Федеральное государственное автономное образовательное учреждение высшего образования «**Уральский федеральный университет** имени первого Президента России Б. Н. Ельцина»

Отчет по лабораторной работе №2.2

дисциплины «Алгоритмы и анализ сложности»

Студент: Волков Илья Евгеньевич  
Группа: РИ-200004 (АТ-08)  
Преподаватель: Рыбкина Мария Николаевна

2022 год.

Оглавление

[Задание 3](#_Toc101813509)

[Теоретическая часть 3](#_Toc101813510)

[Инструкция пользователя 4](#_Toc101813511)

[Инструкция программиста 4](#_Toc101813512)

[Тестирование 5](#_Toc101813513)

[Листинги 6](#_Toc101813514)

[Вывод 8](#_Toc101813515)

# Задание

Лабораторная работа №2.2

Решение задачи о назначениях.  
  
В наиболее общей форме задача формулируется следующим образом:

Имеется некоторое число работ и некоторое число исполнителей. Любой исполнитель может быть назначен на выполнение любой (но только одной) работы, но с неодинаковыми затратами. Нужно распределить работы так, чтобы выполнить работы с минимальными затратами.

# Теоретическая часть

Есть несколько способов решения данной задачи:

1. Найти готовую библиотеку.

2. Найти готовое решение и, основываясь на нем, написать свое.

3. Полностью написать все с нуля самостоятельно.

Я выбрал комбинацию второго и третьего способа. До выполнения лабораторной работы я не знал ни одного алгоритма решения задачи о назначениях, однако в интернете нашел алгоритм, называемый Венгерским. Я взял его за основу, потому не могу сказать, что все написал самостоятельно.

# Инструкция пользователя

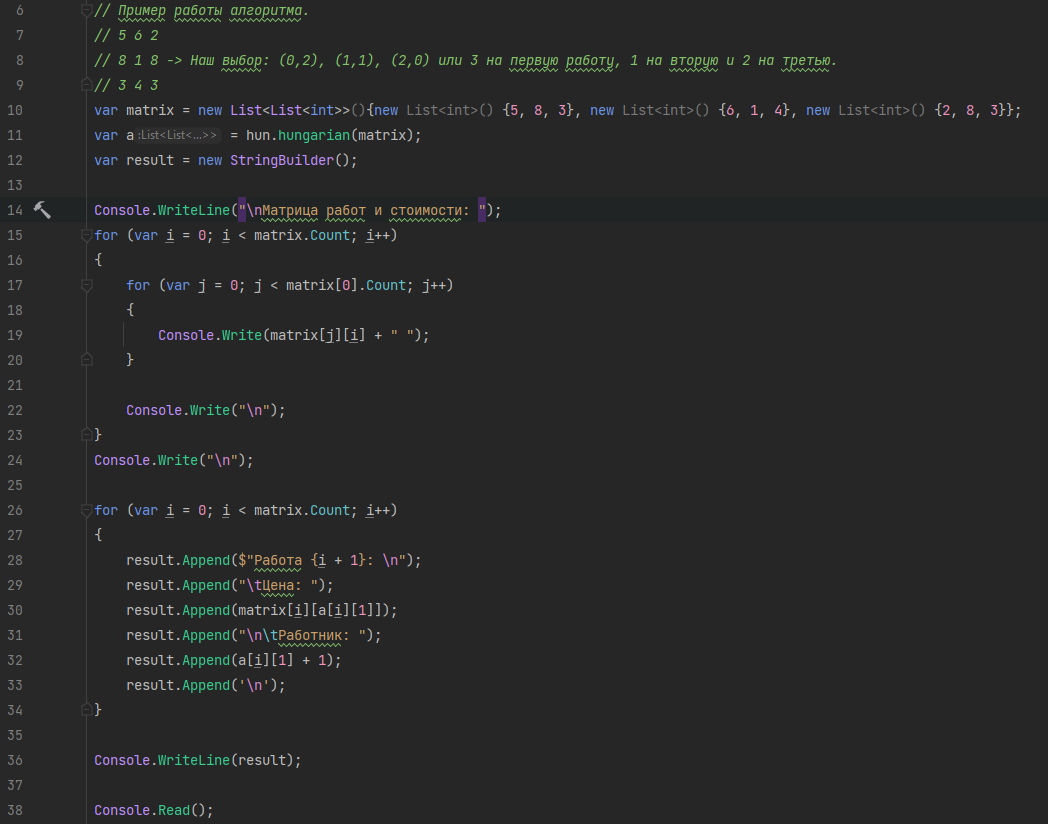
Для запуска софта используется файл HungaryanAlgoritm.exe.  
Откроется окно с примером работы программы. В ПО нет пользовательского интерфейса, т.е. кроме как кодом управлять им нельзя.

# Инструкция программиста

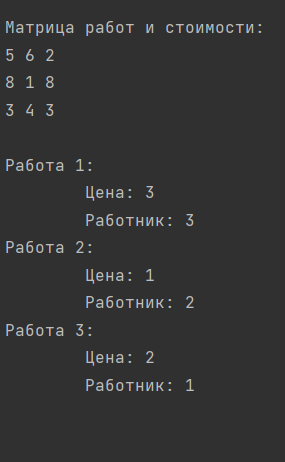
Для запуска софта используется файл HungaryanAlgoritm.exe.   
Для запуска решения используется файл HungaryanAlgoritm.sln (или HungaryanAlgoritm.csproj). Метод Венгерского алгоритма реализован в файле Hungaryan.cs. Работать с методом можно из любой точки программы, например, из Program.cs. Пример работы с методом представлен в Program.cs.

# Тестирование

1.



Вывод:



# Листинги

Листинг класса Hungaryan.cs:

namespace HungaryanAlgoritm.classes;  
  
public class Hungaryan  
{  
 public List<List<int>> hungarian(List<List<int>> matrix)  
 {  
 var height = matrix.Count;  
 var width = matrix[0].Count;  
   
 var u = new List<int>(height); *// Значения, вычитаемые из строк.* var v = new List<int>(width); *// Значения, вычитаемые из столбцов.* for (var x = 0; x < height; x++)  
 u.Add(0);  
 for (var x = 0; x < width; x++)  
 v.Add(0);  
   
 *// Индекс помеченной клетки в каждом столбце.* var zeroIndexes = new List<int>(width);  
 for (var x = 0; x < width; x++)  
 zeroIndexes.Add(-1);  
   
 for (var i = 0; i < height; i++)  
 {  
 var links = new List<int>(width); *// Тут будут храниться ссылки на пред. столбец.* var mins = new List<int>(width); *// Тут хранится минимальное значение в текущем столбце.* var visited = new List<int>(width); *// Массив посещенных стобцов.* for (int a = 0; a < width; a++)  
 {  
 links.Add(-1);  
 mins.Add(int.**MaxValue**);  
 visited.Add(0);  
 }  
   
 var markedI = i; *// i последнего посещенного элемента* var markedJ = -1; *// j последнего посещенного элемента* var j = 0;  
   
 while (markedI != -1)  
 {  
 j = -1;  
   
 for (var j1 = 0; j1 < width; j1++)  
 if (visited[j1] != 1)  
 {  
 *// Находим минимальный элемент в строке.  
 // Добавляем ссылку на пред. элемент в графе.  
 // В j добавим индекс столбца с самым маленьким минимумом.* if (matrix[markedI][j1] - u[markedI] - v[j1] < mins[j1])  
 {  
 mins[j1] = matrix[markedI][j1] - u[markedI] - v[j1];  
 links[j1] = markedJ;  
 }  
 if (j == -1 || mins[j1] < mins[j])  
 j = j1;  
 }  
   
 *// Произведем манипуляции со строками и столбцами так, чтобы обнулился (markIndices[links[j]], j)  
 // (markIndices[links[j]], j) - это будущий ноль, по которому мы востановим минимумы.* for (var j1 = 0; j1 < width; j1++)  
 if (visited[j1] == 1)  
 {  
 u[zeroIndexes[j1]] += mins[j];  
 v[j1] -= mins[j];  
 }  
 else  
 {  
 mins[j1] -= mins[j];  
 }  
 u[i] += mins[j];  
   
 visited[j] = 1;  
 markedJ = j;  
 markedI = zeroIndexes[j];  
 }  
   
 *// Пройдем по найденной цепочке клеток, поставим отметки на неотмеченные клетки.* for (; links[j] != -1; j = links[j])  
 zeroIndexes[j] = zeroIndexes[links[j]];  
 zeroIndexes[j] = i;  
 }  
   
 *// Востанавливаем ответ.* var result = new List<List<int>>();  
 for (var j = 0; j < width; j++)  
 if (zeroIndexes[j] != -1)  
 result.Add(new List<int>() { j, zeroIndexes[j]});  
 return result;  
 }  
}

# Вывод

Я сумел реализовать Венгерский метод решения задачи о назначениях. Программа работает достаточно быстро, хотя есть способы ее оптимизации. В процессе выполнения я узнал о Венгерском методе, применил его на практике. Эта лабораторная работа показалась мне одной из самых сложных среди выполненных мной.