函数：

ord()

chr()

.capitalize() 单词首字母大写

.upper()

.lower()

.isupper()

.isdecimal()判断是否是数字

pow()求幂

时间复杂度：<https://ics.uci.edu/~pattis/ICS-33/lectures/complexitypython.txt>

isdigit()判断是否为数字

l2=[x for \_,x in sorted(zip(l1,l2),key=lambda pair:pair[0])]按照l1对应顺序排l2

**format**(scores,'.1f')保留1位小数

print(‘%1f’%float(a))同上

pythontutor.com

leecode.com

常见区间问题：

1. 区间合并，数量：左端升序，右端越界加一改变界为右端
2. 不相交区间：右端升序，左端越界加一，界改为右端
3. 区间选点：右端升序，左端越界加一，界改为右端
4. 区间覆盖（数量最少）：左端升序，左端界内右端最大加一，界改为右端
5. 区间分组组内不相交：左端升序，创现有组：右端在某组区间内不能放，区间外能放且更新该区间，无处可放则更新组，用heapq（每次只用和最小的比）

Form recviz import recviz(递归可视化)

from functools import lru\_cache 递归变dp

@lru\_cache(maxsize=128)

下面一行接函数

递推时不能用list数据类型，会冲突

01背包：每个元素只能选1或0次：倒着选取

完全背包：每个元素选取次数不限，正着选取

Max 和 min的初始化不要总是无脑，有时候初始值要思考一下

From collections import defaultdict

A = defaultdict(list) 不用判断key是否在dict中

From collections import deque双端队列

Import heapq 使用堆

Heapify

heappushpop

康托尔展开：mod n！表示前面有多少个数比它小

01背包recap

for i in range(1, n + 1):

for l in range(0, W - w[i] + 1):

f[l + w[i]] = max(f[l] + v[i], f[l + w[i]])

和

for i in range(1, n + 1):

for l in range(W, w[i] - 1, -1):

f[l] = max(f[l], f[l - w[i]] + v[i])

区别，想明白，很重要

import operator   
op = {"+": operator.add, "-": operator.sub, "\*": operator.mul, "/": operator.truediv}

eval(‘1+3’) output:4

eval(‘sum[1,2,3]’) output:6

oop编程：

class fraction:

def \_\_init\_\_(self,num1,num2):

self.top = num1

self.down = num2

def \_\_add\_\_(self,other)

分子计算，分母计算

return fraction(分子，分母)

def \_\_str\_\_(self):

return str(self.up)+’/’+str(self.down)（保证可以print）

def \_\_sub\_\_(减法)

def \_\_mul\_\_(乘法)

from itertools import permutations

a = 'abc' #对字符串进行permutations排列组合

for i in permutations(a,3):

x = ''.join(i)

print (x,end=' ')

print ('\n------------------------------------')

itertools.product([a,b],repeat=3)

iterable1 **=** [1, 2]

iterable2 **=** ['a', 'b']

result **=** list(itertools**.**product(iterable1, iterable2))

from copy import deepcopy

最长上升子列

from bisect import bisect\_left

n = int(input())

l1 = list(map(int,input().split()))

dp=[float("inf")]\*n

for i in l1:

    dp[bisect\_left(dp,i)]=i

print(bisect\_left(dp,float("inf")))