3 第二讲 作业记录（一）

笔记本： 浙江大学《数据结构》
创建时间： 2025/3/18 22:43 **更新时间：** 2025/4/6 16:13
作者： panhengye@163.com
URL： https://pintia.cn/problem-sets/1873565885118418944/exam/problems/type/7?...

题目：02-线性结构2 一元多项式的乘法与加法运算

提交结果

题目	用户	提交时间
02-线性结构2	飞翔的小师弟	2025/03/18 19:53:10
编译器	**内存**	**用时**
Python (python3)	3140 / 65536 KB	15 / 200 ms
状态	**分数**	**评测时间**
答案正确	20 / 20	2025/03/18 19:53:11

评测详情

测试点	提示	内存(KB)	用时(ms)	结果	得分
0	sample换个数字	3072	15	答案正确	12 / 12
1	同类项合并时有抵消	3084	15	答案正确	4 / 4
2	系数和指数取上限，结果有零多项式	3140	15	答案正确	2 / 2
3	输入有零多项式和常数多项式	2904	15	答案正确	2 / 2

【代码】

```
# 一元多项式求和及求积

def read_data():

    nums = list(map(int, input().split()))
    n = nums[0]
    poly = {}
    for i in range(1, 2*n, 2):
        coef, exp = nums[i], nums[i+1]
        poly[exp] = coef
    return poly

def add_poly(poly1, poly2):

    result = poly1.copy()  # 使用copy()函数，避免修改原多项式

    for exp, coef in poly2.items():
        if exp in result:
            # 如果指数相同，则系数相加
            result[exp] += coef
```

```
            # 如果系数相加后为0，删除该项
            if result[exp] == 0:
                del result[exp]
        else:
            # 如果指数不同，则加一个新的项
            result[exp] = coef

    return result

def multiply_poly(poly1, poly2):

    result = {}

    # 遍历两个字典
    for exp1, coef1 in poly1.items():
        for exp2, coef2 in poly2.items():

            # 计算新项的指数和系数
            new_exp = exp1 + exp2 # n^m做乘法运算时，是将两个指数相加
            new_coef = coef1 * coef2 # 两个系数相乘

            # 将结果保存到对应的指数位置
            if new_exp in result:  # 如果指数已经存在，则相加
                result[new_exp] += new_coef
            else:  # 如果指数不存在，则直接赋值
                result[new_exp] = new_coef

    # 删除系数为0的项
    result = {exp: coef for exp, coef in result.items() if coef != 0}

    return result

def print_poly(poly):
    # 如果多项式为空，输出零多项式
    if not poly:
        print("0 0")
        return

    # 将多项式按指数从高到低排序
    sorted_poly = sorted(poly.items(), key=lambda x: x[0], reverse=True)

    # 构建输出字符串
    result = []
    for exp, coef in sorted_poly:
        # 只输出非零系数项
        if coef != 0:
            result.append(f"{coef} {exp}")

    # 连接所有项，用空格分隔 (这部分应该在循环外)
    output = " ".join(result)

    # 如果所有项都被移除(所有系数都为0)
    if not output:
        print("0 0")
    else:
        print(output)

def main():

    # 1. 已知数据分两行输入，则构建两个一元多项式
    poly1, poly2 = read_data(), read_data()

    # 2. 设计一个加法函数，求两个一元多项式的和
    poly_sum = add_poly(poly1, poly2)

    # 3. 设计一个乘法函数，求两个一元多项式的积
    poly_product = multiply_poly(poly1, poly2)

    # 4. 设计输出函数，将一元多项式输出为标准形式
    print_poly(poly_product)
```

```
    print_poly(poly_sum)

main()
```

【整体思路】
这段 Python 代码的主要功能是实现一元多项式的求和与求积操作，并将结果以标准形式输出。代码通过几个函数的协作完成这些任务，具体步骤如下：

1. **读取输入**：从用户输入中读取两个一元多项式的系数和指数信息。
2. **多项式求和**：设计一个函数来计算两个一元多项式的和。
3. **多项式求积**：设计一个函数来计算两个一元多项式的积。
4. **输出结果**：将多项式的和与积以标准形式输出。

【关键点解析1】

```
for i in range(1, 2*n, 2):
        coef, exp = nums[i], nums[i+1]
        poly[exp] = coef
```

由于观察到输入值的第一项是项数，后面是（系数 指数）成对出现的，所以通过 for 循环，以步长为 2 遍历整数列表，每次取出一个系数 coef 和一个指数 exp，并将其存储在字典 poly 中，其中指数作为键，系数作为值

【关键点解析2】

```
result = {exp: coef for exp, coef in result.items() if coef != 0}
```

使用字典推导式删除 result 中系数为 0 的项，简洁高效。否则将写成这样：

```
new_result = {}
for exp, coef in result.items():
    if coef != 0:
        new_result[exp] = coef
result = new_result
```

【关键点解析3】

```
sorted_poly = sorted(poly.items(), key=lambda x: x[0], reverse=True)
```

- items() 是字典的一个方法。调用该方法时，会返回一个由键值对组成的视图对象，每一个键值对都以元组的形式呈现
- sorted() 属于 Python 的内置函数，其作用是对可迭代对象进行排序操作，并且返回一个新的已排序列表。sorted() 函数的基本语法如下
- lambda x: x[0]是一个匿名函数，其功能是接收一个参数 x（在本代码里，x 代表一个键值对元组），并返回该元组的第一个元素（即键）

```
sorted(iterable, key=None, reverse=False)
```

- iterable：这是需要排序的可迭代对象，在本代码中为 poly.items()。
- key：它是一个可选参数，为一个函数，此函数会作用于可迭代对象的每个元素，并且依据函数的返回值来进行排序。在本代码中，key=lambda x: x[0] 表示按照每个键值对元组的第一个元素（也就是键）进行排序。
- reverse：同样是可选参数，为布尔值。若设置为 True，则进行降序排序；若设置为 False（默认值），则进行升序排序。在本代码中，reverse=True 意味着按照键进行降序排序

示例代码

```
poly = {'a': 1, 'b': 2, 'c': 3}
sorted_poly = sorted(poly.items(), key=lambda x: x[0], reverse=True)
print(sorted_poly)
```

上述代码会输出 [('c', 3), ('b', 2), ('a', 1)]，这表明字典 poly 已经按照键进行了降序排序