ЗАТВЕРДЖУЮ
Завідувач кафедри управління проектами, інформаційних технологій та телекомунікацій, д.т.н., професор
Є. В. Мартин
«_ _ »_____20___ р.

МЕТОДИЧНА РОЗРОБКА

для проведення практичного заняття з студентами (курсантами) 1-го курсу спеціальності 122 «Комп'ютерні науки» з дисципліни «Основи програмування та алгоритмізація»

Практичне заняття 4.10: «Реалізація алгоритму пошуку порядкової статистики»

Мета: здобути навики щодо реалізації алгоритму пошуку порядкової статистики.

Час: 2 годин

Місце заняття: комп'ютерна лабораторія

Матеріальне забезпечення:
- ПК з відповідним ПЗ та доступом до мережі Internet;
- Методичні рекомендації щодо виконання практичної роботи.

Після практичного заняття студенти (курсанти) повинні:
вміти: реалізовувати алгоритм пошуку порядкової статистики довільного елемента в масиві.
знати: особливості застосування алгоритму пошуку порядкової статистики на основі методу декомпозиції.

Література:
1. Віртуальне навчальне середовище ЛДУ БЖД «Віртуальний університет» [Електронний ресурс]. – Доступний з : http://virt.ldubgd.edu.ua/course/view.php?id=1672

План заняття:
1. Складання тестових завдань за темою попередньої лекції.
2. Реалізація алгоритму пошуку порядкової статистики мовою програмування Java.

1. Складання тестових завдань за темою попередньої лекції.
Для складання тестових завдань необхідно перейти у відповідний розділ дисципліни «Основи програмування та алгоритмізація» у Віртуальному навчальному середовищі (ННІ цивільного захисту/спеціальність «Комп'ютерні науки»/бакалавр/І

курс/II семестр/Основи програмування та алгоритмізація/Тема 4. Алгоритми/Контрольна частина/Тестове завдання за темою 4.9.)

Слід звернути увагу, що час та період складання тестового завдання обмежено.

2. Реалізація алгоритму пошуку порядкової статистики мовою програмування Java.

Перед виконання практичного завдання слід повторити лекцію «Пошук порядкових статистик», та нагадати особливості роботи методу декомпозиції. Реалізацію алгоритму слід проводити з дотриманням таких вказівок: вибір опорного елементу та значення шуканої порядкової статистики має відбуватись випадково (на відміну від попереднього завдання – алгоритму швидкого сортування). Розглянемо код алгоритму з детальним аналізом усіх методів:

```
import java.util.*;

public class Statistics {

// далі відтворено основний метод алгоритму пошуку порядкової статистики із передачею йому в якості аргументів масиву, початкового та кінцевого елементів, а також значення шуканої статистики. Основна мета методу - циклічне визначення опорного елемента k та занурення у відповідний масив (правий чи лівий) залежно від результату порівняння шуканої статистики n та індексу опорного елементу k на чергової ітерації

      public static void nth_element(int[] arr, int first, int last, int n) {
            while (true) { // запуск циклу до поки умова істинна
                  int k = randomizedPartition(arr, first, last); // запуск методу розбиття та сортування масиву із записом одержаного значення (середини масиву) у змінну k
                  if (n < k) // умова: якщо шукана статистика менша за середину масиву
                        last = k; // встановлення в якості останнього елементу масиву його середнє значення (значення опорного елементу), іншими словами занурення в ліву частину. Таким чином на черговій ітерації розбиття та сортування буде проводитись на проміжку від початку до середини масиву відносно його вихідного розміру
                  else if (n > k) // умова: якщо шукана статистика більша за середину масиву
                        first = k + 1; // встановлення в якості першого елементу масиву його середнє значення+1 (наступне значення після опорного елементу), іншими словами занурення в праву частину. Таким чином на черговій ітерації розбиття та сортування буде проводитись на проміжку від середини до кінця масиву відносно його вихідного розміру
                  else
                        return; // вихід з методу після останньої стадії рекурсивного розв’язку (кількість елементів масиву ==1)
            }
      }

// далі відтворено метод вибору випадкових значень rnd. Застосування методу необхідне для подальшого спрощення процедури випадкового вибору опорного елементу та шуканої порядкової статистики (відповідно до умов завдання)
```

```
	static Random rnd = new Random();

// далі по ходу відтворення алгоритму представлено вже відомий метод циклічного розбиття та сортування масиву (на основі QuickSort) із деякими модифікаціями та удосконаленнями

	static int randomizedPartition(int[] arr, int first, int last) {
		swap(arr, first + rnd.nextInt(last - first), last - 1); // звернення до методу заміни двох елементів масиву між собою (сам метод відтворено далі). Метод в якості аргументів приймає масив, випадково згенероване значення з масиву та останній елемент масиву-1. Випадкове значення, яке передається цьому методу – це є опорний елемент (його рандомізований вибір). А сам метод swap потрібен для того, щоб розмістити випадково обраний опорний елемент в останню комірку масиву. Для випадкового вибору опорного елементу використано раніше написаний метод rnd
		int separator = arr[last - 1]; // запис у змінну separator значення з передостанньої комірки масиву
		int i = first - 1; // введення індексу, що розмежовує менші та більші значення за опорний елемент, та його встановлення перед початком масиву (аналогічно як у методі швидкого сортування)
		for (int j = first; j < last; j++) // запуск циклу для перебирання усіх елементів масиву
			if (arr[j] <= separator) // порівняння на черговій ітерації поточного елемента масиву з значенням змінної separator
				swap(arr, ++i, j); // у випадку виконання умови відбувається звернення до методу swap для заміни поточного елементу масиву j з черговим елементом i (більше значення міняється місцями з меншим)
		return i; // повернення крайньої комірки з елементом меншим опорного (крайня комірка між меншими та більшими елементами відносно опорного)
	}

// далі представлено метод заміни між собою більших та менших за значеннями елементів масиву у порядку зростання (метод заміни)

	static void swap(int[] arr, int i, int j) {
		int temp = arr[i]; // створення тимчасової змінної temp для збереження значення із комірки i
		arr[i] = arr[j]; // запис в комірку i значення з комірки j
		arr[j] = temp; // запис в комірку j значення з тимчасової змінної temp
	}

// далі основний рушій – тестовий (головний) метод. В методі ініційовано створення масиву з 10 елементів та його рандомізоване наповнення. Також в методі реалізовано випадковий вибір шуканої порядкової статистики та запуск основного методу алгоритму nth_element із передачею йому необхідних аргументів

	public static void main(String[] args) {
		int[] arr = new int[10];
		for (int i = 0; i < arr.length; i++) {
			arr[i] = rnd.nextInt(10); // створення масиву на 10 елементів та його циклічне наповнення випадковими значеннями в проміжку 0-10. Для випадкового вибору значення i-го елемента масиву використано раніше написаний метод rnd
		}
		System.out.println(Arrays.toString(arr));
```

```
            int k = rnd.nextInt(10); // випадковий вибір шуканої порядкової статистики (у випадку пошуку конкретного елемента, статистику можна задавати вручну)
            nth_element(arr, 0, arr.length, k); // запуск основного методу пошуку порядкової статистики із передачею йому в якості аргументів масиву, початкового та кінцевого елементу масиву та значення шуканої порядкової статистики
            System.out.println("Порядкова статистика масиву з індексом " + k + " містить значення: " + arr[k]);
        }
}
```

Для закріплення пройденого матеріалу в якості практичного завдання необхідно переписати реалізований алгоритм із внесенням певних коректив. Отже **умова завдання**: в основному методі `nth_element` цикл `while` замінити на цикл `for`; кількість масивів для пошуку порядкових статистик має обиратись випадково в проміжку 0-4. Також передбачити циклічний вивід у консоль усіх випадково згенерованих масивів та результатів пошуку випадкової порядкової статистики до кожного масиву. Кількість елементів у масиві також має обиратись випадково та при кожному запуску програми генерувати різні значення.

Код написаної програми необхідно завантажити у віртуальне навчальне середовище в категорію «Реалізація алгоритму пошуку порядкової статистики». Слід зауважити, що період виконання завдання обмежений.

Методичну розробку розробив:
Заступник начальника кафедри УПІТтаТ
майор служби цивільного захисту О.В. Придатко

Методична розробка обговорена на засіданні кафедри УПІТтаТ
Протокол № ____ від "___" _____ 20__ р.