1. Write a program to sort a list of N elements using Selection sort Technique.

selection.py - C:/Users/user/Pictures/ada/selection.py (3.13.5)

File Edit Format Run Options Window Help

```
def selection_sort(arr):
 for i in range(len(arr)):
     min_index=i
     for j in range(i+1,len(arr)):
         if arr[j]<arr[min_index]:
             min_index=j
     arr[i],arr[min_index]=arr[min_index],arr[i]
if __name__=="__main__":
 input_list=[64,25,12,22,11,8,1]
 print("Original list",input_list)
 selection_sort(input_list)
 print("Sorted list",input_list)
```

IDLE Shell 3.13.5

File Edit Shell Debug Options Window Help

```
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (AMD64
)] on win32
Enter "help" below or click "Help" above for more information.
>>>
=============== RESTART: C:/Users/user/Pictures/ada/selection.py ===============
Original list [64, 25, 12, 22, 11, 8, 1]
Sorted list [1, 8, 11, 12, 22, 25, 64]
>>>
=============== RESTART: C:/Users/user/Pictures/ada/selection.py ===============
Original list [64, 25, 12, 22, 11, 8, 1]
Sorted list [1, 8, 11, 12, 22, 25, 64]
>>>
```

2. Write a program to implement the DFS and BFS algorithm for a graph.

DFS

DFS.py - C:\Users\user\Pictures\ada\DFS.py (3.13.5)

File Edit Format Run Options Window Help

```
# Write a program to implement the DFS and BFS algorithm for a graph
# DFS
def dfs(graph, start, visited = None):
    if visited is None:
        visited=set()
    visited.add(start)
    print(start,end=" ")
    for neighbour in graph[start]:
        if neighbour not in visited:
            dfs(graph,neighbour,visited)
graph={
    'A':['B','C'],
    'B':['D'],
    'C':['E','F'],
    'D':[],
    'E':[],
    'F':[]
    }

print("\n DFS order:")
dfs(graph,'A')
```

IDLE Shell 3.13.5

File Edit Shell Debug Options Window Help

```
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
================== RESTART: C:\Users\user\Pictures\ada\DFS.py ==================

 DFS order:
A B D C E F
```

BFS

DFS.py - C:\Users\user\Pictures\ada\DFS.py (3.13.5)

File Edit Format Run Options Window Help

```python
# Write a program to implement the DFS and BFS algorithm for a graph
# BFS

from collections import deque
def bfs(graph, start):
    visited=set()
    queue = deque([start])
    visited.add(start)
    while queue:
        vertex=queue.popleft()
        print(vertex,end=" ")
        for neighbour in graph[vertex]:
            if neighbour not in visited:
                queue.append(neighbour)
                visited.add(neighbour)

graph={
    'A':['B','C'],
    'B':['D'],
    'C':['E','F'],
    'D':[],
    'E':[],
    'F':[]
    }
print("\n BFS order:")
bfs(graph,'A')
```

IDLE Shell 3.13.5

File Edit Shell Debug Options Window Help

```
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
================== RESTART: C:\Users\user\Pictures\ada\DFS.py ==================

 BFS order:
A B C D E F
```

3. Write a program to find minimum and maximum value in an array using Divide and conquer.

minmax.py - C:\Users\user\Pictures\ada\minmax.py (3.13.5)

File Edit Format Run Options Window Help

```
def find_min_max(arr,low,high):
    if low ==high:
        return arr[low],arr[low]
    mid=(low+high)//2
    min1,max1 = find_min_max(arr,low,mid)
    min2,max2 = find_min_max(arr,mid+1,high)
    return min(min1,min2),max(max1,max2)
arr=[3,1,9,7,2,8,99]
low = 0
high = len(arr)-1
print("min and max values are:")
min_val,max_val = find_min_max(arr,low,high)
print("minimum:",min_val)
print("maximum:",max_val)
```

IDLE Shell 3.13.5

File Edit Shell Debug Options Window Help

```
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
================= RESTART: C:\Users\user\Pictures\ada\minmax.py ================
min and max values are:
minimum: 1
maximum: 99
>>>
```

4. Write a test program to implement Divide and Conquer Strategy Eg: Quick sort algorithm for sorting list of integers in ascending order.

quicksort.py - C:\Users\user\Pictures\ada\quicksort.py (3.13.5)

File Edit Format Run Options Window Help

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr  # Base case: Already sorted

    pivot = arr[len(arr) // 2]  # Choose a pivot(middle)element
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]

    return quick_sort(left) + middle + quick_sort(right)

# Example usage:
my_list = [3, 1, 9, 7, 6, 4,2, 8 ]
print("Sorted lists are:")
sorted_list = quick_sort(my_list)
print(sorted_list)
```

IDLE Shell 3.13.5

File Edit Shell Debug Options Window Help

```
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
=============== RESTART: C:\Users\user\Pictures\ada\quicksort.py ===============
Sorted lists are:
[1, 2, 3, 4, 6, 7, 8, 9]
>>>
```

5. Write a program to implement Merge sort algorithm for sorting a list of integers in ascending order.

mergesort.py - C:\Users\user\Pictures\ada\mergesort.py (3.13.5)

File Edit Format Run Options Window Help

```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr  # Base case: Already sorted or empty
    # Split the list into two halves
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]
    # Recursively sort both halves
    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)
    # Merge the sorted halves
    return merge(left_half, right_half)

def merge(left, right):
    merged = []
    left_index, right_index = 0, 0
    while left_index < len(left) and right_index < len(right):
        if left[left_index] < right[right_index]:
            merged.append(left[left_index])
            left_index += 1
        else:
            merged.append(right[right_index])
            right_index += 1
    # Append any remaining elements (if any)
    merged.extend(left[left_index:])
    merged.extend(right[right_index:])
    return merged

# Example usage:
my_list = [3, 1, 9, 7,5,4,6, 2, 8]
print("Sorted lists are:")
sorted_list = merge_sort(my_list)
print(sorted_list)
```

IDLE Shell 3.13.5

File Edit Shell Debug Options Window Help

```
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
=============== RESTART: C:\Users\user\Pictures\ada\mergesort.py ===============
Sorted lists are:
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
```

6. Sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n>5000, and record the time taken to sort.

mergewithtime.py - C:/Users/user/Pictures/ada/mergewithtime.py (3.13.5)

File Edit Format Run Options Window Help

```python
import random
import time
# Merge Sort Function
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]
        merge_sort(left_half)
        merge_sort(right_half)
        i = j = k = 0
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

        while i < len(left_half):
            arr[k] = left_half[i]
            i+= 1
            k += 1
        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1
# Function to generate a random list of n elements

def generate_random_list(n):
    return [random.randint(1, 100000) for _ in range(n)]

# Perform the sorting and measure time

n_values = [100, 500, 1000, 5000, 10000, 50000, 100000]

for n in n_values:
    arr = generate_random_list(n)
    start_time = time.time()
    merge_sort(arr)
    end_time = time.time()
    elapsed_time = end_time-start_time
    print(f"Time taken to sort {n} elements: {elapsed_time:.6f} seconds")
```

IDLE Shell 3.13.5

File Edit Shell Debug Options Window Help

```
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:
15:46) [MSC v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more inf
ormation.
>>>
============= RESTART: C:/Users/user/Pictures/ada/mer
gewithtime.py =============
Time taken to sort 100 elements: 0.000829 seconds
Time taken to sort 500 elements: 0.003650 seconds
Time taken to sort 1000 elements: 0.010769 seconds
Time taken to sort 5000 elements: 0.062801 seconds
Time taken to sort 10000 elements: 0.126267 seconds
Time taken to sort 50000 elements: 0.311920 seconds
Time taken to sort 100000 elements: 0.730165 seconds
>>>
```

Ln: 12 Col: 0

7. Sort a given set of N integer elements using Quick Sort method and compute its complexity. Run the program for varied values of n >5000 and record the time taken to sort.

p1.py - C:\Users\user\Pictures\ada\p1.py (3.13.5)

File Edit Format Run Options Window Help

```python
# Sort a given set of n integer elements using Quick Sort method and compute its time complexity.
# Run the program for varied values of n > 5000 and records the time taken to sort.

#Quick sort function
import random
import time
def quick_sort(arr):
    if len(arr)<=1:
        return arr
    pivot = arr[random.randint(0,len(arr)-1)]
    lesser =[]
    equal=[]
    greater =[]
    for element in arr:
        if element < pivot:
            lesser.append(element)
        elif element == pivot:
            equal.append(element)
        else:
            greater.append(element)
    return quick_sort(lesser)+ equal +quick_sort(greater)

# Function to generate a random list of n elements
def generate_random_list(n):
    return [random.randint(1,10000) for _ in range(n)]

# Perform the sorting and measure time
n_values = [100,500,1000,5000,10000,50000,100000]

for n in n_values:
    arr = generate_random_list(n)
    start_time=time.time()
    sorted_arr = quick_sort(arr)
    end_time = time.time()
    elapsed_time = end_time-start_time
    print(f" time taken to sort {n} elements : { elapsed_time:.6f} seconds")
```

IDLE Shell 3.13.5

File Edit Shell Debug Options Window Help

```
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MS
C v.1943 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
================== RESTART: C:\Users\user\Pictures\ada\p1.py =
================
 time taken to sort 100 elements : 0.000513 seconds
 time taken to sort 500 elements : 0.004973 seconds
 time taken to sort 1000 elements : 0.010229 seconds
 time taken to sort 5000 elements : 0.038046 seconds
 time taken to sort 10000 elements : 0.041799 seconds
 time taken to sort 50000 elements : 0.129872 seconds
 time taken to sort 100000 elements : 0.230875 seconds
>>>
```

Ln: 12 Col: 0

8. Write a program to perform Travelling Salesman Problem.

tsp.py - C:/Users/user/Pictures/ada/tsp.py (3.13.5)

File Edit Format Run Options Window Help

```
import itertools
def calculate_total_distance(path,cities):
    total_distance = 0
    for i in range(len(path)-1):
        city1=path[i]
        city2=path[i+1]
        total_distance+=cities[city1][city2]
    total_distance+=cities[path[-1]][path[0]]
    return total_distance
def traveling_salesman_bruteforce(cities):
    num_cities=len(cities)
    all_cities=set(range(num_cities))
    shortest_path=None
    shortest_distance=float('inf')
    for path in itertools.permutations(all_cities):
        distance=calculate_total_distance(path,cities)
        if distance<shortest_distance:
            shortest_distance=distance
            shortest_path=path
    return shortest_path,shortest_distance
if __name__=="__main__":
    cities =[
        [0,29,20,21],
        [29,0,15,17],
        [20, 150,0,28],
        [21,17,28,0]
    ]
    shortest_path,shortest_distance=traveling_salesman_bruteforce(cities)
    print ("Shortest TSP path",shortest_path)
    print("Shortest TSP distance:",shortest_distance)
```

IDLE Shell 3.13.5

File Edit Shell Debug Options Window Help

```
Python 3.13.5 (tags/v3.13.5:6cb20a2, Jun 11 2025, 16:15:46) [MSC v.1943 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
================== RESTART: C:/Users/user/Pictures/ada/tsp.py ==================
Shortest TSP path (0, 3, 1, 2)
Shortest TSP distance: 73
>>>
```