**Phần 1: Cơ bản Về Giải Thuật**

**Bài 1: giới thiệu về giải thuật**

**Thuật toán Euclid**

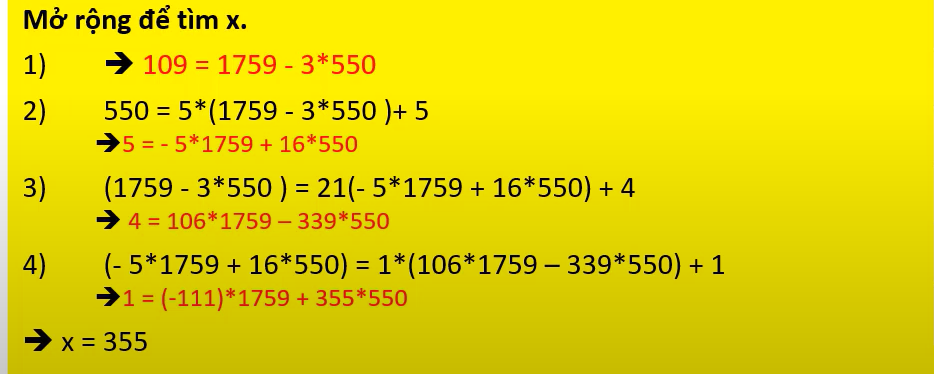
Giả sử cần tìm x sao cho

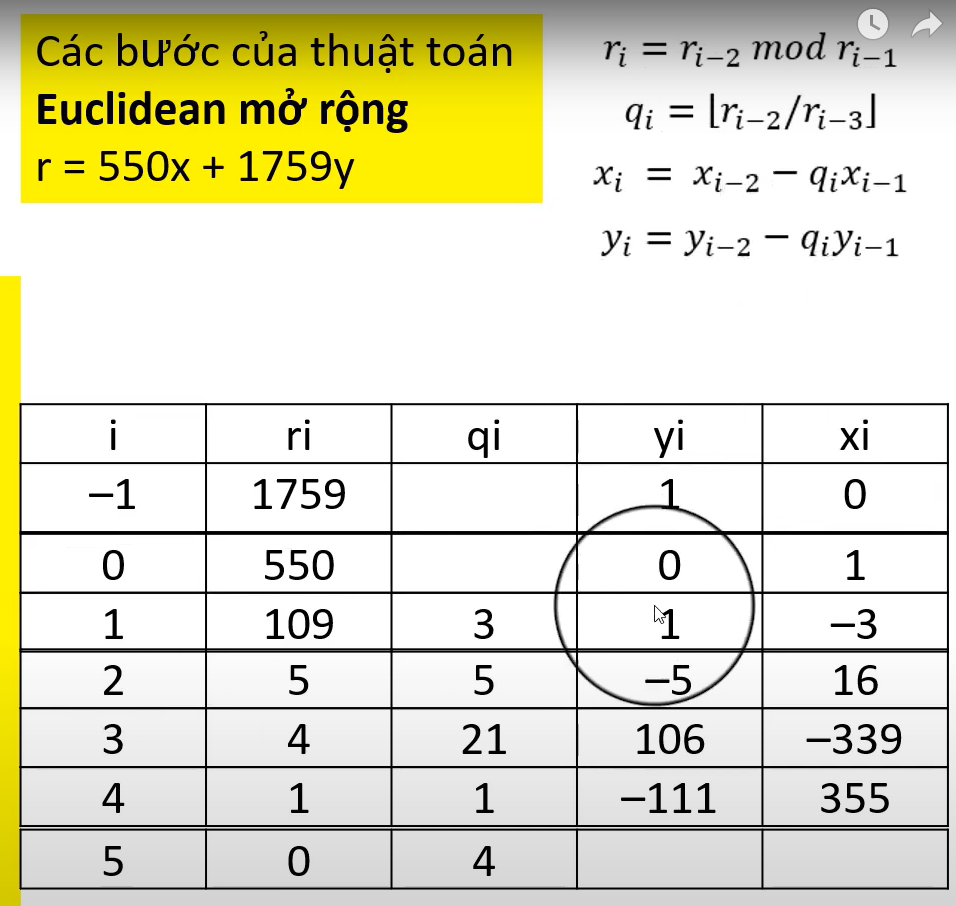
550x = 1 (mod 1759), hay nói cách khác là cần tìm biểu diễn

550x + 1759y = 1

Ta sẽ tìm USCLN:

* Biểu diễn số lớn qua số nhỏ: **1759** = 3 \* **550** + 109
* Biểu diễn số nhỏ qua số dư: **550** = 5 + 109 + 5
* Biểu diễn **109** qua số dư của biểu thức ở trên:: **109** = 21 \* 5 + 4
* Tiếp tục: 5 = 4 \* 1 + 1
* Tiếp tục: 4 = 4 \* 1 + 0





Thuật toán Euclid dùng để xác định các hệ số trong đẳng thức Bezout

Bổ đề Bezout: nếu **d** là ước số chung lớn nhất của hai số nguyên **a** và **b**, thì sẽ tồn tại hai số nguyên **x** và **y** sao cho:

d = a\*x + b \*y

**--------------- --------------- --------------- ---------------**

**Thuật toán nổi bọt**

package com.example.sort**;**import java.util.Scanner**;**public class BubbleSort {  
 static void swap(int[] arr**,** int i**,** int j) {  
 var temp = arr[i]**;** arr[i] = arr[j]**;** arr[j] = temp**;** }  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*)**;** System.*out*.print("Enter the number of elements you want to store: ")**;** int n = sc.nextInt()**;** int[] array = new int[n]**;** System.*out*.println("Input your array from keyboard:")**;** for (int i = **0;** i < n**;** i++) {  
 array[i] = sc.nextInt()**;** }  
  
 for (int i = **0;** i < array.length**;** i++) {  
 for (int j = array.length - **1;** j > i**;** j--) {  
 if (array[j] < array[j - **1**]) {  
 *swap*(array**,** j**,** j- **1**)**;** }  
 }  
 }  
  
 System.*out*.println("New array after sorted")**;** for (int j : array) {  
 System.*out*.println(j)**;** }  
 }  
}

**--------------- --------------- --------------- ---------------**

**Độ chính xác của thuật toán**

**--------------- --------------- --------------- ---------------**

**Giới thiệu về cấu trúc dữ liệu**

Trong khoa hoc máy tính, cấu trúc dữ liệu là một cách tổ chức dữ liệu trong máy tính sao cho nó có thể được sử dụng một cách hiệu quả cho các thuật toán tác động đến nó

Trong việc thiết kế chương trình, việc chọn cấu trúc dữ liệu là vấn đề quan trọng, kinh nghiệm cho thấy khó khăn của việc triển khai chương trình, chất lượng và hiệu năng của kết quả cuối cùng phụ thuộc rất nhiều vào việc chọn CTDL phù hợp

**Quiz**

**1 A**

**2 A**

**3 A + B**

**4 B + D**

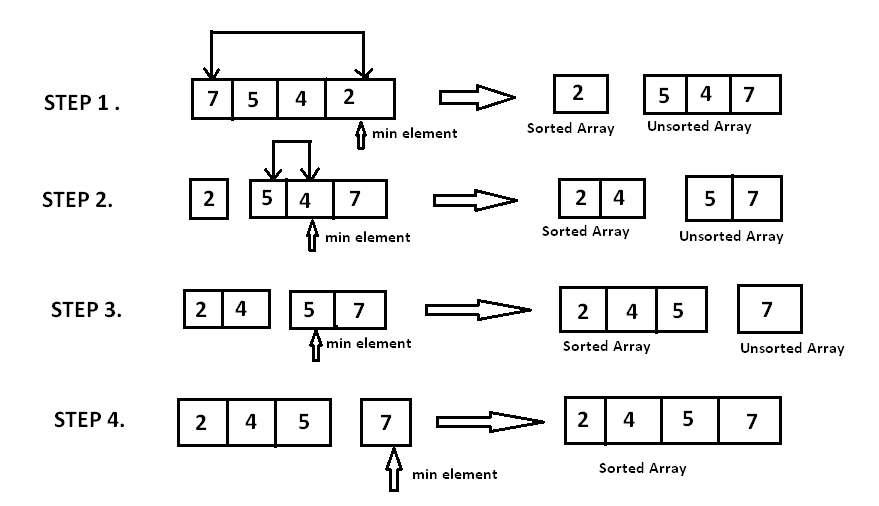
**5 A + B + C**

**--------------- --------------- --------------- --------------- ---------------**

**Thuật toán tìm kiếm và sắp xếp cơ bản**

**Sắp sếp chọn (selection sort)**

Danh sách chứa các phần tử sẽ được chia làm hai phần, phần bên trái là phần được sắp xếp (sort list) và phần bên phải là phần chưa được sắp xếp (unsorted list). Phần tử nhỏ nhất trong list sẽ được chọn và tráo đổi với vị trí đầu tiên, tiếp đến là phần tử nhỏ thứ 2 sẽ được đặt ở vi jtris thử 2 …

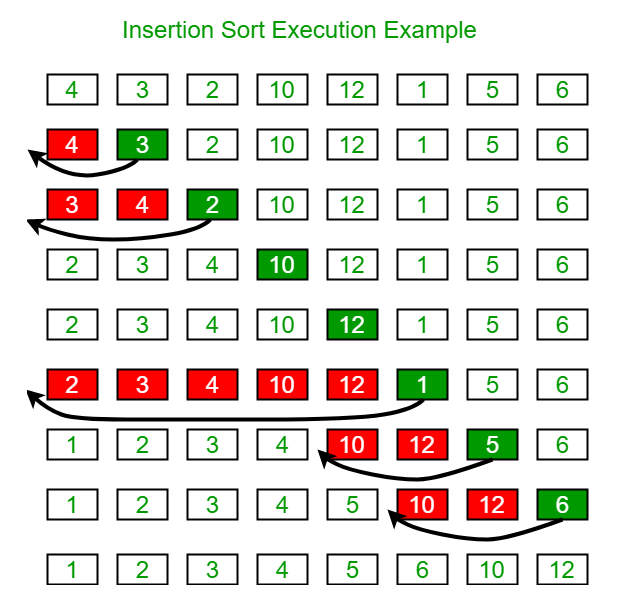


**--------------- --------------- --------------- ---------------**

**Thuật toán chèn (Insertion sort)**

Sắp xếp chèn là thuật toán sắp xếp khi đã có một dãy đã có thử tự, chèn thêm 1 phần tử vào vị trí thích hợp của dãy số đã sắp xếp sao cho dẫy số đó vẫn là dãy có thứ tự.

\



**--------------- --------------- --------------- ---------------**

**Sắp xếp tại chỗ**

**In place (tại chỗ) sorting algorithms** là những thuật toán mà khi chúng ta sắp xếp dữ liệu trong mảng mà không cần sử dụng thêm bộ nhớ bổ sung, điều này có nghĩa là chúng ta sẽ không cần sử dụng các biến tạm để đổi vị trí các element, như vậy tức là có một vùng bộ nhớ đã được sử dụng ngoài dữ liệu lưu trữ của mảng. Các implementations của chúng ta là IN PLACE, có điều nếu chúng ta sử dụng một lượng không đổi bộ nhớ thêm, như trong ví dụ của chúng ta, chúng ta chỉ sử dụng tạm thời một biến – đó có thể đã được định nghĩa một lần bên ngoài vòng lặp và tái sử dụng, việc sắp xếp này là in place

**--------------- --------------- --------------- ---------------**

**Sắp xếp với Stable và Unstable**

Khi sắp xếp các phần tử trong 1 mảng có rất nhiều phần tử có giá trị sắp xếp giống nhau, vậy sau khi sắp xếp mà trình tự của các phần tử **có giá trị giống nhau không thay đổi** thì gọi là stable, ngược lại là unstable