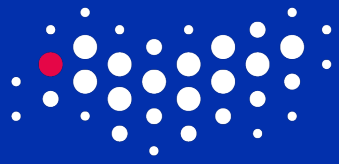


УНИВЕРСИТЕТ ИТМО

Лекция 4
Динамическое программирование. Метод «разделяй и властвуй».

«**Разделяй и властвуй**» в информатике — схема разработки алгоритмов, заключающаяся в рекурсивном разбиении решаемой задачи на две или более подзадачи того же типа, но меньшего размера, и комбинировании их решений для получения ответа к исходной задаче; разбиения выполняются до тех пор, пока все подзадачи не окажутся элементарными.

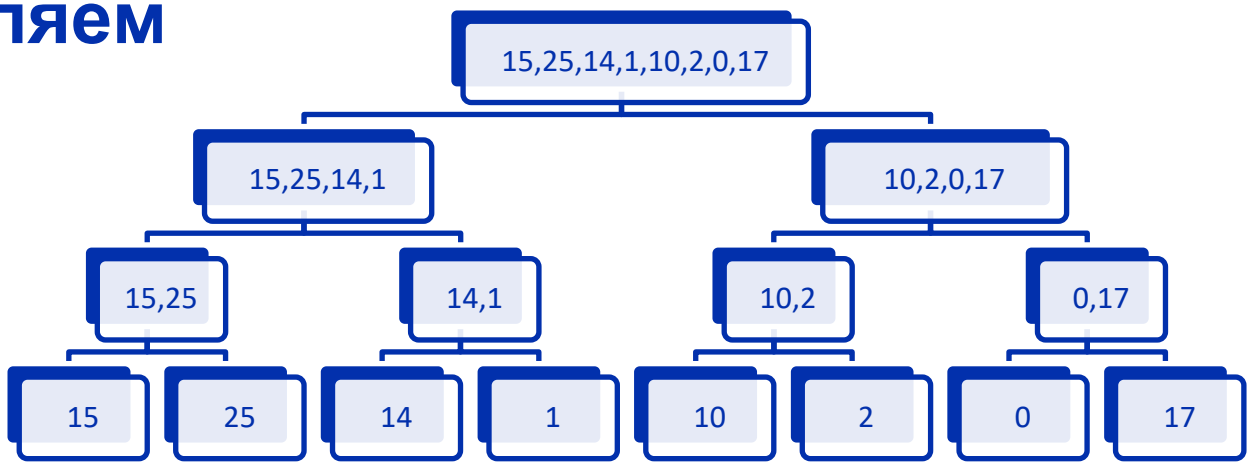
Этапы решения задачи

1. Разделяй! Задача разделяется на две или более задачи меньшей размерности, решаемых ***независимо*** одна от другой
2. Покоряй! Каждая из полученных подзадач решается рекурсивно
3. Соединяй! Из решений подзадач компонуется решение исходной задачи

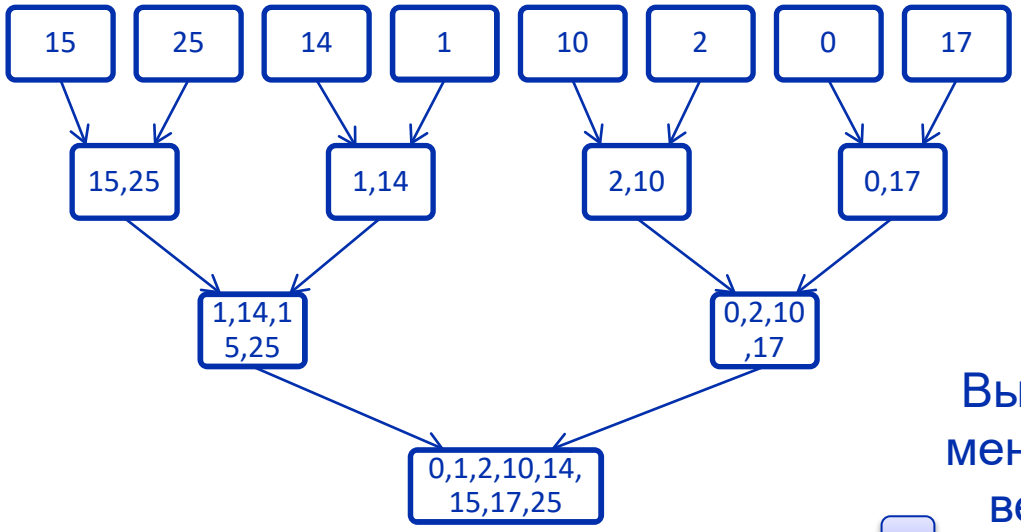
Пример задачи.
Отсортировать массив по возрастанию.
Решение.

1. Делим массив на две примерно равные части
2. Сортируем каждую из них по возрастанию
3. Сливаем два отсортированных массива в один

Разделяем



Покоряем и соединяем



Количество действий пропорционально сумме количеств элементов соединяемых массивов

Выбираем меньший из верхних

Динамическое программирование в теории управления и теории вычислительных систем — способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. В этом случае время вычислений, по сравнению с «наивными» методами, можно значительно сократить.

- Задача разбивается на вспомогательные задачи меньшей размерности
- Эти вспомогательные задачи не являются независимыми, т.е. **разные вспомогательные задачи используют решение одних и тех же подзадач**
- Решения вспомогательных задач запоминаются в таблице, чтобы не тратить время на их повторное решение
- Из решений перекрывающихся вспомогательных задач получается решение исходной задачи

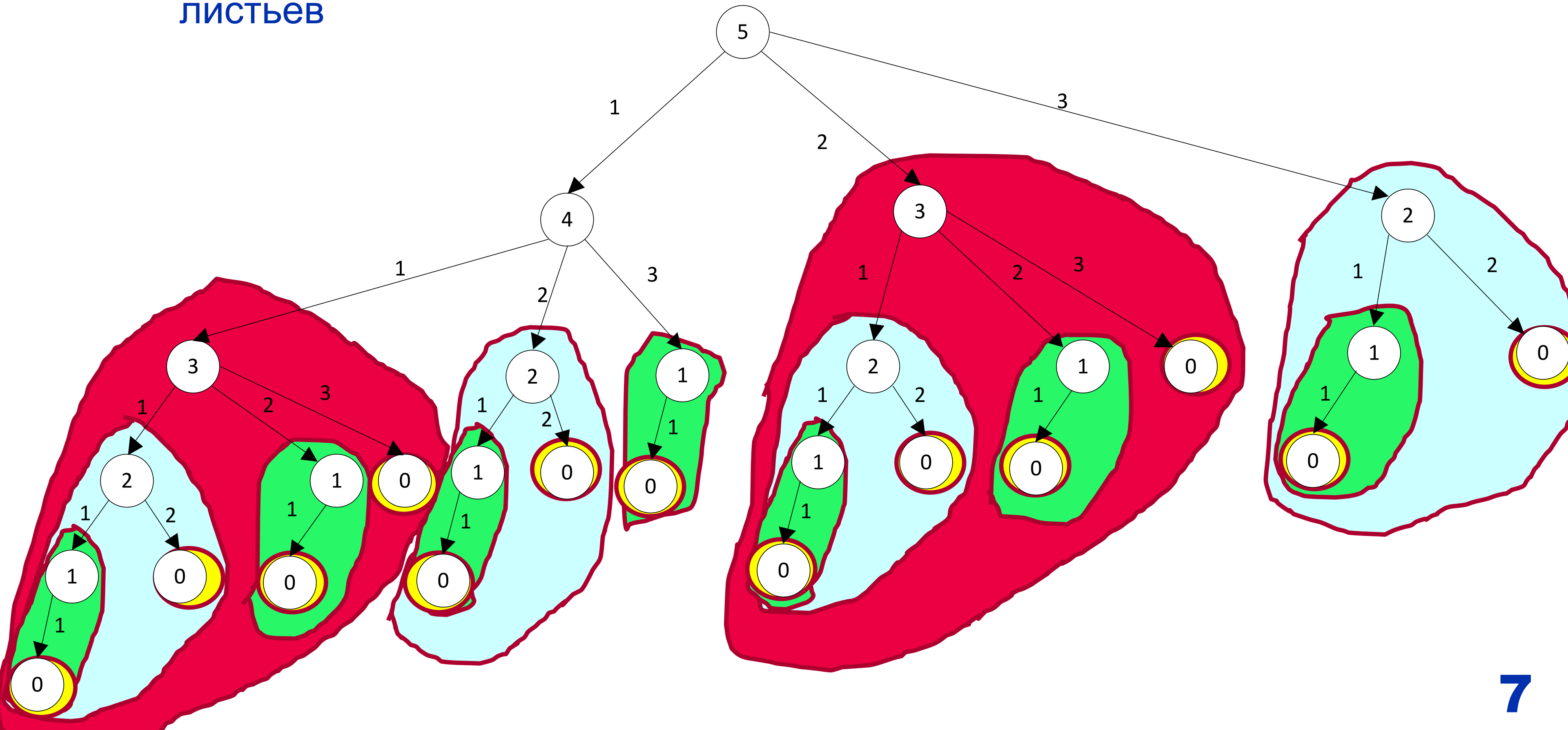
Пример задачи

Фишка может двигаться только вперед по полю длины N. Длина хода фишки не более K. Найти число различных вариантов ходов, при которых фишка может пройти поле от начала до конца. Например, при N=3, K=2 — возможные пути: (1,1,1), (1,2), (2,1), т.е. возможны 3 варианта.

Начальное положение фишки — первая клетка поля, конечное — первая клетка за пределами поля.

Анализируем

- Пусть длина поля N=5, максимальная длина хода K=3
- Представим в виде дерева размер поля, ещё не пройденного фишкой, включая поле, на котором она стоит.
- Лист дерева соответствует фишке, вышедшей за пределы поля.
- Ответ на вопрос задачи – количество путей от корня дерева до его листьев



Восходящее динамическое программирование

- Находим решение задачи в простейшем случае.

  Существует единственный путь из листа в него же самого: $c_0 = 1$
- Находим рекуррентное соотношение, связывающее решения различных подзадач.

$$c_p = \sum_{i=1}^{\min(p,K)} c_{p-i}$$

  Складываем ответы для подзадач, на которые распадается задача размерности ***p****. Учитываем, что размерность задачи не может быть отрицательной.*
- Все c_i храним в линейном массиве
- Для $p = 1, 2, \ldots, N$ вычисляем c_p.
- Ответ содержится в c_N

Нисходящее динамическое программирование

- Пишется рекурсивная подпрограмма, использующая то же рекуррентное соотношение

$$c_p = \sum_{i=1}^{\min(p,K)} c_{p-i}$$

- Перед выполнением вычислений проверяется, не решалась ли ещё эта же задача
- Если решалась, сразу выдаётся ответ, иначе решается, а ответ запоминается