## САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

# ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1 по курсу «Алгоритмы и структуры данных» Тема: Сортировка слиянием. Метод декомпозиции Вариант 20

Выполнила:

Толстухина Ксения Александровна

K3139

Проверил:

Афанасьев А.В.

Санкт-Петербург

<mark>2024 г.</mark>

## Содержание отчета

Содержание отчета	2
Задачи по варианту	3
Задача №1. Сортировка слиянием.	3
Задача №5. Представитель большинства	7
Задача №7. Поиск максимально подмаслила за линейное время	10
Дополнительные задачи	14
Задача №4. Бинарный поиск	14
Задача №3. Число инверсий	17
Задача №8. Умножение многочленов	21
Вывод	23

#### Задачи по варианту

#### Задача №1. Сортировка слиянием.

Текст задачи.

Напишите программу сортировки слиянием и проверьте ее, создав несколько рандомных массивов.

```
def merge sort(nums):
    if len(nums) > 1:
        mid = len(nums) // 2
        left = nums[:mid]
        right = nums[mid:]
        merge sort(left)
        merge sort(right)
        i = j = k = 0
        while i < len(left) and j < len(right):</pre>
             if left[i] < right[j]:</pre>
                 nums[k] = left[i]
                 i += 1
             else:
                 nums[k] = right[j]
             k += 1
        while i < len(left):</pre>
            nums[k] = left[i]
             i += 1
            k += 1
        while j < len(right):</pre>
            nums[k] = right[j]
```

```
j += 1
    k += 1

return nums

with open('output.txt', 'w') as f:
    file = open('input.txt')
    n = int(file.readline())
    nums = list(map(int, file.readline().split()))
    f.write(' '.join(map(str, merge_sort(nums))))
```

Для начала я написала функцию - рекурсивный алгоритм сортировки вставками, для его реализации я делю список пополам, таким образом создаю два списка right и left. Затем я применяю этот же алгоритм для деления списков right и left, до тех пор, пока не получу список, длина которого равна 1. Дальше происходит слияние списков. Я ввожу три индекса, для правого, левого и изначального. После я перебираю элементы и правого, и левого списка, сравнивая каждый элемент между собой. Если число из списка left меньше, чем число из списка right, я вставляю его на на позицию к и увеличиваю индексы левого и текущего массива, в обратном случае в итоговый список отправляется число из правого и увеличиваются нужные индексы на 1. Этот алгоритм занесен в цикл while и будет работать до тех пор, пока в любом из списков не кончатся элементы. На случай, если в левом или правом списке еще остались элементы, я прописываю еще два цикла while, в которых просто добавляю все оставшееся в нужный список. После этого я читаю данные из файла и записываю в результирующий файл результат работы функции.

Для проверки я генерирую рандомные списки.

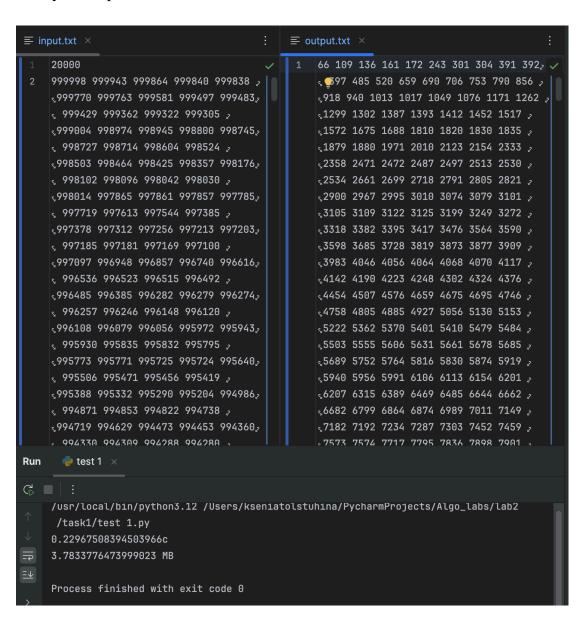
Результат работы кода на примерах из текста задачи:

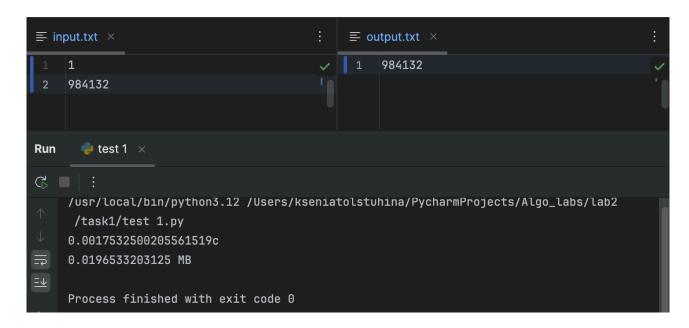
```
    input.txt
                                        ■ output.txt
                                             15 167 223 324 376 391 542 550 586 5962
    783747 554672 639792 625107 868760 2
                                             s 663 811 921 926 972 977 988 997 1040 2
    <765368 451362 921828 134099 5575922</p>
                                             1130 1164 1191 1222 1422 1438 1469 2
    844254 942025 734168 220661 2
                                             1557 1676 1729 1757 1841 2112 2144
    s350241 72971 841010 284873 900834 2
                                             s 2209 2369 2475 2500 2538 2675 2726 ¿
    490128 805438 127498 802431 589607<sub>2</sub>
                                             2796 2800 2805 2835 2838 2840 2942 2
    916171 541239 772191 208439 >
                                             2949 2987 3006 3115 3119 3127 3152 >
    842940 109297 175150 571501 1479282
                                             3234 3389 3434 3453 3507 3597 3638 2
                                             3671 3754 3873 3893 4204 4488 4522
    s630766 227520 572347 64585 963753 ¿
                                             4553 4564 4592 4595 4715 4852 5001 2
    5488971 485701 688121 786417 4853572
                                             5018 5024 5062 5087 5089 5102 5125 ¿
    s 680712 399771 404414 72810 477358<sub>2</sub>
                                             5206 5327 5353 5457 5470 5558 5630 >
    37588 510215 833627 999986 49670 »
                                             5695 5754 5783 6132 6144 6570 6576 »
    <284981 267708 758108 630125 2269802</p>
                                             6628 6774 6782 6811 6978 7260 7291 >
    s 306752 376 254741 308502 288206 2
                                             7394 7625 7628 7751 7752 7977 8095 2
    426016 11224 474900 401719 104457 >
                                             8159 8245 8266 8306 8339 8354 8376 >
    592926 234984 701900 68701 320160 <sub>2</sub>
                                             8434 8524 8557 8559 8628 8693 9106 2
    <116324 408505 791634 107249 7547122</p>
                                             9350 9384 9402 9407 9439 9511 9583
                                             9688 9743 9832 9834 9875 9877 9886 2
    s 952326 618599 608121 319157 >
    s352290 398496 615372 862717 294929<sub>2</sub>
                                             s9922 10036 10286 10344 10663 10744 ¿
    5 998068 549223 656107 88506 7319762
                                             10759 10782 10792 10810 10902 10953 »
    s 810262 747030 372136 241538 ¿
                                             s10995 11089 11095 11200 11224 11229 ¿
    5,466091 330834 812791 673482 660992<sub>2</sub>
                                             ,11332 11355 11454 11632 11638 11673 <sub>2</sub>
    . 497638 722274 956964 988693 .
                                            .11691 11857 12016 12057 12104 12128 3
    🏺 test 1 🗵
Run
    /usr/local/bin/python 3.12 \ /Users/ksenia tolstuhina/Pycharm Projects/Algo\_labs/lab 2
     /task1/test 1.py
    0.14203095802804455c
    1.320887565612793 MB
    Process finished with exit code 0
 \equiv input.txt \times
                                                     \equiv output.txt \times
                                                          6 10 48 54 55 295 329 416 435 484 513 2
      18018
      590254 13280 588086 840115 721688 2
                                                          562 610 726 781 903 1009 1069 1138 2
      679505 32001 200965 148408 513955 2
                                                          1197 1209 1217 1224 1232 1253 1266 >
      s 238564 297905 935197 753659 904152
                                                          $1468 1479 1501 1545 1689 1694 1697 2
      s 668387 566246 272303 292229 ¿
                                                          $1823 1863 1906 1990 2124 2204 2215 2
      883499 361686 371796 189807 1914742
                                                          $2906 2912 2956 2988 3027 3034 3082 2
      s 625450 945221 731746 62537 198256<sub>2</sub>
      s 370145 883539 460197 201223 ¿
                                                          s3188 3211 3368 3381 3420 3469 3473 2
      7779949 884432 7791 528082 884466 2
                                                          3490 3504 3622 3627 3728 3734 3761 2
      s411844 719363 129168 474399 258663<sub>2</sub>
                                                          s3785 3830 3847 3899 4066 4114 4162 2
      s 317815 77651 414913 8466 166243 >
                                                          <4198 4249 4355 4449 4504 4576 4631 >
                                                          34772 4906 4908 4957 4977 5027 5031 2
      691104 757480 648838 100589 36175 >
                                                          s5046 5085 5120 5141 5175 5391 5461 2
      944923 885473 103877 762920 423091<sub>2</sub>
      s 267943 189744 60838 501781 593907<sub>2</sub>

    5511 5619 5620 5693 5703 5890 5894 
    2

      207038 119539 928339 60475 2806452
                                                          5901 5909 5974 6126 6133 6181 6199 2
      s 55018 168539 205951 642316 219155<sub>2</sub>
                                                          s6212 6258 6334 6403 6432 6522 6565 ¿
      s6572 6578 6610 6639 6661 6663 6715 ¿
      s 560801 125759 970490 189583 ¿
                                                          s6760 6834 6893 6895 6960 6973 7057 2
      226411 714310 385482 960451 2492762
                                                          7086 7093 7104 7137 7255 7387 7631 2
      611667 297705 820737 144922 >
                                                          7633 7664 7695 7701 7731 7749 7775 >
      57791 7808 7851 7952 7957 8031 8072 2
      s 587609 793442 270661 961670 2
                                                          s8242 8272 8275 8436 8466 8586 8716 2
      634331 861369 517923 577356 3938072
                                                          8747 8750 8764 8805 8929 9012 9049 2
      201859 599462 154332 222777 2
                                                          -9084 9114 9278 9301 9360 9465 9538 -
Run
        🏓 test 1 🛛 🔻
      /usr/local/bin/python3.12 /Users/kseniatolstuhina/PycharmProjects/Algo_labs/lab2
       /task1/test 1.py
      0.3264776249998249c
      1.7098779678344727 MB
```

Process finished with exit code  $\theta$ 





	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.001753250020556151 9c	0.0196533203125 MB
Пример из задачи	0.14203095802804455c	1.320887565612793 MB
Пример из задачи	0.3264776249998249c	1.7098779678344727 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.22967508394503966c	3.7833776473999023 MB

Вывод по задаче: Алгоритм работает достаточно быстро, как в среднем случае, так и наихудшим случае.

#### Задача №5. Представитель большинства

Текст задачи.

Правило большинства - это когда выбирается элемент, имеющий больше половины голосов. Допустим, есть последовательность A элементов a1, a2, ...Оп, и нужно проверить, содержит ли она элемент, который появляется больше, чем n /2 раз.

```
def majority(a):
    dic = {}
    for i in a:
        dic[i] = dic.get(i, 0) + 1

    for k, v in dic.items():
        if v > len(a) // 2:
            return 1
    return 0

with open('output.txt', 'w') as f:
```

```
file = open('input.txt')

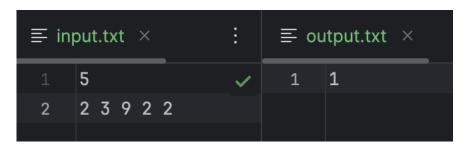
n = int(file.readline())

list_input = list(map(int, file.readline().split()))

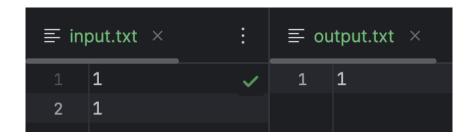
f.write(str(majority(list_input)))
```

Я использую словарь для записи количества чисел, потом прохожу по его элементам и если какое либо значение больше чем n // 2, то возвращаю 1, если таких значений нет, 0.

Результат работы кода на примерах из текста задачи:



≡ in	put.txt ×	:	<b>≡</b> οι	ıtput.txt ×	
1	4	~	1	0	
2	1 2 3 4				



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000211582984775304 8 c	0.018624305725097656 MB
Пример из задачи	0.000185542041435837 75 c	0.01867961883544922 MB
Пример из задачи	0.000286249967757612 47 c	0.018675804138183594 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.1273922499967739 с	10.944957733154297 MB

Вывод по задаче: данная программа работает за линейное время и не превышает лимит памяти.

#### Задача №7. Поиск максимально подмаслила за линейное время

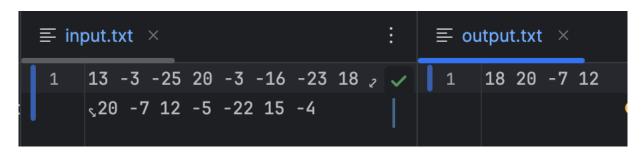
Текст задачи.

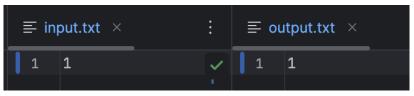
Можно найти максимальный подмассив за линейное время, воспользовавшись следующими идеями. Начните с левого конца массива и двигайтесь вправо, отслеживая найденный к данному моменту максимальный подмассив. Зная максимальный подмассив массива A[1..j], распространите ответ на поиск максимального подмассива, заканчивающегося индексом j+1, воспользовавшись следующим наблюдением: максимальный подмассив массива A[1..j+1] представляет собой либо максимальный подмассив массива A[1..j], либо подмассив A[i..j+1] для некоторого  $1 \le i \le j+1$ . Определите максимальный подмассив вида A[i..j+1] за константное время, зная максимальный подмассив, заканчивающийся индексом j

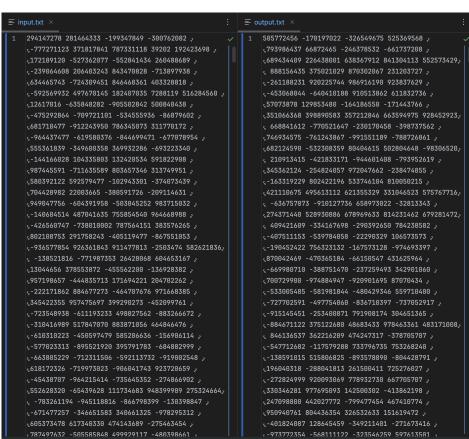
```
def find max(a):
    \max sum, summ = 0, 0
    for i in range(len(a)):
        if summ == 0:
            start indx = i
        summ += a[i]
        if max sum < summ:</pre>
            max sum = summ
            end indx = i
        if summ < 0:
            summ = 0
    return a[start indx:end indx + 1]
with open('output.txt', 'w') as file:
    f input = list(map(int,
open('input.txt').readline().split()))
    a = find max(f input)
    file.write(' '.join(map(str, a))
```

Ввожу две переменные максимальную сумму и текущую. Прохожу по файлу и в случае, если текущая сумма равна нулю, то запоминаю начальный индекс(с которого будет в дальнейшем начинаться счет). Далее каждую итерацию я прибавляю к текущей сумме элемент. Далее идет проверка на максимальность, если максимальная сумма меньше текущей, то я обновляю данные макс суммы и запоминаю индекс, как конечный. Если сумма меньше нуля, то она не может быть наибольшей, поэтому я обнуляю текущую сумму и счет начинается сначала. В конце я вывожу массив от начального, до конечного индекса(с максимальной суммой)

Результат работы кода на примерах из текста задачи:







	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	5.8833000366576016e-0 5 секунд	0.005827903747558594 MB
Пример из задачи	9.850000060396269e-05 секунд	0.006003379821777344 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.07804337500056135 секунд	3.817852020263672 MB

Вывод по задаче: алгоритм работает на больших значениях достаточно быстро

#### Дополнительные задачи

#### Задача №4. Бинарный поиск

Текст задачи.

В этой задаче вы реализуете алгоритм бинарного поиска, который позволяет очень эффективно искать (даже в огромных) списках при условии, что список отсортирован. Цель - реализация алгоритма двоичного (бинарного) поиска.

```
file = open('input.txt')
n, mass = int(file.readline()), list(map(int,
file.readline().split()))
k, mass find = int(file.readline()), list(map(int,
file.readline().split()))
def binary search(mas, what find):
    1 = 0
    r = len(mas) - 1
    while 1 <= r:
        mid = (1 + r) // 2
        if what find == mas[mid]:
            return mid
        elif what find < mas[mid]:</pre>
            r = mid - 1
        else:
            1 = mid + 1
    return -1
```

```
def play(mas, maas_find):
    return [binary_search(mas, maas_find[i]) for i in
range(len(maas_find))]

with open('output.txt', 'w') as f:
    f.write(' '.join(map(str, play(mass, mass_find))))
```

Внутри функции ввожу две переменные - левая и правая граница. Далее, до тех пор пока мы не добрались до конца списка, мы находим середину списка, в случае, если элемент по середине равен искомому элементу, возвращаем индекс этого элемента. Если элемент посередине больше, чем искомый, то мы сдвигаем правую границу до серединного индекса, в обратном случае тоже самое с левой границей. Далее я в функции play перебираю элементы списка чисел, которые нужно найти и записываю в список найденные индексы, либо -1(если элемента в списке нет).

Результат работы кода на примерах из текста задачи:

≡ in	put.txt ×	:	≡ output.txt ×
1	5	~	1 20-10-1
2	1 5 8 12 13		
3	5		
4	8 1 23 1 11		

```
      ≣ input.txt ×
      ⋮
      ≣ output.txt ×

      1
      1
      •
      1
      0

      2
      1
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
      0
```



	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000125499999739986 38 секунд	0.01835346221923828 MB
Пример из задачи	9.595899973646738e-05 секунд	0.006002426147460937 5 MB
Верхняя граница диапазона значений входных данных из текста задачи	1.2827624579995245 секунд	12.541053771972656 MB

Вывод по задаче: алгоритм бинарного поиска работает быстрее и эффективнее других алгоритмов поиска.

#### Задача №3. Число инверсий

Текст задачи.

Инверсией в последовательности чисел A называется такая ситуация, когда i < j, а A[i] > A[j]. Количество инверсий в последовательности в некотором роде определяет, насколько близка данная последовательность к отсортированной. Например, в сортированном массиве число инверсий равно 0, а в массиве, сортированном наоборот - каждые два элемента будут составлять инверсию (всего n(n-1)/2).

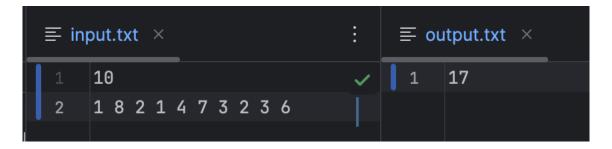
Дан массив целых чисел. Ваша задача - подсчитать число инверсий в нем. Подсказка: чтобы сделать это быстрее, можно воспользоваться модификацией сортировки слиянием.

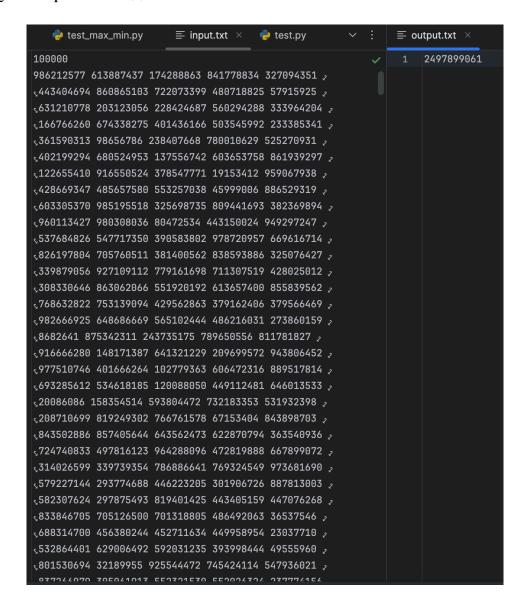
```
def search(mass, copy mass, 1, mid, r):
    i = k = 1
    j = mid + 1
    count inf = 0
    while i <= mid and j <= r:</pre>
        if mass[i] <= mass[j]:</pre>
            copy mass[k] = mass[i]
            i += 1
        else:
            copy mass[k] = mass[j]
            j += 1
            count inf += (mid - i + 1)
        k += 1
    while i <= mid:
        copy mass[k] = mass[i]
        k += 1
```

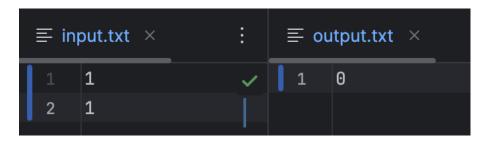
```
i += 1
    for i in range(l, r + 1):
        mass[i] = copy mass[i]
    return count inf
def search_inversions(mass, copy_mass, 1, r):
    if r <= 1:
       return 0
   mid = (1 + r) // 2
    count inf = 0
   count_inf += search_inversions(mass, copy_mass, 1, mid)
    count inf += search inversions(mass, copy mass, mid +1,r)
    count inf += search(mass, copy mass, 1, mid, r)
    return count inf
with open('output.txt', 'w') as f:
    file = open('input.txt')
   n = int(file.readline())
   nums = list(map(int, file.readline().split()))
    nums copy = nums.copy()
    f.write(str(search inversions(nums, nums copy, 0, n -
```

Функция search в которую я передаю массив, вспомогательный массив, левую границу, середину, правую границу. Дальше, таким же алгоритмом, как в задаче 1 прохожу по массиву, сортирую его и в отдельную переменную записываю число инверсий на каждом шаге, если они есть. Циклом for переношу значения из вспомогательного массива в основной. Функция возвращает целое число - инверсий. Вторая функция делит массив на подмассивы и рекурсивно находит инверсии в каждом.

Результат работы кода на примерах из текста задачи:







	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000226374999328982 08 секунд	0.018799781799316406 MB
Пример из задачи	0.000113666999823180 96 секунд	0.006013870239257812 5 MB
Верхняя граница диапазона значений входных данных из текста задачи	2.17517908300033 секунд	8.967170715332031 MB

Вывод по задаче: алгоритм сортировки с использованием сортировки слиянием работает намного быстрее, чем алгоритм перебора элементов массива в двух циклах.

#### Задача №8. Умножение многочленов

Текст задачи.

Даны два многочлена. Нужно получить их произведение.

Листинг кода.

```
def mul_polynomials(a, b, n):
    res = [0] * (n + n - 1)
    for cof_a in range(n):
        for cof_b in range(n):
            res[cof_a + cof_b] += a[cof_b] * b[cof_a]
    return res

with open('output.txt', 'w') as f:
    file = open('input.txt')
    n = int(file.readline())
    a, b = list(map(int, file.readline().split())),
list(map(int, file.readline().split()))
    f.write(' '.join(map(str, mul_polynomials(a, b, n))))
```

Текстовое объяснение решения.

Прохожу по коэффициентам и в результирующий список записываю нужные коэф(каждый член одного, на каждый член второго, затем складываю полученные произведения)

Результат работы кода на примерах из текста задачи:

≡ in	put.txt ×	:	≡ ou	tput.tx	t ×	
1	3	<b>~</b>	1	15 13	33	9 10
2	3 2 5					
3	5 1 2					

	Время выполнения	Затраты памяти
Пример из задачи	9.604099977877922e-05 секунд	0.00600433349609375 MB

Вывод по задаче: программа работает корректно

### Вывод

В этой лабораторной работе я научилась писать различные алгоритмы, включая алгоритм сортировки слиянием и метод декомпозиции, так же вспомнила такие алгоритмы, как бинарный поиск или поиск максимального подмассива.