**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**імені ІГОРЯ СІКОРСЬКОГО”**

**ЗВІТ**

**з лабораторної роботи №1**

**з навчальної дисципліни “Програмування складних алгоритмів”**

**Тема:**

**ОЦІНКА СКЛАДНОСТІ АЛГОРИТМІВ**

**Виконав студент групи ТР–12**

Каркушевський Владислав

Лабораторну роботу захищено

з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Перевірив доцент кафедри**

\_\_\_\_\_\_\_\_\_\_\_ Андрій ОНИСЬКО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 року

**Київ 2022**

**Мета:** Ознайомлення з оцінкою складності алгоритмів та її характеристиками

**Теоретична частина**

**Алгоритм** – це скінченна послідовність команд, які треба виконати над вхідними даними для отримання результату.

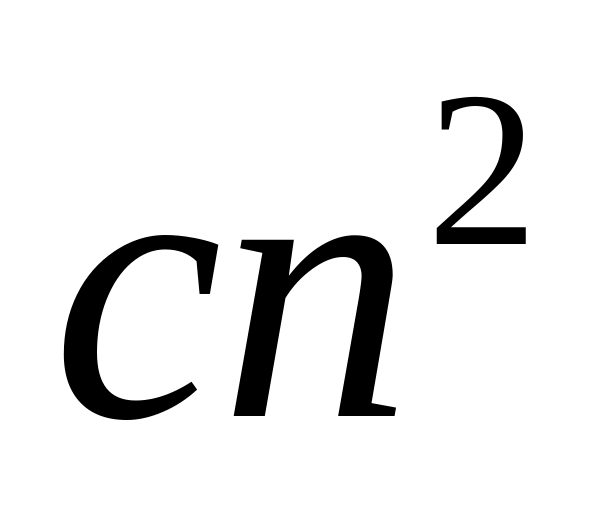
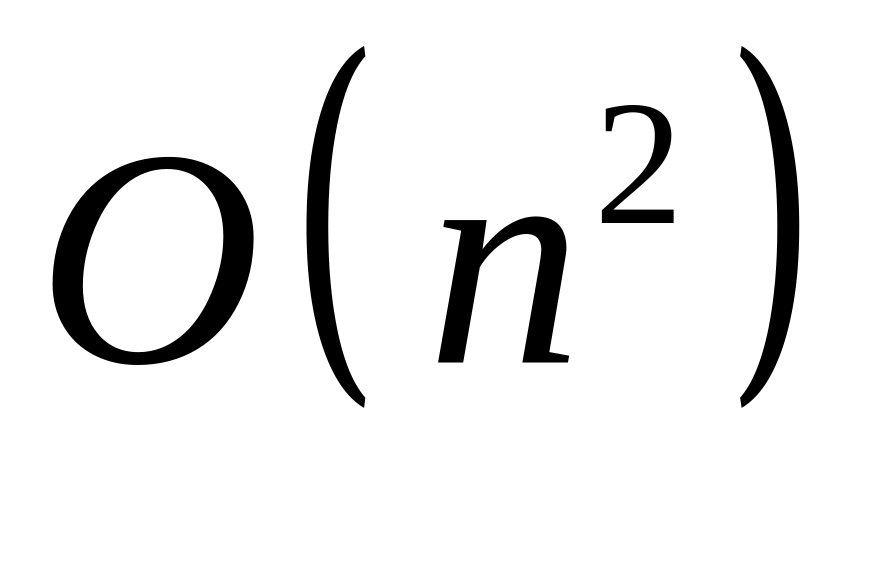
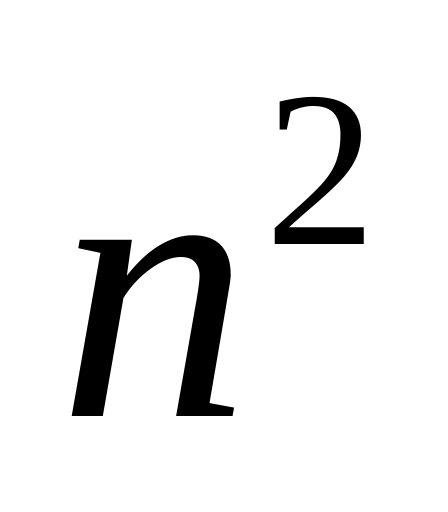
Існують такі способи описання алгоритмів:

•словесний,

•формульний,

•графічний,

•алгоритмічною мовою

Для оцінки ефективності роботи алгоритму необхідно мати певні критерії. Одним із таких критеріїв – час роботи алгоритму, але він не єдиний. Іншою важливою характеристикою алгоритму є *асимптотична складність алгоритму*. Як раз асимптотична складність алгоритму визначає у підсумку розмір задачі, яку можна розв’язати за допомогою такого алгоритму. Якщо алгоритм обробляє вхідні дані розміром за час  , де - деяка постійна величина, то *складність* такого алгоритму буде  (читається «порядку  ). Складність алгоритму – це кількість кроків, які необхідно виконати, щоб досягти заданого результату. Часова складність алгоритму - це затрати часу  на обробку входу розміром .

**Характеристики алгоритму**

* Чіткість і точність
* Чітко визначені вхідні дані
* Чітко визначені результати
* Алгоритм повинен бути скінченним
* Алгоритм повинен бути розумним і простим

Розгляд вхідних даних великого розміру і оцінка порядку зростання часу роботи алгоритму приводять до поняття **асимптотичної складності** алгоритму.

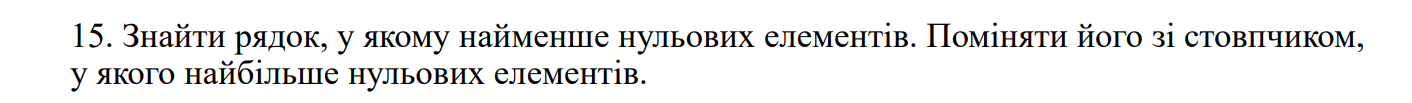
Виділяють такі основні класи алгоритмів:

* логарифмічні
* лінійні
* поліноміальні
* експоненційні

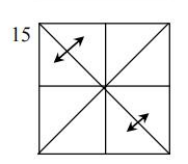
Експоненційні алгоритми часто пов’язані з перебором різних варіантів розв’язку. Алгоритми без циклів і рекурсивних викликів мають константу складність. Визначення складності алгоритму в основному зводиться до аналізу циклів і рекурсивних викликів.

**Завдання**

**1 завдання**

****

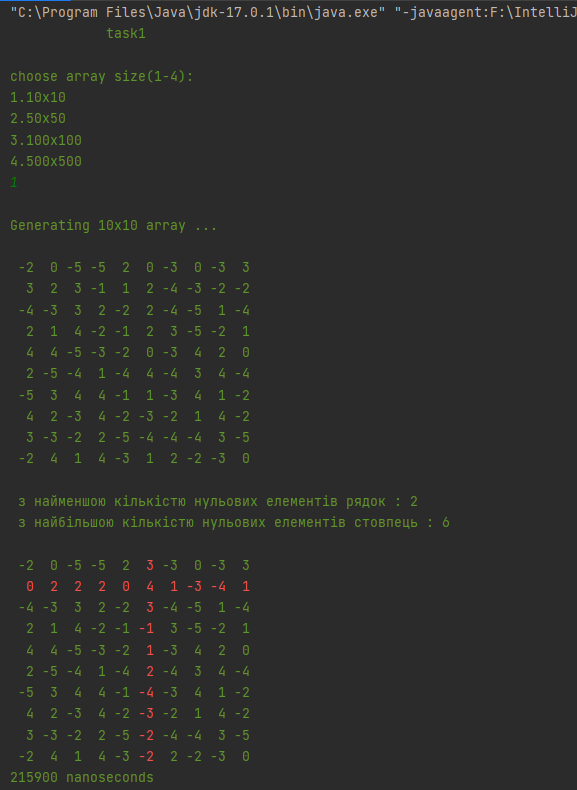
**2 завдання**

****

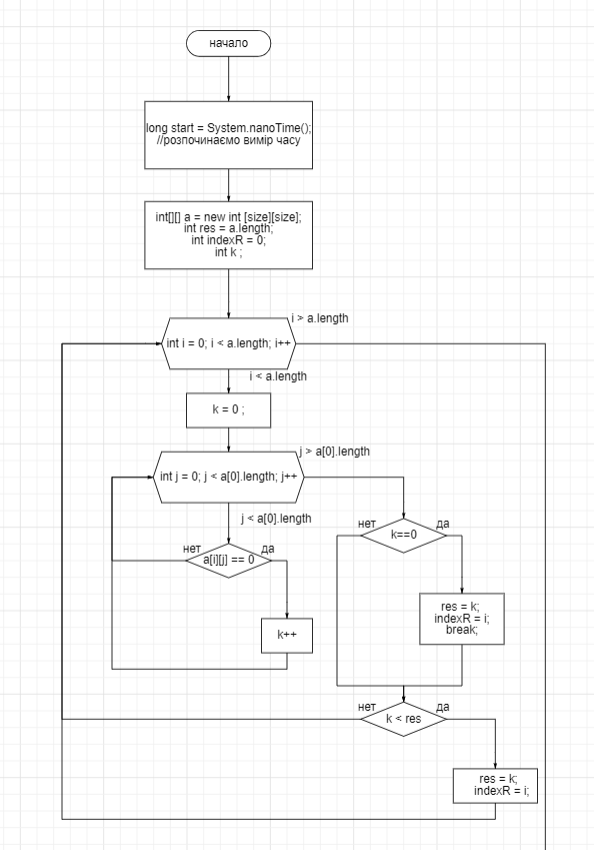
**Результати виконання**

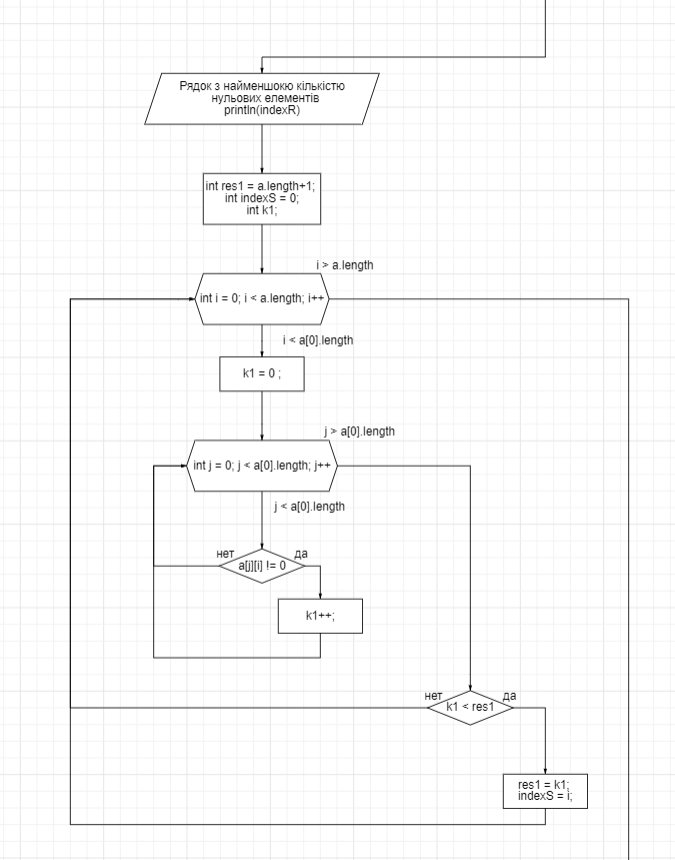
**import** java.util.Scanner;//імпортуємо сканер  
  
**public class** lab1 {  
 **public static void** main(String[] args) {//викликаємо головний метод main  
  
 Scanner sc = **new** Scanner(System.*in*);//викликаємо сканер  
  
 System.*out*.print("\u001B[32m");  
 System.*out*.println("\t\t\ttask1\n");  
 System.*out*.println("choose array size(1-4):\n1.10x10\n2.50x50\n3.100x100\n4.500x500");  
  
 **int** size = sc.nextInt();//за допомогою сканера вибираємо розмір матриці  
  
 size = *info*(size);  
 **int**[][] a = *makeMatrix*(size);//викликаємо метод для створення матриці  
 *outputMatrix*(a);//виводимо матрицю  
 *task1*(a);//викликаємо метод з 1 завданням  
  
  
 System.*out*.println("\n\t\t\ttask2\n");  
 System.*out*.println("choose array size(1-4):\n1.10x10\n2.50x50\n3.00x100\n4.500x500");  
  
 **int** size1 = sc.nextInt();//за допомогою сканера вибираємо розмір матриці  
  
 size1 = *info*(size1);  
 **int**[][] b = *makeMatrix*(size1);//викликаємо метод для створення матриці  
 *outputMatrix*(b);//виводимо матрицю  
 *task2*(b);//викликаємо метод з 2 завданням  
  
 }  
  
 **public static int**[][] task1(**int**[][] a) {  
 **long** start = System.*nanoTime*();//розпочинаємо вимір часу  
  
 **int** res = a.length;  
 **int** indexR = 0;  
 **int** k ;  
  
 **for** (**int** i = 0; i < a.length; i++) {//створюємо цикл для знаходження рядка у якому найиенше нульових елементів  
 k = 0;  
 **for** (**int** j = 0; j < a[0].length; j++) {  
 **if** (a[i][j] == 0) {  
 k++;  
 }  
 }  
 **if**(k==0){  
 res = k;  
 indexR = i;  
 **break**;  
 }  
 **if** (k < res) {  
 res = k;  
 indexR = i;  
 }  
 }  
  
 System.*out*.printf(" з найменшою кількістю нульових елементів рядок : %d\n", indexR+1);  
  
 **int** res1 = a.length+1;  
 **int** indexS = 0;  
 **int** k1;  
 **for** (**int** i = 0; i < a.length; i++) {//створюємо цикл для знаходження стовпця у якому найбільше нульових елементів  
  
 k1 = 0;  
 **for** (**int** j = 0; j < a[0].length; j++) {  
 **if** (a[j][i] != 0) {  
 k1++;  
 }  
 }  
 **if** (k1 < res1) {  
 res1 = k1;  
 indexS = i;  
 }  
 }  
 System.*out*.printf(" з найбільшою кількістю нульових елементів стовпець : %d\n\n" ,indexS+1);  
  
 **for** (**int** i = 0; i < a.length; i++) {//створюємо цикл для заміни знайдених рядка зі стовпцем  
 **int** c = a[indexR][i];  
 a[indexR][i] = a[i][indexS];  
 a[i][indexS] = c;  
  
 }  
 **long** finish = System.*nanoTime*();//закінчуємо вимір часу  
  
 **for**(**int** i = 0 ; i < a.length ; i++){//створюємо цикл для різнобарвного виведення матриці(червоний колір-змінені елементи )  
 **for**(**int** j = 0 ; j < a[0].length ; j++){  
 **if**( i == indexR || j == indexS){  
 System.*out*.print("\u001B[31m");  
 }**else**{  
 System.*out*.print("\u001B[32m");  
 }  
 System.*out*.printf("%3d",a[i][j]);  
  
  
 }  
 System.*out*.println();  
 }  
  
 System.*out*.print("\u001B[32m");  
  
 System.*out*.printf( "%d nanoseconds", finish-start);//виведення часу перестановки матриці  
  
 **return** a ;  
  
 }  
  
 **public static int** [][] task2(**int**[][] b){  
 **long** start1 = System.*nanoTime*();//розпочинаємо вимір часу  
  
 **int** limLeft = (b.length/2)-1;  
 **int** limRight = b.length/2 ;  
  
 **for**(**int** i = 0 ; i< b.length/2;i++){//створення циклу для перестановки елементів згідно з варіантом  
  
 **for**(**int** j = 0 ; j< b[0].length/2;j++){  
 **if**(i == j){  
 **break**;  
 }**else** {  
 **int** mov = b[limLeft- i][limLeft-j];  
 b[limLeft- i][limLeft-j] = b[limLeft- j][limLeft-i];  
 b[limLeft- j][limLeft-i] = mov;  
  
 **int** lol = b[limRight + i][limRight + j];  
 b[limRight + i][limRight + j]=b[limRight + j][limRight + i];  
 b[limRight + j][limRight + i] = lol;  
 }  
  
 }  
 }  
 **long** finish1 = System.*nanoTime*();//закінчуємо вимір часу  
  
 **for**(**int** i = 0 ; i < b.length/2;i++){//створюємо цикл для різнобарвного виведення матриці(червоний колір-змінені елементи )  
 **for**(**int** j = 0 ; j < b[0].length/2;j++ ){  
 **if**(i==j){  
 System.*out*.print("\u001B[32m");  
 }**else** {  
 System.*out*.print("\u001B[31m");  
 }  
 System.*out*.printf("%3d", b[i][j]);  
 }  
 **for**(**int** k = b[0].length/2; k < b[0].length; k++){  
 System.*out*.print("\u001B[32m");  
 System.*out*.printf("%3d", b[i][k]);  
 }  
 System.*out*.println();  
 }  
  
 **for**(**int** q = b.length/2 ; q < b.length;q++){  
 **for**(**int** r = 0; r < b[0].length/2; r++){  
 System.*out*.print("\u001B[32m");  
 System.*out*.printf("%3d", b[q][r]);  
 }  
 **for**(**int** w = b[0].length/2 ; w < b[0].length;w++ ){  
 **if**(q==w){  
 System.*out*.print("\u001B[32m");  
 }**else** {  
 System.*out*.print("\u001B[31m");  
 }  
 System.*out*.printf("%3d", b[q][w]);  
 }  
 System.*out*.println();  
 }  
  
  
 System.*out*.printf( "%d nanoseconds", finish1-start1);//виведення часу перестановки матриці  
 **return** b;  
 }  
  
 **public static void** outputMatrix(**int**[][] b){//створюємо метод для виведення матриці  
  
 **for**(**int** i = 0 ; i< b.length;i++){  
 **for**(**int** j = 0 ; j< b[0].length;j++ ){  
 System.*out*.printf("%3d", b[i][j]);  
 }  
 System.*out*.println();  
 }  
 System.*out*.println();  
 }  
  
 **public static int**[][] makeMatrix( **int** size ){//створюємо метод для створення матриці  
  
 **int**[][] a = **new int** [size][size];  
  
  
 **for**(**int** i = 0 ; i< a.length; i++){  
 **for**(**int** j = 0 ; j< a[0].length;j++){  
 a[i][j] = -5 + (**int**) (Math.*random*() \* 10);//заповнюємо матрицю числами в діапазоні [-5;5]  
 }  
 }  
  
 **return** a;  
 }  
  
 **public static int** info(**int** size ){//створюємо метод для вибору розміру матриці  
  
 **if**(size == 1){  
 size = 10;  
 System.*out*.println("\nGenerating 10x10 array ...\n");  
 }**else if**(size == 2 ){  
 size = 50;  
 System.*out*.println("\nGenerating 50x50 array ...\n");  
 }**else if**(size == 3 ){  
 size = 100;  
 System.*out*.println("\nGenerating 100x100 array ...\n");  
 }**else if**(size == 4 ){  
 size = 500;  
 System.*out*.println("\nGenerating 500x500 array ...\n");  
 }**else**{  
 System.*out*.println("error");  
 System. *exit*(0);  
 }  
  
 **return** size;  
 }  
  
}

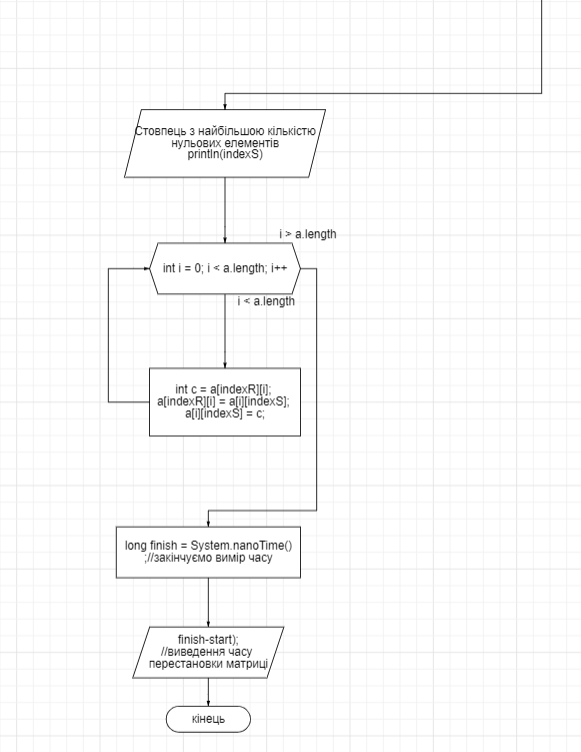
**Скріншоти виконання 1 завдання**

****

**Блок схема до 1 завдання**

****

****

****

**9x^2 + 6x^2 + 4x = O(x^2)**

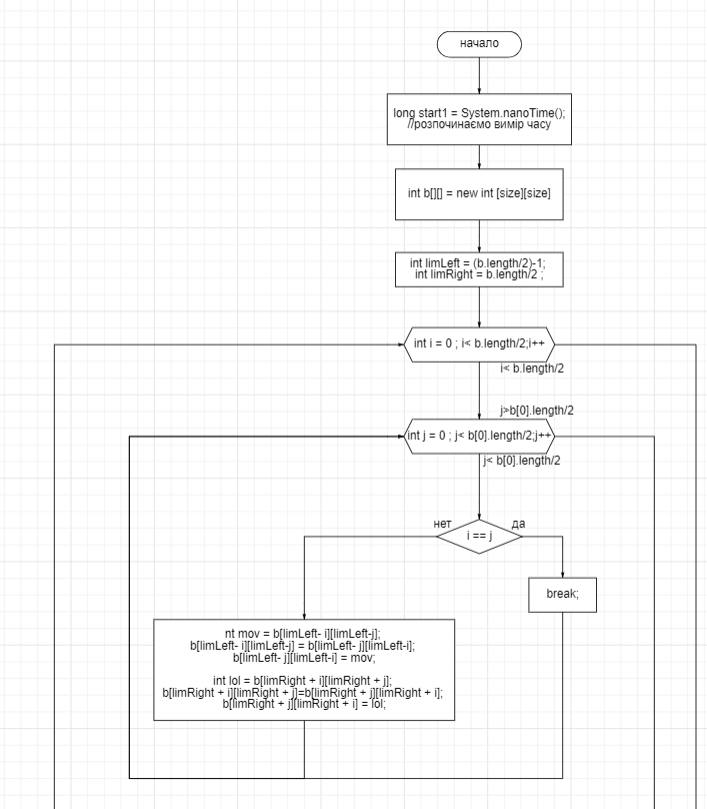
**-––**

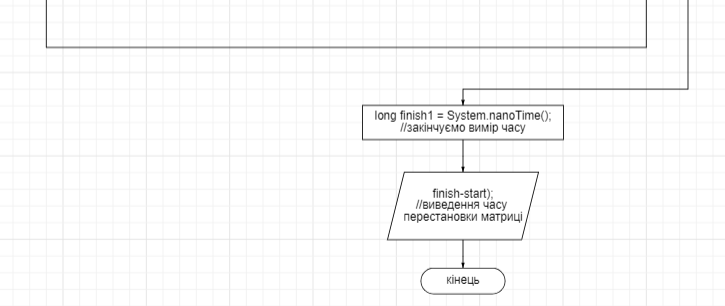
|  |  |  |
| --- | --- | --- |
| **Розмір матриці** | **Кількість кроків** | **Час виконання(нс)** |
| **10** | **1 540** | **215900** |
| **50** | **37 700** | **788500** |
| **100** | **150 400** | **869800** |
| **500** | **3752000** | **10211500** |

**Скріншоти виконання 2 завдання**

****

**Блок схема до 2 завдання**

****

****

**25 \* х/2 \* х/2 = 25/4 \* х^2 = O(x^2)**

|  |  |  |
| --- | --- | --- |
| **Розмір матриці** | **Кількість кроків** | **Час виконання(нс)** |
| **10** | **625** | **3400** |
| **50** | **15 625** | **39100** |
| **100** | **62 500** | **103400** |
| **500** | **1 562 500** | **3227300** |

**Висновок**

На цій лабораторній роботі ми ознайомилися з оцінкою складності алгоритмів та її характеристиками. Взнали що таке алгоритм, способи описання алгоритму, характеристики алгоритмів, основні класи алгоритмів, познайомилися з асимптотичною складністю. Виконали 2 завдання згідно варіанту, побудували блок схеми до завдань, та оцінили часову складність алгоритму.