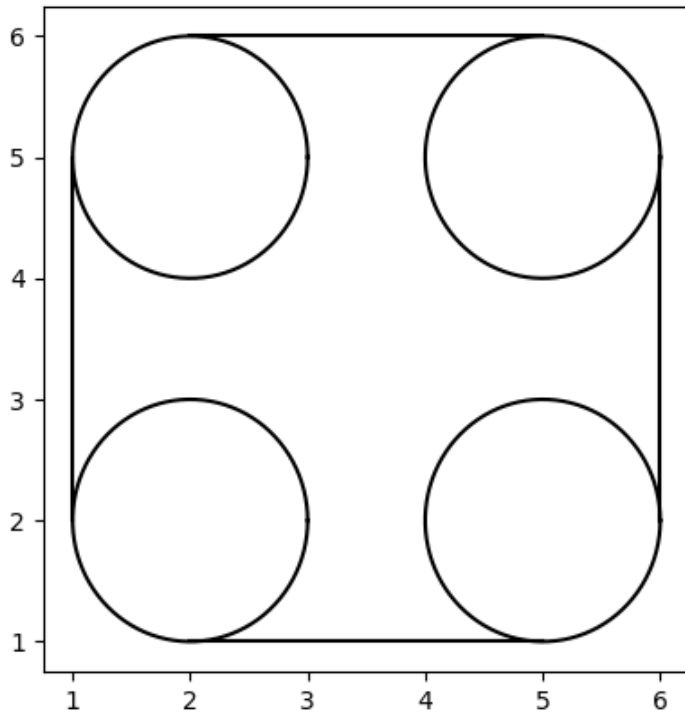
F 熊爷别笑了

给定平面上一些圆心，给出统一的半径，求最短的线将所有点包起来。

解题思路

假设半径为 0，即只要求一个凸包，算凸包长度。凸包可以理解为用一根绳子将平面上一些点勒紧捆住，具体内容需要自己学习。

由于半径统一为 R ，故最终答案只是在凸包基础上多围了一个周长为 $2\pi R$ 的圆。



```
import math

def makehull(nd, cmp) -> list:
    hull = [nd[0], nd[1]]
    for i in range(2, len(nd)):
        while len(hull) >= 2 and cmp(
            math.atan2(hull[-1][1] - hull[-2][1],
                hull[-1][0] - hull[-2][0]),
            math.atan2(nd[i][1] - hull[-1][1],
                nd[i][0] - hull[-1][0])):
            hull.pop()
        hull.append(nd[i])
    return hull
```

```

def calc(hull) -> float:
    return sum([
        math.sqrt((hull[i][0] - hull[i + 1][0])**2 +
            (hull[i][1] - hull[i + 1][1])**2)
        for i in range(len(hull) - 1)
    ])

T = int(input().strip())
for t in range(T):
    n, w = input().strip().split(' ')
    n, w = int(n), float(w)
    nd = []
    for i in range(n):
        x, y = [float(x) for x in input().strip().split(' ')]
        nd.append((x, y))
    if n > 1:
        nd.sort()
        uphull = makehull(nd, lambda x, y: x < y)
        downhull = makehull(nd, lambda x, y: x > y)
        print(calc(uphull) + calc(downhull) + math.tau * w)
    else:
        print(math.tau * w)

```


H Pause! Heal! Explosion!

与敌人对打，敌我血量为 A, C ，敌人每回合攻击 B 点伤害，自己有多种技能分为三类分别为攻击、控制、治疗技能，求能剩下的最大血量。

解题思路

我们发现有两种无解的技能。

如果一种治疗技能的使用能快过敌方消耗自己的速度，如果能攒到释放此技能的精力，则必然会无限释放，直到血量达到无穷大。

如果一种控制技能能让敌方无限僵直，同时你能抽出空余时间干其他事情，那么必然能打败敌方；

- 如果存在治疗技能，则必然会先抽空回血回到无穷大；
- 如果不存在治疗技能，则有可能会使用此技能无限僵直，也有可能在攒够释放此技能的精力前打倒敌人。

如果不存在这些无解的技能，或者不能攒够精力释放这些无解的技能，则必然只会用攻击技能硬碰硬。此处即是背包问题的思路，用动态规划实现。

```
#include<cstdio>
#include<vector>
#include<algorithm>
using namespace std;
inline void MIN(int &x,const int y) {
    (x>y)&&(x=y);
}

int T,A,B,C,n,t,x,y,xt2,xt3,failtime,healflag;
int F[1000005];
vector<pair<int,int> > hit;

int main() {
    scanf("%d",&T);
    for (; T; --T) {
        scanf("%d%d%d%d",&A,&B,&C,&n);
        hit.clear();
        xt2=xt3=failtime=(C+B-1)/B+1;
        healflag=0;
        F[0]=0;
        for (int i=1; i<=A; ++i) F[i]=failtime;
        for (int i=1; i<=n; ++i) {
            scanf("%d%d",&t,&x,&y);
            //t=1 hit y, guarenteed there is one
            //t=2 stop y
            //t=3 heal y
            switch (t) {
                case 1:
                    hit.push_back(make_pair(x,y));
```

```

        break;
    case 2:
        if (x<y) MIN(xt2,x);
        break;
    case 3:
        healflag=1;
        if (x*B<y) MIN(xt3,x);
        break;
    }
}
if (xt3<failtime||xt2<failtime&&healflag) {
    puts("inf");
    continue;
}
sort(hit.begin(),hit.end());
for (auto ht:hit) {
    int x=ht.first,y=ht.second;
    if (y>=A) MIN(F[A],x);
    else {
        for (int i=y; i<=A; ++i) MIN(F[i],F[i-y]+x);
        for (int i=A-y; i<=A; ++i) MIN(F[A],F[i]+x);
    }
    for (int i=A-1; i>=0; --i) MIN(F[i],F[i+1]);
}
MIN(xt2,F[A]);
if (xt2<failtime)
    printf("%d\n",C-(xt2-1)*B);
else puts("0");
}
}

```

I 小学生也能秒解的题

给出5行字符串，随后翻转底3行，最后根据每 5×5 个字符破译为 1 个字符。

解题思路

题目提供的大数据 `the quick brown fox jumps over the lazy dog` 涵盖了 `a-z` 所有字母，可以借助此数据得到所有字母的密码。

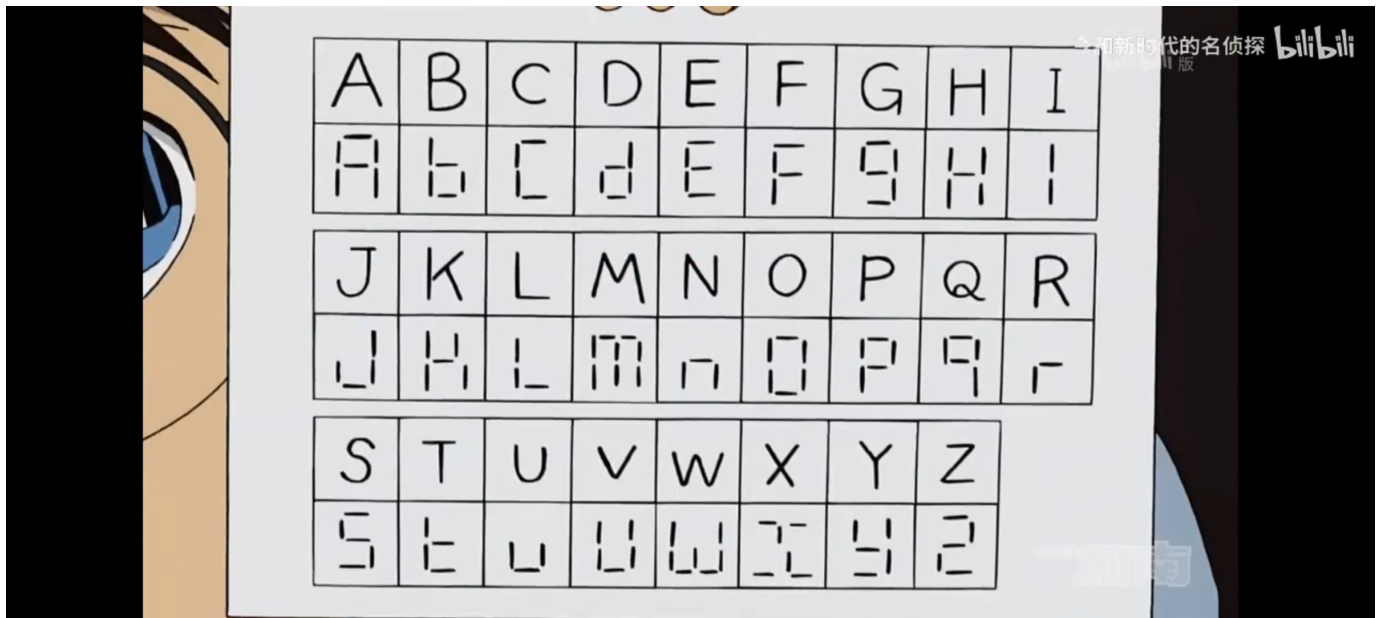
可以自己做表，也可以直接用此数据隐式做表。参考 Lanly 的隐式做表法。

```

};
auto check = [&](int l){
    for(int i = 0; i < ans.size(); ++ i){
        if (ok(i, l))
            return ans[i];
    }
    assert(l == 0);
};
for(int i = 0; i < n; ++ i)
    res += check(i);
cout << res << '\n';
return 0;
}

```

这是最后的表



A	B	C	D	E	F	G	H	I
A	b	C	d	E	F	G	H	I
J	K	L	M	N	O	P	Q	R
J	H	L	m	n	O	P	q	r
S	T	U	V	W	X	Y	Z	
S	t	u	v	w	x	y	z	

J Abyssal Echoes

给定 $x \oplus t, x \oplus y \oplus t, y \oplus t$ 求 x, y, t 。

解题思路

异或有着可逆的性质: $a \oplus b \oplus b = a$ 。

故 $t = (x \oplus t) \oplus (x \oplus y \oplus t) \oplus (y \oplus t), x = (x \oplus t) \oplus t, y = (y \oplus t) \oplus t$ 。

```
n=int(input().strip())

for i in range(n):
    xt, xyt, ty=[int(x) for x in input().strip().split(' ')]
    t=xt^xyt^ty
    print(xt^t,ty^t,t)
```

K 晚餐铃

来听听饭哥的曲子吧。

给一串序列，将其划分为连续的段，各段的最值求和得到 ans ，使 ans 尽可能大。

解题思路

若所有数都是负数，则所有数归为一段， ans 为最大值。

否则所有正数单独成段，然后把负数包进正数中，得到最终 ans 为所有正数之和。

```
T=int(input().strip())

for i in range(T):
    n=int(input().strip())
    a=list(map(int,input().strip().split(' ')))
    g=[i for i in a if i>0]
    print(sum(g) if len(g)>0 else max(a))
```

L 晚餐铃

来听听饭哥的曲子吧。

给一串序列，将其划分为连续的段，各段的最值求和得到 ans ，使 ans 尽可能小。

解题思路

首先压缩序列，所有正数必然在同一段中，取最大值，所有负数尽量单独成段，取和，最后序列会成为 $\cdots + - + - + - \cdots$ 的正负交替结构。

随后可以通过动态规划选出合适的段得到最优答案， DP_i 表示以 i 为结尾的正负交替序列的最小答案。其中答案必然也为正负交替结构。

故转移方式如下：

- 正到负转移 $+(---+)-$ 其中下标部分应满足不存在数大于最前方的 $+$ 。
- 负到正转移 $-(++--)+$ 其中下标部分应满足不存在数大于最后方的 $+$ 。

找到这样的正负交替结构使得答案最优即可。

奇技淫巧

如果想更方便处理边界问题，可以在序列前方加上 $+0-0$ ，在序列后方加上 $+0-0$ ，取 $+0\cdots-0$ 的序列即可。

M 梦与公交站

点 u_i 向 $[L_i, R_i]$ 中所有倍数为 k_i 的点连长度为 w_i 的边，建好图后求单源最短路。

解题思路

当 $k = 1$ 时可以用线段树辅助区间连边，参考CF786B。

具体为所有点作为线段树的叶子，然后线段树的枝干往下连接长度为0的边。随后叶子按给出的规定向枝干连边。

当 $k > 1$ 时同理，对于 $k = 1 \cdots n$ 各建一棵线段树即可，线段树空间大小为 $O(n \log n)$ 。