

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ  
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ  
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №2  
по курсу «Алгоритмы и структуры данных»  
Тема: Тема работы  
Вариант 11

Выполнил:  
Тимаков Егор  
К3141

Проверила:  
Артамонова В.Е.

Санкт-Петербург  
2024 г.

## **Содержание отчета**

<b>Содержание отчета</b>	<b>2</b>
<b>Задачи по варианту</b>	<b>3</b>
Задача №1. Сортировка слиянием	3
Задача №3. Число инверсий	8
Задача №4.(Замена задачи №7). Бинарный поиск	12
<b>Вывод</b>	<b>15</b>

## Задачи по варианту

### Задача №1. Сортировка слиянием

1. Используя псевдокод процедур Merge и Merge-sort из презентации к Лекции 2 (страницы 6-7), напишите программу сортировки слиянием на Python и проверьте сортировку, создав несколько случайных массивов, подходящих под параметры:

- Формат входного файла (input.txt). В первой строке входного файла содержится число  $n$  ( $1 \leq n \leq 2 \cdot 10^4$ ) — число элементов в массиве. Во второй строке находятся  $n$  различных целых чисел, по модулю не превосходящих  $10^9$ .

- Формат выходного файла (output.txt). Одна строка выходного файла с отсортированным массивом. Между любыми двумя числами должен стоять ровно один пробел.

- Ограничение по времени. 2сек.

- Ограничение по памяти. 256 мб.

2. Для проверки можно выбрать наихудший случай, когда сортируется массив размера 1000,  $10^4$ ,  $10^5$  чисел порядка  $10^9$ , отсортированных в обратном порядке; наилучший, когда массив уже отсортирован, и средний. Сравните, например, с сортировкой вставкой на этих же данных.

3. Перепишите процедуру Merge так, чтобы в ней не использовались сигнальные значения. Сигналом к остановке должен служить тот факт, что все элементы массива L или R скопированы обратно в массив A, после чего в этот массив копируются элементы, оставшиеся в непустом массиве. или перепишите процедуру Merge (и, соответственно, Merge-sort) так, чтобы в ней не использовались значения границ и середины - p, r и q.

Код:

```
import os
import psutil
from datetime import datetime

def merge(leftArray, rightArray):
    answer = []
    i = 0
```

```

r = 0
while l < len(leftArray) and r < len(rightArray):
    if leftArray[l] < rightArray[r]:
        answer.append(leftArray[l])
        l += 1
    else:
        answer.append(rightArray[r])
        r += 1
if l < len(leftArray):
    answer += leftArray[l:]
if r < len(rightArray):
    answer += rightArray[r:]
return answer

def mergeSort(array):
    if len(array) == 1:
        return array
    middle = len(array) // 2 # делим массив на 2 для разбиения
    на части и съём в merge
    return merge(merge_sort(array[:middle]),
merge_sort(array[middle:]))

process = psutil.Process(os.getpid())
starttime = datetime.now()

InputFile = open('input.txt')
length = InputFile.readline()
string = InputFile.readline().split()
InputFile.close()
array = []
for i in string:
    array.append(int(i))

OutputFile = open('output.txt', 'w')
OutputFile.write(str(merge_sort(array))[1:-1].replace(", ", ""))

print(datetime.now() - starttime)

```

```
print(process.memory_info().rss / (1024 * 1024))
```

Объяснение:

Для реализации этого алгоритма я написал две функции: merge и mergeSort. в merge мы сравниваем элементы и меньший суем в итоговый массив. в mergeSort мы попросту делим массив и передаем его в функцию merge.

Результат работы кода на максимальных значениях:

input.txt:

```
1 20000
946193558 48684695 -864770688 687225092 949617685 501985907 301738734 -509983384 277191785 947181250 -320965922 -720987462 -838338566 651502836 -594202065
-166845459 725611234 842050668 286527186 -924647289 -679548085 -308272590 856339768 -195604823 861562176 29268669 -892492554 -127127143 -690177447 189158537
-882984272 -912545657 -601783466 642064823 229405771 569471294 -447698143 -270585735 -610343500 -776022998 -832812075 867680991 660577695 883986481 -361554186
631871132 -765609280 170758705 -532861388 -136866357 -999523922 -245859952 435099824 -992371142 -468017682 678742541 863191345 344789624 -212880236 -604504677
186578331 -915577889 265439458 -217357753 921093329 937199544 868907163 -629767240 -473064284 657163316 -251704357 757473575 620322943 -455788246 320944440
-964325877 28802653 -501410673 860017969 116353482 -382660430 383603734 -941581775 -322480350 -700069940 107832642 -669343245 88736737 -509473857 -64189760
732556463 694668741 -891083371 487448115 36102835 185059619 -644367635 297541316 403067178 -166331491 -679801328 577121806 -297723221 -487107292 -278799995
-514462810 540155572 -218294392 -153673807 -509875146 -718178877 -519059634 672040126 -975285627 13796104 -222265499 -987320116 809579116 422097947 81316627
670042140 783762516 300539334 281316214 884928918 346862396 833414386 349469267 -222847391 -678153402 -957901551 190684517 -34796275 -711381059 -424290574
486506833 521294749 -37635323 601939551 -57286431 -711632689 104959351 -301642649 -196676586 182575850 -782585905 808823238 -724493693 353439514 -405208142
-867100516 913968001 696470347 -117384166 880779446 -715240877 -832393446 -750186261 123565452 -763434102 -877900070 -576685885 524456966 373047975 367284305
773389542 98963075 -428792811 -977391186 748240996 -836622257 -560763898 787294197 -676746632 -784399233 -490458489 94326530 -286335389 -771556485 -973651099
955863128 -274981165 -394123128 -35375709 702635629 -527461966 921737596 68658399 75594293 -563559643 586419615 -129972388 -123486468 171560041 588384218 290829722
633670617 -493925801 190187062 567705685 209219377 -945809906 866222583 -136261100 315099917 -208970291 -852214991 419041468 -238325660 335748873 175114830
-299389737 -584718443 877491999 140909074 -898787895 13802637 327162399 -565701332 223520128 674851889 -72338725 622080714 -413981670 984275104 616525800 95544010
948376712 -860866574 770457523 -336020279 -455374596 -3390423 929428860 65039089 151504993 48830015 334802489 -371385227 -644654098 721662110 257527974 72014309
-983946208 -250535889 -12534864 276155104 298409971 -674207859 -213320130 -151497377 506195796 -188061514 -313651944 445788711 -975800686 603424651 475626307
-845691249 -693112766 -889395397 -335643662 -868984776 556070137 775490588 -219451011 -420831204 -27343827 117297298 631440610 249431850 -682476837 423745260
-562471943 -719475708 -55847187 -385624236 757966202 -949313380 707518095 286468832 -779066735 -291288612 535966994 -705579328 -955162712 605962274 -133572084
811228529 4763852 -389374364 -223696376 769630981 459403218 -754234019 986627993 480695735 -394551841 89561945 692391055 -104663836 -894444636 793921488
-451038339 -442935825 695004709 -136714695 122540700 -584221760 -666074128 -54566828 -480170347 -529019213 -528084447 90084521 671470571 923885174 325425621
963464639 -306568499 630841517 -849267382 -824683439 -282186537 368030417 976323144 -981631422 -23648413 344683511 687415667 700790163 711370120 460923050 -33599701
-259826633 932177058 -631075785 -855985919 875164988 126663000 -578226216 195468040 -761991614 597574668 210861452 -521698329 -371055893 -656857888 -964106488
157193194 182948292 802510245 -222771963 17463460 41647036 -723009834 -873296062 795255553 -864570072 598612593 258268562 -39747587 322514603 659134046 -173040008
-724318153 804793696 163689016 -443463833 752643081 -840849340 526293973 -369407267 921022942 -824942252 -46330668 312036692 13991379 829239184 324366121 312987206
```

output.txt:

```
1 -999986647 -999951354 -999923212 -999781220 -999735929 -999579676 -999523922 -999386232 -999332913 -998869039 -998770386 -998703344 -998601673 -998559306 -998554345
-998269211 -998277071 -998268409 -998059379 -997998518 -997463344 -997198110 -997139989 -996941973 -996936322 -996836386 -996790634 -996772975 -996772217
-996749338 -996740951 -996740861 -996504685 -996377628 -996259420 -995681730 -995618966 -995563627 -995512794 -995406321 -995336276 -995226575 -994799922
-994665020 -994662120 -994471095 -994414541 -994393937 -994308999 -994260116 -994207697 -994195302 -994186930 -994184127 -994088177 -993998541 -993742459
-993719296 -993643602 -993359507 -993284542 -993208631 -993136313 -993103688 -993017428 -992670240 -992573101 -992542361 -992371142 -992351284 -992286111
-992139817 -992026072 -992023628 -991559932 -991550014 -991468652 -991359570 -991255217 -991122475 -991121321 -991014205 -990951605 -990949378 -990924911
-990917365 -990814828 -990728123 -990551012 -990489853 -990473180 -990438972 -990210387 -990187046 -990071916 -990014398 -990010684 -989839479 -989464776
-989117617 -988886989 -988764807 -988696413 -988692475 -988253903 -988223339 -988043742 -988016624 -9879909306 -987650810 -987531547 -987405828 -987404286
-987361573 -987320116 -987309406 -987274216 -986943629 -986786239 -986695567 -986646084 -986503101 -986363954 -986231910 -986194601 -986112672 -986097013
-985845017 -985713413 -985712469 -985562698 -985521749 -985474276 -985335870 -985262318 -985130103 -984917739 -984732526 -984584967 -984566285 -984467540
-984219068 -983984914 -983964208 -983905777 -983868293 -983801834 -983668979 -983342391 -983127839 -983169323 -983123060 -983118660 -982960096 -982731108
-982636494 -982206544 -982126122 -982106816 -981901654 -981683961 -981631422 -981444356 -981426639 -981417066 -981339344 -981217827 -981011380 -980879210
-980712634 -980684443 -980580373 -980285032 -980194284 -980170064 -980095037 -979995895 -979989859 -979985780 -979946547 -979901835 -979845356 -979774570
-979583064 -979388160 -979331870 -979313582 -979235051 -979229489 -978917506 -978879726 -978752319 -978674331 -978649936 -978648320 -978642792 -978603047
-978566673 -978519412 -978418737 -978250553 -978220589 -978140313 -978087827 -978082652 -978064778 -977727181 -977545014 -977391626 -977391186 -977346336
-977304895 -977280251 -977238813 -977079049 -976961811 -976905862 -976905786 -976846335 -976798962 -976667328 -976583432 -976559311 -976443525 -976093192
-975874120 -975813520 -975800686 -975795437 -975658549 -975540261 -975455093 -975285627 -975130429 -975027009 -974797366 -974794724 -974787632 -974775812
-974707040 -974567438 -974284582 -974274305 -974252037 -974159061 -973902897 -973867527 -973744061 -973651099 -973427179 -973397022 -973288112 -973219052
-973203883 -973051415 -972980065 -972963007 -972959134 -972693224 -972445799 -972375754 -972361123 -972344477 -972328626 -972259180 -972256420 -971937756
-971899537 -971836113 -971833068 -971794117 -971780004 -971242341 -971080345 -971077233 -971062301 -971035842 -971009571 -970977179 -970944014 -970881596
-970845296 -970818090 -970800426 -970744835 -970513830 -970507238 -970422910 -970409145 -970387769 -970308687 -970120430 -970111432 -969937637 -969893388
-9698881208 -969736701 -969606475 -969512681 -969500564 -969384331 -969167217 -969082488 -968987557 -968973588 -968950332 -968851598 -968831403 -968810015
-968698721 -968461253 -968432236 -968295266 -968126848 -967997283 -967909202 -967878677 -967580694 -967559294 -967507232 -967234605 -967204368 -967117946
-967064136 -967063863 -966829305 -966811686 -966744794 -9666645730 -966412151 -966382704 -966367017 -966297902 -966190701 -966182310 -965937527 -965834151
-965793238 -965799047 -965738022 -965636993 -965342662 -965280427 -965258579 -964948243 -964902542 -964867549 -964834062 -964778908 -964592555 -964482027
-964370155 -964325877 -964252961 -964247890 -964177801 -964165946 -964157497 -964146978 -964117683 -964106488 -964083662 -963686858 -963575600 -963571940
```

Результат работы кода при минимальных значениях:

1	1
2	10

```
1 10
```

input.txt:

[illegible]

6

1	-999988242 -999965016 -999962124 -999951608 -999945500 -999850502 -999848130 -999803421 -999767091 -999755721 -999750145 -999696759 -999635484 -999613542 -999600022
2	-999576294 -999575604 -999558605 -999517482 -999490021 -999459042 -999455547 -999454666 -999454413 -999428073 -999389816 -999378017 -999359565 -999351327
3	-999340655 -999296969 -999296647 -999286142 -999266243 -999242187 -999210791 -999189997 -999134412 -999134180 -999125340 -999115845 -999106592 -999091042
4	-999059259 -998968426 -998954572 -998879335 -998877607 -998878088 -998848550 -998836220 -998825167 -998816234 -998802788 -998776983 -998772569 -998736492
5	-998734786 -998684163 -998680546 -998663012 -998604939 -998597131 -998595608 -998583772 -998537765 -998525324 -998521307 -998472840 -998449632 -998443747
6	-998428614 -998420414 -998398186 -998353504 -998320431 -998317926 -998313233 -998300155 -998268386 -998258053 -998256621 -998238545 -998225715 -998223470
7	-998222840 -998184556 -998180169 -998175010 -998163031 -998124613 -998116222 -998097918 -998082250 -998075279 -998064741 -998007564 -997998042 -997922044
8	-997895098 -997883480 -997877344 -997820641 -997818935 -997807116 -997782566 -997780301 -997763557 -997754301 -997734806 -997722421 -997716850 -997683581
9	-997677519 -997654005 -997637045 -997587672 -997573517 -997566065 -997505571 -997502479 -997492371 -997485709 -997439976 -997437873 -997352333 -997344836
10	-997234153 -997176074 -997137546 -997137394 -997093042 -997091068 -997090745 -996997785 -996978470 -996970234 -996902501 -996866000 -996835669 -996833751
11	-996833608 -996817809 -996808740 -996790948 -996790222 -996700971 -996697773 -996695480 -996687901 -996664712 -996642758 -996628849 -996564261 -996535196
12	-996533284 -996503403 -996487802 -996484483 -996442209 -996427650 -996422642 -996407067 -996404239 -996383513 -996382278 -996379763 -996355551 -996353454
13	-996303764 -996271462 -996262561 -996243143 -996209812 -996197972 -996141350 -996134562 -996125219 -996090039 -996086824 -996086394 -996083518 -996067523
14	-996059905 -996059474 -996045473 -996023493 -995972528 -995969908 -995935158 -995919598 -995916097 -995906436 -995900904 -995874472 -995849675 -995842337
15	-995832683 -995830184 -995821483 -995813088 -995803385 -995771732 -995696962 -995695423 -995667500 -995649034 -995641663 -995631999 -995622426 -995610892
16	-995533703 -995510886 -995504423 -995476970 -995448001 -995442966 -995438025 -995436553 -995421436 -995418795 -995418692 -995399178 -995385249 -995373028
17	-995345393 -995297067 -995265981 -995231739 -995227682 -995181983 -995173362 -995141928 -995129557 -995123545 -995082360 -995077873 -994981833 -994973369
18	-994955995 -994904041 -994879228 -994870145 -994833610 -994810873 -994798433 -994788306 -994781946 -994724598 -994723153 -994712466 -994701906 -994687535
19	-994683647 -994661178 -994655282 -994649419 -994646225 -994636903 -994621589 -994575436 -994564367 -994554355 -994545815 -994516142 -994503540 -994496503
20	-994478889 -994458662 -994441273 -994434008 -994405831 -994400874 -994335252 -994324060 -994323453 -994321342 -994239132 -994191369 -994171877 -994170597
21	-994161019 -994159860 -994106881 -994095734 -994092920 -994083849 -994073286 -994049208 -994019915 -994017031 -993992776 -993990737 -993983134 -993965790
22	-993963639 -993916886 -993909260 -9939084941 -993866840 -993861059 -993832247 -993830796 -993823076 -993817392 -993796604 -993776158 -993748636 -993734649
23	-993707618 -993637629 -993626271 -993622797 -993615764 -993597899 -993589990 -993589861 -993573408 -993557126 -993540597 -993532256 -993505603 -993444804
24	-993432680 -993428846 -993418261 -993416729 -993414969 -993396484 -993371466 -993366670 -993366477 -993360386 -993355032 -993352357 -993345122 -993312067
25	-993305169 -993294539 -993293922 -993273565 -993258583 -993251941 -993221908 -993204982 -993201752 -993197051 -993185240 -993156676 -993147118 -993137254
26	-993108653 -993096555 -993038391 -993030377 -993021586 -992995832 -992994596 -992954211 -992907164 -992893655 -992887714 -992840335 -9928335401 -992802294

## Проверка задачи на время выполнения и затраты памяти

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.001	15
Верхняя граница диапазона значений входных данных из текста задачи	0.034	16
При $10^5$ элементов отсортированных в обратном порядке.	0.260	26

Вывод по задаче:

При решении этой задачи пришлось изучить алгоритм сортировки слиянием

### 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 отсортированном массиве число инверсий равно 0, а в массиве, отсортированном 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.

- Формат входного файла (input.txt). В первой строке входного файла содержится число  $n$  ( $1 \leq n \leq 10^5$ ) — число элементов в массиве. Во второй строке находятся  $n$  различных целых чисел, по модулю не превосходящих  $10^9$ .

- Формат выходного файла (output.txt). В выходной файл надо вывести число инверсий в массиве.

- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.

Код (Без рекурсии):

```
import os
import psutil
from datetime import datetime
```

```
def merge(leftArray, rightArray):
    answer = []
    global inv
    l = 0
    r = 0
    while l < len(leftArray) and r < len(rightArray):
        if leftArray[l] < rightArray[r]:
            answer.append(leftArray[l])
            l += 1
            inv += 1
        else:
            answer.append(rightArray[r])
            r += 1
```



```

if l < len(leftArray):
    answer += leftArray[l:]
if r < len(rightArray):
    answer += rightArray[r:]
return answer

```

```

def merge_sort(array):
    if len(array) == 1:
        return array
    middle = len(array) // 2
    return merge(merge_sort(array[:middle]), merge_sort(array[middle:]))

```

```

process = psutil.Process(os.getpid())
startTime = datetime.now()

```

```

InputFile = open('input.txt')
length = int(InputFile.readline())
string = InputFile.readline().split()
InputFile.close()
array = []
for i in string:
    array.append(int(i))

```

```

OutputFile = open('output.txt', 'w')
inv = 0
merge_sort(array)
OutputFile.write(str(inv))
OutputFile.close()

```

```

print(datetime.now() - startTime, process.memory_info().rss / (1024 * 1024))

```

Объяснение:

В данном алгоритме мы модифицировали алгоритм сортировки слиянием добавляя в функции Merge глобальную переменную inv, которая считает число инверсий.

Результат работы программы на примере из задачи:

input.txt:

1	10
2	1 8 2 1 4 7 3 2 3 6

output.txt:

1	17
---	----

Результат работы кода на максимальных значениях:

input.txt:

1	100000
2	886565522 -587626674 -758567816 56070078 480001815 614656780 -463454809 585268171 221195139 -759436206 -281059728 633302777 143500614 -943568968 197225948 137816545 , 292840502 -422627694 273523329 165005005 938648689 -211871863 -692206893 150267476 480708942 816519155 -813669762 675005442 442874896 -755182288 -108378436 , 730456169 292587984 -730712575 -488743707 -953058463 477188209 -315322331 -659660991 770022931 156570934 -51539113 -992027982 -727208476 -983471134 449059556 , 173683380 133319953 -748948873 -446718870 653408212 -290788541 -975354523 -41150011 -725673755 37195907 -363594578 -556728081 138502465 -120502823 796425500 , 993240205 602393818 -918207939 456253556 -777720825 205988162 -216527643 -864109004 -34386201 -800374989 -375393825 -478344783 -198105544 -780939247 -140153730 , 761731718 -103599365 962429131 988387972 945643830 -361383179 152111380 384552891 -404164768 685030535 489324144 -430513628 -541925784 -824076779 -543237746 , 538798636 336175942 -962688195 856977077 492286070 10895820 692395723 436930228 749938599 -244759382 174280228 326991444 -225012678 618773021 692155456 98537694 , 741141998 714216035 945929531 590918731 -792114660 344598531 -569742859 -332274302 620005899 -724193434 228693877 310076118 160920628 7901168 285339506 -770681562 , 819769121 -744846555 841463585 -668733523 795319883 199509299 728158005 -137924228 951773944 -9208602 -159075521 -604533358 -875671261 -670828987 81251932 , 137774710 -50380145 -49302033 156368790 418998649 -534760643 455139778 379007160 566338843 118431504 -322317922 -997443448 512275200 -775456304 -27425997 , 766994026 -713573960 -423895684 784688391 -438020405 -69795166 943413445 544240704 -166719415 112827518 277656488 -956787320 149347196 -989095850 979580547 , 187336348 -390683316 -180466683 173221391 -861618134 881200975 718198325 -329643067 380177125 -559104063 -911248618 -328203312 140259520 84233477 -395781516 , 34729996 -195955246 828860946 -634158929 -20522546 679388652 -473791215 779610581 645378279 -598991310 -367698693 -694238967 -489450655 -828964320 -21960608 , 96959082 104516944 -318109587 588571830 -424617439 -702087639 669298992 445926173 656639891 323764109 938355779 204976704 649355342 -844753921 -79015701 984712316 , 369136440 937381121 751280975 -387341020 -251871980 670212724 -613044963 524886763 -994703216 917257076 997141784 -174931084 -515891062 -178171337 -890589561 , 427977792 274050927 793772018 721434917 799151893 357427389 208521519 515916500 -979527247 -452241397 -947995190 -41132223 748216600 -82807385 -384889663 , 858916062 582614190 -74616508 -558332611 811991834 402979744 626702703 -850287591 691297713 725622712 8776604 -398908365 -925181990 -601706336 -599868435 , 555313897 -119374999 514476621 -990459741 445437248 -964931734 402542533 -945042746 864648020 -633764307 885344722 -164511364 -888488714 592440744 718936084 , 511711008 -586665974 -447733316 331857272 -17940299 -312966575 33222105 -147636973 -787893768 376994184 828062443 -69938083 690840604 304407057 -230073277 , 651169976 -403875579 -47877357 -198457112 -709377327 120695455 -897073064 448864744 -841803692 28347836 882574271 -394497270 -49135693 -748245910 -513828115 , 486570393 -609768667 -605836660 -922694442 468011249 -125373751 800075274 617483207 -899581402 -632518082 925980848 -106840916 290220320 -251152150 679253678 , 79609162 731223646 -155654230 842496573 852349765 -296278092 -605912655 867352383 945875545 -643729227 373188926 -565512677 -619106914 -400450416 -197283669 , 745710852 614229891 311953995 314611010 -583813219 515337169 -655116064 989160139 -364036915 585806271 976049230 22326648 95148440 691388166 374010894 937832335 , 545770674 758304950 297854397 -668165256 -336845016 833113559 797207661 374152949 -971258632 -926743788 519732331 82052554 142614697 -602573511 -979934991 , 433605040 890493670 -258779567 -815476389 739413070 727428379 462240072 -526306884 421774069 -845394810 -921828440 -843007236 856694365 804718802 -188769890 ,

output.txt:

1	2499461989
---	------------

Результат работы кода на минимальных значениях:

input.txt

1	1
2	2

output.txt:

1	0
---	---

Проверка задачи на время выполнения и затраты памяти:

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.001	14
Пример из задачи	0.001	14
Верхняя граница диапазона значений входных данных из текста задачи	0.43	26

Вывод по задаче:

В данной задаче мы реализовали подсчет инверсий с помощью сортировки слиянием.

## Задача №4(Замена задачи №7). 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.

- Формат входного файла (input.txt). В первой строке входного файла содержится число  $n$  ( $1 \leq n \leq 10^5$ ) — число элементов в массиве, и последовательность  $a_0 < a_1 < \dots < a_{n-1}$  из  $n$  различных положительных целых чисел в порядке возрастания,  $1 \leq a_i \leq 10^9$  для всех  $0 \leq i < n$ . Следующая строка содержит число  $k$ ,  $1 \leq k \leq 10^5$  и  $k$  положительных целых чисел  $b_0, \dots, b_{k-1}$ ,  $1 \leq b_j \leq 10^9$  для всех  $0 \leq j < k$ .

- Формат выходного файла (output.txt). Для всех  $i$  от 0 до  $k - 1$  вывести индекс  $0 \leq j \leq n - 1$ , такой что  $a_i = b_j$  или -1, если такого числа в массиве нет.

- Ограничение по времени. 2сек.
- Ограничение по памяти. 256 мб.

Код:

```
import os
import psutil
from datetime import datetime

def Solution(array, value, start, stop):
    if start > stop:
        return -1
    else:
        mid = (start + stop) // 2
        if value == array[mid]:
            return mid
        elif value < array[mid]:
            return Solution(array, value, start, mid - 1)
        else:
            return Solution(array, value, mid + 1, stop)
```

```

process = psutil.Process(os.getpid())
startTime = datetime.now()

InputFile = open('input.txt')
lengthA = InputFile.readline()
arrayA = sorted([int(x) for x in InputFile.readline().split()])
lengthB = InputFile.readline()
arrayB = [int(x) for x in InputFile.readline().split()]

Outputfile = open('output.txt', 'w')
answer = ""
for i in range(len(arrayB)):
    x = Solution(arrayA, arrayB[i], 0, len(arrayA) - 1)
    arrayB[i] = x

for i in arrayB:
    answer += str(i) + ' '
Outputfile.write(f'{answer}')

print(datetime.now() - startTime, process.memory_info().rss / (1024 * 1024))

```

Объяснение:

В данной программе мы также реализуем алгоритм бинарного поиска в функции Solution. В функции находим среднее значение, и если элемент меньше/больше среднего значения, то убираем большую/меньшую часть массива и рекурсивно вызываем функцию, пока среднее значение не будет равно элементу.

Результат работы программы на примере из задачи:

input.txt:

1	5
2	1 5 8 12 13
3	5
4	8 1 23 1 11

output.txt:

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

Результат работы кода на максимальных значениях:

input.txt:

```
1      10000
2      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
3      10000
4      847 6295 8892 9882 4486 6347 5900 3093 4977 5475 625 8557 3137 5099 2644 2746 9377 6150 8376 7384 2935 3668 2753 908 1016 2672 1506 7497 8219 9157 1029 8484 2338 589 6
```

output.txt:

```
1      4811 2426 9228 12 3776 7291 1190 5124 323 5218 6205 9937 4626 3603 3698 9825 7977 3946 723 9242 2086 4344 3054 6377 6405 1770 5642 1658 2955 2440 7416 8748 1808 34
```

Результат работы кода на минимальных значениях:

input.txt:

```
1      1
2      12
3      1
4      3
```

output.txt:

```
1      -1
```

Проверка задачи на время выполнения и затраты памяти:

	Время выполнения (сек)	Затраты памяти (Мб)
Нижняя граница диапазона значений входных данных из текста задачи	0.001	14
Пример из задачи	0.001	14
Верхняя граница диапазона значений входных данных из	0.037	15

текста задачи		
---------------	--	--

Вывод по задаче:

При выполнении данной задачи, потребовалось изучить алгоритм бинарного поиска

## **Вывод**

В данной лабораторной я изучил алгоритм сортировкой слиянием и реализовал его. Также я изучил и реализовал алгоритм бинарного поиска.