Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №4**

з дисципліни «[Організація обчислювальних процесів](http://wiki.kpi.ua/index.php/%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F%20%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D1%85%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%96%D0%B2_(20402040))»

на тему «Дослідження принципів проектування та роботи динамічного та статичного планування»

Виконав:

студент ІІI курсу

групи ІО-53

Лісовий В. О.

Київ – 2018

№ЗК=5318

Вариант = 18 mod 13 + 1 = 6

6. Реалізувати модель просторового планувальника на основі алгоритму перетворення матриці зв'язності з вибудовуванням "1" по головній діагоналі.

**Описание алгоритма**

1. Поиск строки матрицы, где сумма элементов минимальна.
2. Поменять местами первую строку матрицы с той что найдена в п1.
3. Поиск столбца с минимально сумой элементов среди тех, у которых первый элемент равен 1.
4. Поменять местами первый столбец матрицы с тем что найден в п3.
5. Заменить все элементы первого столбца и первой строки 0 кроме первого элемента, который равен 1.
6. Заявку номер которой равен номеру столбца, отправить на ресурс номер которого равен номеру строки.
7. Повторить операции 1-6 для подматрицы, которую получаем из матрицы п5 исключение первого столбца и строки, но выполнять операции обмена между столбцами и строками для самой начальной матрицы.

Временная сложность для первой итерации О1(n)=2(n2+n)

Для i-ой Оi(n)=2((n-i)2+(n-i))

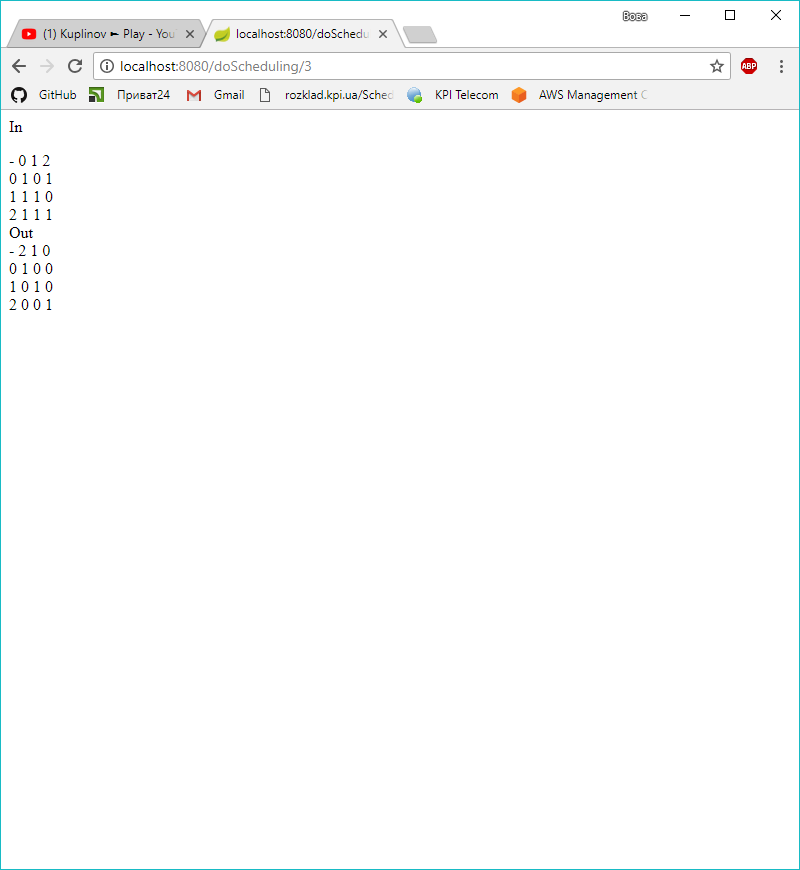
О(n)=2n(n2+n)=n3+n2

**Сирцевий код**

**package** lab4.service;  
  
**import** java.util.Arrays;  
**import** java.util.Random;  
  
**public class** SchedulerService {  
  
 **private int**[] **cpu**;  
 **private int**[] **task**;  
 **private int**[][] **startMatrix**;  
 **private int**[][] **matrix**;  
  
 **public** SchedulerService(**int** n) {  
 **this**.**cpu** = **new int**[n];  
 **this**.**task** = **new int**[n];  
 **this**.**startMatrix** = **new int**[n][];  
 **this**.**matrix** = **new int**[n][];  
 **for** (**int** i = 0; i < n; i++) {  
 **cpu**[i] = i;  
 **task**[i] = i;  
 }  
 **this**.randMatrix();  
 **for** (**int** i = 0; i < n; i++)  
 **startMatrix**[i] = **matrix**[i].clone();}  
  
 **private void** randMatrix() {  
 Random r = **new** Random();  
 **for** (**int** i = 0; i < **matrix**.**length**; i++)  
 **matrix**[i] = r.ints(0, 2).limit(**matrix**.**length**).toArray();  
 }  
  
 **public void** doScheduling() {  
 **for** (**int** itr = 0; itr < **matrix**.**length**; itr++) {  
 *// find row with min sum* **int** minRowIndex = itr;  
 **int** minRowSum = Integer.***MAX\_VALUE***;  
 **for** (**int** i = itr; i < **matrix**.**length**; i++) {  
 **int** sum = 0;  
 **for** (**int** j = itr; j < **matrix**.**length**; j++)  
 sum += **matrix**[i][j];  
 **if** (sum < minRowSum) {  
 minRowSum = sum;  
 minRowIndex = i;  
 }  
 }  
  
 *// swap first row with min sum row* **this**.swapRows(itr, minRowIndex, **this**.**matrix**);  
 **this**.swap(itr, minRowIndex, **this**.**cpu**);  
  
  
 *// find col that starts with 1 with min sum* **int** minColIndex = itr;  
 **int** minColSum = Integer.***MAX\_VALUE***;  
 **for** (**int** i = itr; i < **matrix**.**length**; i++) {  
 **if** (**matrix**[itr][i] == 1) {  
 **int** sum = 0;  
 **for** (**int** j = itr; j < **matrix**.**length**; j++) {  
 sum += **matrix**[j][i];  
 }  
 **if** (sum < minColSum) {  
 minColSum = sum;  
 minColIndex = i;  
 }  
 }  
 }  
  
 *// swap first col with min sum col that starts with 1* **this**.swapCols(itr, minColIndex, **this**.**matrix**);  
 **this**.swap(itr, minColIndex, **this**.**task**);  
 **for** (**int** i = itr + 1 ; i < **matrix**.**length**; i++) {  
 **matrix**[itr][i] = 0;  
 **matrix**[i][itr] = 0;  
 }  
 }  
 }  
  
 **private void** swapRows(**int** i, **int** j, **int**[][] matrix) {  
 **int**[] buf = matrix[i].clone();  
 matrix[i] = matrix[j].clone();  
 matrix[j] = buf;  
 }  
  
 **private void** swapCols(**int** i, **int** j, **int**[][] matrix) {  
 **for** (**int** k = 0; k < matrix.**length**; k++) {  
 **this**.swap(i, j, matrix[k]);  
 }  
 }  
  
 **private void** swap(**int** i, **int** j, **int**[] arr) {  
 **int** buf = arr[i];  
 arr[i] = arr[j];  
 arr[j] = buf;  
 }  
  
 **public int**[] getCpu() {  
 **return cpu**;  
 }  
  
 **public int**[] getTask() {  
 **return task**;  
 }  
  
 **public int**[][] getStartMatrix() {  
 **return startMatrix**;  
 }  
  
 **public int**[][] getMatrix() {  
 **return matrix**;  
 }  
}

**Результати тестування програми**

N=3



N=5

