# Лабораторная работа № 8 по курсу дискретного анализа: жадные алгоритмы

Выполнил студент группы М8о-207Б-18 МАИ *Токарев Никита*.

## Условие

азработать жадный алгоритм решения задачи, определяемой своим вариантом. Доказать его корректность, оценить скорость и объём затрачиваемой оперативной памяти. Реализовать программу на языке C или C++, соответсвующую построенному алгоритму. Формат входных и выходных данных описан в варианте задания.
Вариант: на первой строке заданы два числа, N и p $> 1$, определяющие набор монет некоторой страны с номиналами $p_0, p_1, ..., p_{N-1}$. Нужно определить наименьшее количество монет, которое можно использовать для того, чтобы разменять заданную на второй строчке сумму денег M $<= 2^{32}$ - 1 и распечатать для каждого i-го номинала на i-ой строчке количество участвующих в размене монет. Кроме того, нужно обосновать почему жадный выбор неприменим в общем случае (когда номиналы могут быть любыми) и предложить алгоритм, работающий при любых входных данных.

## Метод решения

Основным определение(правило) жадного влгоритма: на каждом шаге алгоритм делает оптимальный выбор. Таким образом для решения моей задачи можно применить жданый алгориитм: номиналы упорядочены по возрастанию. Если же номиналы могут быть любыми то первое, что мне приходит в голову, это отсортирвать список с номиналами по возрастанию и применить мой жадный алгоритм.

1. Создаю массив с номиналами и массив количества использованных монет для выбранного номинала.

2. По условию задачи номиналы подаются на вход в порядке возрастания, поэтому идя с монеты максимального номинала к минимальному я считаю максимальное количество воможных используемых монет и отнимаю от данной суммы денег. Таким образом данный шаг является оптимальным.

Таким образом сложность подсчета наименьшего количества монет равна O(n), где n - количесво возможных наминалов.

## Описание программы

Для решения поставленной задачи я реализовал структуру TInformation, которая содержит три поля, конструктор, деструктор и один метод вывода результата. В данном

классе полями являются массив из возможных номиналов, массив значений используемых монет, а также переменная, определяющая размер этих двух массивов. В конструкторе я динамически выделяю память для массивов и инициализирую их, а в деструкторе соотвественно происходит удаление динамических массивов. Алгоритм представлен ниже:

```
while(number) { // данная сумма монет
  tempcount = number / table.values[n - 1]; // подсчет максимального числа возможных используемых монет
  number = number - tempcount * table.values[n - 1]; // изменение суммы
  table.counts[n - 1] = tempcount;
  n–;
}
```

В моей программе 5 статических переменных типа long long и два динамических массива типа long long, размер которых равен количеству возможных номиналов.

## Тест производительности

Данный алгоритм работает со сложностью O(n), где n - максимальное количество номиналов. Для данного алгоритма мне довольно сложно создать тесты, которые могли бы продемонстрировать сложность. Также можно было бы сравнить время работы с наивным алгоритмом, однако реализация наивного алгоритма намного сложнее реализации данного.

## Выводы

Хотелось бы отметить, что и динамическое программирование, и жадные алгоритмы основываются на свойстве оптимальности для подзадач, поэтому возможны ситуации в которых применимы как динамическое программирование, так и динамическое программирование. Жадный алгоритм — алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.