Модуль 3 Динамическое программирование
Лекция 3.5 Задача о рюкзаке

Эта лекция посвящена ещё одной задаче, с которой уже встречались — задаче о рюкзаке.

Задача о рюкзаке
Имеется n предметов, для которых известны веса w_i и стоимости c_i, $i = 1, \ldots, n$. Рюкзак вмещает вес не более W. Нужно набрать в него предметы максимальной суммарной стоимости.

Мы уже обсуждали эту задачу в модуле про жадные алгоритмы и научились решать непрерывную задачу о рюкзаке, в которой предметы можно дробить на части. Тем не менее, для дискретной задачи, в которой дробить предметы нельзя, жадное решение не всегда является оптимальным, и мы пока не знаем, как её решать.

На этой лекции мы решим дискретную задачу о рюкзаке методом динамического программирования. Для этого нам нужно перейти от одной задачи, в которой мы имеем n предметов и рюкзак вместимости W, к серии подзадач, таких, чтобы получать решение большой задачи по задачам меньшего размера.

Предположим, что мы можем использовать из n предметов только первые i (с индексами от 1 до i) и имеем рюкзак вместимости j, $0 \leqslant i \leqslant n$, $0 \leqslant j \leqslant W$. Для каждой пары (i, j) будем считать величину d_{ij} — максимальную суммарную стоимость, которую можно набрать, если разрешается использовать только первые i предметов и рюкзак имеет вместимость j. Состоянием динамики будет пара чисел (i, j).

Определим переходы между состояниями. Пусть мы имеем оптимальное решение для состояния (i, j). Посмотрим, берём ли мы в этом решении в рюкзак последний предмет (с индексом i). Если берём — значит, мы пришли в состояние (i, j) из состояния $(i - 1, j - w_i)$. Это состояние, в котором мы можем использовать только первые $(i - 1)$ предметов и вместимость рюкзака меньше j на вес i-го предмета w_i. Либо мы можем прийти в состояние (i, j) из состояния $(i - 1, j)$, если мы не берём в рюкзак i-й предмет и фактически используем только первые $(i - 1)$ предметов. Получаем схему на рис. 1 и следующую формулу для переходов:

$$d_{ij} = \max(d_{i-1,j-w_i} + c_i, d_{i-1,j}). \tag{1}$$

На основе формулы (1) можно написать программу (см. рис. 2).

В программе последовательно перебираются состояния (i, j), и для каждого из них вычисляется ответ по формуле (1). Делается проверка, что величина j-w[i] неотрицательна, то есть вес последнего предмета помещается в рюкзак,

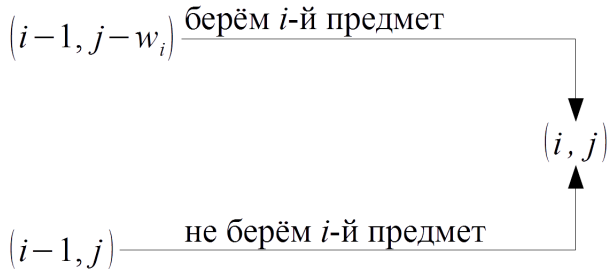

Рис. 1. Переходы динамики

```
for (int i = 1; i <= n; i++)
    for (int j = 0; j <= W; j++)
    {
        d[i][j] = d[i-1][j];
        if (j - w[i] >= 0)
            d[i][j] = max(d[i][j], d[i-1][j-w[i]] + c[i]);
    }
```

Рис. 2. Решение задачи о рюкзаке

потому что только в этом случае мы можем положить этот предмет. Иначе у нас только один вариант, когда мы этот предмет не кладём.

Предварительно необходимо инициализировать все значения d при $i = 0$. Мы не можем использовать ни один из предметов, значит, максимальная стоимость будет равна нулю для всех j.

Упражнение 3.5.1
Оцените время работы описанного решения.

На примере задачи о рюкзаке мы разберём подход, который называется *динамикой «вперёд»*. При построении переходов динамики (см. рис. 1) мы определяли, из каких состояний можно прийти в состояние (i, j). Можно поступить наоборот. Пусть мы стоим с состоянии (i, j) и уже посчитали d_{ij}. Определим, в какие состояния мы можем перейти. Мы можем положить в рюкзак $(i + 1)$-й предмет (при этом добавить w_{i+1} к вместимости рюкзака) и перейти в состояние $(i + 1, j + w_{i+1})$. Или можем не брать $(i + 1)$-й предмет и перейти в состояние $(i + 1, j)$ (см. рис. 3).

Основываясь на этой идее, можно написать код на рис. 4.

Мы также перебираем состояния в циклах по i и по j.

for (**int** i = 0; i < n; i++)
    for (**int** j = 0; j <= W; j++)

Проверяем, можно ли к j можно добавить вес $(i + 1)$-го предмета.

if (j + w[i + 1] <= W)

Берём состояние, в которое мы перейдём и обновляем в нем ответ.

d[i + 1][j + w[i + 1]] = max(d[i + 1][j + w[i + 1]], d[i][j] + c[i + 1]);

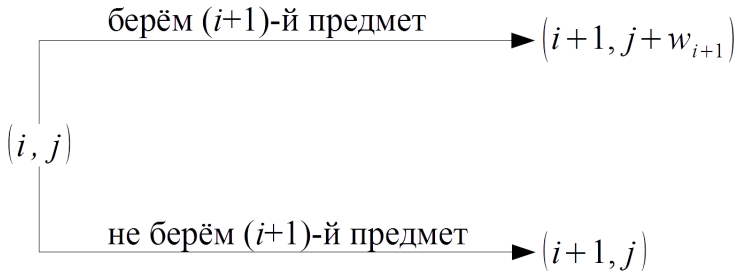

Рис. 3. Переходы вперёд

```
for (int i = 0; i < n; i++)
    for (int j = 0; j <= W; j++)
    {
        if (j + w[i + 1] <= W)
            d[i + 1][j + w[i + 1]] = max(d[i + 1][j + w[i + 1]],
                                         d[i][j] + c[i + 1]);
        d[i + 1][j] = max(d[i + 1][j], d[i][j]);
    }
```

Рис. 4. Решение задачи о рюкзаке динамикой «вперёд»

Изначально массив d заполнен нулями. Мы присваиваем значение в нужной ячейке максимуму из него самого и значения, которое получится при переходе из состояния (i, j). Это случай, когда мы берём $(i + 1)$-й предмет, поэтому добавляется его стоимость.

Затем мы аналогично переходим в состояние $(i+1, j)$ и обновляем значение для него.

d[i + 1][j] = max(d[i + 1][j], d[i][j]);

Код на рис. 4 заполнит массив d теми же значениями, что и предыдущая программа на рис. 2. Оценка времени работы будет такая же. Только в нашем новом решении происходят переходы вперёд. Нельзя сказать, что этот метод принципиально лучше или хуже, но иногда при решении олимпиадной задачи проще догадаться до динамики «вперёд», чем «назад», поэтому лучше знать оба подхода.

Таким образом, мы научились решать динамикой дискретную задачу о рюкзаке, которая не решается жадностью. На примере этой задачи мы разобрали реализацию динамики «вперёд».