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

Задача 1

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

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

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

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

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

H=23

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 1 | 21 |  | 0 (1) |
| **2** | 0 (15) |  | 15 | 66 | 82 |
| **3** | 0 (0) | 0(1) |  | 83 | 47 |
| **4** | 15 | 54 | 1 |  | 0(1) |
| **5** | 78 | 53 | 38 | 0(104) |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **5** |
| **1** |  | 1 | 21 | 0 (1) |
| **2** | 0 (15) |  | 15 | 82 |
| **3** | 0 (0) | 0(1) |  | 47 |
| **4** | 15 | 54 | 1 | 0(1) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **5** | **di** |
| **1** |  | 1 | 21 | 0 (1) | 0 |
| **2** | 0 (15) |  | 15 | 82 | 0 |
| **3** | 0 (0) | 0(1) |  | 47 | 0 |
| **4** | 15 | 54 | 1 | 0(1) | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **5** |
| **1** |  | 1 | 21 | 0 (1) |
| **2** | 0 (15) |  | 15 | 82 |
| **3** | 0 (0) | 0(1) |  | 47 |
| **4** | 15 | 54 | 0(30) | 0(1) |
| **dj** | 0 | 0 | 0 | 0 |

H2=23

H2\*=127

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **5** |
| **1** |  | 1 | 21 | 0 (48) |
| **2** | 0 (15) |  | 15 | 82 |
| **3** | 0 (0) | 0(1) |  | 47 |
| **4** | 15 | 54 | 0(0) | 0(0) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** |
| **2** | 0 (15) |  | 15 |
| **3** | 0 (0) | 0(1) |  |
| **4** | 15 | 54 | 0(0) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **di** |
| **2** | 0 (15) |  | 15 | 0 |
| **3** | 0 (0) | 0(1) |  | 0 |
| **4** | 15 | 54 | 0(0) | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** |
| **2** | 0 (15) |  | 15 |
| **3** | 0 (0) | 0(1) |  |
| **4** | 15 | 54 | 0(0) |
| **dj** | 0 | 0 | 0 |

H3=23

H3\*=72

|  |  |  |  |
| --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** |
| **2** | 0 (15) |  | 15 |
| **3** | 0 (0) | 0(54) |  |
| **4** | 15 | 54 | 0(30) |

|  |  |  |
| --- | --- | --- |
| **Город** | **1** | **3** |
| **2** | 0 (15) | INF |
| **4** | 15 | 0(30) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Город** | **1** | **3** | **di** |
| **2** | 0 (15) | INF | 0 |
| **4** | 15 | 0(30) | 0 |
| **dj** | 0 | 0 |  |

H4=23

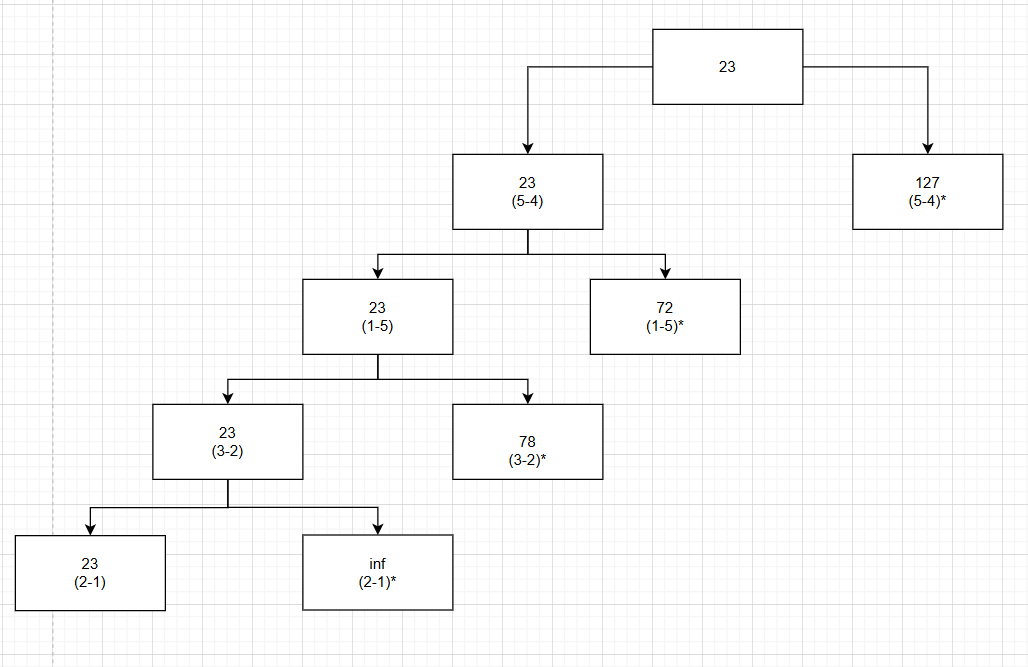
H4\*=78

|  |  |  |
| --- | --- | --- |
| **Город** | **1** | **3** |
| **2** | 0 (inf) | INF |
| **4** | 15 | 0(inf) |

|  |  |
| --- | --- |
| **Город** | **3** |
| **4** | 0(inf) |

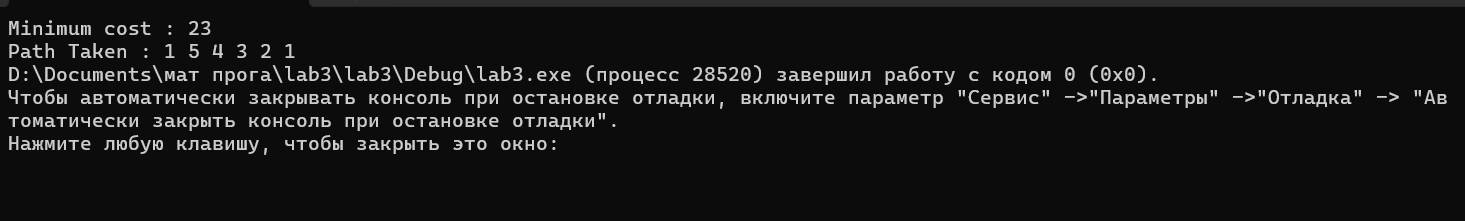
H5=24

H5\*=inf



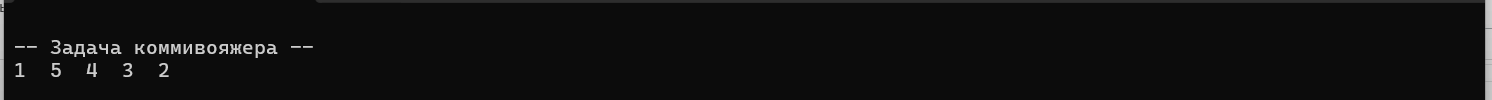
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Город** | **1** | **2** | **3** | **4** | **5** |
| **1** |  | 2 | 22 |  | 1 |
| **2** | 1 |  | 16 | 67 | 83 |
| **3** | 3 | 3 |  | 86 | 50 |
| **4** | 18 | 57 | 4 |  | 3 |
| **5** | 92 | 67 | 52 | 14 |  |

H=1+1+3+4+14=23



|  |
| --- |
| #include<iostream>  using namespace std;  const int N = 5;  int final\_path[N + 1];  bool visited[N];  int final\_res = INT\_MAX;  void copyToFinal(int curr\_path[])  {  for (int i = 0; i < N; i++)  final\_path[i] = curr\_path[i];  final\_path[N] = curr\_path[0];  }  int firstMin(int adj[N][N], int i)  {  int min = INT\_MAX;  for (int k = 0; k < N; k++)  if (adj[i][k] < min && i != k)  min = adj[i][k];  return min;  }  int secondMin(int adj[N][N], int i)  {  int first = INT\_MAX, second = INT\_MAX;  for (int j = 0; j < N; j++)  {  if (i == j)  continue;  if (adj[i][j] <= first)  {  second = first;  first = adj[i][j];  }  else if (adj[i][j] <= second &&  adj[i][j] != first)  second = adj[i][j];  }  return second;  }  void TSPRec(int adj[N][N], int curr\_bound, int curr\_weight,  int level, int curr\_path[])  {    if (level == N)  {    if (adj[curr\_path[level - 1]][curr\_path[0]] != 0)  {    int curr\_res = curr\_weight +  adj[curr\_path[level - 1]][curr\_path[0]];  if (curr\_res < final\_res)  {  copyToFinal(curr\_path);  final\_res = curr\_res;  }  }  return;  }  for (int i = 0; i < N; i++)  {    if (adj[curr\_path[level - 1]][i] != 0 &&  visited[i] == false)  {  int temp = curr\_bound;  curr\_weight += adj[curr\_path[level - 1]][i];  if (level == 1)  curr\_bound -= ((firstMin(adj, curr\_path[level - 1]) +  firstMin(adj, i)) / 2);  else  curr\_bound -= ((secondMin(adj, curr\_path[level - 1]) +  firstMin(adj, i)) / 2);  if (curr\_bound + curr\_weight < final\_res)  {  curr\_path[level] = i;  visited[i] = true;    TSPRec(adj, curr\_bound, curr\_weight, level + 1,  curr\_path);  }  curr\_weight -= adj[curr\_path[level - 1]][i];  curr\_bound = temp;    memset(visited, false, sizeof(visited));  for (int j = 0; j <= level - 1; j++)  visited[curr\_path[j]] = true;  }  }  }  void TSP(int adj[N][N])  {  int curr\_path[N + 1];    int curr\_bound = 0;  memset(curr\_path, -1, sizeof(curr\_path));  memset(visited, 0, sizeof(curr\_path));  for (int i = 0; i < N; i++)  curr\_bound += (firstMin(adj, i) +  secondMin(adj, i));    curr\_bound = (curr\_bound & 1) ? curr\_bound / 2 + 1 :  curr\_bound / 2;  visited[0] = true;  curr\_path[0] = 0;  TSPRec(adj, curr\_bound, 0, 1, curr\_path);  }  int main()  {    int adj[N][N] = { {0, 2, 22, 0, 1},  {1, 0, 16, 67,83},  {3, 3, 0, 86,50},  {18, 57, 4, 0,3},  {92,67,52,14,0}  };  TSP(adj);  printf("Minimum cost : %d\n", final\_res);  printf("Path Taken : ");  for (int i = 0; i <= N; i++)  printf("%d ", final\_path[i]+1);  return 0;  } |

Листинг кода решения задачи коммивояжера методом вервей и границ на языке C++



Решение задачи при помощи генератора перестановок