欧拉筛

是质数 = [True]*(n+1)

质数 = [] 共:个数组

```
for i in range (2, n+1):
    if 是质数[i] = True:
        质数.append i
    for p in 质数
        if i*p > n: 超界了
            break
        质数[i*p] = False
        if i%p == 0: 优化:直接用
            break
```

质数 ——> 便是 2~n的质数表

判断一个数是不是质数(6倍数)

↳一切大于6的数均写为6k±1

n:

① if n<=1 ; return False

② if n=2 or n=3: 是

③. if n%2==0 or n%3==0 : 不是

```
for i in range (5, ⌊√n⌋+1, 6):
    if n%i==0 or n%(i+4)==0: ↳import math
        return 不是            math.sqrt(n)
return 是.
```

语法点

① f"{a:.2f}", 字符串格式化 (习先用一下float())

↳浮点数 注意"-0.000"优化为"0.000"

② "间隔".join(列表), 字符串拼接

③A.index(a), 找A中首次出现a.

④ 双指标判断

list.sort(key = lambda x: (x[0], x[1]))

⑤ 创建多出之副本(深拷贝)

A为一个二维表

from copy import deepcopy.

B = deepcopy(A)

⑥ try: .... except EOFError

⑦ from math import gcd

gcd(a,b) ⇒ 最大公因数.

↳最小公倍数*最大因数=a*b.

abs() ⇒ 绝对值.

⑧ ASCII码:

字母 —ord()→ ASCII码

ASCII码 —chr()→ 字母

a~z: 97~122

A~Z: 65~90

技巧

① from collections import Counter

Counter(list), 返回一个字典

② 大于10^6, 不可穷举

③ from collections import deque

A = deque([a])

④ ⌊n⌋ : n//1,

⌈n⌉ : -((-n)//1)

⑤. 二分查找一般要先排序.

例题整理.

DP

最佳买单: for i in 列表:
```
    for j in c.copy()
        if j<b: c.add(i+j)
        else: continue
```
↓
print c.(sort) 的最大的(大括号中)

时间调度问题
- 畜栏保留问题: 用堆, 维护最小
- 区间调度:
  → 这些调度问题便开一个堆吧

八皇后: 回溯法: 学会用递归

Greedy

一维 kadane 算法: 计算最大子段和.
```
for num in nums
    current_sum = max(num, current_sum + num)
    max_sum = max(max_sum, current_sum)
```

对于二维的情况, 就化为一维 (先计算前缀和)

DFS & BFS

迷宫要求存储出路径: 用 BFS, 三元组 (x, y, 路径表[]).

棋盘问题: 回溯法

变换的迷宫: 三维 visited

快速把握 三参数 (t, x, y)
时间

注意事项

边遍历边修改不好.
对于集合, 可以用 for i in A.copy()

int(str, n)
字符串 → 进制.

for key, value in dict.items()

dict.get(key, default)
↳ 可选 default = 0

list(zip(a, b))
↳ 合并列表 (元组), (a_1, b_1) (a_2, b_2) ...

math.log(m, n) ⇒ $\log_n m$, 不给n, 默认 ln

math.pow(m, n) ⇒ m^n

@lru_cache

---

import calendar
日历

calendar.month(年, 月).
返回一个月份日历字符串

calendar.isleap(年). 返回布尔值, 是否闰年

calendar.leapdays(年1, 年2) 返回 1~2 间闰年数

calendar.weekday(年, 月, 日). 返回这天周几
0 ~ 6
周一 ~ 日

计数: from collections import Counter
n个元素 返回字典

---

from itertools import permutations.

全排列
list(permutations(表))

combinations

全组合
combinations(表, 数)
↳ eg 2
全 2元组合

WA: 先查特殊情况

边界情况, 循环条件, 正反便可以.

拷贝 copy

递归优化

△ sys.setrecursionlimit

△ functools.lru_cache.

简单题我直接多个循环

Heapq 堆

import heapq.

list.

heapq.heapify(lst)

heap.heappop (lst)

heapq.heappush (lst, element)

① 优先级调度问题
② Top K 问题
③ 合并多个有序序列
④ 滑动窗口
⑤ 最大化
⑥ 模拟
⑦ 数据流

Max 函数

① 一般 max 函数取二元

三元的话用 max(max(a,b), c)

② max(列表, key=lambda x: x[0])

可行

多重背包

```
k=1
while k <= 物品数:
    for j in range(N, price*k-1, -50)
        dp[j] = min(dp[j], dp[j-price*k]+k)
    max_count -= k
    k *= 2
```

或者 int(math.log(number, 2))

求是2的几次方

逃生指南 ⑧ 分类 Dp QQ Dp

1. 除法用地板除得到整数
2. 编进全部识
3. 边界情况，取很小的情况
4. return 的位置与逻辑
5. 贪心是否最优
6. 正反是列反 eg (装箱子, 字符串长度
7. 开背包可开 2 个

优化: ① 过程中 mod

② 过程中除整数, eg "万", ÷1000 记得除

③ 数字的整时用 +−×÷ 好用快

④ 分类, 提前跳出循环

逃生指南 ⑨ 回溯: 选择, 探索, 回溯全

⑩ 深拷贝, import copy

deep_copy = copy.deepcopy()

Date /

区间5讲　　以下七为数轴由左向右走的方法

一、区间合并

① 左端点排序

② 记忆右端点，遍历区间，若 new_左 > 右端点

更新 右端点 且 num += 1

二、选择不相交的区间

① 右端点排序

② 记忆右端点，遍历区间，若 new_左 > 右端点

更新右端点 且 num += 1

三、区点选点

① 右端点排序，初始点为首区间的右端点

② 记忆点，遍历区间，若 new_左 > 记忆点

更新记忆点 且 num += 1

四：区间覆盖

① 左端点排序，<坐标归一>，

② 记忆右端点，在所有 new_左 < 记忆点、区间内 <盖一个到表>

找到右端点最大的

更新右端点，且 num += 1

五　区间分组 <互不相交>

① 左端点 排序，

② 遍历所有区间 → 无组 or 不可被入现有组中 → 是 → 新盖组 num += 1

无组 or 不可被入现有组中 → 否 → 放旧组 continue

bisect

```python
import bisect
# 创建一个已排序的列表
sorted_list = [1, 3, 3, 6, 7, 9]
# 使用 bisect_left 查找元素应插入的位置
insert_index = bisect.bisect_left(sorted_list, 4)
print("Insert at index:", insert_index)
# 使用 insort_left 插入元素并保持有序
bisect.insort_left(sorted_list, 4)
print("Updated list:", sorted_list)
```

质因数分解

```python
from math import sqrt
pFact, limit, check, num = [], int(sqrt(n))+1, 2, n
for check in range(2, limit):
    while num % check == 0
    ...
if num>1
    pFact.append(num)
```

reduce：累积

```python
import functools
numbers = [1, 2, 3, 4, 5]
# 使用 reduce 计算累积乘积
product = functools.reduce(lambda x, y: x * y, numbers)
```

math.gcd(x,y) 最大公约

product：笛卡尔积

```python
from itertools import product
# 创建两个可迭代对象
colors = ['red', 'blue']
numbers = [1, 2]
# 生成它们的笛卡尔积
cartesian_product = list(product(colors, numbers))
# 创建一个可迭代对象
colors = ['red', 'blue']
# 生成它们的重复笛卡尔积
repeat_cartesian_product = list(product(colors, repeat=3))
```

注意

1.读题仔细

*wrong answer 先看看有没有弱智错误

2.边界情况：例如空数组、数组中只有一个元素、最大值和最小值等

3.循环条件和边界： 检查循环的条件和边界是否正确。确保不会死循环

4.变量名千万别重(复杂题不要单字母命名大法)

5.逻辑错误：尝试分析代码中的逻辑错误，可以手动模拟算法的执行步骤，或者使用示例输入数据来验证。

6.标志性打印语句：在代码中插入标志性的打印语句，以追踪代码的执行流程，找到问题所在。

（但是提交前别忘了注释掉）(ctrl+/)

7.正难则反：乌鸦坐飞机，剪绳子

8.确保输入和输出格式正确

9.回想一下有没有类似题做过，但不要生搬硬套

10.心态要稳，两个小时其实并不短，不急不躁正常发挥就问题不大

f"{a:.2f}"

中国剩余定理，同时用 $x_1 * x_2 * \ldots$

桶：n个数 → 数列合并

bisect:

from bisect import bisect_right　　from bisect import bisect_left

bisect_right(A, x)　→　第一个大于等于x的值

A中找到第一个大于x的值,返回其索引.　　→ 若有x,返回x

若A中存在x,则会返回其右　　若全小于x,返回 len(A)

若A中 max(A) < x
　则返回 len(A)

↳用于二分查找
　↳题目对斗的双牛 → 2个主要函数 ① 可在给定距离下布置
　　　　　　　　　　　　　② 二分找出最大距离
　↳三数之和 → ① 先sort ② 去重(跳过), ③ 若大于零,跳过.
　　　　　　④ [i] 左 右
　　　　　　　固定 找范围.

回文串

Manacher,找出回文串数量.
d₁[i]　d₂[i]　and 若有长为l的回文串,则 l-2, l-4 也是回文串
↓　↓
i为中心 长度为奇数 回文串数　i为中心 长度为偶数 回文串数
↓　↑
先算这个 → 再推这个

最长上升子序列: 二分法降低时间复杂度

宠物小精灵 — 二维费用背包，

↓

多目标优化 → 《举一反三》，n维也可以

0,1背包 $f(i,j)$ → 二维 $f(i,j,k)$ → $V[i]$, $M[i]$

↓ 前 i 个物品

s.t. 一个参数总和小于 j，一个参数总和小于 k

⇒ 三维化二维

⇓

奇怪的Dp：

dp[i][j] 表示捕获 i 个精灵，它在剩余体力 j 时，剩余精灵球最大

（可互换）

注，max只可有2个

注意倒序枚举

土豪购物

以第 i 个商品为结尾，不放回，最大值

$dp1[i] = \max(dp1[i-1]+a[i], a[i], 0)$

以第 i 个商品为结尾，可放回最大值

$dp2[i] = \max(dp1[i-1], dp2[i-1]+a[i], a[i], 0)$

采药。

一般 dp[x] 代表时间 x 可拿到的最大价值

↕

时空为 dp[x] 表示最大价值的最小时间

注，用二维数组才可得到最佳方案

→ 用回溯的方法，记录。

多重背包（0-1背包变种）

物品有（件数）→ 多套循环一个

↘ 朴素方法

↳ 二进制优化。

背包物品太多了 → 降低时间复杂度

↓

二进制，

2^0 2^1 2^3 …

↓ ↓ ↓

1 2 4 …

↳ 可以组合成任意数

↓ 0/1 组合

行拆解时 → $S[i] = 2^0 + 2^1 + 2^2 + \dots + Rem$

↓ 剩余数

行拆解为多个，

改 $W[i] \leq C[i]k$

代码 dp表：初始化为 −∞

```
dp[0] = 0
for i in range(7)
    cur_price = price[i]
    cur_num = nums[i]
    k=1
    while cur_num >0
        use_num = min(cur_num, k)
        cur_num -= use_num
        for j in range(n ~ 长度)
            dp[j] = min(dp[j],
                        dp[j - 求价]
                        + use_num)
        k *= 2.
```

宠物小精灵 — 二维费用背包

语法点：

⑨. 长字符串

a = "abc"

a[3:] 返回""空

a[4:] 也返回""空，不会报错

⑩. sys.stdin.read() 一次性读入

⑪ 二分查找，m = 左+(右-左)//2.

⑫. ①字典的知识！ 字典可嵌套 dic 为一字典

dic[键] ⇒ 返回值

②添加键值对 = 修改键值对

③ 初始字符串列顺序便已加入顺序：

④删除键值对：del dic[键]

⑤. 用 get() 访问值（不存在返回 None）

get(键, 返回值) 也可自定义 ← 可无

⑥. 遍历所有键值对，for a,b in dic.items()：

⑦. 遍历键 for i in dic.keys()

⑧ 遍历所有值 for i in dic.values() → 返回列表（自己加一下吧）

可直接用max什么的

⑬. bin() 十转二 (0b)

oct() 十转八 (0o)

hex() 十转十六 (0x)

int() 其它转十

eg int('1011', 2)=10

注意是字符串

也可以 0b

⑭ .str.find(a)，找字符

找到返索引

找不到返"-1"

⑮. str.title() 每个单词首字母大写

⑯. 反转列表 不用 reverse. ← 新建

↳ 用 list(reversed(A))

或 A[::-1]

dp / dfs.

① 不要省空间 直接开二维甚至三维

↳ 有时候开 2 个 DP 表甚至 3 个，像数列那题

② 要注意边界值与初始值

③. dp 本质：数学归纳法

④ 有时 range(0, n)，有时 range(n, 1, -1)

⑤ 有时 return max(dp)

对字典排序 按键

newdic = {k : dic[k] for k in sorted(dic)}

↳ 直接 sort 按键

加 key = lambda x : x[1]

x[1] 当成元组了

优化合集

(优化) 前缀和：预处理思想，降低时间

eg [1, 5, 9, 4, 3, 3]

↓ 先算出每一前段求和

[0, 1, 6, 15, 19, 22, 25]

O(n×m) ⇒ O(n+m)

(优化) 提前算出 n=sum[...]

(优化) 某些情况（dfs 什么的）提前 <剪枝> continue or break.

(优化) in 数组 改为 in 集合 or in 元组

(优化) @cache.

(优化) 改二分查找（前提已排好）

(优化) 双指针技术把数列倒置.

Heapq

① import heapq

heapq.heapify(列表) ⇒ 把C最小的数列到0号

heapq.heappop(表)

heapq.heappush(表)

函数

△ enumerate

↳ enumerate(iterable, start=0)

↳ 默认为0

应用: for index, thing in enumerate(表)

↓ 索引 ↓ 对应元素

---

② permutations

① import itertools

② itertools.permutations(列表, 长度)

↑ 返回列表

↳ 默认为len(列表)

具体再加个list()更好

返回含多个元组的列表

---

③. Dp ⇔ dfs + @lru_cache()

def ___.

↓

from functools import lru_cache

@lru_cache(maxsize = ___)

↳ 默认为None, 无限类

↳ 若MLE, 写4096, → 2048 → 1024

用T换M.

递归

三法则 ① n=1时 return
② 改变n且使n变小
③ 调用自身.

扩递归深度

```
import sys
sys.setrecursionlimit(n)
```

找递归定义

一般只有一个参量,但有限可以参量中可加上去

二分查找.

```
from bisect import bisect_left
```

o bisect_left(a, x) (a: 列表)

找出 a 列表中第一个大于等于x的值.

返回它的索引

~~bisect_right: 第a[i]>x 最左一边~~

bisect_right(list, num)

找第一个 list 中大于num 的索引

若无,则返回长度

o insert(索引, 值)

在该索引处插入x

dijkstr 算法

```
import heapq  堆
directions = [ ~ ]
ditu = []
```

距离(最大值) = [inf … inf ; … ; inf … inf] → (x1, y1) 为 "0"

```
queue = [ ]
heapq.heappush(queue, (0, x1, y1))   # 开始点
while heapq:  → queue.
    t, x, y = heapq.heappop(queue).
    if x = 终点_x. y = 终点_y :
        ~
        break
    for dx, dy in directions
        x', y' = x+dx, y+dy
        if (x', y') 在界内.
            ~
            if ~ :
                ...
                heapq.heappush.(queue.(s, x, y
```

求最长递减子序列 ⟺ 反转后求最长递增子序列

bisect_right(a, x, lo=0, hi=len(a))

返回 a中第一个大于x的位置索引,

如果都小于 x, 则返回 len(a)

往年题目

① 计算机最重要的软件为 OS 负责 资源管理 和 任务调度
↳操作系统

② 主存管理：8位，每个地址的访问时间与地址无关 时间局限性 → 分时技术

③ 1GHz = 1×10^9 Hz. M ⇒ 10^6. K ⇒ 10^3

④ 收发邮件协议：POP/SMT

⑤ TCP/IP：网络通信 公共协议 LAN：局域网 WAN：广域网

⑥ 浮点数 不一定完全精确

⑦ 断点 ⇒ 内存丢失

⑧ IPV6 下一代IP协议：128位二进制 技术

冯·诺依曼结构 → 总线连接
- 运算器、控制器 } CPU { CU 控制器、ALU 算术 → 核心硬件设备
- 存储器
- 输入设备
- 输出设备
（5T）

3T
- CPU<中央处理器>
- 存储器
- 输入/输出设备

⑨ 二进制乘法同十进制 除法：化十进制

```
   101
×   11
------
   101
  101
------
  1111
```

2^{10} 2^{20} 2^{30} → 约为 10^3，10^6，10^9

KB
MB
GB
TB
⇒ 全是字节 不是位
1B = 8位 → 字节

⑩ 计算题，对于位运算，

30FPS：每秒30帧

压缩率为 X：1，60X
↳ 保存颜色

⑪ 复位 与 1111 0000 进行 & or | or "∧"

⑫ 主板是主机箱 主要部分，其它都与主板相连

⑬ 声波离散化采样在时间与波形高度上有先后划分

类政治
一读便知对错

① 一个局域网内把各台计算机连在一起的是 集线器 ⟶ 交换机
各局域网连一起

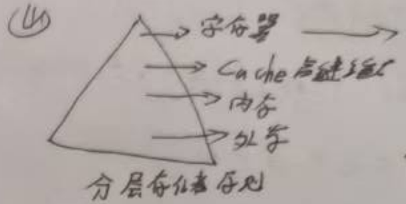
② 空间局部性原理，时间局部性

True + 1 = 2   False or 5 = 5

True * 5 = 0

③ CPU 由
- 寄存器
- 中断处理器
- 算术逻辑运算器（ALU）
- 程序控制器

数据总线
控制 ~
地址 ~ ⟶ 决定主存地址空间大小



④ 8G ↓ 内存   2T ↓ 硬盘   8核 ↓ CPU

⑤ 防火墙：内外网隔离技术

⑤ IP6：128位   IP4：32位
↳ 无符号 0,1.

⑥. Chrome 不是操作系统

⑦ 北大网为B类地址，2+14位+16位

⑧ "=1" 为 "或" 即 V

⑨ 用 文件 和 文件系统 组织管理硬件上的信息

⑩ DNS 查询网址

⑪ 摩尔律：18个月，增加 1 倍

⑫ 图灵，冯·诺伊曼

⑬ 互联网协议 TCP, UDP, HTTP, ICMP, DNS

⑭ 基本控制结构：顺序结构，分支结构，循环结构

⑮ 主板 --- 插座配置 --- 外部设备
网卡
显卡

⑯ 内存单位 KB, MB, GB, TB
2^{40}

⑫. 主存储器 主要访问方式

RAM：随机访问存储器

地址总线宽度
- 决定主存储器地址空间大小
- 所需二进制位数

指针型变量：存放存储单元地址的变量

⑬ 主存 — CPU

存储总线
- 数据总线：传输数据(双向) 一般为 32位 or 64位
- 地址总线：传输地址 } 单向
- 控制总线：传输控制信号 } 单向

⑭ 主存储器：易失特性

⑮ 正在运行的程序在计算机内存(RAM)中

⑯ CPU 负责程序的解释和运行

⑰ 图灵机：一条存储带，读写头，控制器
- 存储带：两端无限长；带小方格
- 读写头 → 可左/右移动
- 控制器 → 状态寄存器

读写头 → 读 ⟹ 找到对应指令（自己已知）⟹ 进行运动作
- ① 写入~
- ② 改状态
- ③ 左/右移

⑱ 补码：一种操作：取反加一　负数首位强制改为1

⑲ 浮点数二进制表示：单精度：32位二进制。 双精度：64位二进制

⑳ 1 word = 2 字节 Byte = 16 bit(位)

㉑. GB2312-80. 汉字16位(2字节)，首位0，存储用汉字内码：首位强改为1（每字节）

㉒ IP地址八位一段

㉓ http://主机名：端口/路径

莫尼玛王 PPT 米肯尼

① \n 为换行

字符串以"~~\n"结尾

②第一台电子计算机：ENIAC，火炮弹轨迹

③ 电子管 → 晶体管 → 集成电路 → 大规模集成电路

④ 通信协议. HTTP 超文本传输 POP3：邮件接收
TCP 传输控制 SMTP：邮件发送
IP：互联网 FTP：文件传输

⑤ IPV4：32位二进制 　北大为B类
↓
IPV6：128位

⑥ 网络分类：有线，无线.

⑥ 总线标准：PCI，PCIe，ISA …

⑦负数二进制表示 补码， eg -5：1 111,1010
(按位取反再添)

⑧ GB2312-80 中，2个字节表示一个汉字 <汉字内容>

⑨ 声音：，时长 t (s)
频率 x (Hz) 内存 = t · X · a/8 · 声道(单位为B) 立体声为双声道
量化 a (位)
↓
换算B 化为 KB，MB

⑩ 图灵机 开始，走向，…… 结尾.

当前状态 当前符号 写入符号，移动方向 新状态

⑪ CPU → 算术逻辑运算器 运算
→ 寄存器组 ：暂存，高速存储
→ 中断处理器 ：中断当前程序调用外部新程序
→ 程序控制器 ：分析指令，控制其它部件

⑫ CPU主要指标：主频(Hz)，字长，运算速度(MIPS)

⑬ 图灵机：一条存储带，读写头，控制器

⑭ 时钟频率 (单位：Hz)
指令周期：完成一指令所需时钟周期

$V = \frac{1}{C}$，注意单位

⑮ 总线宽 = 总线宽×时钟频率 / 传输周期

⑯ 主板 — 连接 — 外围设备
\ 总线、CPU, Memory

CPU — 主存
cache
↓
高速缓存

⑰ 半导体存储器 { RAM 随机存取存储器；ROM 只读存储器 }

⑱ 主存：易失特性.

计概. 笔考.

1. ASCII编码 ⟹ 从0开始算的 → chr(), ord(), upper() lower()

① 位(bit): 最小数据单位, 0 or 1.

8位 = 1字节 (byte)

→ 用一串8位二进制数表示每个符号. 注, 1位为校对

→ 8位: 1位是校对, → 0~127, 本名为2^7

A → 65. a → 97

+32 = 2^5

2. 虚拟地址空间

计算机基础架构: 处理器, 存储指令数据的内存, 输入和输出设备

冯·诺依曼架构

CPU ⟹ 算法 控制

RAM + 磁盘 ⟹ 比比存储

键盘 + 鼠标 + 显示器 ⟹ 交互

操作系统与数据
↓
共享
+
共
内核虚拟存储器

3. 图灵机.

一条存储带 + 一个控制器

↓ 双向无限延长; ↓ 双向读写头移动

4. 进制转化 $()_2$, $()_8$, $()_{10}$

十 ⟹ 其他进制, ① 整数部分: 连除 --- 取余

② 小数部分: 连乘

十六进制 A ⟹ 10

eg $(0.625)_{10}$ $\xrightarrow{\times 2}$ $(1.25)_{10}$ → $(0.25)_{10}$ $\xrightarrow{\times 2}$ (0.5) $\xrightarrow{\times 2}$ 1

1 0 1

从小数点向右写 即 $(0.101)_2$

eg $(0.638)_{10}$ ×8, ×8, ×8 ...

$(0.5044)_8$ → 位数可能变 → 看需要保留的位数

二 ⟷ 4 ⟷ 8 ⟷ 十六

很简单, eg 二 ⟷ 十六

一次只看4位, 4位 对应 1位

不要去0 尤其是"内存中表示形式"

5. 数据存储

位 — 位模式 → 一个序列几个位的位模式 → 存数: 即存为二进制前面补0. → 会溢出 $0\sim 2^n-1$

整数 — 定点表示法

— float → 浮点

8位 = 1字节

6. 二进制补码

有符号整数表示

即 n位

□□□…□

↓ n-1位

符号

0 ⇒ 非负

1 ⇒ 负

反码.

简单各个位反转

2次反转变得原数

补码（二进制补码存储）

① first. 从右复制位，直到有1被复制

② 再反转其它位

即

1000 —— 对它反码，它不动

10000000 对应 -2^8

最小的

1字节可表示 -2^8 ~ 2^8-1

→ 规定

二进制补码可作 int ⇒ n位二进制数

若 int 为负数 ⇒ 其由取补码存储

等输出时，若最左位为1，再取补码输出 ⇒ 输出十进制 int

若最左位为0 ⇒ 输出十进制 int

因此存储时用的是补码

存实数呢

3部分

① 符号 ② 位移量 ③ 定点数

1.32

→ 如 7.425

↓

[illegible] 科学计数法

基于二进制可以使用.

如 $(101\underbrace{00000\ldots0}_{n-2个})_2 = (1.01\times2^n)_2$

7. 计算机组成：三大类 CPU，主存，输入/输出设备

CPU → ALU，控制元，寄存器

连接器有 数据总线，地址总线，控制总线

SCSI

火线

USB

8. 逻辑运算

非	与	或	异或	移位运算
[illegible]	[illegible]	[illegible]	"^"	保持位数不变
!a	A∧B	∨	不同 return 1	扔几个且补几个0
	交	并	同 return 0	
	都1才1	有1⇒1	⊕	
		全0⇒0	异或	

>> 1

<< 1

9. 摩尔定律

per 18 months ⇒ 集成电路芯片晶体管数 ×2