**Вариант 7**

Решите следующую задачу о рюкзаке, записав результаты рекурсии динамического программирования в таблицу. Напишите алгоритм обратного хода.

max 3x1 + 8x2 + 13x3

s.t. 2x1 + 3x2 + 5x3 ≤ 9

x1, x2, x3 ≥ 0 — целые.

**Решение задачи:**

Для решения данной задачи мы будем использовать подход с заполнением таблицы, где каждая ячейка представляет собой максимальную стоимость, которую можно достичь с учетом ограничения по весу рюкзака.

**Решение С++:**

#include <iostream>

#include <vector>

void knapsack(int capacity, const std::vector<int>& weights, const std::vector<int>& values) {

int n = weights.size();

// Создаем таблицу для записи результатов

std::vector<std::vector<int>> dp(n + 1, std::vector<int>(capacity + 1, 0));

// Заполняем таблицу

for (int i = 1; i <= n; ++i) {

for (int w = 1; w <= capacity; ++w) {

if (weights[i - 1] <= w) {

// Если текущий предмет может быть помещен в рюкзак

// выбираем максимум между добавлением предмета и его невключением

dp[i][w] = std::max(values[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);

}

else {

// Если текущий предмет не может быть помещен в рюкзак

// просто присваиваем значение предыдущей ячейки

dp[i][w] = dp[i - 1][w];

}

}

}

// Выводим таблицу

std::cout << "Таблица результатов:" << std::endl;

for (int i = 0; i <= n; ++i) {

for (int w = 0; w <= capacity; ++w) {

std::cout << dp[i][w] << " ";

}

std::cout << std::endl;

}

// Выводим результаты

std::cout << "Максимальная стоимость: " << dp[n][capacity] << std::endl;

// Алгоритм обратного хода

int remainingCapacity = capacity;

std::vector<int> selectedItems;

for (int i = n; i > 0; --i) {

if (dp[i][remainingCapacity] != dp[i - 1][remainingCapacity]) {

// Если значение в текущей ячейке отличается от значения в предыдущей ячейке

// значит предмет был выбран, добавляем его в выбранные предметы

selectedItems.push\_back(i - 1);

remainingCapacity -= weights[i - 1];

}

}

// Выводим выбранные предметы

std::cout << "Выбранные предметы: ";

for (int item : selectedItems) {

std::cout << item << " ";

}

std::cout << std::endl;

}

int main() {

// Пример использования

setlocale(LC\_ALL, "Russian");

std::vector<int> weights = { 2, 3, 5 };

std::vector<int> values = { 3, 8, 13 };

int capacity = 9;

knapsack(capacity, weights, values);

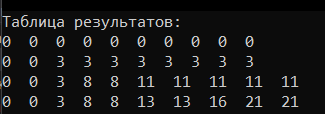
return 0;

}

Данные вводятся вручную.

Вывод осуществляется в консоль.

Код содержит комментарии для большего понимания алгоритма.

****

****