Design analysis and algorithm

Lab 10

Divide And Conquer Nearest Neighbour

Name: Naomi George

Slot: L29

Reg No: 19bce7572

Course code: CSE3004

Nearest Neighbour:

```
import math
import copy
class Point():
    def __init__(self, x, y):
        self.x = x
        self.y = y
def dist(p1, p2):
    return math.sqrt((p1.x - p2.x) *
                     (p1.x - p2.x) +
                     (p1.y - p2.y) *
                     (p1.y - p2.y))
def bruteForce(P, n):
    min_val = float('inf')
    for i in range(n):
        for j in range(i + 1, n):
            if dist(P[i], P[j]) < min_val:
                min_val = dist(P[i], P[j])

    return min_val
def stripClosest(strip, size, d):
    min_val = d
    for i in range(size):
        j = i + 1
        while j < size and (strip[j].y -
                            strip[i].y) < min_val:
            min_val = dist(strip[i], strip[j])
            j += 1

    return min_val
def closestUtil(P, Q, n):
    if n <= 3:
        return bruteForce(P, n)
    mid = n // 2
    midPoint = P[mid]
    Pl = P[:mid]
    Pr = P[mid:]
    dl = closestUtil(Pl, Q, mid)
    dr = closestUtil(Pr, Q, n - mid)

    d = min(dl, dr)
    stripP = []
    stripQ = []
    lr = Pl + Pr
    for i in range(n):
        if abs(lr[i].x - midPoint.x) < d:
```

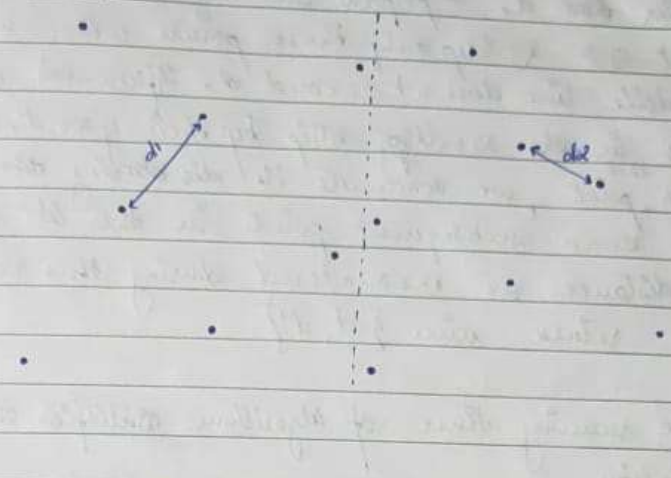
```
            stripP.append(lr[i])
        if abs(Q[i].x - midPoint.x) < d:
            stripQ.append(Q[i])

    stripP.sort(key = lambda point: point.y)
    min_a = min(d, stripClosest(stripP, len(stripP), d))
    min_b = min(d, stripClosest(stripQ, len(stripQ), d))

    return min(min_a,min_b)
def closest(P, n):
    P.sort(key = lambda point: point.x)
    Q = copy.deepcopy(P)
    Q.sort(key = lambda point: point.y)
    return closestUtil(P, Q, n)

P = [Point(2, 3), Point(12, 30),
     Point(40, 50), Point(5, 1),
     Point(12, 10), Point(3, 4)]
n = len(P)
print("The smallest distance is",closest(P, n))
```

Output:

```
The smallest distance is 2.23606797749979
```

Analysis:

Nearest Neighbour:

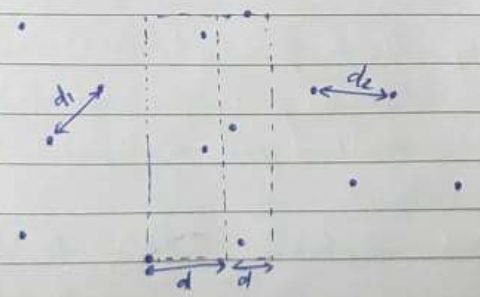


To solve this we first split the given 'n' points by an appropriately chosen vertical line into 2 halves S_1 & S_2 of size $n/2$.

By making 2 recursive calls for the sets S_1 & S_2, we find the minimum distances d_1 & d_2 in these subsets.

Let $d = \min\{d_1, d_2\}$

but this way takes a lot of time thus not providing us the goal. What we do is, part the strip into $d \times d$ squares & show that each such square contains at most four input points.



then we sort n points by their x-coordinate & then split the resulting sorted list into 2 halves S_1 & S_2 of size $n/2$.

Calculate min of d_1 & d_2 distances & check whether it is

Smaller than d. To perform such a check, we filter the initial point set & keep only those points whose x-distance to the middle line does not exceed d. Afterwards we sort the set of points in the resulting strip by their y coordinates & scan the list. For each point, we compute its distance to the seven subsequent points in this list of points. For any distance we encountered during this scan, afterwards, we return $\min\{d, d'\}$

The running time of algorithm satisfies the recurrence relation,

$$T(n) = 2T(n/2) + O(n \log n)$$

The $O(n \log n)$ term comes from sorting the points in the strip by their y-coordinates at every iteration.