```
import timeit
import random
import matplotlib.pyplot as plt

#input array elements
def Input(Array, n):
    #iterating till the range
    for i in range(0, n):
        ele = random. randrange(1, 50)
        #adding the elements
        Array.append(ele)

#divide function
def partition(Array, low, high):
    i = (low - 1)
    pivot = Array[high] #pivot element
    for j in range(low, high):
        # if current element is smaller
        if Array[j] <= pivot:
            # increment
            i = i + 1
            Array[i], Array[j] = Array[j], Array[i]
    Array[i + 1], Array[high] = Array[high], Array[i + 1]
    return (i + 1)


#quick sort
def quickSort(Array, low, high):
    if low < high:
        #index
        pi = partition(Array, low, high)
        #sort the partitions
        quickSort(Array, low,pi - 1)
        quickSort(Array,pi + 1, high)

#main code
N = []
CPU = []
```

```
trail = int(input("Enter no. of trails: "))
for t in range(0, trail):
    Array = []
    print("--------> TRAIL NO:", t + 1)
    n = int(input("Enter number of elements: "))
    Input(Array, n)
    start = timeit.default_timer()
    quickSort(Array, 0, n - 1)
    times = timeit.default_timer() - start
    print("Sorted Array")
    print(Array)
    N.append(n)
    CPU.append(round(float(times)* 1000000,2))

print("N CPU")
for t in range(0, trail):
    print(N[t], CPU[t])

#plotting graph
plt.plot(N, CPU)
plt.scatter(N, CPU, color="red", marker="*", s=50)
plt.xlabel('Array Size - N')
plt.ylabel('CPU Processing Time')
plt.title('quick Sort Time efficiency')
plt.show()
```