Этот модуль посвящён жадным алгоритмам. Мы разберём, что это такое, на примерах задач. Начнём с задачи о размене.

Задача о размене
Нужно набрать 39 рублей минимальным количеством монет 1, 2, 5 и 10 рублей. Можно использовать несколько монет каждого номинала.

Эту задачу можно решить перебором, но проще поступить так, как обычно поступают на кассе в магазине.

Сначала наберём наибольшую сумму, не превосходящую 39, монетами по 10 рублей. Потом, если осталось больше 5 рублей, возьмём 5 рублей. Нам останется добрать 4 рубля. Мы их наберём монетами по 2 рубля, и монеты по одному рублю нам не понадобятся (см. рис. 1). Алгоритм состоит в том, чтобы брать как можно больше монет большего номинала.

Рис. 1. Ответ к задаче о размене

Разберём реализацию этого алгоритма (см. рис. 2).

```
int change(int n, vector<int> a, int s)
{
    int cnt = 0;
    for (int i = 0; i < n; i++)
    {
        cnt += s / a[i];
        s %= a[i];
    }
    return cnt;
}
```

Рис. 2. Жадное решение задачи о размене

Пусть n — число доступных номиналов монет, в векторе a содержатся сами номиналы. Будем считать, что они заданы в порядке убывания. И s — сумма, которую мы хотим набрать. Функция возвращает минимальное число монет, которое для этого понадобится.

int change(**int** n, vector<**int**> a, **int** s)

Создадим переменную cnt, чтобы в ней считать результат. Сначала ей присваивается ноль.

int cnt = 0;

Затем мы перебираем в цикле монеты.

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

Чтобы определить, сколько раз мы можем взять монету номинала a[i], чтобы получить сумму, не превосходящую s, используем деление с остатком: s / a[i]. Результат будет равен количеству монет, которое мы возьмём, и мы его прибавим к ответу cnt.

cnt += s / a[i];

После этого мы заменяем значение s на его остаток от деления на a[i], то есть уменьшаем сумму на то число, которое уже взяли.

s %= a[i];

Например, если сумма равна 39, и мы сначала рассматриваем монеты номиналом 10 рублей, то $39/10 = 3$. Значит, мы возьмём 3 монеты по 10 рублей, и остаток будет 9 рублей, которые нам нужно добрать.

Это жадный алгоритм. Он на каждом шаге пытается действовать оптимальным образом: взять как можно больше монет наибольшего достоинства. Но всегда ли такой алгоритм даст оптимальный ответ на задачу в целом?

Упражнение 2.1.1
Нужно набрать сумму s минимальным количеством монет номиналами из вектора a. В каких случаях описанный жадный алгоритм **не** даст правильного ответа на задачу?

1. $s = 26$, $a = [10, 5, 2, 1]$.
2. $s = 19$, $a = [5, 3, 2]$.
3. $s = 21$, $a = [4, 2, 1]$.
4. $s = 25$, $a = [10, 8, 6, 1]$.

Получается, что жадное решение задачи о размене — не всегда оптимальное. Оптимизация на каждом шаге не всегда позволяет достичь оптимального результата в целом.

В примере на рис. 3 нужно набрать сумму 25 монетами номиналов 10, 8, 6, 1. Жадный алгоритм возьмёт две монеты по 10 рублей, и дальше уже не сможет использовать монеты 8 и 6, придётся брать 5 монет по рублю. Всего потребуется 7 монет. При этом существует другое решение задачи: взять по одной монете каждого номинала, тогда как раз получится сумма 25. Будут использованы всего 4 монеты, что более оптимально.

Как правило, основная проблема с жадными алгоритмами не в их реализации, а в оптимальности и корректности их работы. Здесь сложно выделить

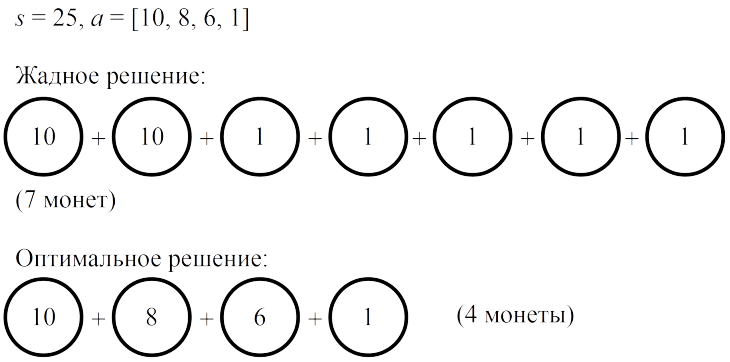

Рис. 3. Пример, на котором жадное решение — не оптимальное

какой-то общий принцип. Каждая олимпиадная задача очень индивидуальна. В дальнейшем мы разберём несколько классических задач, которые решаются жадными алгоритмами, с доказательством корректности этих алгоритмов.