Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Кафедра ОТ

­­

**Лабораторна робота № 2**

з дисципліни

АЛГОРИТМИ ТА МЕТОДИ ОБЧИСЛЕНЬ

**Тема:** «Обчислювальна складність алгоритмів сортування»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Новотарський М. А. |  | студент 2-го курсу  гр. ІП-53 ФІОТ  Сулима Олександр Сергійович |

2017

**Мета:** Закріплення навичок практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритмів сортування .

**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.

**Номер завдання за списком: 17**



**Теоретичні відомості:**

Метод використовує той факт, що число перестановок у масиві може різко

скоротитися, якщо міняти місцями елементи, що перебувають на великій

відстані. Він реалізує метод «розділяй і пануй». Для цього звичайно в середині

масиву вибирається один елемент (опорний). Далі ліворуч від опорного

розташовують усі елементи, менше його, а праворуч – більше. Потім той же

приймання застосовують до половинок масиву. Процес закінчується, коли в

частинах масиву залишиться по одному елементу.

В алгоритмі використовуються два різноспрямовані процеси. Перший

виконується від початку масиву й шукає елемент, більший опорного. Другий -

працює з кінця й шукає елемент, менший опорного. Як тільки такі елементи

знайдені, проводиться їхній обмін місцями. Далі пошук триває з того місця, де

процеси зупинилися.

Таким чином, коли процеси зустрічаються, будь-який елемент у першій

частині менше кожного в другий. Це значить, що їх уже порівнювати один з

одним не прийде. Залишається тільки провести таку ж операцію стосовно

отриманих половин, і так далі, поки в черговій частині масиву не залишиться

один елемент. Це буде означати, що масив відсортований. Очевидно, що

найбільш зручний спосіб реалізації розглянутого методу – рекурсивний.

**Блок-схема алгоритму:**

min = 0; min < arr.length - 1; min++

Так Ні

least = min;

least = j;

int j = min + 1; j < arr.length; j++

Так

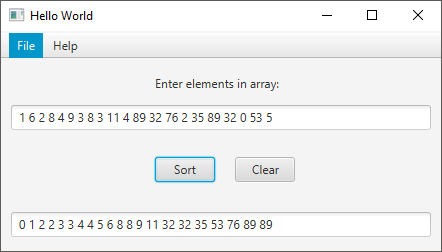
Ні

int tmp = arr[min];  
arr[min] = arr[least];  
arr[least] = tmp;

**Текст програми**

public void sortArray() {  
  
 symbols = textFieldEnterElement.getText();  
  
 try {  
 arr = Arrays.*stream*(symbols.split(" ")).mapToInt(Integer::*parseInt*).toArray();  
 for (int min = 0; min < arr.length - 1; min++) {  
 int least = min;  
 for (int j = min + 1; j < arr.length; j++) {  
 if (arr[j] < arr[least]) {  
 least = j;  
 }  
 }  
 int tmp = arr[min];  
 arr[min] = arr[least];  
 arr[least] = tmp;  
 }  
  
 for (int i = 0; i < arr.length; i++) {  
 res += arr[i];  
 res += " ";  
 }  
 textFieldResult.setText(res);  
 } catch (Exception e) {  
 JOptionPane.*showMessageDialog*(null, "Please, enter correct data!");  
 }  
  
  
}

**Результати виконання програми**



Аналіз результатів:

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Висновок**

Дана програма сортує масив швидким методом сортування, виводить на екран відсортований масив і обчислює час виконання алгоритму.