МЕЖГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «????????????»

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ РЕШЕНИЯ ТРАНСПОРТНОЙ ЗАДАЧИ ВЕНГЕРСКИМ МЕТОДОМ

Курсовая работа

по дисциплине «Системный анализ и исследование операций»

КР.1-53 01 02.10028412.15

Исполнитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Самусев Д.А. АСОИ-181

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Плиско И. Г.

(подпись)

Дата допуска к защите \_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_

Могилев 2020

МЕЖГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Факультет Электротехнический

«Утверждаю»

Заведующий кафедрой\_\_\_\_\_\_\_\_\_\_

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

З А Д А Н И Е

на курсовую работу по дисциплине «Системный анализ и исследование операций» студенту Самусев Д.А. гр. № АСОИ-181

1. Тема курсовой работы: Разработка приложения для решения транспортной задачи венгерским методом

2. Срок сдачи законченной работы: 18.05.2020

3. Исходные данные к курсовой работе:

Среда разработки: на выбор студента.

Разработать приложение для решения следующего типа задачи. Составить приложение для решения транспортной задачи венгерским методом.

4. Перечень подлежащих разработке вопросов:

Введение

1 Постановка задачи

2 Проектирование программного модуля

3 Реализация программного модуля

4 Тестирование программного модуля

Заключение

Список использованных источников

5. На проверку предоставляются пояснительная записка, исходные тексты

программ и исполняемые файлы на электронном носителе.

Руководитель курсовой работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Плиско И. Г.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Самусев Д. А.

Содержание

[Введение 3](#_Toc34772304)

[1 Постановка задачи 4](#_Toc34772305)

[1.1 Математическая модель задачи 4](#_Toc34772306)

[1.2 Входные данные 5](#_Toc34772307)

[1.3 Выходные данные 5](#_Toc34772309)

[1.4 Обработка ошибок 5](#_Toc34772311)

[2 Проектирование программного модуля 6](#_Toc34772313)

[2.1 Разработка структурной диаграммы программного модуля и её описание 6](#_Toc34772314)

[2.2 Разработка пользовательского интерфейса 6](#_Toc34772315)

[3 Реализация программного модуля 7](#_Toc34772317)

[3.1 Код программы 7](#_Toc34772318)

[4 Тестирование программного модуля 8](#_Toc34772319)

[Заключение 9](#_Toc34772320)

[Литература 10](#_Toc34772321)

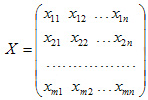
Введение

Под названием “транспортная задача” объединяется широкий круг задач с единой математической моделью. Классическая транспортная задача – задача о наиболее экономном плане перевозок однородного продукта или взаимозаменяемых продуктов из пунктов производства в пункты потребления, встречается чаще всего в практических приложениях линейного программирования. Линейное программирование является одним из разделов математического программирования – области математики, разрабатывающей теорию и численные методы решения многомерных экстремальных задач с ограничениями. Огромное количество возможных вариантов перевозок затрудняет получение достаточно экономного плана эмпирическим или экспертным путем. Применение математических методов и вычислительных в планировании перевозок дает большой экономический эффект. Транспортные задачи могут быть решены симплексным методом, однако матрица системы ограничений транспортной задачи настолько своеобразна, что для ее решения разработаны специальные методы. Эти методы, как и симплексный метод, позволяют найти начальное опорное решение, а затем, улучшая его получить оптимальное решение. В зависимости от способа представления условий транспортной задачи она может быть представлена в сетевой (схематичной) или матричной (табличной) форме. Транспортная задача может также решаться с ограничениями и без ограничений. В данной курсовой работе будут рассматриваться математическая постановка транспортной задачи линейного программирования - венгерский метод.

1 Постановка задачи

* 1. Математическая модель задачи

Переменными (неизвестными) транспортной задачи являются xij , i=1,2,...,m j=1,2,...,n - объемы перевозок от i-го поставщика каждому j-му потребителю. Эти переменные могут быть записаны в виде матрицы перевозок:



Так как произведение Cij\*Xij определяет затраты на перевозку груза от i-го поставщика j-му потребителю, то суммарные затраты на перевозку всех грузов равны:



По условию задачи требуется обеспечить минимум суммарных затрат.  
Следовательно, целевая функция задачи имеет вид:



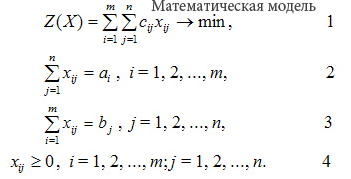
Система ограничений задачи состоит из двух групп уравнений. Первая группа из m уравнений описывает тот факт, что запасы всех m поставщиков вывозятся полностью и имеет вид:



Вторая группа из n уравнений выражает требование удовлетворить запросы всех n потребителей полностью и имеет вид:



Учитывая условие неотрицательности объемов перевозок, математическая модель выглядит следующим образом:



В рассмотренной модели транспортной задачи предполагается, что суммарные запасы поставщиков равны суммарным запросам потребителей, т.е.:



Такая задача называется задачей с **правильным балансом**, а модель задачи **закрытой**. Если же это равенство не выполняется, то задача называется задачей с **неправильным балансом**, а модель задачи — **открытой**. **Математическая формулировка транспортной задачи** такова: найти переменные задачи X=(xij), i=1,2,...,m; j=1,2,...,n, удовлетворяющие системе ограничений (цифра 2 на математической модели) , (3), условиям неотрицательности (4) и обеспечивающие минимум целевой функции (1).

* 1. Входные данные

Входными данными являются количество грузов для каждого транспорта, потребность каждой точки назначения и стоимость перевозки каждым транспортом в каждую точку назначения.

* 1. Выходные данные

Выходными данными являются значение целевой функции (минимальная стоимость перевозок) и матрица распределения транспортов по точкам назначения с необходимым количеством ресурсов.

* 1. Обработка ошибок

В процессе работы, приложение может работать некорректно при введении некорректных данных, таких как отрицательные или строковые значения в качестве входных данных. Для предотвращения подобных исключительных ситуаций разработан механизм, не позволяющий введение подобных данных.

2 Проектирование программного модуля

* 1. Разработка структурной диаграммы программного модуля и её описание
  2. Разработка пользовательского интерфейса

Пользовательский интерфейс представляет собой React js приложение, взаимодействующее с .Net приложением. Пользовательский интерфейс вначале принимает на вход размерность матрицы цен (количество строк и столбцов) - рисунок 1.

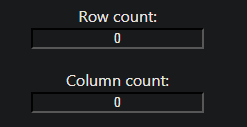


Рисунок 1

Затем заполняется матрица цен, массив имеющихся ресурсов (правый столбец) и массив потребностей (массив под матрицей) – рисунок 2.

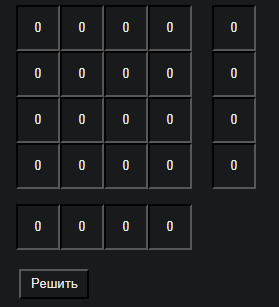


Рисунок 2

После нажатия кнопки “решить” будет предоставлено решение указанной задачи – рисунок 3.



Рисунок 3

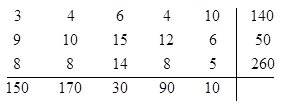
3 Реализация программного модуля

3.1 Код программы

Код программы находится в приложении А.

4 Тестирование программного модуля

Для тестирования возьмём следующее условие:



Матрица цен имеет размерность 3x5. Введём данные соответствующие поля (рисунок 4).

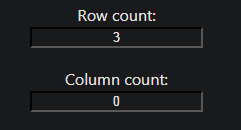


Рисунок 4

Далее заполним соответствующее данные в появившейся матрице (рисунок 5) и нажмём на кнопку “решить”.

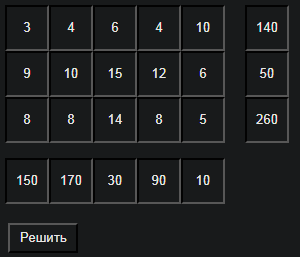


Рисунок 5

После нажатия на кнопку появляется матрица распределения ресурсов и значение целевой функции (рисунок 6).

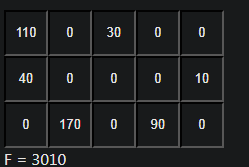
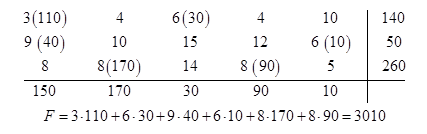


Рисунок 6

Сравним решение с предоставленным решением в примере:



Решение, предоставленное выше, совпадает с решением, которое предоставляет приложение. Таким образом, можно сделать вывод, что приложение работает корректно.

Заключение

Была проделана работа, в результате которой реализована программа, позволяющая решать транспортную задачу венгерским методом. Программа позволяет автоматизировать производственный процесс, что позволяет сократить время на трудоёмкую работу. Т.е. данное приложение позволяет экономить время. Так же программа обладает графическим отображением результатов, что обеспечивает лучшее понимание и освоение программы. Этот факт говорит о том, что сотрудники могут самостоятельно и быстро обучится использовать данное приложение.

Литература

1. http://htmlbook.ru

2. https://docs.microsoft.com

3. http://professorweb.ru

4. https://metanit.com/sharp

5. <https://stackoverflow.com>

6. <http://www.grandars.ru/student/vysshaya-matematika/model-zadachi-lineynogo-programmirovaniya.html>

7. <https://math.semestr.ru/simplex/lec_lp_tip.php>

8. <https://math.semestr.ru/transp/index.php>

9. <https://mathminsk.com/sample/12.aspx>

10. <http://kontromat.ru/?page_id=5163>

11. <https://function-x.ru/zadacha_transportnaja_raspredelitelnyi_metod.html>

Приложение А

Startup.cs

namespace WebApplication  
**{** public class Startup  
 {  
 public Startup(IConfiguration configuration)  
 {  
 Configuration = configuration;  
 }  
  
 public IConfiguration Configuration { get; }  
  
 // This method gets called by the runtime. Use this method to add services to the container.  
 public void ConfigureServices(IServiceCollection services)  
 {  
 services.AddCors(options => options.AddPolicy("CorsPolicy",  
 builder =>  
 {  
 builder.AllowAnyMethod().AllowAnyHeader().WithOrigins("http://localhost:3000").AllowCredentials();  
 }));  
 services.AddControllers();  
 }  
  
 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.  
 public void Configure(IApplicationBuilder app, IWebHostEnvironment env)  
 {  
 if (env.IsDevelopment())  
 {  
 app.UseDeveloperExceptionPage();  
 }  
  
 app.UseCors("CorsPolicy");  
   
 app.UseRouting();  
  
 app.UseAuthorization();  
   
 app.UseEndpoints(endpoints => { endpoints.MapControllers(); });  
 }  
 }  
**}**

Program.cs

namespace WebApplication  
**{** public class Program  
 {  
 public static void Main(string[] args)  
 {  
 CreateHostBuilder(args).Build().Run();  
 }  
  
 public static IHostBuilder CreateHostBuilder(string[] args) =>  
 Host.CreateDefaultBuilder(args)  
 .ConfigureWebHostDefaults(webBuilder => { webBuilder.UseStartup<Startup>(); });  
 }  
**}**

Distributer.cs

namespace WebApplication  
**{** class Distributer  
 {  
 private int[,] \_zeroMatrix;  
 private int[] \_goods;  
 private int[] \_needs;  
 private List<DistributedGood> \_distributedGoods = new List<DistributedGood>();  
 private readonly ArrayHelper \_arrayHelper;  
  
 public Distributer()  
 {  
 \_arrayHelper = new ArrayHelper();  
 }  
  
 public List<DistributedGood> Distribute(int[,] matrix, int[] goods, int[] needs)  
 {  
 \_arrayHelper.SimplifyMatrix(matrix);  
 SetZeroMatrix(matrix);  
 \_goods = new int[goods.Length];  
 \_needs = new int[needs.Length];  
 Array.Copy(needs, \_needs, needs.Length);  
 Array.Copy(goods, \_goods, goods.Length);  
  
 int requiredZeroesCount = 1;  
  
 while (true)  
 {  
 while (true)  
 {  
 List<int> columnNumbers =  
 \_arrayHelper.FindColumnNubersWithRequiredZeroesCount(requiredZeroesCount, \_zeroMatrix);  
  
 if (columnNumbers.Count != 0)  
 {  
 DistributeByColumns(columnNumbers);  
  
 if (DistributionIsOptimal())  
 {  
 return \_distributedGoods;  
 }  
  
 if (DistributionIsImpossible())  
 {  
 OptimizeMatrix(matrix, \_goods.ToList().IndexOf(\_goods.Max()));  
 Array.Copy(needs, \_needs, needs.Length);  
 Array.Copy(goods, \_goods, goods.Length);  
 SetZeroMatrix(matrix);  
 \_distributedGoods.Clear();  
 }  
   
 requiredZeroesCount = 1;  
 continue;  
 }  
  
 List<int> rowNumbers =  
 \_arrayHelper.FindRowNubersWithRequiredZeroesCount(requiredZeroesCount, \_zeroMatrix);  
  
 if (rowNumbers.Count != 0)  
 {  
 DistributeByRows(rowNumbers);  
  
 if (DistributionIsOptimal())  
 {  
 return \_distributedGoods;  
 }  
  
 if (DistributionIsImpossible())  
 {  
 OptimizeMatrix(matrix, \_goods.ToList().IndexOf(\_goods.Max()));  
 Array.Copy(needs, \_needs, needs.Length);  
 Array.Copy(goods, \_goods, goods.Length);  
 SetZeroMatrix(matrix);  
 \_distributedGoods.Clear();  
 }  
   
 requiredZeroesCount = 1;  
 continue;  
 }  
  
 if (DistributionIsImpossible())  
 {  
 OptimizeMatrix(matrix, \_goods.ToList().IndexOf(\_goods.Max()));  
 Array.Copy(needs, \_needs, needs.Length);  
 Array.Copy(goods, \_goods, goods.Length);  
 SetZeroMatrix(matrix);  
 \_distributedGoods.Clear();  
 requiredZeroesCount = 1;  
 }  
  
 break;  
 }  
   
 requiredZeroesCount++;  
 }  
 }  
  
 private void OptimizeMatrix(int[,] matrix, int rowNumber)  
 {  
 int min = 10000;  
  
  
 for (int columnNumber = 0; columnNumber < matrix.GetLength(1); columnNumber++)  
 {  
 if (matrix[rowNumber, columnNumber] < min && matrix[rowNumber, columnNumber] != 0)  
 {  
 min = matrix[rowNumber, columnNumber];  
 }  
 }  
  
 for (int columnNumber = 0; columnNumber < matrix.GetLength(1); columnNumber++)  
 {  
 matrix[rowNumber, columnNumber] -= min;  
 }  
  
 for (int columnNumber = 0; columnNumber < matrix.GetLength(1); columnNumber++)  
 {  
 if (matrix[rowNumber, columnNumber] < 0)  
 {  
 for (int row = 0; row < matrix.GetLength(0); row++)  
 {  
 matrix[row, columnNumber] += min;  
 }  
 }  
 }  
 }  
   
 private bool DistributionIsImpossible()  
 {  
 for (int rowNumber = 0; rowNumber < \_zeroMatrix.GetLength(0); rowNumber++)  
 {  
 for (int columnNumber = 0; columnNumber < \_zeroMatrix.GetLength(1); columnNumber++)  
 {  
 if (\_zeroMatrix[rowNumber, columnNumber] == 0)  
 {  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
  
 private bool DistributionIsOptimal()  
 {  
 return \_needs.ToList().Sum() == 0;  
 }  
  
 private void SetZeroMatrix(int[,] matrix)  
 {  
 \_zeroMatrix = new int[matrix.GetLength(0), matrix.GetLength(1)];  
  
 for (int rowNumber = 0; rowNumber < matrix.GetLength(0); rowNumber++)  
 {  
 for (int columnNumber = 0; columnNumber < matrix.GetLength(1); columnNumber++)  
 {  
 if (matrix[rowNumber, columnNumber] == 0)  
 {  
 \_zeroMatrix[rowNumber, columnNumber] = 0;  
 }  
 else  
 {  
 \_zeroMatrix[rowNumber, columnNumber] = 1;  
 }  
 }  
 }  
 }  
  
 private void DistributeByColumns(List<int> columnNumbers)  
 {  
 foreach (int columnNumber in columnNumbers)  
 {  
 for (int rowNumber = 0; rowNumber < \_zeroMatrix.GetLength(0); rowNumber++)  
 {  
 if (\_zeroMatrix[rowNumber, columnNumber] == 0)  
 {  
 Disrtibute(rowNumber, columnNumber);  
 }  
 }  
 }  
 }  
  
 private void DistributeByRows(List<int> rowNumbers)  
 {  
 foreach (int rowNumber in rowNumbers)  
 {  
 for (int columnNumber = 0; columnNumber < \_zeroMatrix.GetLength(1); columnNumber++)  
 {  
 if (\_zeroMatrix[rowNumber, columnNumber] == 0)  
 {  
 Disrtibute(rowNumber, columnNumber);  
 }  
 }  
 }  
 }  
  
 private void Disrtibute(int rowNumber, int columnNumber)  
 {  
 if (\_needs[columnNumber] <= \_goods[rowNumber])  
 {  
 \_distributedGoods.Add(new DistributedGood(rowNumber, columnNumber, \_needs[columnNumber]));  
 \_goods[rowNumber] -= \_needs[columnNumber];  
 \_needs[columnNumber] = 0;  
 }  
 else  
 {  
 \_distributedGoods.Add(new DistributedGood(rowNumber, columnNumber, \_goods[rowNumber]));  
 \_needs[columnNumber] -= \_goods[rowNumber];  
 \_goods[rowNumber] = 0;  
 }  
  
 CrossMatrixColumnOrRow();  
 }  
  
 private void CrossMatrixColumnOrRow()  
 {  
 for (int rowNumber = 0; rowNumber < \_goods.Length; rowNumber++)  
 {  
 if (\_goods[rowNumber] == 0)  
 {  
 for (var columnNumber = 0; columnNumber < \_zeroMatrix.GetLength(1); columnNumber++)  
 {  
 \_zeroMatrix[rowNumber, columnNumber] = 1;  
 }  
 }  
 }  
  
 for (int columnNumber = 0; columnNumber < \_needs.Length; columnNumber++)  
 {  
 if (\_needs[columnNumber] == 0)  
 {  
 for (var rowNumber = 0; rowNumber < \_zeroMatrix.GetLength(0); rowNumber++)  
 {  
 \_zeroMatrix[rowNumber, columnNumber] = 1;  
 }  
 }  
 }  
 }  
 }  
**}**

DistributedGood.cs

namespace WebApplication  
**{** public class DistributedGood  
 {  
 public int Row { get; set; }  
 public int Column { get; set; }  
 public int Amount { get; set; }  
  
 public DistributedGood(int row, int column, int amount)  
 {  
 Row = row;  
 Column = column;  
 Amount = amount;  
 }  
 }  
**}**

Data.cs

namespace WebApplication  
**{** public class Data  
 {  
 public List<int[]> Matrix { get; set; }  
 public int[] Goods { get; set; }  
 public int[] Needs { get; set; }  
 }  
**}**

ArrayHelper

namespace WebApplication  
**{** public class ArrayHelper  
 {  
   
 public void SimplifyMatrix(int[,] matrix)  
 {  
 int min;  
  
 for (int rowNumber = 0; rowNumber < matrix.GetLength(0); rowNumber++)  
 {  
 min = 10000;  
   
 for (int columnNumber = 0; columnNumber < matrix.GetLength(1); columnNumber++)  
 {  
 if (matrix[rowNumber, columnNumber] < min)  
 {   
 min = matrix[rowNumber, columnNumber];  
 }  
 }  
  
 for (int columnNumber = 0; columnNumber < matrix.GetLength(1); columnNumber++)  
 {  
 matrix[rowNumber, columnNumber] -= min;  
 }  
 }  
  
 for (int columnNumber = 0; columnNumber < matrix.GetLength(1); columnNumber++)  
 {  
 min = 10000;  
  
 for (int rowNumber = 0; rowNumber < matrix.GetLength(0); rowNumber++)  
 {  
 if (matrix[rowNumber, columnNumber] < min)  
 {  
 min = matrix[rowNumber, columnNumber];  
 }  
 }  
   
 for (int rowNumber = 0; rowNumber < matrix.GetLength(0); rowNumber++)  
 {  
 matrix[rowNumber, columnNumber] -= min;  
 }  
 }  
 }  
  
 public List<int> FindColumnNubersWithRequiredZeroesCount(int requiredZeroesCount, int[,] matrix)  
 {  
 var columnNubersWithZeroes = new List<int>();  
  
 for (int columnNuber = 0; columnNuber < matrix.GetLength(1); columnNuber++)  
 {  
 int actualZeroesCount = 0;  
  
 for (int rowNumber = 0; rowNumber < matrix.GetLength(0); rowNumber++)  
 {  
 if (matrix[rowNumber, columnNuber] == 0)  
 {  
 actualZeroesCount++;  
 }  
 }  
  
 if (actualZeroesCount == requiredZeroesCount)  
 {  
 columnNubersWithZeroes.Add(columnNuber);  
 }  
 }  
  
 return columnNubersWithZeroes;  
 }  
  
 public List<int> FindRowNubersWithRequiredZeroesCount(int requiredZeroesCount, int[,] matrix)  
 {  
 var rowNubersWithZeroes = new List<int>();  
  
 for (int rowNumber = 0; rowNumber < matrix.GetLength(0); rowNumber++)  
 {  
 int actualZeroesCount = 0;  
  
 for (int columnNuber = 0; columnNuber < matrix.GetLength(1); columnNuber++)  
 {  
 if (matrix[rowNumber, columnNuber] == 0)  
 {  
 actualZeroesCount++;  
 }  
 }  
  
 if (actualZeroesCount == requiredZeroesCount)  
 {  
 rowNubersWithZeroes.Add(rowNumber);  
 }  
 }  
  
 return rowNubersWithZeroes;  
 }  
 }  
**}**

Home.cs

namespace WebApplication.Controllers  
**{** [Route("api/[controller]")]  
 [Produces("application/json")]  
 [ApiController]  
 public class Home : Controller  
 {  
  
 [HttpGet]  
 public IActionResult GetSln()  
 {  
 return Ok();  
 }  
   
 [HttpPut]  
 public IActionResult GetSolution([FromBody] Data data)  
 {  
 var a = 0;  
 int[,] matrix = CopyMatrix(data);  
   
 var goods1 = new [] {140, 50, 260};  
 var needs1 = new [] {150, 170, 30, 90, 10};  
 var matrix1 = new [,]  
 {  
 { 3, 4, 6, 4, 10},  
 { 9, 10, 15, 12, 6},  
 { 8, 8, 14, 8, 5}  
 };  
   
 var distributer = new Distributer();  
 List<DistributedGood> result = distributer.Distribute(matrix, data.Goods, data.Needs);  
   
 return Ok(result);  
 }  
  
 public int[,] CopyMatrix(Data data)  
 {  
   
 int[,] matrix = new int[data.Goods.Length, data.Needs.Length];  
   
 for (var i = 0; i < matrix.GetLength(0); i++)  
 {  
 for (var j = 0; j < matrix.GetLength(1); j++)  
 {  
 matrix[i, j] = data.Matrix[i][j];  
 }  
 }  
  
 return matrix;  
 }  
 }  
**}**

**SolutionWindow.js**

export default class SolutionWindow extends React.Component {

constructor(props) {

super(props);

this.state = {

rowCount: 0,

columnCount: 0,

matrix: null,

goods: null,

needs: null,

result: null,

};

this.handleMatrixChange = this.handleMatrixChange.bind(this);

this.handleInputChange = this.handleInputChange.bind(this);

this.handleGoodsChange = this.handleGoodsChange.bind(this);

this.handleNeedsChange = this.handleNeedsChange.bind(this);

this.solveMatrix = this.solveMatrix.bind(this);

}

handleInputChange(event) {

let target = event.target;

let value = target.value;

let name = target.name;

if (isNaN(parseInt(value))) {

this.setState({

[name]: 0,

})

} else {

this.setState({

[name]: parseInt(value),

})

}

}

handleMatrixChange(event) {

let target = event.target;

let value = target.value;

let name = target.name;

let row = parseInt(name[0]);

let column = parseInt(name[1]);

let matrix = this.state.matrix;

if (isNaN(parseInt(value))) {

matrix[row][column] = 0;

} else {

matrix[row][column] = parseInt(value);

}

this.setState(() => {

return {};

});

}

handleGoodsChange(event) {

let target = event.target;

let value = target.value;

let name = target.name;

let goods = this.state.goods;

if (isNaN(parseInt(value))) {

goods[parseInt(name)] = 0;

} else {

goods[parseInt(name)] = parseInt(value);

}

this.setState(() => {

return {};

});

}

handleNeedsChange(event) {

let target = event.target;

let value = target.value;

let name = target.name;

let needs = this.state.needs;

if (isNaN(parseInt(value))) {

needs[parseInt(name)] = 0;

} else {

needs[parseInt(name)] = parseInt(value);

}

this.setState(() => {

return {};

});

}

initializeMatrix() {

if (this.state.matrix == null) {

let matrix = [];

for (let row = 0; row < this.state.rowCount; row++) {

matrix[row] = [];

for (let column = 0; column < this.state.columnCount; column++) {

matrix[row][column] = 0;

}

}

this.setState(() => {

return {

matrix: matrix

};

});

}

}

initializeGoods() {

if (this.state.goods == null) {

let goods = this.state.goods;

goods = [];

for (let i = 0; i < this.state.rowCount; i++) {

goods[i] = 0;

}

this.setState(() => {

return {

goods: goods

};

});

}

}

initializeNeeds() {

if (this.state.needs == null) {

let needs = this.state.needs;

needs = [];

for (let i = 0; i < this.state.columnCount; i++) {

needs[i] = 0;

}

this.setState(() => {

return {

needs: needs

};

});

}

}

initializeResult(data) {

this.setState(() => {

return {

result: data

};

});

}

solveMatrix() {

let data = {

matrix: this.state.matrix,

goods: this.state.goods,

needs: this.state.needs,

};

fetch('http://localhost:5000/api/home',

{

method: "put",

body: JSON.stringify(data),

headers:

{

'Accept': 'application/json',

'Content-Type': 'application/json',

}

})

.then(response => {

if (response.status >= 200 && response.status < 300) {

return response.json().then((data) => this.initializeResult(data));

}

})

}

render() {

let columnCount;

let rowCount;

let matrix;

let goods;

let needs;

let solve;

let resultMatrixForView;

let functionResult = null;

if (this.state.rowCount == 0 || this.state.columnCount == 0) {

rowCount =

<div className="rowCount">

Row count:

<input onChange={this.handleInputChange}

className="rowCountInput" name="rowCount"

value={this.state.rowCount}/>

</div>;

columnCount =

<div className="columnCount">

Column count:

<input onChange={this.handleInputChange}

className="columnCountInput"

name="columnCount"

value={this.state.columnCount}/>

</div>;

}

if (this.state.rowCount > 0 && this.state.columnCount > 0) {

this.initializeMatrix();

if (this.state.matrix != null) {

matrix = this.state.matrix.map((row, rowId) => (

<div className="row" name="row" row={row}>

{row.map((elem, id) => (

<input

className="elem"

name={rowId + "" + id}

onChange={this.handleMatrixChange}

value={elem}

/>

))}

</div>

));

}

}

if (this.state.matrix != null) {

this.initializeGoods();

if (this.state.goods != null) {

goods = this.state.goods.map((good, id) =>

<input onChange={this.handleGoodsChange} name={id} className="elem" value={good}/>

);

}

}

if (this.state.matrix != null) {

this.initializeNeeds();

if (this.state.needs != null) {

needs = this.state.needs.map((need, id) =>

<input onChange={this.handleNeedsChange} name={id} className="elem" value={need}/>

);

}

}

if (this.state.matrix != null) {

solve = <button

className="solveButton"

onClick={this.solveMatrix}>

??????</button>

}

if (this.state.result != null) {

let resultMatrix = [];

for (let row = 0; row < this.state.rowCount; row++) {

resultMatrix[row] = [];

for (let column = 0; column < this.state.columnCount; column++) {

resultMatrix[row][column] = 0;

}

}

let result = 0;

for (let i = 0; i < this.state.result.length; i++) {

let elem = this.state.result[i];

resultMatrix[elem.row][elem.column] = elem.amount;

result += this.state.matrix[elem.row][elem.column] \* elem.amount;

functionResult =

<div>

F = {result}

</div>

}

resultMatrixForView = resultMatrix.map(row => (

<div className="row">

{

row.map(elem => (

<input

className="elem"

onChange={this.handleMatrixChange}

value={elem}

/>

))}

</div>

));

}

return (

<div className="app">

<div>

{rowCount}

{columnCount}

</div>

<div className="inline">

<div className="matrix">{matrix}</div>

<div className="goods">{goods}</div>

</div>

<div className="needs">{needs}</div>

<div>

{solve}

</div>

<div className="resultMatrix">

{resultMatrixForView}

</div>

<div className="functionResult">

{functionResult}

</div>

</div>

);

}

}