|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 | |  |  | ( | Суриков Н. С. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Былинка М. И. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

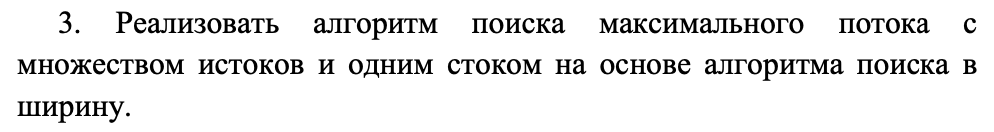
**Цель:** формирование практических навыков создания алгоритмов решения оптимизационных задач.

**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:**

**

**Листинг программы:**

1 *#include* *<iostream>*

2 *#include* *<vector>*

3 *#include* *<queue>*

4 *#include* *<climits>*

5 *#include* *<cstring>*

6

7 class MaxFlow {

8 private:

9 int numVertices;

10 std::vector<std::vector<int> > capacity;

11 std::vector<std::vector<int> > adjList;

12

13 *// Поиск увеличивающего пути*

14 bool bfs(int source, int sink, std::vector<int>& parent) {

15 std::fill(parent.begin(), parent.end(), -1);

16 std::queue<int> q;

17 q.push(source);

18 parent[source] = source;

19

20 while (!q.empty()) {

21 int current = q.front();

22 q.pop();

23

24 for (size\_t i = 0; i < adjList[current].size(); ++i) {

25 int next = adjList[current][i];

26 if (parent[next] == -1 && capacity[current][next] > 0) {

27 parent[next] = current;

28 if (next == sink) return true;

29 q.push(next);

30 }

31 }

32 }

33

34 return false;

35 }

36

37 public:

38 MaxFlow(int vertices) : numVertices(vertices) {

39 capacity.assign(vertices, std::vector<int>(vertices, 0));

40 adjList.resize(vertices);

41 }

42

43 void addEdge(int from, int to, int cap) {

44 capacity[from][to] += cap;

45 adjList[from].push\_back(to);

46 adjList[to].push\_back(from); *// Для остаточной сети*

47 }

48

49 *// Алгоритм Эдмондса-Карпа для поиска максимального потока*

50 int findMaxFlow(int source, int sink) {

51 int maxFlow = 0;

52 std::vector<int> parent(numVertices);

53

54 while (bfs(source, sink, parent)) {

55 *// Найти минимальную пропускную способность на пути*

56 int flow = INT\_MAX;

57 for (int v = sink; v != source; v = parent[v]) {

58 int u = parent[v];

59 flow = std::min(flow, capacity[u][v]);

60 }

61

62 *// Обновить остаточную сеть*

63 for (int v = sink; v != source; v = parent[v]) {

64 int u = parent[v];

65 capacity[u][v] -= flow;

66 capacity[v][u] += flow;

67 }

68

69 maxFlow += flow;

70 }

71

72 return maxFlow;

73 }

74 };

75

76 int main() {

77 int numVertices, numEdges, numSources, sink;

78

79 std::cout << "Введите количество вершин: ";

80 std::cin >> numVertices;

81 std::cout << "Введите количество ребер: ";

82 std::cin >> numEdges;

83

84 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);

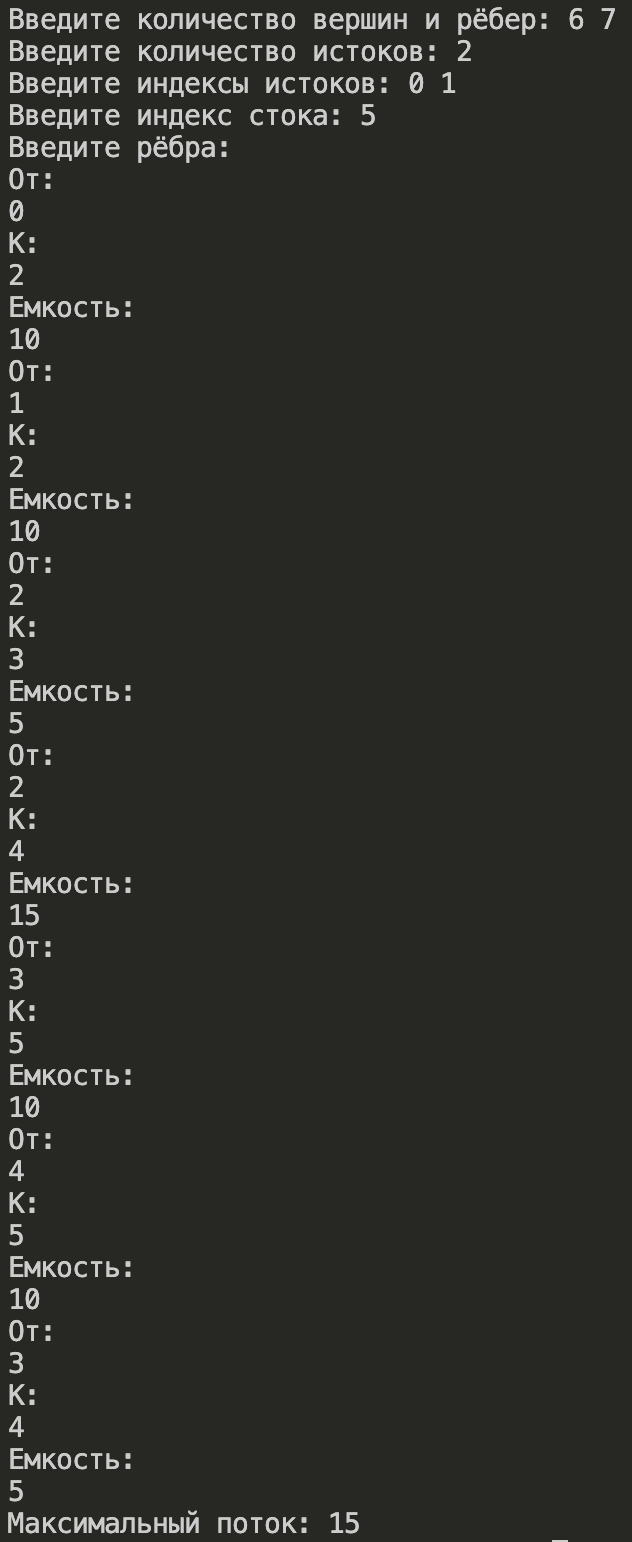
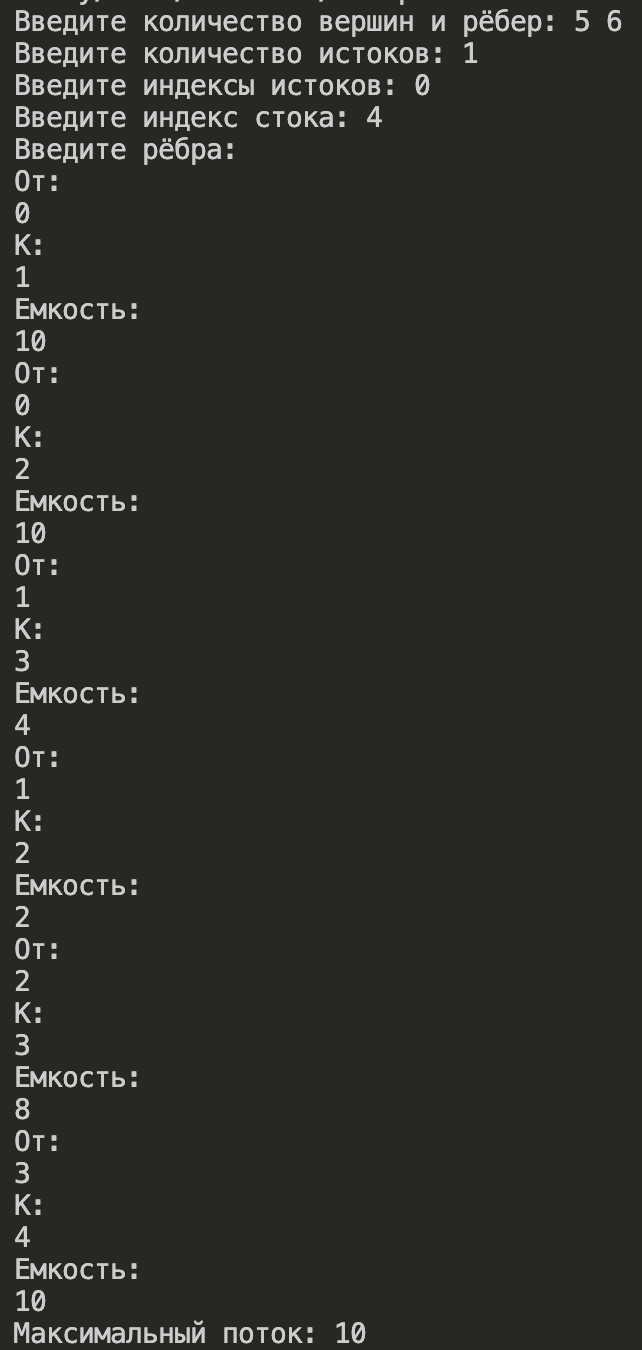
112 std::cout << "Максимальный поток: " << maxFlow << std::endl;

113

114 return 0;

115 }

**Результат работы:**

****

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