

CCCS 314 – Design and Analysis of Algorithms

LAB 2

[PLO K2, CLO 2.2]

Topic: Brute Force Algorithms

1. Brute Force: String Matching
2. Brute Force: Closest-Pair Problem
3. Validation of String-Matching Code
4. Validation of Closest-Pair Problem's Code

Student Name: Khalid Nimri

Student ID: 2140145

Exercise 1: (Brute Force: String Matching)

PART A: The brute force algorithm for string matching is given below:

ALGORITHM *BruteForceStringMatch*($T[0..n-1]$, $P[0..m-1]$)

```
//Implements brute-force string matching
//Input: An array  $T[0..n-1]$  of  $n$  characters representing a text and
//       an array  $P[0..m-1]$  of  $m$  characters representing a pattern
//Output: The index of the first character in the text that starts a
//        matching substring or  $-1$  if the search is unsuccessful
for  $i \leftarrow 0$  to  $n - m$  do
     $j \leftarrow 0$ 
    while  $j < m$  and  $P[j] = T[i + j]$  do
         $j \leftarrow j + 1$ 
    if  $j = m$  return  $i$ 
return  $-1$ 
```

Write a code to implement this algorithm in the language of your choice. **Paste your complete code here:**

```
def bruteForceStringMatch(T, P):
    n = len(T)
    m = len(P)
    for i in range(n-m+1):
        j=0
        while j < m and T[i+j] == P[j]:
```

```

j += 1 if j == m:

return i

return -1

n = input("Enter text string: ")
m = input("Enter pattern string: ")
result = bruteForceStringMatch(n, m)
print("Pattern found at index:", result)

```

PART B: Run your code for the following inputs:
T = akfkdfdsfsfsdfhsdfhsdfhsdfhsdfhsdfhsdfhsdfjksd
P = dsfh

Write the output of your code below and attach its output as screenshot.

The screenshot shows the OnlineGDB web IDE interface. The code editor displays the following Python code:

```

1 def bruteForceStringMatch(T, P):
2     n = len(T)
3     m = len(P)
4     for i in range(n-m+1):
5         j = 0
6         while j < m and T[i+j] == P[j]:
7             j += 1
8         if j == m:
9             return i
10    return -1
11
12 n = input("Enter text string: ")
13 m = input("Enter pattern string: ")
14 result = bruteForceStringMatch(n, m)
15 print("Pattern found at index:", result)
16
17

```

The input section shows the following inputs:

```

Enter text string: akfkdfdsfsfsdfhsdfhsdfhsdfhsdfhsdfhsdfjksd
Enter pattern string: dsfh

```

The output section shows the following output:

```

Pattern found at index: 23

```

The status bar at the bottom indicates: "...Program finished with exit code 0. Press ENTER to exit console."

Exercise 2: (Brute Force: Closest Pair)

PART A: The brute force algorithm for finding closest pair of points in 2D space is given below:

ALGORITHM *BruteForceClosestPair(P)*

//Finds distance between two closest points in the plane by brute force

//Input: A list P of n ($n \geq 2$) points $p_1(x_1, y_1), \dots, p_n(x_n, y_n)$

//Output: The distance between the closest pair of points

$d \leftarrow \infty$

for $i \leftarrow 1$ **to** $n - 1$ **do**

for $j \leftarrow i + 1$ **to** n **do**

$d \leftarrow \min(d, \text{sqrt}((x_i - x_j)^2 + (y_i - y_j)^2))$ //sqrt is square root

return d

Write a code to implement this algorithm in the language of your choice. **Paste your complete code here:**

```
import math

def bruteForceClosestPair(points):
    n = len(points)
    min_dist = float('inf')
    for i in range(n):
        for j in range(i+1, n):
            dist = math.sqrt((points[i][0]-points[j][0])**2 +
                             (points[i][1]-points[j][1])**2)
            if dist < min_dist:
                min_dist = dist
                closest_pair = (points[i], points[j])
    return closest_pair, min_dist

# Example usage
points = [(1,2), (3,4), (5,6), (7,8)]
closest_pair, min_dist = bruteForceClosestPair(points)
print("Closest pair:", closest_pair) print("Distance:",
min_dist)
```

PART B: Run your code for the following inputs:

$p = [(3, 2), (3, 6), (7, 12), (3, 1), (9, 18), (8, 56)]$;

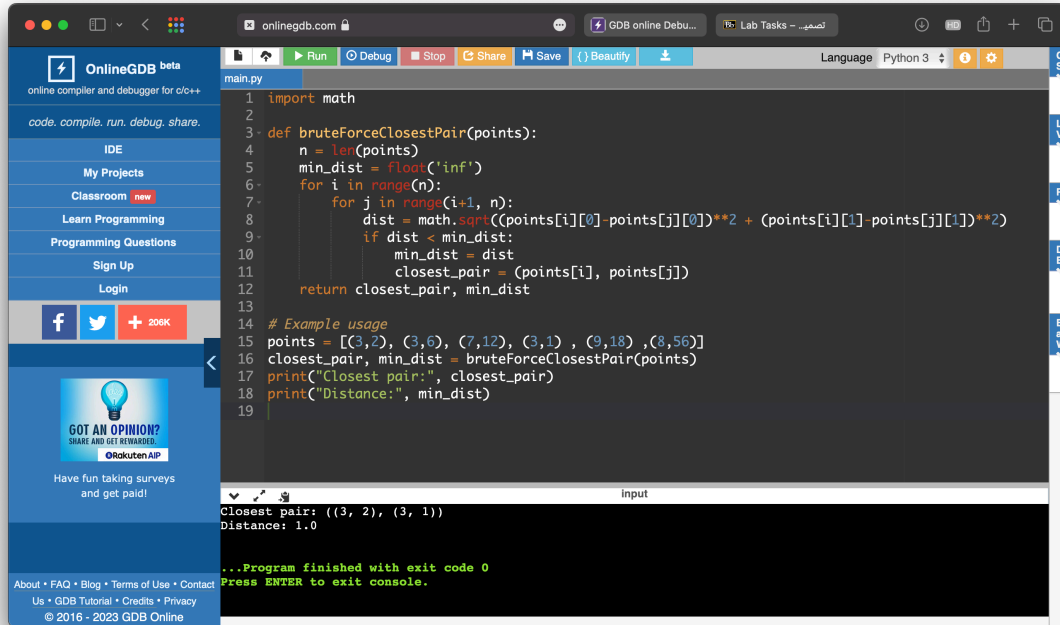
Write the output of your code below and **attach its output as screenshot**.

the closest points are:

?

and their distance is:

?



```
1 import math
2
3 def bruteForceClosestPair(points):
4     n = len(points)
5     min_dist = float('inf')
6     for i in range(n):
7         for j in range(i+1, n):
8             dist = math.sqrt((points[i][0]-points[j][0])**2 + (points[i][1]-points[j][1])**2)
9             if dist < min_dist:
10                 min_dist = dist
11                 closest_pair = (points[i], points[j])
12     return closest_pair, min_dist
13
14 # Example usage
15 points = [(3,2), (3,6), (7,12), (3,1), (9,18), (8,56)]
16 closest_pair, min_dist = bruteForceClosestPair(points)
17 print("Closest pair:", closest_pair)
18 print("Distance:", min_dist)
19
```

Input

Closest pair: ((3, 2), (3, 1))
Distance: 1.0

...Program finished with exit code 0
Press ENTER to exit console.

Exercise 3 and 4:

You will run the above codes on given inputs and write your answers in the provided space on blackboard.