**Лабораторная работа 3. Метод ветвей и границ. Задача коммивояжера и методы её решения.**

**ЦЕЛЬ РАБОТЫ:** освоить общие принципы решения задач методом ветвей и границ, решить задачу о коммивояжере данным методом, сравнить полученное решение задачи с комбинаторным методом перестановок.

**Примечание:** Задания и вопросы со знаком (\*), выполняются в необязательном порядке, но их выполнение поощряется.

**Задание 1.** Сформулировать условие задачи коммивояжера с параметром. Для этого:

* принять элементы матрицы расстояний равными:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** | C:\Users\mysha\AppData\Local\Temp\ksohtml13632\wps1.jpg | 2 | 22 | C:\Users\mysha\AppData\Local\Temp\ksohtml13632\wps2.jpg | 1 |
| **2** | 1 | C:\Users\mysha\AppData\Local\Temp\ksohtml13632\wps3.jpg | 16 | 67 | 83 |
| **3** | 3 | 3 | C:\Users\mysha\AppData\Local\Temp\ksohtml13632\wps4.jpg | 86 | 50 |
| **4** | 18 | 57 | 4 | C:\Users\mysha\AppData\Local\Temp\ksohtml13632\wps5.jpg | 3 |
| **5** | 92 | 67 | 52 | 14 | C:\Users\mysha\AppData\Local\Temp\ksohtml13632\wps6.jpg |

где *n* – номер варианта или номер по журналу;

**Задание 2.** Решить сформулированную задачу методом ветвей и границ.

**Примечание**: отчет по лабораторной работе должен быть выполнен в MS Word и должен содержать:

* Название лабораторной работы;
* условие (матрица расстояний);
* ход решения (граф решения, обоснование ветвления и вычисление границ **для всех этапов**) – можете оформить в табличном виде;
* решение (если их несколько, то все решения).

**Ход решения:**

Имеем 5 городов, построим матрицу расстояний между городами:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 2 | 22 | INF | 1 |
| 2 | 1 | INF | 16 | 67 | 83 |
| 3 | 3 | 3 | INF | 86 | 50 |
| 4 | 18 | 57 | 4 | INF | 3 |
| 5 | 92 | 67 | 52 | 14 | INF |

Находим минимальное значение в каждой строке (di) и выписываем его в отдельный столбец:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 2 | 22 | INF | 1 | 1 |
| 2 | 1 | INF | 16 | 67 | 83 | 1 |
| 3 | 3 | 3 | INF | 86 | 50 | 3 |
| 4 | 18 | 57 | 4 | INF | 3 | 3 |
| 5 | 92 | 67 | 52 | 14 | INF | 14 |
| **22** |

Производим приведение строк – из каждого элемента в строке вычитаем соответствующее значение найденного минимума (di).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | INF | 1 | 21 | INF | 0 | 1 |
| 2 | 0 | INF | 15 | 66 | 82 | 1 |
| 3 | 0 | 0 | INF | 83 | 47 | 3 |
| 4 | 15 | 54 | 1 | INF | 0 | 3 |
| 5 | 78 | 53 | 38 | 0 | INF | 14 |
| **22** |

Находим минимальные значения в каждом столбце (dj). Эти минимумы выписываем в отдельную строку.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | | | 3 | 4 | | 5 |
| 1 | INF | 1 | | | 21 | INF | | 0 |
| 2 | 0 | INF | | | 15 | 66 | | 82 |
| 3 | 0 | 0 | | | INF | 83 | | 47 |
| 4 | 15 | 54 | | | 1 | INF | | 0 |
| 5 | 78 | 53 | | | 38 | 0 | | INF |
| 0 | | 0 | 1 | | | 0 | 0 | | 1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | | | 3 | 4 | | 5 |
| 1 | INF | 1 | | | 20 | INF | | 0 |
| 2 | 0 | INF | | | 14 | 66 | | 82 |
| 3 | 0 | 0 | | | INF | 83 | | 47 |
| 4 | 15 | 54 | | | 0 | INF | | 0 |
| 5 | 78 | 53 | | | 37 | 0 | | INF |
| 0 | | 0 | 1 | | | 0 | 0 | | 1 |

Вычитаем из каждого элемента матрицы соответствующее ему минимальные значения в каждом столбце dj.

Тогда корневой вершиной будет

**f=22+1=23.**

Для каждой нулевой клетки получившейся преобразованной матрицы находим «оценку». Полученную оценку записываем рядом с нулем, в скобках.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |
| **1** | INF | 1 | 20 | INF | 0(1) |
| **2** | 0(14) | INF | 14 | 66 | 82 |
| **3** | 0(0) | 0(1) | INF | 83 | 47 |
| **4** | 15 | 54 | 0(14) | INF | 0(0) |
| **5** | 78 | 53 | 37 | 0(103) | INF |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |  |
| **1** | INF | 1 | 20 | INF | 0 | 0 |
| **2** | 0 | INF | 14 | 66 | 82 | 0 |
| **3** | 0 | 0 | INF | 83 | 47 | 0 |
| **4** | 15 | 54 | 0 | INF | 0 | 0 |
| **5** | 78 | 53 | 37 | ∞ | INF | 37 |
|  | 0 | 0 | 0 | 66 | 0 | 103 |

Выбираем нулевую клетку с наибольшей оценкой. Будем рассматривать дугу (5,4). Так как удаление дуги (5,4) позволяет получить самую большую константу приведения, т.е. увеличение нижней границы. Для этого заменим вес дуги (5,4) на знак “INF.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |
| **1** | INF | 1 | 20 | INF | 0(1) |
| **2** | 0(14) | INF | 14 | 66 | 82 |
| **3** | 0(0) | 0(1) | INF | 83 | 47 |
| **4** | 15 | 54 | 0(14) | INF | 0(0) |
| **5** | 78 | 53 | 37 | 0(103) | INF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **5** |
| **1** | INF | 1 | 20 | 0 |
| **2** | 0 | INF | 14 | 82 |
| **3** | 0 | 0 | INF | 47 |
| **4** | 15 | 54 | 0 | INF |

Видим, что матрица уже является приведенной.

Значит снова для каждой нулевой клетки получившейся преобразованной матрицы находим «оценку». Полученную оценку записываем рядом с нулем, в скобках.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **5** |
| **1** | INF | 1 | 20 | 0(48) |
| **2** | 0(14) | INF | 14 | 82 |
| **3** | 0(0) | 0(1) | INF | 47 |
| **4** | 15 | 54 | 0(29) | INF |

Далее будем рассматривать дугу (1,5). Так как удаление дуги (1,5) позволяет получить самую большую константу приведения, т.е. увеличение нижней границы. Для этого заменим вес дуги (1,5) на знак “INF.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **5** |
| **1** | INF | 1 | 20 | 0 |
| **2** | 0 | INF | 14 | 82 |
| **3** | 0 | 0 | INF | 47 |
| **4** | 15 | 54 | 0 | INF |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **5** |
| **1** | INF | 1 | 20 | 0 |
| **2** | 0 | INF | 14 | 82 |
| **3** | 0 | 0 | INF | 47 |
| **4** | 15 | 54 | 0 | INF |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **2** | **3** |
| **2** | 0 | INF | 14 |
| **3** | 0 | 0 | INF |
| **4** | INF | 54 | 0 |

Cнова для каждой нулевой клетки получившейся преобразованной матрицы находим «оценку». Полученную оценку записываем рядом с нулем, в скобках.

Для того, чтобы кольцо не замкнулось раньше времени, ставим INF в позиции (4,1).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **2** | **3** |
| **2** | 0(14) | INF | 14 |
| **3** | 0(0) | 0(54) | INF |
| **4** | INF | 54 | 0(68) |

Далее будем рассматривать дугу (4,3). Так как удаление дуги (4,3) позволяет получить самую большую константу приведения, т.е. увеличение нижней границы. Для этого заменим вес дуги (3,2) на знак “INF.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **2** | **3** |
| **2** | 0 | INF | 14 |
| **3** | 0 | 0 | INF |
| **4** | INF | 54 | 0 |

Ту строку и тот столбец, где образовалось два знака «INF», полностью вычеркиваем.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **1** | **2** | **3** |
| **2** | 0 | INF | 14 |
| **3** | 0 | 0 | INF |
| **4** | INF | 54 | 0 |

|  |  |  |
| --- | --- | --- |
|  | **1** | **2** |
| **2** | 0 | INF |
| **3** | INF | 0 |

**φ=23+0=23**

**φ=22+1=23**

**R**

**φ=23+0=23**

**R (5, 4) (1, 5)**

**R (5,4)**

**R (5, 4) (1, 5)**

**(4, 3)**

**φ=23+48=71**

**φ=23+54=77**

**φ=23+0=23**

**R (5, 4)**

**φ=23+103=126**

**R (5, 4)**

**(1, 5)**

**R**

**(2, 1)**

**(3, 2)**

У нас остаются два маршрута (2,1) и (4,3)

**(5,4) (1,5), (4,3), (2,1), (3,2)**

**Решение: (1,5), (5,4), (4,3), (3,2), (2,1)**

**Задание 2\*.** Предложить решение задачи коммивояжера методом вервей и границ на языке C++. Код, анализ и вывод включить в отчет.

**Задание 3.** Проверить полученное решение при помощи генератора перестановок (см. лаб. 2, задание 5.1.) и включить копию экрана с решением в отчет.

#include <iostream>  
#include <vector>  
#include <algorithm>  
#include <limits>  
#include <iomanip>  
  
using namespace std;  
  
const int INF = numeric\_limits<int>::max();  
  
vector<int> solveTravelingSalesmanProblem(const vector<vector<int>>& distances, int& totalDistance) {  
 int n = distances.size();  
 vector<int> cities(n);  
 for (int i = 0; i < n; i++) {  
 cities[i] = i + 1;  
 }  
  
 vector<int> bestPath;  
 int bestDistance = INF;  
  
 // Генерация всех перестановок городов  
 do {  
 int distance = 0;  
 bool validPath = true;  
  
 // Вычисление длины пути  
 for (int i = 0; i < n - 1; i++) {  
 int city1 = cities[i] - 1;  
 int city2 = cities[i + 1] - 1;  
 if (distances[city1][city2] == INF) {  
 validPath = false;  
 break;  
 }  
 distance += distances[city1][city2];  
 }  
  
 // Замыкаем путь к начальному городу  
 if (validPath && distances[cities[n - 1] - 1][cities[0] - 1] != INF) {  
 distance += distances[cities[n - 1] - 1][cities[0] - 1];  
 if (distance < bestDistance) {  
 bestDistance = distance;  
 bestPath = cities;  
 }  
 }  
 } while (next\_permutation(cities.begin() + 1, cities.end()));  
  
 // Замыкание пути к начальному городу  
 bestPath.push\_back(bestPath[0]);  
 totalDistance = bestDistance;  
 return bestPath;  
}  
  
int main() {  
 vector<vector<int>> distances = **{** {INF, 2, 22, INF, 1},  
 {1, INF, 16, 67, 83},  
 {3, 3, INF, 86, 50},  
 {18, 57, 4, INF, 3},  
 {92, 67, 52, 14, INF}  
 **}**;  
  
 int n = distances.size();  
  
 // Вывод номеров городов сверху  
 cout << " ";  
 for (int i = 1; i <= n; i++) {  
 cout << setw(7) << left << i;  
 }  
 cout << endl;  
  
 // Вывод матрицы расстояний и номеров городов слева  
 for (int i = 1; i <= n; i++) {  
 cout << setw(4) << left << i;  
 for (int j = 1; j <= n; j++) {  
 if (distances[i - 1][j - 1] == INF) {  
 cout << setw(7) << left << "INF";  
 } else {  
 cout << setw(7) << left << distances[i - 1][j - 1];  
 }  
 }  
 cout << endl;  
 }  
 cout << endl;  
  
 int optimalDistance;  
 vector<int> optimalPath = solveTravelingSalesmanProblem(distances, optimalDistance);  
  
 cout << "Optimal path: ";  
 for (int city : optimalPath) {  
 cout << city << " ";  
 }  
 cout << endl;  
 cout << "Best path: " << optimalDistance << endl;  
  
 return 0;  
}

