Date: 21.03.2024 Name: Mugilkrishna D U
Ex. No: 5 Reg. No.: 3122225001073

UCS 2403 Design & Analysis of Algorithms

Assignment 5

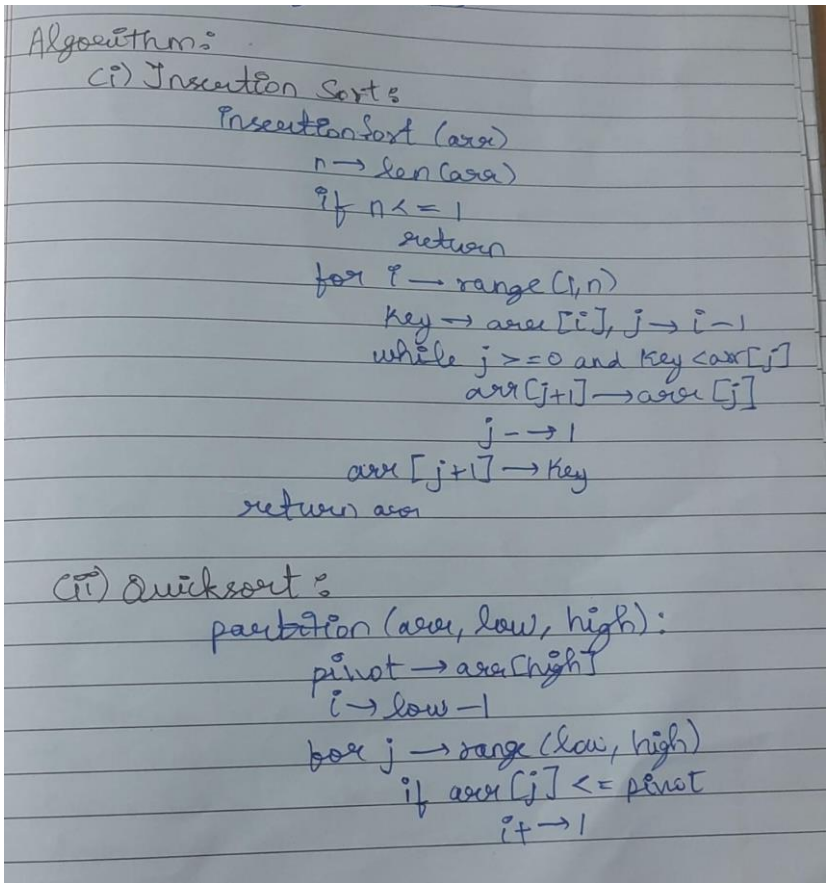
Date of Exercise: 21.03.2024

Aim: To gain understanding and proficiency on the divide and conquer algorithm

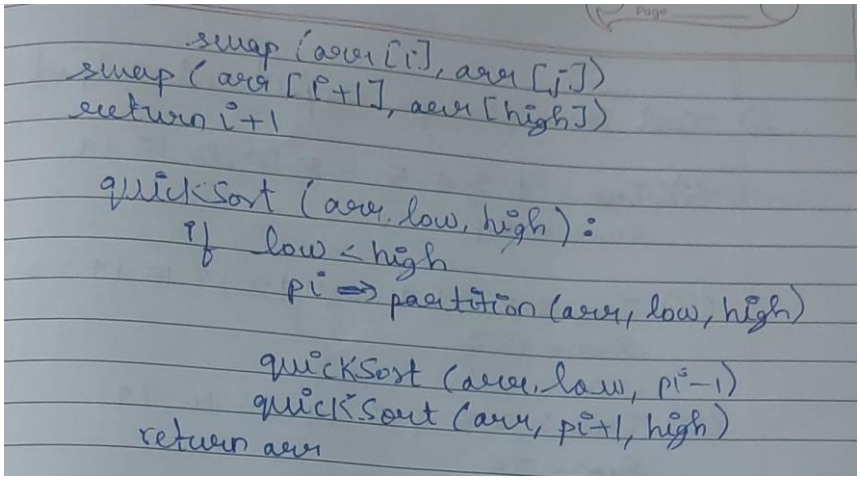
Question 1:

Find the kth smallest element: First, find the kth smallest element in an unsorted list using insertion sort. Next, find the same by modifying the divide-and-conquer algorithm of Quicksort. Compare the time complexities of both the algorithms.

Algorithm:



Name: Mugilkrishna D U
Reg. No.: 3122225001073



Code:

```
#Input list
size = int(input("Enter the size of the List: "))
nums = []

for k in range(size):
    nums.append(int(input("Enter element " + str(k+1) + ": ")))
print("\nList:", nums)

# Insertion Sort
def insertionSort(arr):
    n = len(arr)
    if n <= 1:
        return
    for i in range(1, n):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

print("Sorted List using insertion sort: ",insertionSort(nums))

kthele1 = int(input("\nEnter the value of k: "))
print("The " + str(kthele1) + " smallest element using Insertion Sort algorithm is:
", nums[kthele1-1])

#Quick Sort
def partition(arr, low, high):
    pivot = arr[high]
    i = low - 1
```

```
    for j in range(low, high):
        if arr[j] <= pivot:
            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)

    return arr

sorted_nums_quickSort = quickSort(nums, 0, len(nums) - 1)
print("Sorted List using Quick sort:", sorted_nums_quickSort)

kthele2 = int(input("Enter the value of k: "))
print("The " + str(kthele2) + " smallest element using Quick Sort algorithm is:",
sorted_nums_quickSort[kthele2 - 1])
```

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.1.py"
Enter the size of the List: 6
Enter element 1: 23
Enter element 2: 30
Enter element 3: 12
Enter element 4: 11
Enter element 5: 50
Enter element 6: 45

List: [23, 30, 12, 11, 50, 45]
Sorted List using insertion sort:  [11, 12, 23, 30, 45, 50]

Enter the value of k: 4
The 4 smallest element using Insertion Sort algorithm is:  30
Sorted List using Quick sort: [11, 12, 23, 30, 45, 50]
Enter the value of k: 2
The 2 smallest element using Quick Sort algorithm is: 12
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.1.py"
Enter the size of the List: 4
Enter element 1: 123
```

```
Enter element 2: 43
Enter element 3: 254
Enter element 4: 11

List: [123, 43, 254, 11]
Sorted List using insertion sort:  [11, 43, 123, 254]

Enter the value of k: 2
The 2 smallest element using Insertion Sort algorithm is:  43
Sorted List using Quick sort: [11, 43, 123, 254]
Enter the value of k: 3
The 3 smallest element using Quick Sort algorithm is: 123
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

- **Time Complexity for Insertion Sort:** O(n^2)
- **Time Complexity for Quick Sort:** O(n*logn)

Question 2:

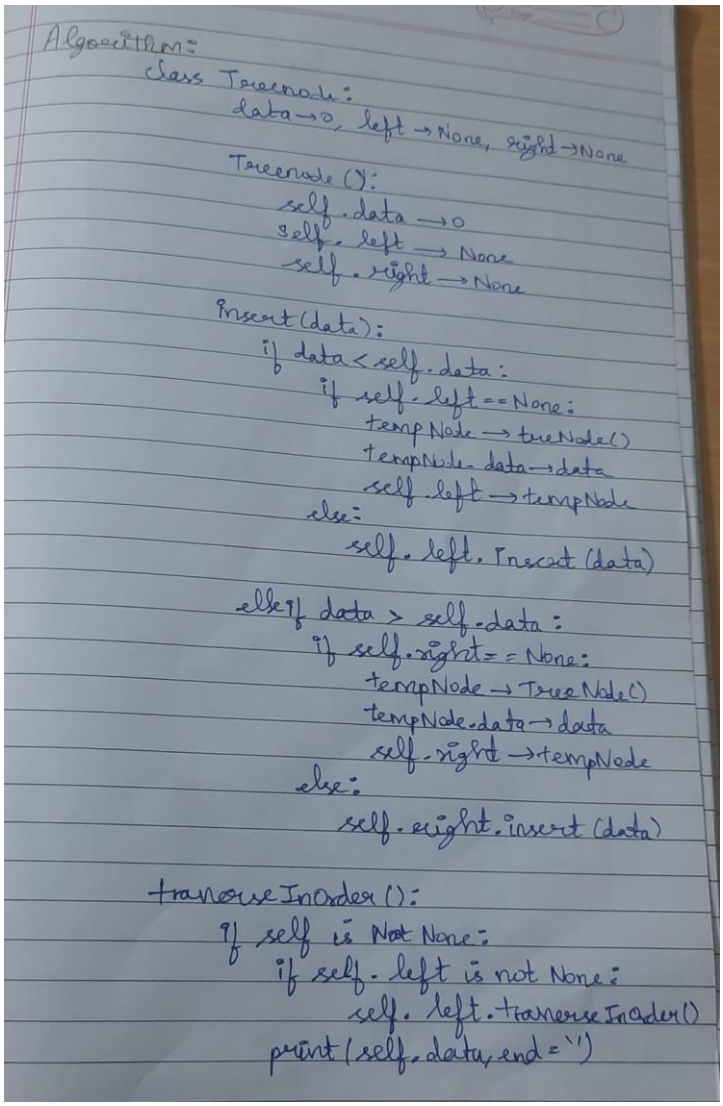
Find the sum of the values in the nodes of a binary tree: Consider the code given below that has to find the sum of the values in the nodes of a binary tree.

```
# Code to populate a tree starts here
import random
class TreeNode:
        def __init__(self):
                self.data = 0
                self.left = None
                self.right = None

        def insert(self, data):
                if data < self.data:
                        if self.left == None:
                                tempNode = TreeNode()
                                self.left = tempNode
                                self.left.data = data
                        else:
                                self.left.insert(data)
                elif data > self.data:
                        if self.right == None:
                                tempNode = TreeNode()
                                self.right = tempNode
                                self.right.data = data
                        else:
                                self.right.insert(data)

        def traverseInOrder(self):
                if self.left != None:
                        self.left.traverseInOrder()
                print(self.data, end=' ')
```

```
        if self.right != None:
            self.right.traverseInOrder()

def createRoot():
    i = random.randint(0, 10)
    rootNode = TreeNode()
    rootNode.data = i
    return rootNode

def createTree():
    rootNode = createRoot()
    numNodes = random.randint(1, 10)
    currentNode = rootNode
    j = 0
    L = []
    while (j <= numNodes):
        newVal = random.randint(1,20)
        if newVal not in L:
            currentNode.insert(newVal)
            L.append(newVal)
        j+=1
    rootNode.traverseInOrder()
    return rootNode
# Code to populate the tree ends here

def getSum(node):
    if node == None:
        return 0
    else:
        leftSum = getSum(node.left)
        rightSum = getSum(node.right)
        return leftSum + rightSum
rootNode = createTree()
print("Sum = ",getSum(rootNode))
```

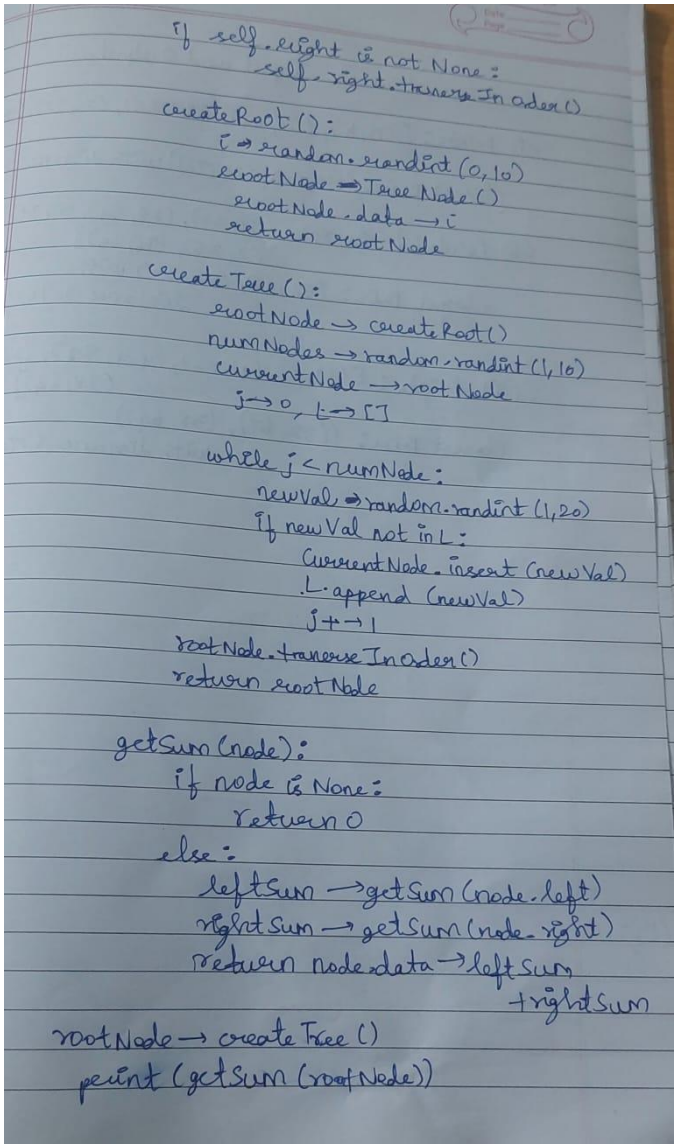
This code is known to have some bugs. Modify the given program to correctly find the sum. Ensure that the number of nodes in the tree and the value in each node are generated randomly.

Algorithm:



```
Algorithm:
    class Treenode:
        data → 0, left → None, right → None

    Treenode ():
        self.data → 0
        self.left → None
        self.right → None

    Insert (data):
        if data < self.data:
            if self.left == None:
                tempNode → treeNode()
                tempNode.data → data
                self.left → tempNode
            else:
                self.left.Insert (data)

        elif data > self.data:
            if self.right == None:
                tempNode → TreeNode()
                tempNode.data → data
                self.right → tempNode
            else:
                self.right.insert (data)

    traverseInOrder():
        if self is Not None:
            if self.left is not None:
                self.left.traverseInOrder()
            print (self.data, end = ' ')
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073



Code:

```
import random
class TreeNode:
    def __init__(self):
        self.data = 0
        self.left = None
        self.right = None
    def insert(self, data):
        if data < self.data:
            if self.left == None:
                tempNode = TreeNode()
                tempNode.data = data
                self.left = tempNode
```

Name: Mugilkrishna D U
Reg. No.: 3122225001073

```
            else:
                self.left.insert(data)
        elif data > self.data:
            if self.right == None:
                tempNode = TreeNode()
                tempNode.data = data
                self.right = tempNode
            else:
                self.right.insert(data)

    def traverseInOrder(self):
        if self is not None:
            if self.left is not None:
                self.left.traverseInOrder()
            print(self.data, end=' ')
            if self.right is not None:
                self.right.traverseInOrder()

    @staticmethod
    def createRoot():
        i = random.randint(0, 10)
        rootNode = TreeNode()
        rootNode.data = i
        return rootNode

    @classmethod
    def createTree(cls):
        rootNode = cls.createRoot()
        numNodes = random.randint(1, 10)
        currentNode = rootNode
        j = 0
        L = []

        while j < numNodes:
            newVal = random.randint(1, 20)
            if newVal not in L:
                currentNode.insert(newVal)
                L.append(newVal)
                j += 1
        rootNode.traverseInOrder()
        return rootNode

    @staticmethod
    def getSum(node):
        if node is None:
            return 0
```

```
        else:
            leftSum = TreeNode.getSum(node.left)
            rightSum = TreeNode.getSum(node.right)
            return node.data + leftSum + rightSum

rootNode = TreeNode.createTree()
print("\nSum =", TreeNode.getSum(rootNode))
```

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.2.py"
1 3 4 5 6 8 10 13 15 19
Sum = 84
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.2.py"
2 5 6 9 10 11 12 13 15 19 20
Sum = 122
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.2.py"
2 3 4 5 14 15 16 19
Sum = 78
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.2.py"
1 2 3 5 7 10 12 17 19 20
Sum = 96
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode>
```

Time Complexity:

- **Time Complexity in Average Case:** O(logn)
- **Time Complexity in Worst Case:** O(n)

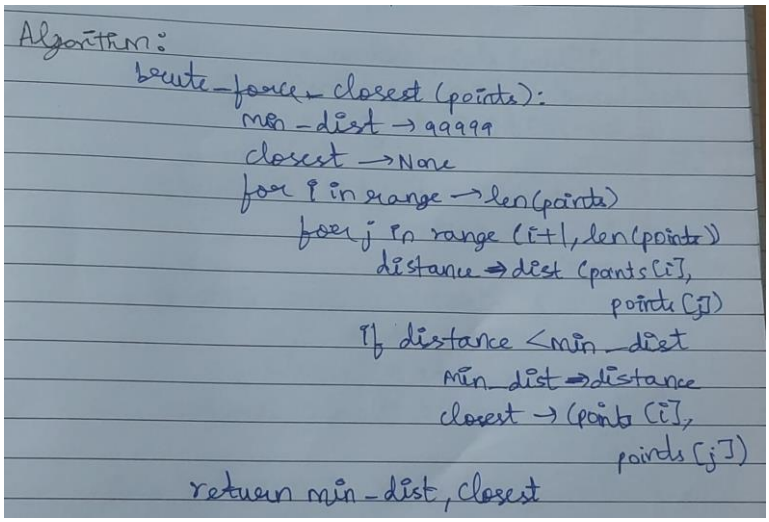
Question 3:

Given a set of points in a 2D plane, find the pair of points with the smallest Euclidean distance between them, using divide-and-conquer strategy. For example. if the given set of points is {(1, 2), (3, 5), (6, 9), (8, 12), (10, 15)}, then the closest pair of points is (3, 5) and (6, 9).

Date: 21.03.2024
Ex. No: 5

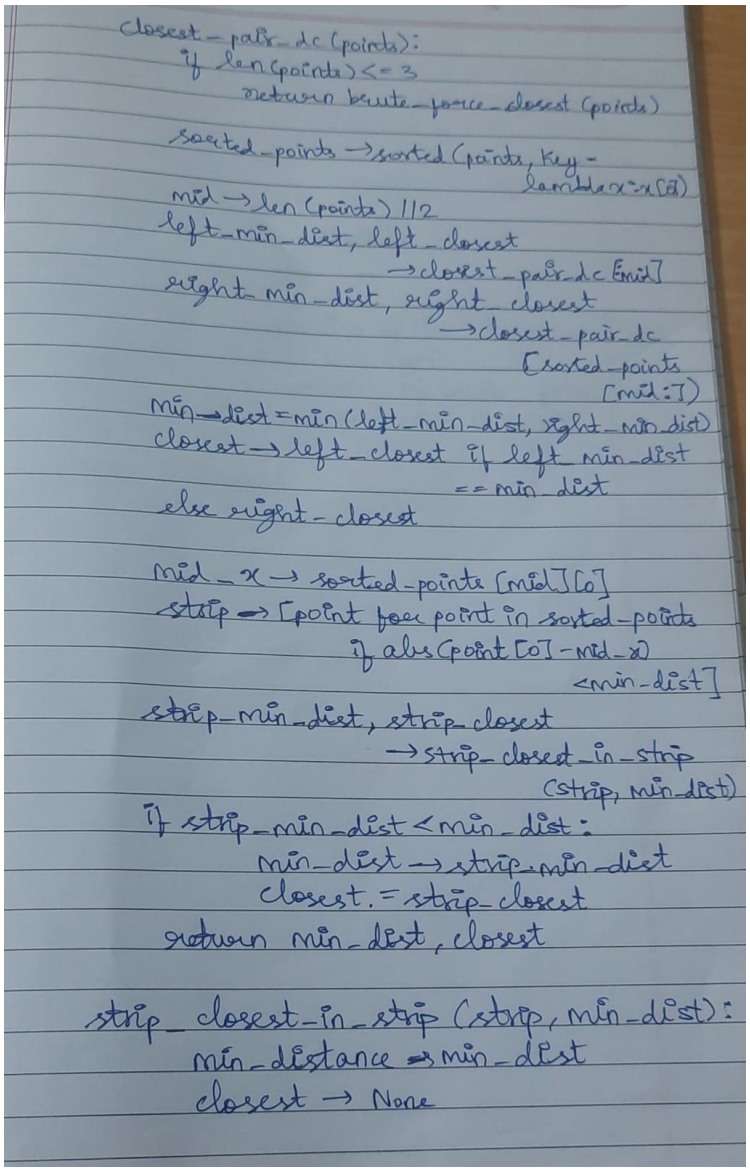
Name: Mugilkrishna D U
Reg. No.: 3122225001073

Algorithm:



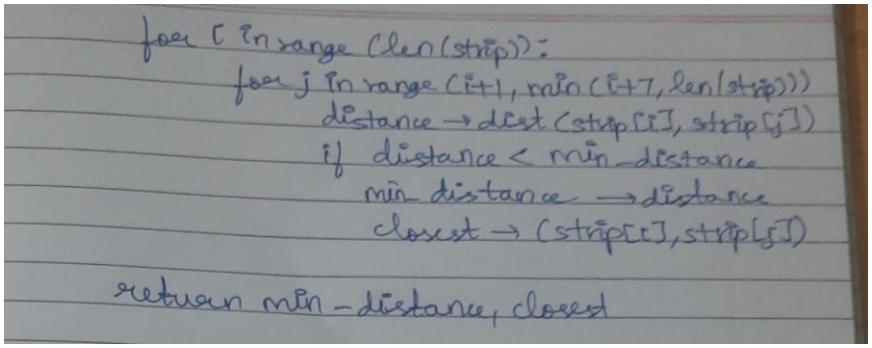
Algorithm:

```
brute_force_closest(points):
    min_dist → 99999
    closest → None
    for i in range → len(points)
        for j in range (i+1, len(points))
            distance → dist(points[i],
                            points[j])
            If distance < min_dist
                min_dist → distance
                closest → (points[i],
                           points[j])
    return min_dist, closest
```



```
closest_pair_dc(points):
    if len(points) <= 3
        return brute_force_closest(points)

    sorted_points → sorted(points, key=
                           lambda x: x[0])
    mid → len(points) // 2
    left_min_dist, left_closest
        → closest_pair_dc[mid]
    right_min_dist, right_closest
        → closest_pair_dc
          [sorted_points
          [mid:])
    min_dist = min(left_min_dist, right_min_dist)
    closest → left_closest if left_min_dist
              == min_dist
    else right_closest

    mid_x → sorted_points[mid][0]
    strip → [point for point in sorted_points
             if abs(point[0] - mid_x)
             < min_dist]
    strip_min_dist, strip_closest
        → strip_closest_in_strip
          (strip, min_dist)
    if strip_min_dist < min_dist:
        min_dist → strip_min_dist
        closest. = strip_closest
    return min_dist, closest

strip_closest_in_strip(strip, min_dist):
    min_distance → min_dist
    closest → None
```

Date: 21.03.2024 Name: Mugilkrishna D U
Ex. No: 5 Reg. No.: 3122225001073



Code:

```
import math

def dist(p1, p2):
    return math.sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

def brute_force_closest(points):
    min_dist = 99999
    closest = None
    for i in range(len(points)):
        for j in range(i+1, len(points)):
            distance = dist(points[i], points[j])
            if distance < min_dist:
                min_dist = distance
                closest = (points[i], points[j])
    return min_dist, closest

def closest_pair_dc(points):
    if len(points) <= 3:
        return brute_force_closest(points)

    sorted_points = sorted(points, key=lambda x: x[0])
    mid = len(points) // 2
    left_min_dist, left_closest = closest_pair_dc(sorted_points[:mid])
    right_min_dist, right_closest = closest_pair_dc(sorted_points[mid:])

    min_dist = min(left_min_dist, right_min_dist)
    closest = left_closest if left_min_dist == min_dist else right_closest

    mid_x = sorted_points[mid][0]
    strip = [point for point in sorted_points if abs(point[0] - mid_x) < min_dist]

    strip_min_dist, strip_closest = strip_closest_in_strip(strip, min_dist)
    if strip_min_dist < min_dist:
        min_dist = strip_min_dist
        closest = strip_closest
```

```
    return min_dist, closest

def strip_closest_in_strip(strip, min_dist):
    min_distance = min_dist
    closest = None
    for i in range(len(strip)):
        for j in range(i+1, min(i+7, len(strip))):
            distance = dist(strip[i], strip[j])
            if distance < min_distance:
                min_distance = distance
                closest = (strip[i], strip[j])
    return min_distance, closest

def generate_random_points(n):
    points = []
    for i in range(n):
        x = random.randint(0, 100)
        y = random.randint(0, 100)
        points.append((x, y))
    return points

# Example usage:
num_points = 10
points = generate_random_points(num_points)
print("Randomly generated points:", points)

min_dist, closest_pair = closest_pair_dc(points)
print("Closest pair:", closest_pair, "with distance:", min_dist)
```

Output:

```
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.3.py"
Randomly generated points: [(75, 14), (65, 54), (82, 68), (50, 86), (92, 89), (43, 63), (21, 72), (0, 76),
 (70, 56), (24, 76)]
Closest pair: ((21, 72), (24, 76)) with distance: 5.0
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.3.py"
Randomly generated points: [(84, 80), (52, 40), (28, 54), (55, 39), (2, 35), (96, 65), (65, 68), (90, 1),
(33, 16), (70, 95)]
Closest pair: ((52, 40), (55, 39)) with distance: 3.1622776601683795
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.3.py"
Randomly generated points: [(25, 7), (92, 7), (88, 40), (77, 82), (80, 31), (62, 52), (97, 25), (50, 38),
(16, 38), (66, 49)]
Closest pair: ((62, 52), (66, 49)) with distance: 5.0
PS C:\Users\Mugilkrishna D U\OneDrive\Desktop\My Files\.vscode> & "C:/Users/Mugilkrishna D U/AppData/Local
/Programs/Python/Python311/python.exe" "c:/Users/Mugilkrishna D U/OneDrive/Desktop/My Files/SSN/SEM4/DESIG
N AND ANALYSIS OF ALGORITHMS/LAB/5.3.py"
Randomly generated points: [(79, 0), (41, 22), (44, 89), (81, 52), (34, 11), (73, 63), (99, 73), (32, 67),
 (38, 64), (53, 66)]
Closest pair: ((32, 67), (38, 64)) with distance: 6.708203932499369
```

Time Complexity:

- **Time Complexity of closed_pair_dc function:** O(n*logn)
- **Time Complexity of brute_force_closest:** O(n^2)

Learning Outcome:

Upon completing this exercise, I have understood the applications of insertion sort and quick sort algorithms in finding the kth smallest element and also the principles of divide and conquer strategy in finding the Euclidean distance and the sum of a binary tree