

### 实验1

第一个函数是老师的穷举法；第二个函数是老师的二分查找函数；第三个函数是 $n^2 \log n$ 复杂度的函数，算法与老师的思路一致。先排序；排完序建立一个二维list（因为三元组可能不止一个）；然后对b从左往右遍历；对c从b+1开始遍历（防止重复），最后对a从c+1开始二分查找即可。（因为如果a从头开始会产生重复的结果，算法效率也会降低。）找完入栈，最后输出，注意边界条件。

```
def exhaustive_search_3Sum(s, x):
    n = len(s)
    for i in range(n - 2):
        for j in range(i + 1, n - 1):
            for k in range(j + 1, n):
                if s[i] + s[j] + s[k] == x:
                    return s[i], s[j], s[k]
    return ()

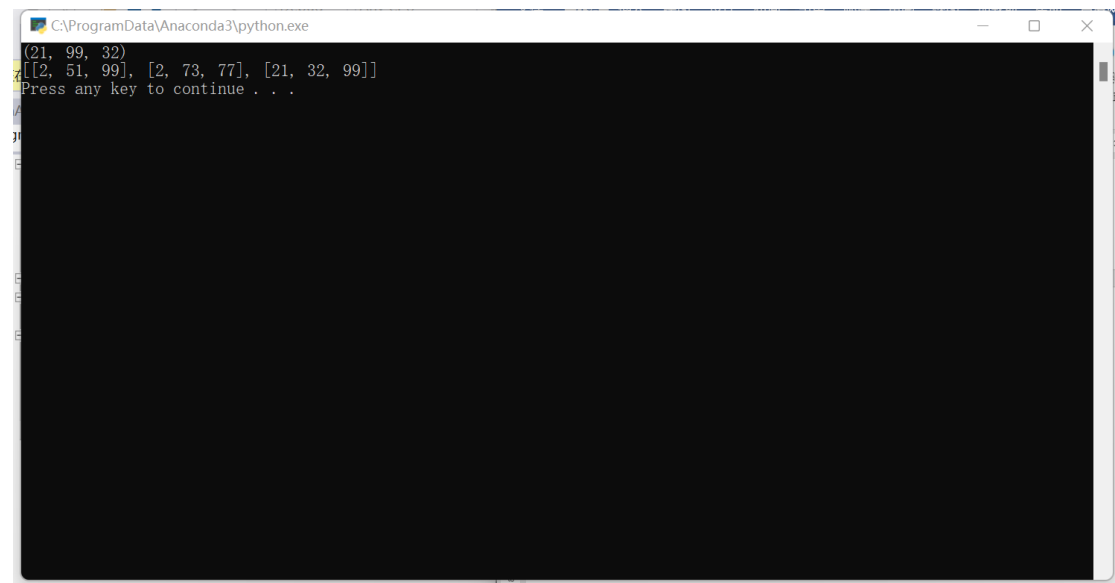
s = [21, 73, 6, 67, 99, 60, 77, 5, 51, 32, 2]
x=152
print (exhaustive_search_3Sum(s, 152))

#teacher's code for test
def binary_search(s, k):
    low = 0; high = len(s) - 1
    while low <= high:
        mid = (high + low) // 2
        if k == s[mid]:
            return mid
        elif k < s[mid]:
            high = mid - 1
        else:
            low = mid + 1
    return -1

#teacher's code for binary_search
def lognnum3(s, x):
    s=sorted(s, key=lambda s:s)
    #print(s)
    n=len(s)
    result=[0 for i in range(0)]for j in range(n^2)]
    pos=0
    for b in range (0,n):
        for c in range(b+1,n):
            a=binary_search(s[c+1:n], x-s[b]-s[c])
            if(a!=-1):
                result[pos].append(s[b])
                result[pos].append(s[c])
                result[pos].append(s[a+c+1])
                pos=pos+1
    return result[0:pos]
```

```
print(lognnum3(s, x))
```

第一行是穷举法结果（穷举一个结果就会 return），第二行是我的算法结果



```
C:\ProgramData\Anaconda3\python.exe
(21, 99, 32)
[[2, 51, 99], [2, 73, 77], [21, 32, 99]]
Press any key to continue...
```

## 实验 2

注意在 vs 环境下要先写六个“，再把内容粘进注释内，否则容易不识别；字符串名称建议用 s1, s2, s3 的形式，不然容易出 bug；调用 timeit 时后面的 %n 在测试程序内以 %d 的形式传入，大家找找 s1, s2, s3 内的 %d 符号就明白了。

```
s1="""
import random
def insertion_sort(s):
    n = len(s)
    for i in range(1, n):
        value = s[i]
        pos = i
        while pos > 0 and value < s[pos - 1] :
            s[pos] = s[pos - 1]
            pos -= 1
        s[pos] = value
alist=[random.random() for i in range (%d)]
insertion_sort(alist)
"""

s2="""
import random
def merge_ordered_lists(s1, s2):
    t = []
    i = j = 0
    while i < len(s1) and j < len(s2):
        if s1[i] < s2[j]:
            t.append(s1[i]); i += 1
        else:
```

```

        t.append(s2[j]); j += 1
    t += s1[i:]
    t += s2[j:]
    return t
def merge_sort(s):
    if len(s) <= 1:
        return s
    mid = len(s) // 2
    left = merge_sort(s[:mid])
    right = merge_sort(s[mid:])
    return merge_ordered_lists(left, right)
alist=[random.random() for i in range (%d)]
merge_sort(alist)
"""

s3="""
import random
def partition(arr,low,high):
    i = ( low-1 )
    pivot = arr[high]
    for j in range(low , high):
        if arr[j] <= pivot:

            i = i+1
            arr[i],arr[j] = arr[j],arr[i]

    arr[i+1],arr[high] = arr[high],arr[i+1]
    return ( i+1 )
def quickSort(arr,low,high):
    if low < high:
        pi = partition(arr,low,high)
        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)
alist=[random.random() for i in range (%d)]
quickSort(alist,0,len(alist)-1)
"""

import timeit
import random
for n in range(100,1100,100):
    print("n",n)
    t=timeit.timeit(stmt=s1 % n,number=10) / 10
    print("insertion_sort:",t)
    t=timeit.timeit(stmt=s2 % n,number=10) / 10
    print("merge_sort:",t)

```

```
t=timeit.timeit(stmt=s3 % n,number=10) / 10
print("quick_sort:",t)
```

```
C:\ProgramData\Anaconda3\python.exe
n 100
insertion_sort: 0.0015745200000000016
merge_sort: 0.0009233400000000058
quick_sort: 0.0007109599999999938
n 200
insertion_sort: 0.0079162299999999991
merge_sort: 0.0025299399999999997
quick_sort: 0.00158656000000000146
n 300
insertion_sort: 0.0162079799999999999
merge_sort: 0.0042917199999999999
quick_sort: 0.00204942999999999966
n 400
insertion_sort: 0.02773249
merge_sort: 0.0047196200000000013
quick_sort: 0.0029600100000000013
n 500
insertion_sort: 0.04855478
merge_sort: 0.0078525599999999986
quick_sort: 0.0040219599999999991
n 600
insertion_sort: 0.06409398
merge_sort: 0.0096096999999999988
quick_sort: 0.0060703600000000008
n 700
insertion_sort: 0.086716029999999997
merge_sort: 0.0089293299999999958
quick_sort: 0.006894599999999995
n 800
insertion_sort: 0.110885220000000003
merge_sort: 0.0100874200000000069
quick_sort: 0.0061700000000000009
n 900
insertion_sort: 0.142469899999999995
merge_sort: 0.0117009599999999937
quick_sort: 0.00698458000000000405
n 1000
insertion_sort: 0.195494080000000001
merge_sort: 0.0127807100000000006
quick_sort: 0.0079095099999999998
Press any key to continue . . .
```

### 实验 3

记得先在pyder命令行下pip install line\_profiler，然后新建一个txt，把以下代码粘入；然后把txt文件改名成merge\_sort.py移进C盘，对命令的参数也稍作更改：

C:\anaconda\Scripts\kernprof -l -v C:\merge\_sort.py

否则会有问题。

```
import time
```

```
@profile
```

```
def merge_ordered_lists(s1, s2):
    t = []
    i = j = 0
    while i < len(s1) and j < len(s2):
        if s1[i] < s2[j]:
            t.append(s1[i]); i += 1
        else:
            t.append(s2[j]); j += 1
    t += s1[i:]
    t += s2[j:]
    return t
```

@profile

```
def merge_sort(s):  
    if len(s) <= 1:  
        return s  
    mid = len(s) // 2  
    left = merge_sort(s[:mid])  
    right = merge_sort(s[mid:])  
    return merge_ordered_lists(left, right)
```

```
s = [21, 73, 6, 67, 99, 60, 77, 5, 51, 32]  
print(merge_sort(s))
```

```
PS C:\> C:\anaconda\Scripts\kernprof -l -v C:\merge_sort.py  
[5, 6, 21, 32, 51, 60, 67, 73, 77, 99]  
Wrote profile results to merge_sort.py.lprof  
Timer unit: 1e-06 s  
  
Total time: 6.58e-05 s  
File: C:\merge_sort.py  
Function: merge_ordered_lists at line 3  
  
Line #      Hits          Time  Per Hit   % Time  Line Contents  
=====
```

Line #	Hits	Time	Per Hit	% Time	Line Contents
3					@profile
4					def merge_ordered_lists(s1, s2):
5	9	3.5	0.4	5.3	t = []
6	9	3.9	0.4	5.9	i = j = 0
7	31	18.1	0.6	27.5	while i < len(s1) and j < len(s2):
8	22	11.1	0.5	16.9	if s1[i] < s2[j]:
9	11	7.8	0.7	11.9	t.append(s1[i]); i += 1
10					else:
11	11	7.0	0.6	10.6	t.append(s2[j]); j += 1
12	9	5.8	0.6	8.8	t += s1[i:]
13	9	5.9	0.7	9.0	t += s2[j:]
14	9	2.7	0.3	4.1	return t

```
  
Total time: 0.0001736 s  
File: C:\merge_sort.py  
Function: merge_sort at line 16  
  
Line #      Hits          Time  Per Hit   % Time  Line Contents  
=====
```

Line #	Hits	Time	Per Hit	% Time	Line Contents
16					@profile
17					def merge_sort(s):
18	19	12.2	0.6	7.0	if len(s) <= 1:
19	10	4.8	0.5	2.8	return s
20	9	5.2	0.6	3.0	mid = len(s) // 2
21	9	14.4	1.6	8.3	left = merge_sort(s[:mid])
22	9	12.1	1.3	7.0	right = merge_sort(s[mid:])
23	9	124.9	13.9	71.9	return merge_ordered_lists(left, right)