**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

**ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ**

**КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ**

**Лабораторна робота № 2**

**Обчислювальна складність алгоритму сортування**

Підготував

студент групи ІП-64, ФІОТ

Вінницький В’ячеслав

Київ – 2018

**Завдання**

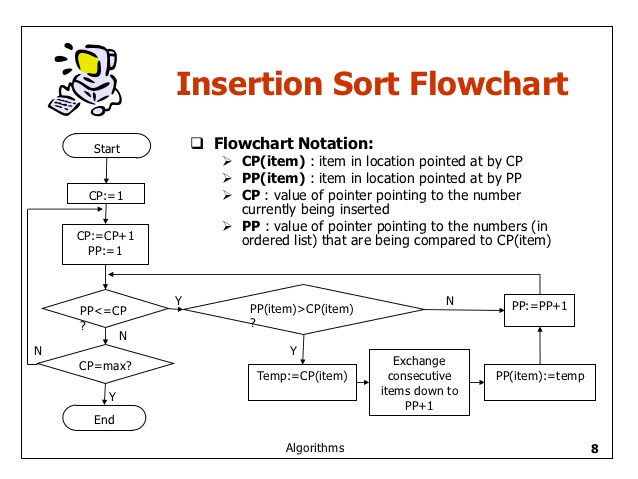
**Завдання**: Використовуючи відповідний до варіанту алгоритм сортування написати програму сортування масиву даних. Застосовуючи дану програму, дослідити часову складність алгоритму сортування та порівняти її з теоретичною алгоритмічною складністю.

**Порядок виконання лабораторної роботи**

1. Написати програму сортування у відповідності з вибраним варіантом.
2. У програмі передбачити вимірювання часу сортування з максимально можливою точністю.
3. Створити 10 тестових масивів даних різного розміру, які підлягають сортуванню.
4. Побудувати графік залежності часу виконання алгоритму від розміру вхідного масиву даних.
5. Побудувати графік теоретично відомої обчислювальної складності.
6. Порівняти одержані графіки часової складності та обчислювальної складності вашого алгоритму сортування.

|  |  |  |
| --- | --- | --- |
| Вар. | Назва алгоритму | Алгоритмічна складність |
| 2 | Алгоритм 2.1 Сортування вставками |  |

**Блок-схема алгоритму**



**Текст програми**

**INDEX.PHP**

<html>

<head>

<meta charset = "UTF-8">

<title>2 Лабораторна</title>

</head>

<body>

<form action = "" method = "get">

<ul class = "form-style-1">

<li>

<label>Число елементів</label>

<input type = "number" name="capacity" class = "field-divided"/>

<input type = "submit" value = "Generate" name="generate">

<!--<input type = "submit" value = "Sort" name="sort">-->

</li>

</ul>

</form>

<?php

ini\_set("display\_errors",1);

//include\_once "Sorting.php";

include\_once "FillArray.php";

// генерируем

if (filter\_input(INPUT\_GET,"generate")) {

echo "<p class='textout'><b>Згенерований масив : </b></p>";

$array = fillArray();

echo json\_encode($array);

}

//выводим

if (filter\_input(INPUT\_GET,"capacity")) {

$size = filter\_input(INPUT\_GET, "capacity");

echo "<p class='textout'><b>Його розмір :</b> $size</p><br>";

}

if (filter\_input(INPUT\_GET,"generate")) {

//сортируем

$start = microtime(true);

sort($array, SORT\_NUMERIC);

$time = microtime(true) - $start;

echo "<p class='textout'><b>Время выполнения :</b> $time секунд</p><p class='textout'><b>Відсортований масив : </b>". json\_encode($array)."</p>";

}

?>

</body>

<link rel="stylesheet" href="css/main.css">

</html>

**FillArray.php**

function fillArray() {

if (isset($\_GET["capacity"]) && isset($\_GET["generate"])) {

$array = array();

for ($i = 0; $i < $\_GET["capacity"]; $i++) {

array\_push($array, mt\_rand(1, 10));

}

return $array;

}

}

**Sorting.php**

<?php

function insertionSort($arr, $length) {

if(isset($\_POST["sort"])) {

// $length = count($arr);

for ($i = 1; $i < $length; $i++) {

$element = $arr[$i];

$j = $i;

while ($j > 0 && $arr[$j - 1] > $element) {

//move value to right and key to previous smaller index

$arr[$j] = $arr[$j - 1];

$j = $j - 1;

}

//put the element at index $j

$arr[$j] = $element;

}

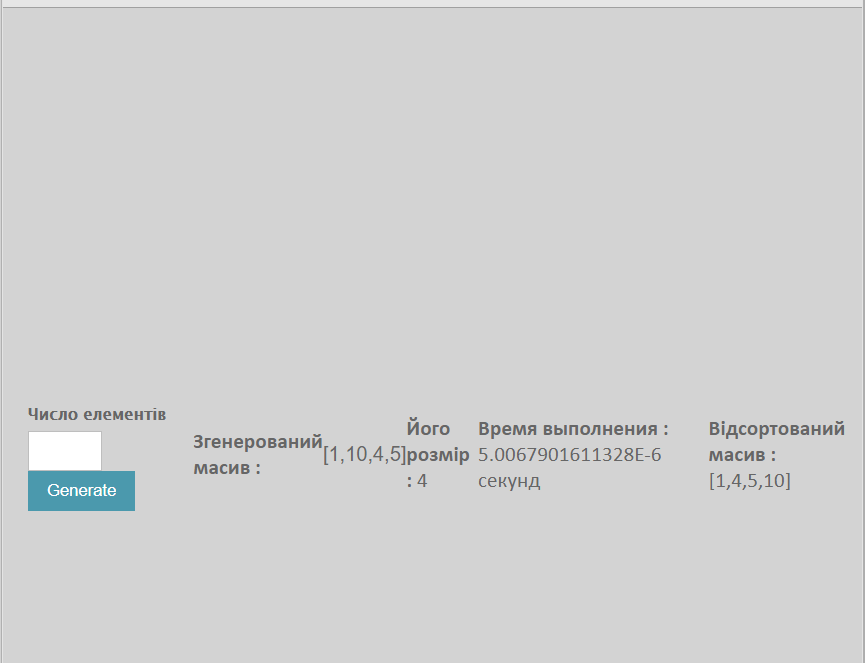
return $arr;

}

}

**Графіки**

**Результати виконання програми**



**Висновки**

На цій лабораторній роботі я створив функція, яка сортує заданий масив методом злиття , побудував графік залежності часу виконання алгоритму від розміру вхідного масиву даних, знайшов побудований графік теоретично відомої обчислювальної складності даного методу. Порівнявши одержані графіки часової складності та обчислювальної складності мого алгоритму сортування, я побачив, що тенденції схожі, але обчислювальна складність ніби більш «різка».