Advanced Algorithms Analysis and Design

By

Nazir Ahmad Zafar

Lecture No 13

Designing Algorithms using Brute Force and Divide & Conquer Approaches

Today Covered

Brute Force

- Finding closest pair in 2-D
- Improved version finding closest pair in 2-D
- Generalization in 3-D and then n-D
- Finding maximal points in n-D

Divide and Conquer

- A General Divide and Conquer approach
- Merge Sort algorithm
- Finding Maxima in 1-D, and 2-D
- Finding Closest Pair in 2-D

The Closest Pair Problem

Finding Closest Pair in 2-D

Problem

The closest pair problem is defined as follows:

- Given a set of n points, determine the two points that are closest to each other in terms of distance. Furthermore, if there are more than one pair of points with the closest distance, all such pairs should be identified.

Input :

is a set of n points

Output

- is a pair of points closest to each other,
- there can be more then one such pairs

Definition: Closest Pair

Distance

- In mathematics, particular in geometry, distance on a given set M is a function $d: M \times M \rightarrow R$, where R denotes the set of real numbers, that satisfies the following conditions:

1. $d(x, y) \geq 0$,
2. $d(x, y) = 0$ if and only if $x = y$.
3. Symmetric i.e.
   $d(x, y) = d(y, x)$.
4. Triangle inequality:
   $d(x, z) \leq d(x, y) + d(y, z)$.

Finding Closest Pair in 2-D

Closest Pair Problem in 2-D

- A point in 2-D is an ordered pair of values (x, y).
- The Euclidean distance between two points

  $P_i = (x_i, y_i)$ and $P_j = (x_j, y_j)$ is

  $d(p_i, p_j) = \text{sqr}((x_i - x_j)^2 + (y_i - y_j)^2)$
- The closest-pair problem is finding the two closest points in a set of n points.
- The brute force algorithm checks every pair of points.
- Assumption: We can avoid computing square roots by using squared distance.
  - This assumption will not loose correctness of the problem.

Brute Force Approach: Finding Closest Pair in 2-D

ClosestPairBF(P)

```
1. mind ← ∞
2. for i ← 1 to n
3. do
    4. for j ← 1 to n
    5. if i ≠ j
    6. do
    7. d ← $((x_i - x_j)^2 + (y_i - y_j)^2)$
    8. if d < mind then
        8. mind ← d
        9. mini ← i
        10.minj ← j
11.return mind, p(mini, minj)
```

Time Complexity

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} c$$

$$= \sum_{i=1}^{n} cn$$

$$= cn^2$$

$$= \Theta(n^2)$$

Improved Version
(The Closest Pair Problem)

Improved Version: Finding Closest Pair in 2-D

ClosestPairBF(P)

```
1.  mind ← ∞
2.  for i ← 1 to n − 1
3.  do
        4. for j ← i + 1 to n
        5. do
        6. d ← $((x_i - x_j)^2 + (y_i - y_j)^2)$
        7. if d < mind then
            8. mind ← d
            9. mini ← i
            10.minj ← j
11.return mind, p(mini, minj)
```

Time Complexity

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c$$

$$= \sum_{i=1}^{n-1} c(n-i)$$

$$= c\left(\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i\right)$$

$$= cn(n-1) - c\frac{(n-1)n}{2}$$

$$= c\left(n^2 - n - \frac{n^2}{2} + \frac{n}{2}\right)$$

$$= c\left(\frac{n^2}{2} - \frac{n}{2}\right) = \Theta(n^2)$$

The Closest Pair Problem 3-D

Finding Closest Pair in 3-D

ClosestPairBF(P)

```
1. mind $\leftarrow \infty$
2. **for** i $\leftarrow$ 1 to n − 1
3. **do**
    4. **for** j $\leftarrow$ i + 1 to n
    5. **do**
    6. d $\leftarrow ((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)$
    7. **if** d < minn **then**
        8. mind $\leftarrow$ d
        9. mini $\leftarrow$ i
        10.minj $\leftarrow$ j
11.**return** mind, p(mini), p(minj)
```

Time Complexity

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c$$

$$= \sum_{i=1}^{n-1} c(n-i)$$

$$= c\left(\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i\right)$$

$$= \Theta(n^2)$$

The Closest Pair Problem n-D

Finding Closest Pair in n-D

```
ClosestPairBF(P)
    1. mind ← $\infty$
    2. for i ← 1 to n − 1
    3. do
        4. for j ← i + 1 to n
        5. do
        6. d ← $((x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{in} - x_{jn})^2)$
        7. if d < minn then
            8. mind ← d
            9. mini ← i
            10.minj ← j
    11.return mind, p(mini), p(minj)
```

Time Complexity

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} cn$$

$$= \sum_{i=1}^{n-1} cn(n-i)$$

$$= c\left(\sum_{i=1}^{n-1} n^2 - \sum_{i=1}^{n-1} in\right)$$

$$= \Theta(n^3)$$

Finding Maximal in n-dimension

Maximal Points

- Maximal Points in 2-D

  A point p is said to be dominated by q if

  $p.x \leq q.x$ and $p.y \leq q.y$

  A point p is said to be maximal if

  $p.x > q.x$ OR $p.y > q.y$

- Maximal Points in n-D

  A point p is said to be dominated by q if

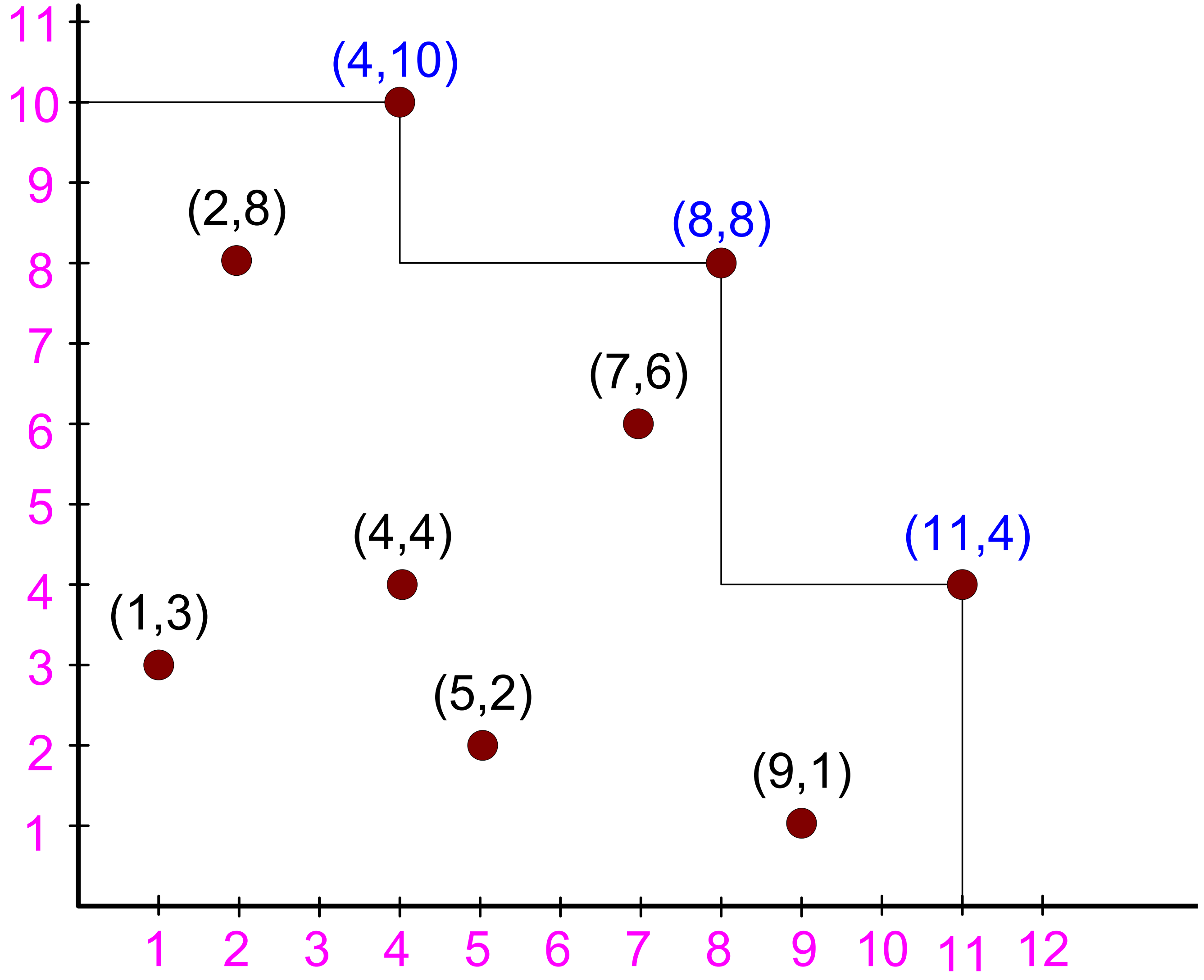
  $p.x_i \leq q.x_i \;\forall\; i = 1, . . ., n$

  A point p is said to be maximal if

  $\exists\; i = 1, . . ., n, \; p.x_i > q.x_i$

  A point is said to be maximal if it is not dominated by any other point.

Example: Maximal Points in 2-Dimension



Problem Statement

Problem Statement:

Given a set of m points, $P = \{p_1, p_2, . . . , p_m\}$, in n-dimension. Our objective is to compute a set of maximal points i.e. set of points which are not dominated by any one in the given list.

Mathematical Description:

Maximal Points =

$$\{ p \in P \mid \forall\ q \in \{p_1, . . . , p_m\}, q \neq p, \exists\ i \in \{1,. . . , n\}\ \&\ p.x_i \geq q.x_j\}$$

Brute Force Algorithm in n-dimension

```
MAXIMAL-POINTS (int n, Point P[1. . . m])
0  A = ∅;
1  for i ← 1 to m          \\ m used for number of points
2  do maximal ← true
3       for j ← 1 to m
4       do
5            if (i ≠ j) &
6            for k ← 1 to n          \\ n stands for dimension
7            do
8                 P[i].x[k] ≤ P[j].x[k]
9                      then maximal ← false; break
10      if maximal
11           then A = A ∪ P[i]
```

Conclusion

- Designing Algorithms using Brute Force approach is discussed
- For Brute Force, formally, the output of any sorting algorithm must satisfy the following two conditions:
  - Output is in decreasing/increasing order and
  - Output is a permutation, or reordering, of input.
- Algorithms computing maximal points can be considered as generalization of sorting algorithms
- Maximal points are useful in Computer Sciences and Mathematics in which at least one component of every point is dominated over all points.
- In fact we put elements in a certain order