

# DS Lab - Hashing

## Problem 01:

**Finding whether an array is a subset of another array.**

### Problem definition:

Suppose, you are given two arrays, namely, `a_array[0...m-1]` and `b_array[0...n-1]`. You need to find whether **b\_array** is a subset of **a\_array**. Both arrays are not in sorted order and the elements in both arrays are **distinct**.

### **Input:**

The first line contains integer  $m$  ( $0 < m < 100$ ) and  $n$  ( $0 < n < m$ ) which corresponds to the size of `a_array` and `b_array`. Then the following two lines contain the input of the elements of the arrays.

### **Output:**

Print a message in the console.

**NOTE:** You must use **HashMap** in this problem.

### **Examples:**

#### **Input:**

```
10 3
12 14 9 7 22 1 0 45 39 5
7 5 6
```

#### **Output:**

```
b_array is a subset of a_array
```

#### **Input:**

9 4

1 2 9 88 41 0 17 11 6

100 52 2 8

**Output:**

b\_array is not a subset of a\_array

**Input:**

4 10

**Output:**

Invalid input

**Input:**

4000 100

**Output:**

Invalid input

### Problem 02:

#### **Most frequent element in an array.**

#### Problem definition:

Given an array: `c_array[0...m-1]` you need to find the most frequent element in it. The elements of the array must be distinct. There can be multiple elements that appear maximum in the array.

#### **Input:**

The first line contains integer  $m$  ( $0 < m < 100$ ) which corresponds to the size of the array. The following line contains the elements of the array.

#### **Output:**

Print the most frequent element of the array as well as its frequency. If there are multiple elements that appear maximum number of times then print only one of them.

**NOTE:** The most efficient solution for this problem is to use hashing. So you must use **HashMap** in this problem.

#### **Examples:**

##### **Input:**

11

0 1 1 2 3 5 8 13 21 34 55

##### **Output:**

Most frequent element: 1

Frequency: 2

**Input:**

-1

**Output:**

Invalid input

**Input:**

0

**Output:**

Invalid input

### Problem 03:

#### **Counting pairs with a given sum.**

#### Problem definition:

Given a set of integers and a number "**sum**", find the pairs of integers in the set whose sum is equal to the given "**sum**". The elements of the set may not be unique.

#### **Input:**

The first line contains integer  $m$  ( $0 < m < 100$ ) which corresponds to the size of the set of integers/array. The following line contains the elements of the array. And the final line is the input of the "**sum**" value.

#### **Output:**

Print the pairs whose sum equals to the "**sum**".

**NOTE:** You must use **Hashmap** in this problem.

#### **Examples:**

##### **Input:**

10

2 5 4 12 9 1 3 17 11 8

13

##### **Output:**

Pairs with sum 13 are: (8, 5), (12, 1), (9, 4), (11, 2)

##### **Input:**

7

2 5 4 2 0 1 3

4

**Output:**

Pairs with sum 4 are: (2, 2), (3, 1), (4, 0)

**Input:**

9

4 -2 2 7 9 1 3 1 0

7

**Output:**

Pairs with sum 7 are: (4, 3), (7, 0), (9, -2)

**Input:**

-1

**Output:**

Invalid input

**Input:**

10

2 5 4 12 9 1 3 17 11 8

-1

**Output:**

No pairs found

Problem 04:

### **Pattern Matching.**

Problem definition:

Given a **text**[0...m-1] and a **pattern**[0...n-1], where ( $m > n$ ) you need to find all the occurrences of that **pattern** in the **text**.

#### **Input:**

The first line contains the **text**. The following line contains the **pattern**.

#### **Output:**

Print at which index/indices the pattern occurs.

**NOTE:** Apply **Rabin-Karp** algorithm to get the solution of this problem. And You must use a Hashmap in this problem.

#### **Examples:**

##### **Input:**

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG  
FOX

##### **Output:**

Pattern found at index: 16

##### **Input:**

AABADBAACABAABADAABAABA

AABA

**Output:**

Pattern found at index: 0

Pattern found at index: 9

Pattern found at index: 16

Pattern found at index: 19

**Input:**

CCACCAACDBA

DBA

**Output:**

Pattern found at index: 8

**Input:**

SAY MY NAME

HEISENBERG

**Output:**

Pattern not found!