**Лабораторна робота №5**

**Спеціальні алгоритми сортування**

**Завдання 1.** «Обрання» алгоритмів

Дано алгоритм сортування Counting Sort та Radix Sort

**Завдання 2.** Складність алгоритмів та їх псевдокоди

**Сортування підрахунком**

Max = max(array) n

valueArray = new array[max] 1

outputArray = new array[inputArray.length] 1

for(i = 0; i < inputArray.length; i++) n

valueArray[inputArray[i]]++ 1

for(I = 0; i< valueArray.Length;i++) n

k = valueArray[i] 1

for(n = 0; n < k; n++) n

list.add(i) 1

outputArray = List.toArray(); n

O = O(n + n +n +n +n +5) = O(4n +5) = O(n)

Клас складності – лінійний.

**Сортування порозрядно**

For(exp = 1; max / exp > 0; exp \*=10) k

Output = array[length] 1

Count = array[10] 1

For(I = 0; I < length; i++) n

Count[(array[i]/exp)%10]++ 1

For(I =0; I < 10; i++) n

Count[i] += count[i-1] 1

For(I = length -1; I >= 0; i--) n

Output[count[(array[i]/exponent)%10]-1] = array[i] 1

Count[array[i]/exponent)%10] -- 1

For(I = 0; i< length;I++) n

Array[i] = output[i] 1

O() = O(k\*(6 + 4n)) = O(k\*n)

Складність алгоритму O(k\*n) к- найбільший розряд числа

**Завдання 3-4.** Створення програми

Нами створена програма, яка реалізує обрані алгоритми та була перевірена правильність її виконання.

**Завдання 5.** Тестові дані

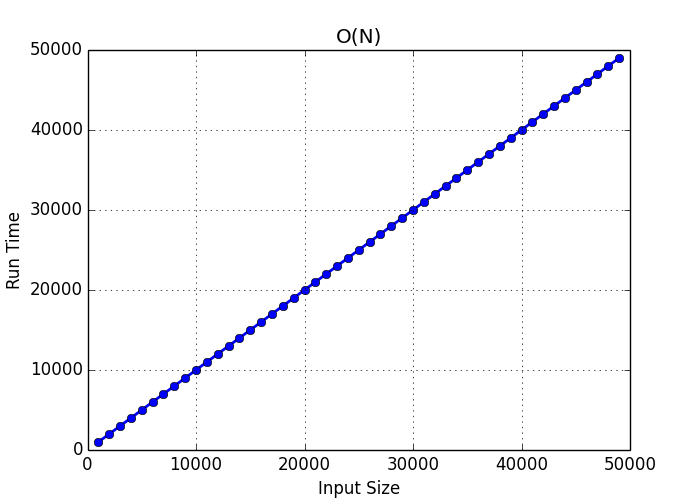
Під час виконання програма генерує заданий діапазон чисел від 0 да 10000

Ми генерували к-сть чисел від 1млн до 10 млн з кроком в 1 млн

**Завдання 6-7.** Час виконання та графіки

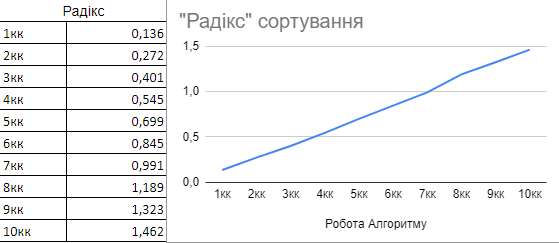
Очікуваний час виконання:

RadixSort та CountingSort:



Час виконання алгоритмів на практиці:

RadixSort:



CountingSort

