**Вариант 7**

Вам дается 8-литровое ведро воды и два пустых ведра, которые могут содержать 5 и 3 литров соответственно. Вы должны разделить воду на две части, разлив воду между ведрами (то есть, чтобы в итоге получилось 4 литра в 8-литровом ведре и 4 литра в 5-литровом ведре).

Какое минимальное количество переливаний воды между ведрами?

**Решение задачи:**

Для решения данной задачи воспользуемся алгоритмом DFS.

#include <iostream>

#include <queue>

#include <set>

#include <utility>

using namespace std;

// Структура для представления состояния ведер

struct BucketsState {

int bucket1;

int bucket2;

BucketsState(int b1, int b2) : bucket1(b1), bucket2(b2) {}

bool operator==(const BucketsState& other) const {

return bucket1 == other.bucket1 && bucket2 == other.bucket2;

}

// Функция для вычисления хэша состояния (для использования в set)

size\_t hash() const {

return bucket1 \* 10 + bucket2; // Простая хэш-функция

}

};

// Функция для переливания воды из одного ведра в другое

BucketsState pourWater(const BucketsState& state, int from, int to, int capacity\_to) {

int amount = min(state.bucket1, capacity\_to - state.bucket2);

BucketsState newState = state;

newState.bucket1 -= (from == 1 ? amount : 0);

newState.bucket2 += (to == 2 ? amount : 0);

return newState;

}

// Функция для поиска минимального количества переливаний

int minPouring(int target) {

queue<pair<BucketsState, int>> q; // Очередь состояний и их глубины

set<size\_t> visited; // Множество для отслеживания посещенных состояний

BucketsState startState(8, 0); // Начальное состояние

q.push({ startState, 0 });

visited.insert(startState.hash());

while (!q.empty()) {

BucketsState currentState = q.front().first;

int depth = q.front().second;

q.pop();

if (currentState.bucket1 == target && currentState.bucket2 == target)

return depth; // Найдено оптимальное решение

// Переливаем воду между ведрами

for (int from = 1; from <= 2; ++from) {

for (int to = 1; to <= 2; ++to) {

if (from != to) {

BucketsState nextState = pourWater(currentState, from, to, to == 2 ? 5 : 3);

if (visited.find(nextState.hash()) == visited.end()) {

q.push({ nextState, depth + 1 });

visited.insert(nextState.hash());

}

}

}

}

}

return -1; // Решение не найдено

}

int main() {

setlocale(LC\_ALL, "Russian");

int target = 4; // Желаемое количество воды в каждом ведре

int minMoves = minPouring(target);

if (minMoves != -1)

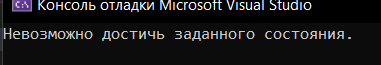
cout << "Минимальное количество переливаний: " << minMoves << endl;

else

cout << "Невозможно достичь заданного состояния." << endl;

return 0;

}



Невозможно разлить 8 литров воды по этим ведрам.