~~因为作业做不完子所以全是原题~~

T1

八月月赛 Div.2 A

P9496

按位判断即可。容易发现答案必定为0或1或2。

显然，如果有一位上 a 为 1 而 b 为 0，则只需在这一位上按位与上 0。同理，如果有一位上a 为 0 而 b 为 1，则只需在这一位上按位或上 1。

对于两个都是 0 或 1 的，显然可以按位与上 1 或按位或上 0。

综上，容易发现最多只需要按位与一个数，然后按位或另一个数，则一定可以变成目标。

```
t = int(input())
for _ in range(t):
    a, b = map(int, input().split())
    x, y = 0, 0
    while a or b:
        if a % 2 == 1 and b % 2 == 0:
            x = 1
        if a % 2 == 0 and b % 2 == 1:
            y = 1
        a //= 2
        b //= 2
        if x and y:
            break
    print(x + y)
```

T2

P3817

贪心即可

```
n, x = map(int, input().split())
lis = [0] + [int(t) for t in input().split()]
s = 0
for i in range(1, n + 1):
    if lis[i] + lis[i - 1] > x:
        s += lis[i] + lis[i - 1] - x
        lis[i] = x - lis[i - 1]
print(s)
```

T3

noip2002 普及组

P1036

深搜，搜到底的时候判断是否为素数即可

```
n, k = map(int, input().split())
lis = [int(x) for x in input().split()]
ans = 0


def isprime(x):
    i = 2
    while i * i <= x:
        if x % i == 0:
            return False
        i += 1
    return True


def dfs(cur, beg, num):
    global ans
    if num == k:
        if isprime(cur):
            ans += 1
        return
    for i in range(beg, n):
        dfs(cur + lis[i], i + 1, num + 1)


dfs(0, 0, 0)
print(ans)
```

T4

noip2004 普及组

P1088

依然是一道深搜。

> 题外话：这个相当于实现了C++中的一个叫 next_permutation的东西

```
n, m = int(input()), int(input())
lis = [int(x) for x in input().split()]

flag = False
num = 0
```

```
visited = [False for _ in range(n)]


def dfs(cur):
    global flag, num, n, m
    if flag:
        return
    if cur >= n:
        num += 1
        if num == m + 1:
            for i in lis:
                print(i, end=" ")
            print("")
            flag = True
        return
    idx = 0
    while idx < n:
        if num == 0:
            idx = lis[cur] - 1
        if not visited[idx]:
            visited[idx] = True
            lis[cur] = idx + 1
            dfs(cur + 1)
            visited[idx] = False
        idx += 1


dfs(0)
```

T5

P1163

二分法求解方程数值解

~~理论上来说数学课也讲过~~

```
eps = 0.0001 # 误差范围最好更精确一位防止出现精度问题
n, m, k = map(int, input().split())


l, r = 0, 5
while r - l > eps:
    mid = (l + r) / 2
    if (1 / (1 + mid)) ** k < (1 - n / m * mid):
        l = mid
    else:
        r = mid
print("%.1f" % (l * 100))
```

T6

八月月赛 Div.2 B

P9497

可以猜测，一定可以选到所有的 $\geq v$ 的数作为独立的列(如果超过 n 个就钦定 n 个)。换句话说，最终的答案就是矩阵中 $\geq v$ 的数的个数与 n 得到最小值。

考虑证明这个结论。

令 $\geq v$ 的数(超过 n 个就钦定 n 个)为 1,否则为 0。只需证明,可以重排行使得每个 1 都独占一列。

逐列考虑。只需证明,对于任何矩阵(不一定是方阵)，如果至少有一个 1,至多有列数个,可以重排行使得第一列恰有一个 1,就可以归纳完成证明。

考虑反证法。

若无法做到,第一列必须有两个 1 ,故有两行全是 1 。这与至多有列数个 1 的假设矛盾!

因此,只需找到有多少个矩阵中的数 $\geq v$ 即可。这可以使用排序和二分实现。时间复杂度 $O((n^2+q)\log n)$。

> 题外话：C++中有个叫 `lower_bound` 的内置库函数可以直接调用
>
> 也许python也有

```python
n, q = map(int, input().split())
lis = []
for _ in range(n):
    lis = lis + [int(x) for x in input().split()]

lis = sorted(lis)


def lowerbound(x):
    l, r = 0, len(lis) - 1
    mid, res = 0, len(lis)
    while l <= r:
        mid = (l + r) // 2
        if lis[mid] < x:
            l = mid + 1
        else:
            res = mid
            r = mid - 1
    return res


for _ in range(q):
    v = int(input())
```

```
print(min(n, len(lis) - lowerbound(v)))
```