Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
*«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)*

ФАКУЛЬТЕТ **ИУК «Информатика и управление»**

КАФЕДРА **ИУК4 «Программное обеспечение ЭВМ, информационные технологии»**

ДОМАШНЯЯ РАБОТА №2

«Решение задач оптимизации при принятии решений»

ДИСЦИПЛИНА: «Типы и структуры данных»

Выполнил: студент гр. ИУК4-31Б _____ (подпись) (Суриков Н. С.) (Ф.И.О.)

Проверил: _____ (подпись) (Былинка М. И.) (Ф.И.О.)

Дата сдачи (защиты): 20.12.24

Результаты сдачи (защиты):

- Балльная оценка: 120
- Оценка:

Калуга, 2024

Цель: формирование практических навыков создания алгоритмов решения оптимизационных задач.

Задачи:

1. Изучить виды задач оптимизации при принятии решений.
2. Изучить основные алгоритмы для решения данных задач.
3. Реализовать алгоритм согласно варианту.

Вариант 3

Формулировка задания:

3. Реализовать алгоритм поиска максимального потока с множеством истоков и одним стоком на основе алгоритма поиска в ширину.

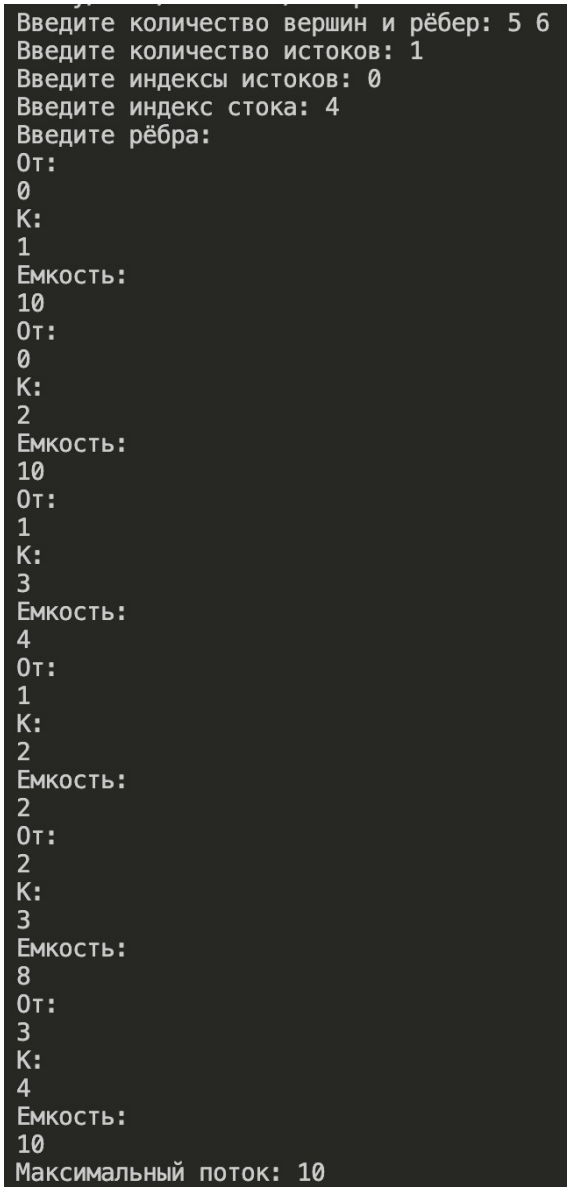
Листинг программы:

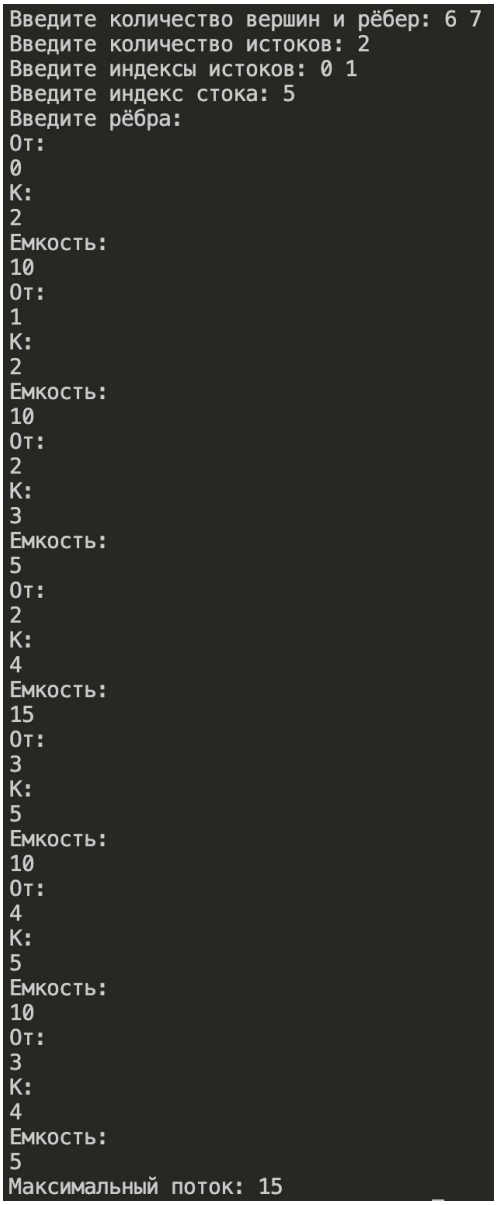
```cpp
#include <iostream>
#include <vector>
#include <queue>
#include <climits>
#include <cstring>

class MaxFlow {
private:
    int numVertices;
    std::vector<std::vector<int> > capacity;
    std::vector<std::vector<int> > adjList;

    // Поиск увеличивающего пути
    bool bfs(int source, int sink, std::vector<int>& parent) {
        std::fill(parent.begin(), parent.end(), -1);
        std::queue<int> q;
        q.push(source);
        parent[source] = source;

        while (!q.empty()) {
            int current = q.front();
            q.pop();

            for (size_t i = 0; i < adjList[current].size(); ++i) {
                int next = adjList[current][i];
                if (parent[next] == -1 && capacity[current][next] > 0) {
                    parent[next] = current;
                    if (next == sink) return true;
                    q.push(next);
```

```
                }
            }
        }

        return false;
    }

public:
    MaxFlow(int vertices) : numVertices(vertices) {
        capacity.assign(vertices, std::vector<int>(vertices, 0));
        adjList.resize(vertices);
    }

    void addEdge(int from, int to, int cap) {
        capacity[from][to] += cap;
        adjList[from].push_back(to);
        adjList[to].push_back(from); // Для остаточной сети
    }

    // Алгоритм Эдмондса-Карпа для поиска максимального потока
    int findMaxFlow(int source, int sink) {
        int maxFlow = 0;
        std::vector<int> parent(numVertices);

        while (bfs(source, sink, parent)) {
            // Найти минимальную пропускную способность на пути
            int flow = INT_MAX;
            for (int v = sink; v != source; v = parent[v]) {
                int u = parent[v];
                flow = std::min(flow, capacity[u][v]);
            }

            // Обновить остаточную сеть
            for (int v = sink; v != source; v = parent[v]) {
                int u = parent[v];
                capacity[u][v] -= flow;
                capacity[v][u] += flow;
            }

            maxFlow += flow;
        }

        return maxFlow;
    }
};

int main() {
    int numVertices, numEdges, numSources, sink;

    std::cout << "Введите количество вершин: ";
    std::cin >> numVertices;
    std::cout << "Введите количество ребер: ";
    std::cin >> numEdges;

    MaxFlow graph(numVertices + 1); // +1 для суперистока
```

```
 85
 86      std::cout << "Введите количество истоков: ";
 87      std::cin >> numSources;
 88
 89      std::vector<int> sources(numSources);
 90      std::cout << "Введите индексы истоков: ";
 91      for (int i = 0; i < numSources; ++i) {
 92          std::cin >> sources[i];
 93                graph.addEdge(numVertices,  sources[i],  INT_MAX);  // Соединяем
суперисток с истоками
 94      }
 95
 96      std::cout << "Введите индекс стока: ";
 97      std::cin >> sink;
 98
 99      std::cout << "Введите рёбра: \n";
100      for (int i = 0; i < numEdges; ++i) {
101          int from, to, cap;
102          std::cout << "От: \n";
103          std::cin >> from;
104          std::cout << "К: \n";
105          std::cin >> to;
106          std::cout << "Емкость: \n";
107          std::cin >> cap;
108          graph.addEdge(from, to, cap);
109      }
110
111      int maxFlow = graph.findMaxFlow(numVertices, sink);
112      std::cout << "Максимальный поток: " << maxFlow << std::endl;
113
114      return 0;
115  }
```

Результат работы:





Вывод: в ходе работы были сформированы практические навыки создания алгоритмов решения оптимизационных задач.