1. map(int,\*\*) 为\*\*应用int
2. map(int,input().split())
3. ord() 把字母转化为ASCII（数字）

chr() 把ASCII转化为字母

1. 使用strip()去除可能存在的空白字符，**只要输入一个字符串的后面都加**
2. /除 //整除 %取余
3. =str()转为字符串形式
4. B[0][::-1]表示选择第一个元素，倒写该元素
5. 要在同一行输入，可以后续用split()分离，然后记得为其转化成各自的形式（eg.float、int）
6. “%.4f”
7. ls.append(n) ls.index
8. =sorted(ls,reverse=True)从大到小
9. Python 3 中的 / 运算符执行的是浮点除法

//是整数除法

1. **求最大公约数**

**法1（原理：两个正整数的最大公约数等于其中较小的数和这两个数相除余数的最大公约数）**

def gcd(a,b):

if b==0:

return a

else:

return gcd(b,a%b)

法2

import math

def gcd(a, b):

return math.gcd(a, b)

法3（暴力）

def gcd(a, b):

smaller = min(a, b)

for i in range(smaller, 0, -1):

if a % i == 0 and b % i == 0:

return i

1. f”{.........}” f"..."：这表明接下来的**字符串**是一个f-string,其中可以包含花括号{}内的**表达式**，这些表达式将被计算并格式化为字符串。

{a:02d}：在花括号内，有一个**表达式a**和一个**格式说明符:02d**。

**前面是对象，后面是处理方式**

a：这是要被格式化为字符串的变量。

:02d：这是一个格式说明符，它指定了如何格式化a的值。(使每个数都是两位数，不够补0）

1. 双for
2. ls.**extend**(...)将这个字符列表中的所有元素添加到列表ls的**末尾**
3. **extend和append区别在于，两者添加[4,5]时，append是直接在末尾加[4,5]，而extend是在末尾添加4,5（加单个整数用append不能用extend）**
4. ls.**count**(x)计算列表ls中字符x出现的次数
5. 检查0-a中的质数从（2，int(a\*\*0.5)+1）即可
6. **使用Python如果超时，可先算出其范围内最大值，然后其后面部分砍掉**
7. 让一个命令执行n次：**for i in range(n):**
8. 空行：print（）
9. **.join(...)** 把**.**前面的作为**分隔符**和**括号内**的相连接，括号内是可迭代对象（如**列表、元组**等）

eg.' '.join(['a', 'b', 'c']) 会返回 'a b c'，其中 ' ' (空格)是分隔符

1. .pop()可以去除列表的一个元素（括号内的），只能去掉一个
2. **enumerate(ls)**：enumerate是内置函数，它接受一个可迭代对象（在这个例子中是列表ls）并返回一个迭代器，迭代器产生**元组**，每个元组包含两个元素：**索引（位置）和值**。例如，**ls = [10, 20, 30]，则enumerate(ls)将产生(0, 10), (1, 20), (2, 30)。**

**Eg.** for index, fruit in enumerate(fruits,10):

1. 标记法
2. 假如一个集合里面有很多（...），len会计算（...）的个数

如果（...）是三个数可以抽象到三维立体，计算有多少个方块

1. 建立一个n\*n的列表：

arr=[[0 for j in range(n)] for i in range(n)]（全是0的n\*n列表）

1. print(arr[i][j],end=" ")
2. 参数 end=" " 指定了在打印完当前元素后不要换行，而是用空格分隔元素。默认情况下，print 函数会在打印完内容后换行，但通过设置 end 参数，我们可以改变这一行为。
3. print()当它位于内层循环的外部，每完成一行的遍历后，都会执行一次这个 print() 函数，从而在**输出中开始新的一行。（新开一行）**
4. 也可以用双for循环建立一个二维画布（要算好画布长宽），然后再挨个点替换
5. 把一个多位数字符化，它会变成一个个单位字符

eg.str(118)=’1’,’1’,’8’

1. 两两对应关系可以用字典dic={},

dic[...]=A表示键...所对应的是A（可以是数字、字符串等等）

1. .intersection(s) .前面的与括号内（列表或元组等）的**交集**
2. 计算列表连续m位数的和：sum(ls[i:i+m])
3. **ls1.sort(reverse=True, key=lambda x: (x[0], x[1]))**这个代码的意思是对列表ls1进行排序（降序），然后先根据（A,B）中的A进行排序，如果A相同，比较B，假如我们第一个A希望按照降序，但是B的时候我们希望用升序，我们只用在B前面加一个**负号**就可以实现
4. “{...}”.format(A)的意思是把A（A是字符串）处理（用{}内...的方法）
5. If ls[i]==y: else:y==ls[i] 可以让y的值随着绿色ls[i]转换
6. 假如一个列表有多个列表，想让每个列表都翻转（整体看就是水平翻转）： for i in ls1: ls2.append**(i[::-1])**
7. **.upper()小转大写 .lower()大转小写 .capitalize()大小写互转**
8. a的ASCII是97，A是65
9. 假如一个字典d={}一开始是空的，可以通过d[s]=v，如果这个s在上面的代码已经存在，就是把上面的s作为键存入，然后v是s所对应的值
10. replace既可以用来替换也可以用来剔除，使用方法：

**.replace(“...”,”...”)**

1. 寻找第一个出现位置：.find(...) ...为寻找的对象，

找不到会输出-1

1. 遍历字典时，要在其名后加.items()

eg.for score, chars in scores.items():

1. 替换一个字符串里面其中一个字符的方法，可以用切片

eg. s1=s[:i]+'K'+s[i+1:] （把s中i位置替换成了K）

1. 用if判断时，如果要用or不能i==1 or 2 or 3，应该是

i==1 or i==2 or i==3

1. lstrip去除开头即左侧，rstrip去除结尾即右侧
2. 填充：.rjust(2,’0’)意为占两个空格位，如果字符串不够两位，则往前面即**左边补0**，如果不想补，则不用写,’0’直接.rjust(2)则前面会补空格\\ljust同理，不过是向后补，即右边（虽然看起来这两个语法与其名字相反，实际上其名字的含义是rjust向右看齐，即以右为定格，反之ljust向左看齐，但实际用起来可以理解成相反）
3. 创建字典**dic=dict(zip(key,val))**，key作为键，val作为值，按顺序一一对应

Eg. #创建两个列表

**keys = [1, 2, 3] values = ['a', 'b', 'c']**

# 使用 zip() 函数将两个列表打包，并创建字典

**dic = dict(zip(keys, values))**

# 打印字典中键为 1 的值

**print(dic[1])**

1. **exec()** 函数会执行传入的字符串作为 **Python 代码，**例如你输入字符串a=3，然后你print(a)的结果就是3
2. 把列表的元素一个个以空格为间隔输出在同一行：

print(' '.join(map(str,ls)))

1. 在if判断时，要判断很多时，可以用all(……)只有括号里面全符合才是真，类似，any(……)只要括号里面有一个符合就是真

eg.if all(i%j!=0 for j in range(2,int(i\*\*0.5)+1)): （判断是否是质数）

1. 假如你要混合输出int和str类型可以print(f'{i}={b}+{j}')其中i,b,j都是int类型，=和+是str类型
2. **判断质数！！！不超时！！！函数**：

**def zs(n):**

if n <= 1:

return False

if n <= 3:

return True

if n % 2 == 0 or n % 3 == 0:

return False

**i = 5**

**while i \* i <= n:**

**if n % i == 0 or n % (i + 2) == 0:**

**return False**

**i += 6**

**return True**

1. **deque** 是 Python 标准库 collections 中的一个双端队列，它支持在**队列的两端**高效地**添加和删除**元素。在这个代码中，我们使用 deque 来实现广度优先搜索（**BFS**）算法。

使用前要导入**from collections import deque**

定义方法eg.a=deque()

其中**a.popleft()**可以从头部取出一个元素，例如x = a.popleft() ，那么就是把队列a中的第一个元素赋给x，并且队列a移除第一个元素

a.append(...)可以在末尾添加元素

1. 字符串匹配可以用**正则表达式**，即**import re**。假如你要查找字符串里面**片段的位置**，可以re.finditer('\\bAlice\\b', s)或者**re.finditer(r'\bAlice\b', s)**，这个s表示原字符串，Alice表示要查找的片段，但是要注意这里finditer返回的只是一个**迭代器**，要换成索引（即位置）要用**.start()**例子如下：

matches = re.finditer(r'\bAlice\b', s)

for match in matches:

start\_index = match.start()

或a = [m.start() for m in re.finditer(r'\bAlice\b', s)]

1. 二分查找可用**bisect**，**要先import bisect**。

l = **bisect.bisect\_left**(a, x1) 这行代码的作用是在有序列表 a 中查找 x1 应该插入的**最左位置**（即大于等于x1），假如相同，也插在相同的最前面，并将该位置的**索引**赋值给变量 l。

类似的**bisect.bisect\_right**则是插在最右位置，即相同的最后一位

举个例子：

列表[2, 3, 4, 5, 6, 7, 8]

x1=3 x2=7

都用**bisect.bisect\_left，**那么x1插在索引为1的位置，7插在索引为5的位置，那么x1到x2有3,4,5,6四个元素，即[3,7)

都用**bisect.bisect\_right，**那么x1插在索引为2的位置，7插在索引为6的位置，那么x1到x2有4,5,6,7四个元素，即(3,7]

如果我们前面用**bisect.bisect\_left**，后面用**bisect.bisect\_right，**那么x1插在索引为1的位置，7插在索引为6的位置，那么x1到x2有3,4,5,6,7五个元素，即[3,7]

1. datetime中**time**不能直接使用timedelta，只能用**datetime或date**
2. 递归思想，首先要有一个终止条件，其次把大问题化成小问题，从而不断递归
3. 前缀和可用**accumulate**

首先 **from itertools import accumulate**(导入）

r=accumulate(a)#a为要进行前缀和的列表

但是，要想前缀和表现出来，还要将r转化为列表

R=list(r)#此时R就是a的前缀和列表

**此外**，accumulate还有其他作用

**一、使用 initial 参数**

from itertools import accumulate

# 定义一个列表

numbers = [1, 2, 3, 4, 5]

# 指定初始值为 10 进行加法累积

result = accumulate(numbers, **initial=10)**

result\_list = list(result)

print(result\_list)# 输出: [10, 11, 13, 16, 20, 25]

就是它会定义初始值，从10开始累加，好比在最前面插入一个初始值然后累加

**二、指定乘法累积操作**

from itertools import accumulate

**import operator#这里要额外导入一个operator**

# 定义一个列表

numbers = [1, 2, 3, 4, 5]

result = accumulate(numbers, **func=operator.mul**)#通过**func=operator.mul指定为乘法**

result\_list = list(result)

print(result\_list) # 输出: [1, 2, 6, 24, 120]

就是先1，然后1\*2、1\*2\*3、1\*2\*3\*4、1\*2\*3\*4\*5

1. 对于幂次方求出来的数取模的时候，要看取得模是否为质数，如果为质数则对模减1。例如mod=10\*\*9+7此时不能pow(2,n,mod)而是要pow(2,n,mod-1)
2. 假如不知道输入多少组数据，可以用while True:try:except break

这样只有出现输入不符合规范，才会停止

1. 二维前缀和公式(前提是下标从1,1开始）

下标从1,1开始，有如下统一公式:

sumi,j= sumi-1,j+sumi,j-1-sumi-1,j-1+ ai,j

二维差分数组公式

上述的sum替换成a，a替换成diff,就可以得到二维差分数组

diffi,j=ai,j-ai-1,j-ai,j-1+ ai-1,j-1

1. 堆队列有**自动排序**功能，可节省时间

要先import heapq

然后heapq.heapify(a) a是列表，这里将a列表转化为堆结构

heapq.heappop(a) 从堆a中弹出最小的元素（可以赋值给别人）

heapq.heappush(a, t) 将t加入堆a

1. 假如你想在while循环结束后，如果在循环中没有break出来，然后你想进行下一项操作，你要用else，否则你直接加的话无论有没有break出来都会执行
2. 双指针可以同向扫描，也可以反向扫描
3. 二分常适用的问题：1.（有单调性的）2.最大值最小化或最小值最大化3.第K小获第K大问题

模版如下：def check(x):

#判断x是否合法, 合法返回True, 否则返回False

pass(判断合法条件)

left, right, ans = 初始化

while left <= right:

mid = (left + right) // 2

if check(mid):

ans = mid

left = mid + 1 #二选一

else:

right = mid - 1 #二选一

print(ans)

1. 二进制位运算：1.&与运算（有0得0）
2. 2.|或运算（有1得1）
3. 3.^异或运算（同0异1）4.~取反5.<<左移（每移一位想等于乘以2）6.>>右移（每移一位想等于除以2）

**这些符号可以直接使用在十进制上面，编译会自动理解为二进制的**

1. 位运算技巧：1.判断奇偶看二进制最后一位，相当于直接判断x&1

2.求x二进制第i位，用(x>>i)&1

3.将二进制的第i位设置为1，用x|(1<<i)

4.将二进制的第i位设置为0，用x&(~(1<<i))

1. 多输出可以先存到一个列表，再输出，比如：新建out=[]，然后out.append()，最后print(**\***out,sep=’\n’)后面是表示换行
2. **加速输入的代码：input=sys.stdin.readline**
3. **import sys**

**sys.setrecursionlimit**(100000)#扩栈：递归层数过大, 需要设置最大栈空间

1. 寻找字典中的键：if i in dic.**keys**():
2. 假如排序内存大、速度慢，用这个方式

import numpy as np

n,m = map(int,input().split())

l = np.fromstring(input(), dtype = np.uint32, sep=' ')

l.sort()

print(l[m])

1. a=tuple(a)将a转化为元组

b=set(b)

c=list(c)

1. 假如输入的数据不确定行数，有可能一行或者多行，可以：

a=[]

while len(a) < n:

line = input().split()#分行读取

a.extend(map(int, line))#记住是extend不是append

1. a.sort(reverse=True,key=lambda x:(**tuple(int(digit) for digit in x)**))

该代码可以让一个列表逐个按里面的元素比较

或者直接a=sorted(lists,reverse=True)

1. from functools import cmp\_to\_key#从functools导入cmp\_to\_key

def compare(x, y):#比较x+y与y+x谁大

return int(y + x) - int(x + y)

n = int(input())

a = list(map(str, input().split()))

a.sort(key=cmp\_to\_key(compare))#规则为cmp\_to\_key(compare)

s = ''.join(a)

print(s)

#cmp\_to\_key（A-B）假如A-B小于0，那么意味着（y+x）小于（x+y），即x应该排在y前面。该程序就是通过对所有元素排一遍然后确定顺序

#cmp\_to\_key(compare)：将 compare 这个传统比较函数转换为 sort() 方法可以使用的 key 函数

1. s.rindex.(a)：在s,从右边开始找a，输出的是a的位置
2. S.index(a,A)：在s，从左往右找a，开始找的位置是A
3. 完全 m 叉树（m 表示每个节点最多有 m 个子节点）有以下常见公式：

**A.节点编号相关**：

若节点编号为 i（根节点编号从 1 开始）， 它的第 j 个子节点编号为 (i - 1) \* m + j + 1（1 ≤ j ≤ m）。

父节点编号：对于节点 i（i > 1），其父节点编号为 ⌊(i - 2) / m⌋ + 1 ，其中⌊ ⌋ 表示向下取整。

**B.深度相关**：

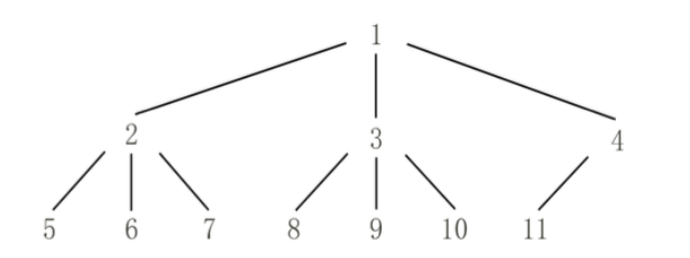
树的深度 h（层数）计算公式：h = ⌈logₘ(n \* (m - 1) + 1)⌉ ，其中n是节点总数，⌈ ⌉ 表示向上取整。

**C.每层节点数相关**：

第 d 层（根节点为第 1 层，d ≥ 1）的节点数为 m^(d - 1) 。

**D.叶子节点数相关：**

叶子节点数 L 的计算公式：L = n - ⌊(n - 1) / m⌋ ，n为节点总数 。



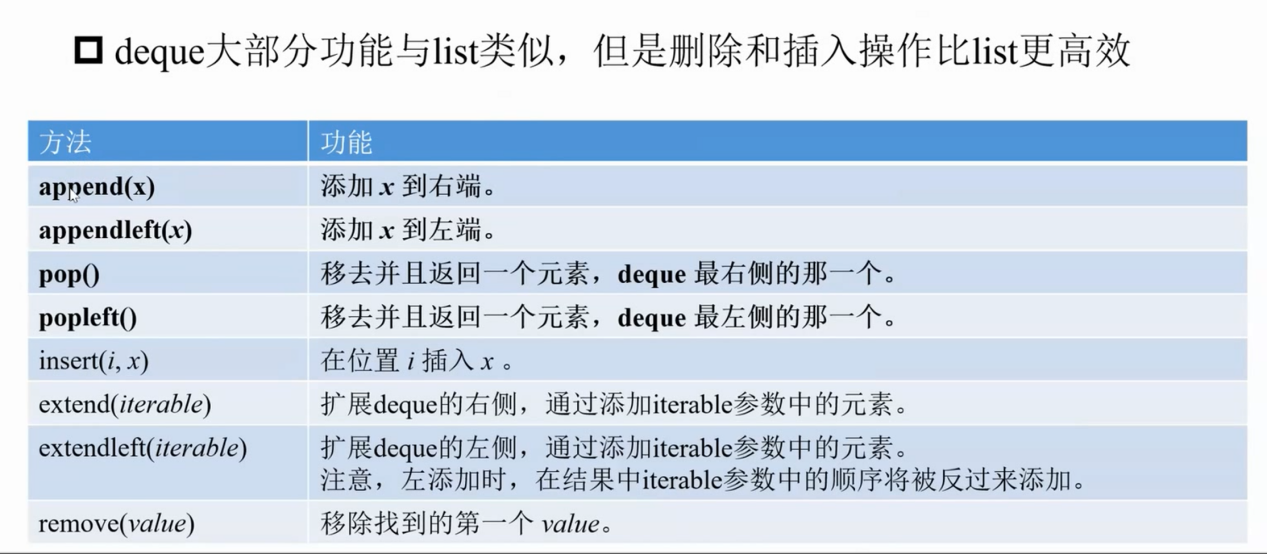
假如一个节点编号为k，那么它的子节点左端等于k\*m-m+2，右端为k\*m+1

1. 差分数组可以有效节约循环时间

当原数组在[l,r]之间要加上x，那么差分数组直接在diff[l]+=x,diff[r+1]-=x

PS:diff数组要比原来的多1个单位，不然r+1有可能超了

1. deque



1. 进制转换：

int\_to\_char = "0123456789ABCDEF"

#十进制数字x转换成k进制

def Ten\_to\_K(k, x):

ans = ""

while x != 0:

ans += int\_to\_char[x % k]

x //= k

return ans[::-1]

1. 其他进制转十进制，直接int(‘661311’,k) k表示k进制
2. file = open('example.txt', 'r')

# 读取文件的全部内容,文件名叫example.txt，r表示只读

content = file.read()

print(content)