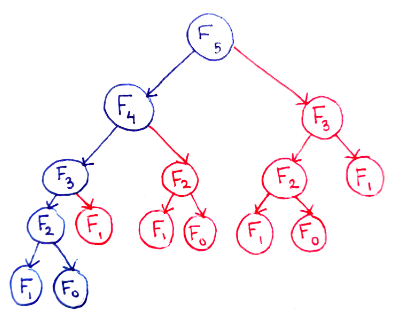
**Билет №7**

**Динамическое программирование. Основные понятия, ДП “вперед” и “назад” на примере задачи о кузнечике.**

Динамическое программирование. Основные понятия.

**Динамическое программирование** — метод решения задачи путём её разбиения на несколько одинаковых подзадач, рекуррентно связанных между собой.

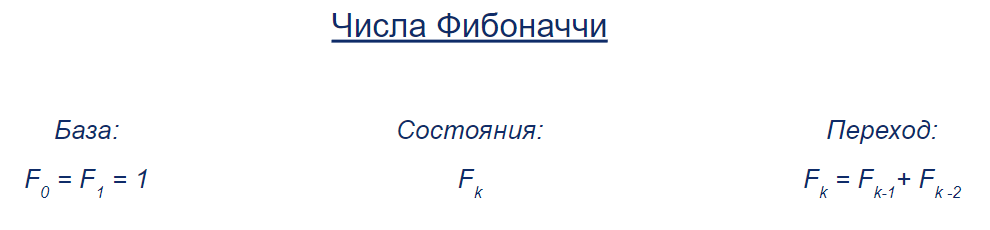
Самым простым примером будут числа Фибоначчи — чтобы вычислить некоторое число в этой последовательности, нам нужно сперва вычислить третье число, сложив первые два, затем четвёртое таким же образом на основе второго и третьего, и так далее.



**База** — известный ответ (или ответы) для задачи при фиксированных начальных условиях.

**Состояния** — подзадачи, на которые мы разбиваем изначальную (основную) задачу.

**Переход** — связь (обычно формула) для получения ответа для одного состояния через другие состояния.



**Определим основные шаги, которые необходимо проверить и продумать, чтобы решить задачу методом динамического программирования:**

1. Что хранится в dp[]? Продумать нужно как тип (Число, строка, пара чисел), так и что из себя представляет это значение (Количество способов, минимальное значение)
2. База dp? Результаты задач, которые мы можем использовать без пересчета. Как правило, эти ответы очевидны и используются в самом начале вычисления ответа
3. Формула пересчета? Она может быть не одна, однако нужно четко определить для каждой задачи, по какой формуле можно вычислить ответ на текущую задачу
4. Порядок пересчета? Необходимо пересчитывать в таком порядке, чтобы во время вычисления мы знали верные ответы для всех тех, которые мы используем в формуле
5. Где брать ответ?

ДП, это своего рода кэш (Caсhe), в который мы складываем предыдущие результаты вычисления этой же задачи.  ДП — термин, который лично у меня никаких правильных ассоциаций не вызывает, я бы назвал этот подход так — “Программирование с сохранением промежуточного результата этой же задачи, но меньшей размерности”

**Задача:** на числовой прямой сидит кузнечик, который может прыгать вправо на одну или на две единицы. Первоначально кузнечик находится в точке с координатой 1. Нужно определить количество различных маршрутов кузнечика, приводящих его в точку с координатой n.

**Шаги для решения задачи про кузнечика:**

1. Что хранится в dp[]? dp[k] - минимальное количество камушек, которое необходимо собрать, чтобы дойти до k-ой клетки
2. База dp? dp[1] = a[1], dp[2] = a[1] + a[2]
3. Формула пересчета? dp[i] = a[i] + min(dp[i - 1], dp[i - 2])
4. Порядок пересчета? i=2 => n
5. Где брать ответ? dp[n]

**ДП “вперед” и “назад” на примере задачи о кузнечике.**

Короче динамика назад, когда мы используем ранее вычисленные значения для i-го элемента, а динамика вперед, когда мы будущие вычисляем на основе i-го элемента(нихрена непонятно)  
на примере кузнечика: a[i]-количество способов дойти до i клетки

динамика назад: dp[i] = a[i] + min(dp[i-1],dp[i-2]);

динамика вперёд:

dp[i+1] = min(dp[i+1],a[i]+dp[i]);

dp[i+2] = min(dp[i+2],a[i]+dp[i]);

**Решение:** Обозначим за K[n] число способов прискакать в клетку с номером n. Тогда K[1] = 1, K[2] = 1. Заметим, что в клетку с большим индексом можно попасть из клеток с номерами меньше либо на 1 (прыжок длины 1), либо на 2 (прыжок длины 2). Тогда K[n] = K[n-1] + K[n-2]

**Асимптотика:** O(n), тк мы проходим по каждому из n.