Динамическое программирование. Часть 1

Динамическое программирование это способ решения сложных задач путём разбиения их на более простые подзадачи.

Ключевая идея в динамическом программировании достаточно проста. Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Часто многие из этих подзадач одинаковы.

Одномерное динамическое программирование.

Обычно у подзадачи есть некоторое количество параметров, которые её определяют. Ответом на подзадачу часто является одно число. Поэтому промежуточные результаты вычисления хранятся в векторе dp - он может быть одномерным, например vector<int> dp, двумерным - vector<vector<int> > dp и так далее. Количество измерений в массиве совпадает с количеством параметров, определяющих подзадачу. Одномерное динамическое программирование, например, включает в себя все те задачи, где размерность массива dp равна 1.

Разберём структуру решения задач динамического программирования. Наша задача состоит в нахождении dp[n] = ?. Мы знаем, что dp[i] связано с меньшими dp некоторые образом, например dp[i] = 3*dp[i-2] - 5*dp[i-1]. При этом нам обычно известно несколько первых dp - например dp[0] = 1 и dp[1] = 2. Тогда мы будем последовательно заполнять массив dp, начиная от меньших i к большим и, таким образом, в итоге мы найдём dp[n]. Важно понимать, что при решении многих задач ни подзадачи, ни формула перехода не даны в явном виде. Ваша задача - выделить то, что является подзадачей и найти формулу перехода, связывающую большие подзадачи с меньшими. Далее разберём типичную задачу одномерного динамического программирования.

Задача 1. Требуется посчитать число последовательностей, состоящих из нулей и единиц длины n, в которых не встречаются две идущие подряд единицы.

Для решения этой задачи методом динамического программирования сведем исходную задачу к подзадачам. То есть подзадачей здесь является нахождение правильной последовательности меньшей длины i, где 1 <= i <= n.

При i = 1, i = 2 ответ очевиден. То есть dp[1] = 2, dp[2] = 3 (подходят последовательности 01, 10, 00). Допустим, что мы уже нашли dp[i – 1] и dp[i – 2]. Посмотрим, какой может быть последовательность длины i. Если последний ее символ равен 0, то первые i – 1 символов могут быть любой правильной последовательностью длины i – 1 (не важно, заканчивается она нулем или единицей, так как последним символом является 0). Таких последовательностей длины i, заканчивающихся на 0 всего dp[i – 1]. Если же последний символ равен 1, то предпоследний символ обязательно должен

быть равен 0 (иначе будет две единицы подряд), а первые i – 2 символа могут быть любыми. Число таких последовательностей равно dp[i – 2].

Таким образом, при i >= 3, dp[i] = dp[i-1] + dp[i-2]. То есть данная задача фактически сводится к нахождению чисел Фибоначчи. Ниже приведёна программа, представляющая собой решение данной задачи.

```
#include<iostream>
using namespace std;

int main()
{
    int n;
    cin >> n;
    vector<int> dp(n+1);
    dp[1] = 2;
    dp[2] = 3;

    for(int i = 3; i <= n i++)
        dp[i] = dp[i-1] + dp[i-2];

    cout << dp[n] << endl;
        return 0;
}
```