МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №4**

по дисциплине: Исследование операций

тема: «Закрытая транспортная задача»

Выполнил: ст. группы ПВ-233

Ситников Алексей Павлович

Проверил:

Вирченко Юрий Петрович

Белгород 2025 г.

**Цель работы:** изучить математическую модель транспортной задачи, овладеть методами решения этой задачи.

**Постановка задачи**

1. Изучить содержательную и математическую постановки закрытой транспортной задачи, методы нахождения первого опорного решения ее системы ограничений. Изучить понятие цикла пересчета в матрице перевозок. Овладеть распределительным методом и методом потенциалов, а также их алгоритмами.

2. Составить и отладить программы решения транспортной задачи распределительным методом и методом потенциалов.

3. Для подготовки тестовых данных решить вручную следующую задачу:

Вариант 13

Изображение выглядит как текст, Шрифт, число, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

**Блок-схемы программ:**

**Метод потенциалов**

**Изображение выглядит как текст, диаграмма, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.**

**Распределительный метод**

**![Изображение выглядит как белый, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.]()Изображение выглядит как текст, Шрифт, снимок экрана, белый

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, диаграмма, снимок экрана, чек

Контент, сгенерированный ИИ, может содержать ошибки.**

**Код программ:  
Метод потенциалов**

#include <iostream>  
#include <windows.h>  
#include <vector>  
  
typedef struct node{  
 int C = 0;//цена  
 int x = 0;//сколько отправили  
 int g = 0;//ля вычисление дельты  
 int flag = 1;//для выяснения свободен ли элемент  
}node;  
  
void inputData(std::vector<std::vector<node>> &matrix, std::vector<int> &A, std::vector<int> &B){  
 std::cout **<<** "Введите матрицу стоймости размером " **<<** matrix.size() **<<** "x" **<<** matrix**[**0**]**.size() **<<** ", где ij элемент это стоимость доставки из i-той базы к j-тому потребителю" **<<** std::endl;  
 for(int i = 0; i < matrix.size(); i++){  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 std::cin **>>** matrix**[**i**][**j**]**.C;  
 }  
 }  
 std::cout **<<** "Введите запасы начиная с первого склада\n";  
 for(int &i : A){  
 std::cin **>>** i;  
 }  
  
 std::cout **<<** "Введите запросы начиная с первого потребителя\n";  
 for(int &i : B){  
 std::cin **>>** i;  
 }  
  
}  
  
  
int finduniq(std::vector<std::vector<int>> &rout, int i, int j){  
 for(auto x : rout){  
 if(x**[**0**]** == i && x**[**1**]** == j){  
 return 0;  
 }  
 }  
 return 1;  
}  
int findRout(std::vector<std::vector<int>> &rout, std::vector<std::vector<node>> &matrix, int \*flag,  
 int start\_i, int start\_j, int current\_i, int current\_j,  
 int prev\_i, int prev\_j, int direction) {  
 // direction: 0 - начальное, 1 - горизонтальное, 2 - вертикальное  
  
 // Если вернулись в начало и цикл не пустой  
 if (current\_i == start\_i && current\_j == start\_j && rout.size() > 1) {  
 \*flag = 1;  
 return 1;  
 }  
  
 // Проверяем, были ли уже в этой точке (кроме стартовой)  
 if (!(current\_i == start\_i && current\_j == start\_j)) {  
 if (!finduniq(rout, current\_i, current\_j)) {  
 return 0;  
 }  
 }  
  
  
  
  
 // Чередуем направления (горизонтальное/вертикальное)  
 if (direction != 1) { // Проверяем по вертикали  
 for (int i = 0; i < matrix.size(); i++) {  
 if (i != current\_i && i != prev\_i && matrix**[**i**][**current\_j**]**.x != 0) {  
 rout.push\_back(**{**current\_i, current\_j**}**);  
  
 if (findRout(rout, matrix, flag, start\_i, start\_j,  
 i, current\_j, current\_i, current\_j, 2)) {  
 return 1;  
 }  
 }  
 }  
 }  
  
 if (direction != 2) { // Проверяем по горизонтали  
 for (int j = 0; j < matrix**[**0**]**.size(); j++) {  
 if (j != current\_j && j != prev\_j && matrix**[**current\_i**][**j**]**.x != 0) {  
 rout.push\_back(**{**current\_i, current\_j**}**);  
  
 if (findRout(rout, matrix, flag, start\_i, start\_j,  
 current\_i, j, current\_i, current\_j, 1)) {  
 return 1;  
 }  
 }  
 }  
 }  
  
 // Если путь не найден - откатываемся  
  
 return 0;  
}  
  
// Функция-обертка для запуска поиска  
void findCycle(std::vector<std::vector<int>> &rout, std::vector<std::vector<node>> &matrix, int \*flag,  
 int start\_i, int start\_j) {  
 \*flag = 0;  
 findRout(rout, matrix, flag, start\_i, start\_j, start\_i, start\_j, -1, -1, 0);  
}  
  
void creatPlan(std::vector<std::vector<node>> &m, std::vector<int> A, std::vector<int> B){  
 //северо-западный угол  
 for(int i = 0; i < m.size(); i++){  
 for(int j = 0; j < m**[**0**]**.size(); j++){  
 if(m**[**i**][**j**]**.flag) {//если элемент доступен  
 if (A**[**i**]** > B**[**j**]**) {  
 m**[**i**][**j**]**.x = B**[**j**]**;  
 A**[**i**]** -= B**[**j**]**;  
 B**[**j**]** = 0;  
 for (auto &row: m) {  
 row**[**j**]**.flag = 0;  
 }  
 } else if (A**[**i**]** < B**[**j**]**) {  
 m**[**i**][**j**]**.x = A**[**i**]**;  
 B**[**j**]** -= A**[**i**]**;  
 A**[**i**]** = 0;  
 for(auto &col : m**[**i**]**){  
 col.flag = 0;  
 }  
 }  
 else{  
 m**[**i**][**j**]**.x = A**[**i**]**;  
 B**[**j**]** = 0;  
 A**[**i**]** = 0;  
 for (auto &row: m) {  
 row**[**j**]**.flag = 0;  
 }  
 }  
 }  
 }  
 }  
}  
//метод потенциалов  
void potencial(std::vector<std::vector<node>> &matrix, std::vector<int> &point, std::vector<int> &u, std::vector<int> &v){  
 while (true) {//пока не будет решение  
 for (int i = 0; i < matrix.size(); i++) {//находим u и v  
 for (int j = 0; j < matrix**[**0**]**.size(); j++) {  
 if (matrix**[**i**][**j**]**.x != 0) {  
 if (i == 0) {  
 v**[**j**]** = matrix**[**i**][**j**]**.C;  
 } else if (u**[**i**]** != 0) {  
 v**[**j**]** = matrix**[**i**][**j**]**.C - u**[**i**]**;  
 } else {  
 u**[**i**]** = matrix**[**i**][**j**]**.C - v**[**j**]**;  
 }  
 }  
 }  
 }  
 int min = **INT\_MAX**;  
 //вычисляем дельты  
 for (int i = 0; i < matrix.size(); i++) {  
 for (int j = 0; j < matrix**[**0**]**.size(); j++) {  
 matrix**[**i**][**j**]**.g = matrix**[**i**][**j**]**.C - u**[**i**]** - v**[**j**]**;  
 if (min > matrix**[**i**][**j**]**.g) {  
 min = matrix**[**i**][**j**]**.g;  
 }  
 }  
 }  
 if (min >= 0) {  
 return;  
 }  
 std::vector<std::vector<int>> rout(1);  
 //делаем перераспределение  
 for (int i = 0; i < matrix.size(); i++) {  
 int isEnd = 0;  
 for (int j = 0; j < matrix**[**0**]**.size(); j++) {  
 if (min == matrix**[**i**][**j**]**.g) {  
 rout**[**0**]**.push\_back(i);  
 rout**[**0**]**.push\_back(j);  
 int flag = 0;  
 findCycle(rout, matrix, &flag, i, j);//находим цикл  
 isEnd = 1;  
 break;  
 }  
 }  
 if(isEnd){  
 break;  
 }  
 }  
 //находим насколько надо перебрать  
 int minC = matrix**[**rout**[**0**][**0**]][**rout**[**0**][**1**]]**.C < matrix**[**rout**[**rout.size()-1**][**0**]][**rout**[**rout.size()-1**][**1**]]**.C ? matrix**[**rout**[**0**][**0**]][**rout**[**0**][**1**]]**.C : matrix**[**rout**[**rout.size()-1**][**0**]][**rout**[**rout.size()-1**][**1**]]**.C;  
   
 //перераспределение  
 for(int i = 0; i < rout.size(); i++){  
 if(i%2==0){  
 matrix**[**rout**[**i**][**0**]][**rout**[**i**][**1**]]**.x += minC;  
 }  
 else{  
 matrix**[**rout**[**i**][**0**]][**rout**[**i**][**1**]]**.x -= minC;  
 }  
 }  
 //опустошаем векторы для повторной работы  
 std::fill(u.begin(), u.end(), 0);  
 std::fill(v.begin(), v.end(), 0);  
 }  
}  
  
void print(std::vector<std::vector<node>> &matrix){  
 //выводим решение  
 for(int i = 0; i < matrix.size(); i++){  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 if(matrix**[**i**][**j**]**.x!=0){  
 std::cout **<<** matrix**[**i**][**j**]**.x **<<** " ";  
 }  
 else{  
 std::cout **<<** " ";  
 }  
 }  
 std::cout **<<** "\n";  
 }  
 int sum = 0;  
 int f = 0;  
 for(int i = 0; i < matrix.size(); i++){  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 if(matrix**[**i**][**j**]**.x!=0){  
 if(f){  
 std::cout **<<** "\* ";  
 }  
 std::cout **<<** matrix**[**i**][**j**]**.x **<<** "\*" **<<** matrix**[**i**][**j**]**.C **<<**" ";  
 f = 1;  
 sum+=matrix**[**i**][**j**]**.x \* matrix**[**i**][**j**]**.C;  
 }  
 }  
 }  
 std::cout **<<** "= " **<<** sum;  
}

int main(){  
 SetConsoleOutputCP(**CP\_UTF8**);  
 int a, b;  
 std::cout **<<** "Введите количество баз и количество потребителей"**<<** std::endl;  
 std::cin **>>** a **>>** b;  
  
 std::vector<std::vector<node>> matrix(a, std::vector<node>(b));  
 std::vector<int> A(a);  
 std::vector<int> B(b);  
 inputData(matrix, A, B);  
 creatPlan(matrix, A, B);  
 std::vector<int> point(2);  
 std::vector<int> u(matrix.size());  
 std::vector<int> v(matrix**[**0**]**.size());  
 potencial(matrix, point, u, v);  
 print(matrix);  
   
  
  
  
  
  
 return 0;  
}

Вывод программы:  
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Распределительный метод**

#include <iostream>  
#include <windows.h>  
#include <vector>  
  
typedef struct node{  
 int C = 0;//цена  
 int x = 0;//сколько отправили  
 int flag = 1;//для выяснения свободен ли элемент  
}node;  
  
void inputData(std::vector<std::vector<node>> &matrix, std::vector<int> &A, std::vector<int> &B){  
 std::cout **<<** "Введите матрицу стоймости размером " **<<** matrix.size() **<<** "x" **<<** matrix**[**0**]**.size() **<<** ", где ij элемент это стоимость доставки из i-той базы к j-тому потребителю" **<<** std::endl;  
 for(int i = 0; i < matrix.size(); i++){  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 std::cin **>>** matrix**[**i**][**j**]**.C;  
 }  
 }  
 std::cout **<<** "Введите запасы начиная с первого склада\n";  
 for(int &i : A){  
 std::cin **>>** i;  
 }  
  
 std::cout **<<** "Введите запросы начиная с первого потребителя\n";  
 for(int &i : B){  
 std::cin **>>** i;  
 }  
  
}  
  
  
int finduniq(std::vector<std::vector<int>> &rout, int i, int j){  
 for(auto x : rout){  
 if(x**[**0**]** == i && x**[**1**]** == j){  
 return 0;  
 }  
 }  
 return 1;  
}  
int findRout(std::vector<std::vector<int>> &rout, std::vector<std::vector<node>> &matrix, int \*flag,  
 int start\_i, int start\_j, int current\_i, int current\_j,  
 int prev\_i, int prev\_j, int direction) {  
 // direction: 0 - начальное, 1 - горизонтальное, 2 - вертикальное  
  
 // Если вернулись в начало и цикл не пустой  
 if (current\_i == start\_i && current\_j == start\_j && rout.size() > 1) {  
 \*flag = 1;  
 return 1;  
 }  
  
 // Проверяем, были ли уже в этой точке (кроме стартовой)  
 if (!(current\_i == start\_i && current\_j == start\_j)) {  
 if (!finduniq(rout, current\_i, current\_j)) {  
 return 0;  
 }  
 }  
  
  
  
  
 // Чередуем направления (горизонтальное/вертикальное)  
 if (direction != 1) { // Проверяем по вертикали  
 for (int i = 0; i < matrix.size(); i++) {  
 if (i != current\_i && i != prev\_i && matrix**[**i**][**current\_j**]**.x != 0) {  
 rout.push\_back(**{**current\_i, current\_j**}**);  
  
 if (findRout(rout, matrix, flag, start\_i, start\_j,  
 i, current\_j, current\_i, current\_j, 2)) {  
 return 1;  
 }  
 }  
 }  
 }  
  
 if (direction != 2) { // Проверяем по горизонтали  
 for (int j = 0; j < matrix**[**0**]**.size(); j++) {  
 if (j != current\_j && j != prev\_j && matrix**[**current\_i**][**j**]**.x != 0) {  
 rout.push\_back(**{**current\_i, current\_j**}**);  
  
 if (findRout(rout, matrix, flag, start\_i, start\_j,  
 current\_i, j, current\_i, current\_j, 1)) {  
 return 1;  
 }  
 }  
 }  
 }  
  
 // Если путь не найден - откатываемся  
  
 return 0;  
}  
  
// Функция-обертка для запуска поиска  
int findCycle(std::vector<std::vector<int>> &rout, std::vector<std::vector<node>> &matrix, int \*flag,  
 int start\_i, int start\_j) {  
 \*flag = 0;  
 return findRout(rout, matrix, flag, start\_i, start\_j, start\_i, start\_j, -1, -1, 0);  
}  
  
void creatPlan(std::vector<std::vector<node>> &m, std::vector<int> A, std::vector<int> B){  
 //северо-западный угол  
 for(int i = 0; i < m.size(); i++){  
 for(int j = 0; j < m**[**0**]**.size(); j++){  
 if(m**[**i**][**j**]**.flag) {//если элемент доступен  
 if (A**[**i**]** > B**[**j**]**) {  
 m**[**i**][**j**]**.x = B**[**j**]**;  
 A**[**i**]** -= B**[**j**]**;  
 B**[**j**]** = 0;  
 for (auto &row: m) {  
 row**[**j**]**.flag = 0;  
 }  
 } else if (A**[**i**]** < B**[**j**]**) {  
 m**[**i**][**j**]**.x = A**[**i**]**;  
 B**[**j**]** -= A**[**i**]**;  
 A**[**i**]** = 0;  
 for(auto &col : m**[**i**]**){  
 col.flag = 0;  
 }  
 }  
 else{  
 m**[**i**][**j**]**.x = A**[**i**]**;  
 B**[**j**]** = 0;  
 A**[**i**]** = 0;  
 for (auto &row: m) {  
 row**[**j**]**.flag = 0;  
 }  
 }  
 }  
 }  
 }  
}  
//распределительный метод  
void raspredelenie(std::vector<std::vector<node>> &matrix, std::vector<int> &point){  
 while (true) {//пока не будет решение  
 std::vector<std::vector<int>> rout(1);  
 //подсчитываем g  
 int min = **INT\_MAX**;  
 int indX = 0;//запоминаем индексы минимального элемента по сумме  
 int indY = 0;  
 for (int i = 0; i < matrix.size(); i++) {  
   
 int flag = 0;  
 for (int j = 0; j < matrix**[**0**]**.size(); j++) {//находим гаммы  
 if (findCycle(rout, matrix, &flag, i, j)) {//находим цикл  
 int sum = 0;  
 for(int k = 0; k < rout.size(); k++){  
 if(k%2==0){//находим гаммы  
 sum+=matrix**[**rout**[**i**][**0**]][**rout**[**i**][**1**]]**.C;  
 }  
 else{  
 sum-=matrix**[**rout**[**i**][**0**]][**rout**[**i**][**1**]]**.C;  
 }  
 }  
 if(min > sum){//запоминаем минимальные  
 min = sum;  
 indX = i;  
 indY = j;  
 }  
 }  
 }  
 }  
 int flag = 0;//минимальный перераспределяем  
 findCycle(rout, matrix, &flag, indX, indY);  
 //находим насколько надо перебрать  
 int minC = matrix**[**rout**[**0**][**0**]][**rout**[**0**][**1**]]**.C < matrix**[**rout**[**rout.size()-1**][**0**]][**rout**[**rout.size()-1**][**1**]]**.C ? matrix**[**rout**[**0**][**0**]][**rout**[**0**][**1**]]**.C : matrix**[**rout**[**rout.size()-1**][**0**]][**rout**[**rout.size()-1**][**1**]]**.C;  
  
 //перераспределение  
 for(int i = 0; i < rout.size(); i++){  
 if(i%2==0){  
 matrix**[**rout**[**i**][**0**]][**rout**[**i**][**1**]]**.x += minC;  
 }  
 else{  
 matrix**[**rout**[**i**][**0**]][**rout**[**i**][**1**]]**.x -= minC;  
 }  
 }  
 if(min >= 0){//если минимальный не отрицательный, то оптимальное решение найдено  
 break;  
 }  
 }  
}  
  
void print(std::vector<std::vector<node>> &matrix){  
 //выводим решение  
 for(int i = 0; i < matrix.size(); i++){  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 if(matrix**[**i**][**j**]**.x!=0){  
 std::cout **<<** matrix**[**i**][**j**]**.x **<<** " ";  
 }  
 else{  
 std::cout **<<** " ";  
 }  
 }  
 std::cout **<<** "\n";  
 }  
 int sum = 0;  
 int f = 0;  
 for(int i = 0; i < matrix.size(); i++){  
 for(int j = 0; j < matrix**[**0**]**.size(); j++){  
 if(matrix**[**i**][**j**]**.x!=0){  
 if(f){  
 std::cout **<<** "\* ";  
 }  
 std::cout **<<** matrix**[**i**][**j**]**.x **<<** "\*" **<<** matrix**[**i**][**j**]**.C **<<**" ";  
 f = 1;  
 sum+=matrix**[**i**][**j**]**.x \* matrix**[**i**][**j**]**.C;  
 }  
 }  
 }  
 std::cout **<<** "= " **<<** sum;  
}  
  
int main(){  
 SetConsoleOutputCP(**CP\_UTF8**);  
 int a, b;  
 std::cout **<<** "Введите количество баз и количество потребителей"**<<** std::endl;  
 std::cin **>>** a **>>** b;  
  
 std::vector<std::vector<node>> matrix(a, std::vector<node>(b));  
 std::vector<int> A(a);  
 std::vector<int> B(b);  
 inputData(matrix, A, B);  
 creatPlan(matrix, A, B);  
 std::vector<int> point(2);  
 raspredelenie(matrix, point);  
 print(matrix);  
 return 0;  
}

Вывод программы:

**Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.**

Решения совпали

**Аналитическое решение методом потенциалов.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поставщик\потребитель | 14 | 15 | 15 | 15 | 15 |
| 28 | 14 | 14 |  |  |  |
| 15 |  | 1 | 14 |  |  |
| 17 |  |  | 1 | 15 | 1 |
| 14 |  |  |  |  | 14 |

Проверка оптимальности методом потенциалов

Вычисление потенциалов:  
v1 = 27, v2 = 6, v3 = 0, v4 = 2, v5 = 14.

u1 = 0, u2 = 19, u3 = 15, u4 = -1

Ɣ13 = 8, Ɣ14 = 10, Ɣ15 = 9, Ɣ21 = -45, Ɣ24 = -10, Ɣ25 = -21, Ɣ31 = -14, Ɣ32 = -2, Ɣ41 = -10, Ɣ42 = 17, Ɣ43 = 19, Ɣ 44 = 4.

Есть отрицательные элементы, минимальный Ɣ21. Строим цикл и получаем:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поставщик\потребитель | 14 | 15 | 15 | 15 | 15 |
| 28 | 13 | 15 |  |  |  |
| 15 | 1 |  | 14 |  |  |
| 17 |  |  | 1 | 15 | 1 |
| 14 |  |  |  |  | 14 |

Проверка оптимальности методом потенциалов

Вычисление потенциалов:  
v1 = 27, v2 = 6, v3 = 45, v4 = 47, v5 = 59.

u1 = 0, u2 = -26, u3 = -30, u4 = -46

Ɣ13 = -37, Ɣ14 = -35, Ɣ15 = -36, Ɣ22 = 45, Ɣ24 = -10, Ɣ25 = -21, Ɣ31 = 31, Ɣ32 = 45, Ɣ41 = 35, Ɣ42 = 62, Ɣ43 = 14, Ɣ 44 = 4.

Есть отрицательные элементы, значит план не оптимальный, минимальный Ɣ13, строим цикл

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поставщик\потребитель | 14 | 15 | 15 | 15 | 15 |
| 28 |  | 15 | 13 |  |  |
| 15 | 14 |  | 1 |  |  |
| 17 |  |  | 1 | 15 | 1 |
| 14 |  |  |  |  | 14 |

v1 = 16, v2 = 6, v3 = 8, v4 = 10, v5 = 22.

u1 = 0, u2 = 11, u3 = 7, u4 = -9.

Ɣ11 = 11, Ɣ14 = 2, Ɣ15 = 1, Ɣ22 = 3, Ɣ24 = 0, Ɣ25 = -21, Ɣ31 = 5, Ɣ32 = 6, Ɣ41 = 9, Ɣ42 = 25, Ɣ43 = 19, Ɣ 44 = 4.

Есть отрицательные элементы, значит план не оптимальный, минимальный Ɣ25, строим цикл

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поставщик\потребитель | 14 | 15 | 15 | 15 | 15 |
| 28 |  | 15 | 13 |  |  |
| 15 | 14 |  |  |  | 1 |
| 17 |  |  | 2 | 15 |  |
| 14 |  |  |  |  | 14 |

v1 = 10, v2 = 6, v3 = 8, v4 = 10, v5 = 21.

u1 = 0, u2 = -9, u3 = 7, u4 = -8.

Ɣ11 = 11, Ɣ14 = 2, Ɣ15 = 2, Ɣ22 = 27, Ɣ23 = 20, Ɣ24 = 12, Ɣ31 = 11, Ɣ32 = 6, Ɣ35 = 1, Ɣ41 = 14, Ɣ42 = 24, Ɣ43 = 18, Ɣ 44 = 3.

Отрицательные элементы отсутствуют, оптимальный план получен

**Подсчёт стоимости:**

15\*6 + 13\*8 + 14\*1 + 1\*12 + 2\*15 + 15\*17+14\*13 = **687**

**Аналитическое решение распределительным методом.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поставщик\потребитель | 14 | 15 | 15 | 15 | 15 |
| 28 | 14 | 14 |  |  |  |
| 15 |  | 1 | 14 |  |  |
| 17 |  |  | 1 | 15 | 1 |
| 14 |  |  |  |  | 14 |

Ɣ13 = с13 - с12 + с22 - с23 = 8–6 + 25–19 = 8

Ɣ21 = с21 – с11 + с12 – с22 = 1–27 + 6–25 = -45 < 0

Ɣ24 = с24 – с34 + с33 - с23 = 11–17 + 15–19 = -10 < 0

Ɣ25 = с25 – с23 + с33 – с35 = 12–19 + 15–29 = -21 < 0

Ɣ32 = с32 - с42 + с43 – с33 = 19–22 + 18–15 = 0

Ɣ43 = с43 – с33 + с35 – с45 = 18–15 + 29–13 = 19

Ɣ44 = с44 – с34 + с35 – с45 = 5–17 + 29–13 = 4

Ɣ21 включаем в решение:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поставщик\потребитель | 14 | 15 | 15 | 15 | 15 |
| 28 | 13 | 15 |  |  |  |
| 15 | 1 |  | 14 |  |  |
| 17 |  |  | 1 | 15 | 1 |
| 14 |  |  |  |  | 14 |

Ɣ13 = с13 - с11 + с21 – с23 = 8–27 + 1–19 = -37 < 0

Ɣ22 = с22 – с21 + с11 – с12 = 25–1 + 27–6 = 45

Ɣ24 = с24 – с34 + с33 - с23 = 11–17 + 15–19 = -10 < 0

Ɣ43 = с43 – с33 + с35 – с45 = 18–15 + 29–13 = 19

Ɣ44 = с44 – с34 + с35 – с45 = 5–17 + 29–13 = 4

Ɣ13 включаем в решение:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поставщик\потребитель | 14 | 15 | 15 | 15 | 15 |
| 28 |  | 15 | 13 |  |  |
| 15 | 14 |  | 1 |  |  |
| 17 |  |  | 1 | 15 | 1 |
| 14 |  |  |  |  | 14 |

Ɣ11 = с11 – с21 + с23 – с13 = 27–1 + 19–8 = 37

Ɣ14 = с14 - с13 + с33 – с34 = 12–8 + 15–17 = 2

Ɣ15 = с15 - с13 + с33 – с35 = 23–8 + 15–29 = 1

Ɣ22 = с22 - с12 + с13 – с23 = 25–6 +8–19 = 8

Ɣ24 = с24 – с23 + с33 – с34 = 17–15 + 18–5 = 15

Ɣ25 = с25 – с23 + с33 – с35 = 12–19 + 15–29 = -21 < 0

Ɣ31 = с31 – с21 + с23 – с33 = 28-1+19-15 = 31

Ɣ32 = с32 – с12 + с13 – с33 = 19-6+8-15 = 6

Ɣ43 = с43 – с33 + с35 – с45 = 18-15+29-13 = 19

Ɣ44 = с44 – с34 + с35 – с45 = 5-17+29-13 = 4

Ɣ25 включаем в решение:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Поставщик\потребитель | 14 | 15 | 15 | 15 | 15 |
| 28 |  | 15 | 13 |  |  |
| 15 | 14 |  |  |  | 1 |
| 17 |  |  | 2 | 15 |  |
| 14 |  |  |  |  | 14 |

Ɣ14 = с14 – с34 + с33 – с13= 12–17 + 15–8 = 2

Ɣ32 = с32 – с12 + с13 – с33 = 19-6+8-15 = 6

Ɣ41 = с41 – с21 + с25 – с45 = 16-1+12-13 = 14

Отрицательных элементов нет, оптимальный план получен.

**Подсчёт стоимости:**

15\*6 + 13\*8 + 14\*1 + 1\*12 + 2\*15 + 15\*17+14\*13 = **687**

Решения совпали.

**Вывод:** после проделанной работы получили программы, которые решают транспортную задачу методом потенциалов и распределительным методом.