Санкт-Петербургский Политехнический Университет

Институт компьютерных наук и технологий

«Высшая школа программной инженерии»

**Отчёт по лабораторной работе**

по дисциплине: «Системный анализ и принятие решений»

Выполнил:

студент гр. в3530904/80322 Мирошниченко В.А.

Руководитель: Амосов В.В.

Санкт-Петербург

2020

**Постановка задачи**

Задача 1.

Дана матрица доходов (решений):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **R1** | **R2** | **R3** | **R4** | **R5** |
| **X1** | 15 | 10 | 0 | -6 | 17 |
| **X2** | 3 | 14 | 8 | 9 | 2 |
| **X3** | 1 | 5 | 14 | 20 | -3 |
| **X4** | 7 | 19 | 10 | 2 | 0 |

Необходимо сравнить варианты решений по следующим критериям:

* Минимаксный критерий;
* Критерий Севиджа;
* Метод Гурвица.

Для метода Гурвица .

Задача 2.

Ежегодный спрос на автомобили марки «Mercedes» в совместном предприятии в 1 половину года задаётся следующим распределением вероятностей:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X (спрос)** | 100 | 150 | 200 | 250 | 300 |
| **P(X)** | 0,15 | 0,4 | 0,3 | 0,1 | 0,05 |

Если автомобиль марки «» продан в 1 половину года, он может быть реализован за 15000 $ к концу года. С другой стороны, автомобили марки «Mercedes» с конвейера (в 1 половину года) продаются по 49000 $ за штуку. Затраты совместного предприятия на 1 автомобиль марки «Mercedes» составляют 25000$.

Определите: какое оптимальное число автомобилей марки «Mercedes» целесообразно заказывать ежегодно и какова оптимальная ожидаемая прибыль.

Матрица решений:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X / Y** | **100** | **150** | **200** | **250** | **300** |
| **100** | 2400 т. $ | 2400 т. $ | 2400 т. $ | 2400 т. $ | 2400 т. $ |
| **150** | 1900 т. $ | . | . | . | . |
| **200** | . | . | . | . | . |
| **250** | . | . | . | . | . |
| **300** | . | . | . | . | . |

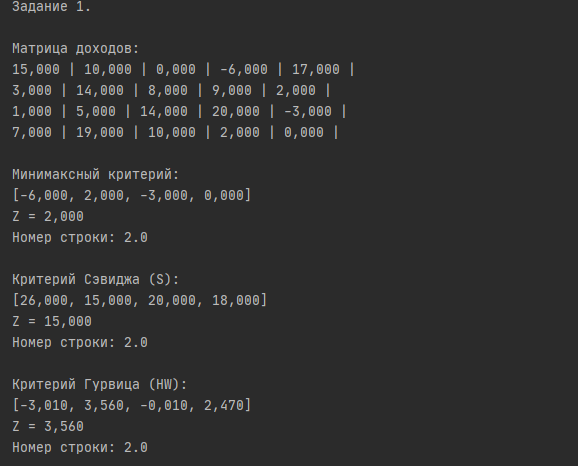
Где X – спрос, а Y – заказ.

Если спрос превышает предложение или равен:

Иначе:

**Результаты работы программы**

Результаты полученные при решении задачи 1:

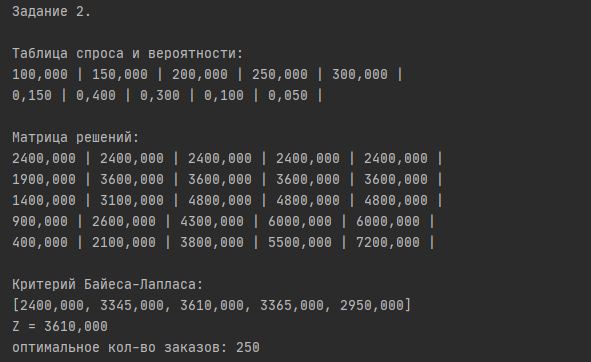


Выводы:

При решении задачи используя минимаксный критерий и критерий Сэвиджа – оптимальной строкой в матрице доходов является строка номер 2.

При решении задачи используя критерий Гурвица при – оптимальной строкой в матрице доходов является срока номер 2.

Результаты полученные при решении задачи 2:



Выводы:

При решении задачи используя критерий Байеса-Лапласа, оптимальным количеством заказа автомобилей марки «Mercedes» будет 250 штук, а максимальная прибыль будет равна 6000 т. $.

**Код программы**

Исходный код программы, написанный на языке программирования Java:

package com.company;  
  
import java.util.Arrays;  
  
public class Main {  
  
 public static void main(String[] args) {  
  
 System.*out*.println("Задание 1.\n");  
  
 double[][] matrix = {{15,10,0,-6,17},  
 {3,14,8,9,2},  
 {1,5,14,20,-3},  
 {7,19,10,2,0}};  
  
  
 *outputMatrix*(matrix, "Матрица доходов:");  
 System.*out*.println("Номер строки: "+*waldMethod*(matrix)+"\n");  
 System.*out*.println("Номер строки: "+*savageMethod*(matrix)+"\n");  
 System.*out*.println("Номер строки: "+*gurvisMethod*(matrix)+"\n");  
  
 System.*out*.println("Задание 2.\n");  
  
 double[][] probabilityMatrix = {{100,150,200,250,300},{0.15,0.4,0.3,0.1,0.05}};  
  
 double[][] answerMat = *getResultMatrix*(probabilityMatrix);  
  
 *outputMatrix*(probabilityMatrix, "Таблица спроса и вероятности:");  
 *outputMatrix*(answerMat, "Матрица решений:");  
  
 int index = *bayesLaplaceMethod*(answerMat, probabilityMatrix[1]);  
 if (index != -1) {  
 System.*out*.printf("оптимальное кол-во заказов: %.0f\n", probabilityMatrix[0][index]);  
 }  
  
  
 }  
//процедура вывода матрицы в консоль  
 private static void outputMatrix(double[][] matrix, String message) {  
 System.*out*.println(message);  
 for (double[] rows: matrix) {  
 for (double element: rows) {  
 System.*out*.printf("%.3f | ", element);  
 }  
 System.*out*.println();  
 }  
 System.*out*.println();  
 }  
//процедура вывода вектора в консоль  
 private static void printVector(double[] vector) {  
 System.*out*.print("[");  
 for (int i = 0; i < vector.length; i++) {  
 if(i != vector.length-1) {  
 System.*out*.printf("%.3f, ", vector[i]);  
 } else {  
 System.*out*.printf("%.3f", vector[i]);  
 }  
 }  
 System.*out*.print("]");  
 System.*out*.println();  
 }  
//Функция поиска индекса  
 private static int findIndexInArray(double[] array, double element) {  
 int index = -1;  
 for (int i = 0; i < array.length; i++) {  
 if(array[i] == element) {  
 index = i;  
 break;  
 }  
 }  
  
 return ++index;  
 }  
//----------------Задание 1-------------------------------  
 //реализация минимаксного критерия  
 private static double waldMethod(double[][] matrix) {  
 System.*out*.println("Минимаксный критерий:");  
 double[] tmp = new double[matrix.length];  
  
 for (int i = 0; i < matrix.length; i++)  
 tmp[i] = Arrays.*stream*(matrix[i]).min().getAsDouble();  
  
 *printVector*(tmp);  
  
 double max = Arrays.*stream*(tmp).max().getAsDouble();  
  
 System.*out*.printf("Z = %.3f\n", max);  
  
 return *findIndexInArray*(tmp,max);  
 }  
//------------------------Метод Сэвиджа (S)---------------------------------------------  
 private static double savageMethod(double[][] matrix) {  
 System.*out*.println("Критерий Сэвиджа (S):");  
  
 double[][] deltaK = new double[matrix.length][matrix[0].length];  
 double[] k = new double[matrix.length];  
  
 for (int i = 0; i < matrix.length; i++)  
 for (int j = 0; j < matrix[0].length; j++)  
 deltaK[i][j] = *getMaxColumn*(matrix, j) - matrix[i][j];  
  
 for(int i = 0; i < k.length; i++)  
 k[i] = Arrays.*stream*(deltaK[i]).max().getAsDouble();  
  
 *printVector*(k);  
 double min = Arrays.*stream*(k).min().getAsDouble();  
 System.*out*.printf("Z = %.3f\n", min);  
  
 return *findIndexInArray*(k,min);  
 }  
 //Получение максимального элемента в столбце матрицы  
 private static double getMaxColumn(double[][] matrix, int index) {  
 double[] column = new double[matrix.length];  
  
 for (int i = 0; i < column.length; i++) {  
 column[i] = matrix[i][index];  
 }  
  
 return Arrays.*stream*(column).max().getAsDouble();  
 }  
//------------------------Метод Гурвица (HW)-------------------------------------------  
  
 private static double gurvisMethod(double[][] matrix) {  
 System.*out*.println("Критерий Гурвица (HW):");  
  
 double[] tmp = new double[matrix.length];  
 double c = 0.87;  
  
 for (int i = 0; i < tmp.length; i++)  
 tmp[i] = c \* Arrays.*stream*(matrix[i]).min().getAsDouble() + (1 - c) \* Arrays.*stream*(matrix[i]).max().getAsDouble();  
  
 *printVector*(tmp);  
 double max = Arrays.*stream*(tmp).max().getAsDouble();  
 System.*out*.printf("Z = %.3f\n", max);  
  
 return *findIndexInArray*(tmp,max);  
 }  
///-----------------------------Задание 2-------------------------------------------  
 //Заполнение таблицы решений (спроса и предложений)  
 private static double[][] getResultMatrix(double[][] array){  
 double[][] answerMat = new double[array[0].length][array[0].length];  
  
 for (int i = 0; i < answerMat.length; i++) {  
 for (int j = 0; j < answerMat[0].length; j++) {  
 if (i == j || i < j) {  
 answerMat[i][j] = 49 \* array[0][i] - 25 \* array[0][i];  
 } else {  
 answerMat[i][j] = 49 \* array[0][j] - 25 \* array[0][i] + 15 \* (array[0][i] - array[0][j]);  
 }  
 }  
 }  
  
 return answerMat;  
 }  
 //Реализация критерия Байеса-Лапласа  
 private static int bayesLaplaceMethod(double[][] resultMatrix, double[] probabilityVector) {  
 System.*out*.println("Критерий Байеса-Лапласа:");  
 double[] e = new double[probabilityVector.length];  
  
 for (int i = 0; i < resultMatrix.length; i++) {  
 for (int j = 0; j < resultMatrix[0].length; j++){  
 e[i] += resultMatrix[i][j] \* probabilityVector[j];  
 }  
 }  
  
 *printVector*(e);  
 double max = Arrays.*stream*(e).max().getAsDouble();  
 System.*out*.printf("Z = %.3f\n", max);  
  
 return *findIndexInArray*(e, max);  
 }  
  
  
}