In [ ]:

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

arr = input("Enter the list of element seperated by spaces:").split()
arr = [int(x)for x in arr]
print("input array:",arr)
n = len(arr)

def selection_sort(arr):
    for i in range (n):
        min_idx = i
        for j in range (i+1,n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i],arr[min_idx] = arr[min_idx],arr[i]
    return arr
sorted_arr = selection_sort(arr)
print("Sorted  Array",sorted_arr)
# time taken to sort
start_time = time.time()
sorted_arr = selection_sort(arr)
end_time = time. time()
print("time taken to start", end_time,"seconds")
```

In [ ]:

```
n_values = [5000,6000,7000,8000]
time_values = []

for n in n_values:
    arr = [random.randint(1,9)for _ in range(n)]
    start_time = time.time()
    sorted_arr =selection_sort(arr)
    end_time = time.time()
    time_taken = end_time-start_time
    print("Time taken to sort",n,"element:",time_taken ,"seconds")
    time_values.append(time_taken)

plt.plot(n_values,time_values,color="green")
plt.xlabel('Number of elements (n)')
plt.ylabel('Time taken (seconds)')
plt.title('Selection sort Time Complexity Analysis')
plt.grid(True)
plt.show ()
```

In [ ]:

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

arr = input("Enter the list if elements seperated by spaces: ").split()
arr = [int(x) for x in arr]
print("Input array: ", arr)
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr)//2
    left = merge_sort(arr[ :mid])
    right = merge_sort(arr[mid: ])
    return merge(left, right)
def merge(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i = i+1

        else:
            result.append(right[j])
            j = j+1

    result.extend(left[i:])
    result.extend(right[j:])
    return result
sorted_arr = merge_sort(arr)
print("Sorted array: ", sorted_arr)

start_time = time.time()
sorted_arr = merge_sort(arr)
end_time = time.time()
print("Time taken to sort: ", end_time-start_time, "seconds")
```

In [ ]:

```
n_values = [5000, 6000, 7000, 8000]
time_values= []
for n in n_values:
    arr = [random.randint(1,9)for _ in range(n)]

    start_time = time.time()
    sorted_arr = merge_sort(arr)
    end_time = time.time()

    time_taken = end_time - start_time
    print("Time taken to sort", n, "elemnts:", time_taken, "seconds")
    time_values.append(time_taken)

plt.plot(n_values, time_values,color="black")
plt.xlabel('Number of elements (n)')
plt.ylabel('Time Taken(seconds)')
plt.title('Merge Sort Time Complexity Analysis')
plt.grid(True)
plt.show()
```

In [ ]:

```
import random
import time
import matplotlib.pyplot as plt
arr = input("Enter the list if elements seperated by spaces: ").split()
arr = [int(x) for x in arr]
print("Input array: ", arr)
```

```
def quick_sort(arr):
     if len(arr) <= 1:
         return arr
     pivot = arr[len(arr)-1]


     left = []
     right = []
     for i in range (len(arr)-1):
         if arr[i] < pivot:
             left.append(arr[i])
         else:
             right.append(arr[i])

     return quick_sort(left) + [pivot] + quick_sort(right)

sorted_arr = quick_sort(arr)
print("Sorted array: ", sorted_arr)
start_time= time.time()
sorted_arr = quick_sort(arr)
end_time = time.time()
print("The time taken to sort ", end_time-start_time, "seconds")
```

In [ ]:

```
n_values = [5000, 6000, 7000, 8000]
time_values= []


for n in n_values:
    arr = [random.randint(1,9)for _ in range(n)]

    start_time = time.time()
    sorted_arr = quick_sort(arr)
    end_time = time.time()

    time_taken = end_time - start_time
    print("Time taken to sort", n, "elemnts:", time_taken, "seconds")
    time_values.append(time_taken)

plt.plot(n_values, time_values,color="purple")
plt.xlabel('Number of elements (n)')
plt.ylabel('Time Taken(seconds)')
plt.title('Quick Sort Time Complexity Analysis')
plt.grid(True)
plt.show()
```

In [ ]:

```
n =int(input("enter the number of vertices in the graph:"))
print("enter the adjacency matrix where each row separeted by spaces:")
graph =[]
for i in range(n):
    row =input().split()
    graph.append([int(x) for x in row])
source =int(input("enter the source vertex:"))
def dijkstra(graph,source):
    V =len(graph)
    dist =[float('inf)]*V
    dist[source] =0
    visited =[false]*V
    for _ in range(V):
           min_dist = float('inf')
           min_index = -1
           for v in range(V):
               if not visited[v] and dist[v] < min_dist:
                   min_dist = dist[v]
                   min_index = v
           if min_index == -1:
               break
           visited[min_index] = True
    for v in range(V):
        if graph[min_index][v] > 0 and not visited[v]:
            dist[v] = min(dist[v],dist[min_index]+ graph[min_index][v])
    return dist
Shortest_Paths = dijkstra(graph,source)
print("shortest distance from node",source,"to each node:")
for i,j in enumerate(shortest_paths):
    print("vertex",i,"distance =",j)
```