1. *Линейные рекуррентные соотношения*

   (a) $a_k = 5a_{k-1} - 6a_{k-2} + 2^{k-2} + 4^k + 3$.

   (b) $F_n = F_{n-1} + F_{n-2}$.

   (c) $\vec{x}_n = A_1\vec{x}_{n-1} + A_2\vec{x}_{n-2} + \ldots + A_k\vec{x}_{n-k}$

   Надо уметь находить явный вид решения рекуррент вида (a) и (b).

   Надо понимать, что однородные линейные рекурренты (например (b), и (c)) сводятся к рекуррентам вида $y_n = Ay_{n-1}$ $(*)$. Эффективный алгоритм вычисления y_n заключается в быстром возведении в степень матрицы A.

   Offtop (надеюсь, что это вам не пригодится) Явный вид решения однородной рекурренты можно искать или стандартным способом через корни хар. многочлена, или представляя ее в виде $(*)$ и диагонализуя матрицу A, переходя в базис из ее собственных векторов. Последний способ выглядит так: $A = SDS^{-1}$, $A^n = SD^nS^{-1}$, $y_n = SD^nS^{-1}y_0$ – явный вид этого вектора получить можно, так как D диагональна. (Кстати говоря, A не всегда диагонализуема и тогда нужно переходить в Жорданов базис, но это вам точно не понадобится).

2. *Полиномиальный алгоритм*

   Задачи бывают разные: задача поиска, распознавание языка, вычисление функции, оптимизации и т.д. Тем не менее все алгоритмы их решающие моделируются моделями вычислений, например машиной Тьюринга (это неформальное утверждение похожее на тезис Черча). Более того, алгоритмы, которые можно реализовать на реальном компьютере, можно смоделировать на машине Тьюринга с *полиномиальным замедлением*. Это значит, что если мы умеем решать какую-то задачу за время $O(T(n))$, то существует машина Тьюринга, которая ее решает за время $O(p(T(n)))$, где p – некоторый полином. Например на семинаре я написал псевдокод, который позволяет находить НОД(a, b) за $O(\log a + \log b)$ арифметических операций, сложность которых, вообще говоря тоже надо выражать через количество битовых операций, но в любом случае итоговая сложность грубо оценивается $O((\log a + \log b)^3)$, то есть полиномом от длины входа. Так как полином от полинома – это полином (кстати именно этим и хороши полиномы), в таком случае мы можем уверенно говорить, что существует и машина Тьюринга, которая находит НОД(a, b) за полином (хотя его степень может возрасти).

   Поэтому для определения понятий полиномиального алгоритма и полиномиально решаемых задач можно использовать машину Тьюринга, понимая при этом, что мы не теряем общности, так как класс задач решаемых за полином на МТ совпадает с классом задач решаемых за полином алгоритмически.

   Определение 1. *Временной сложностью МТ M в худшем случае называется функция*

$$T_M(n) = \max_{x:|x|=n} T_M(x),$$

где $T_M(x)$ – количество переходов M на слове x.

Дальше сузим класс рассматриваемых задач до задач распознавания языка. Напомню, что МТ M распознает язык L (пишется $L = L(M)$), если на любом входном слове, принадлежащем L она останавливается в принимающем состоянии, а на любом входном слове, не принадлежащем L, она останавливается в непринимающем состоянии (МТ останавливается если функция перехода не определена на текущем состоянии и символе).

Определение 2. $\mathbf{DTIME}(T(n)) = \{L \mid \exists MT\ M :\ L = L(M), T_M(n) = O(T(n))\}$

Мало буков и ничего непонятно. Говоря русским языком, язык L входит в класс языков $\mathbf{DTIME}(T(n))$ т. и т.т. когда он распознается некоторой машиной Тьюринга с временной сложностью, не больше чем некоторая константа на $T(n)$.

Определение 3.

$$P = \bigcup_{k=1}^{\infty} \mathbf{DTIME}(n^k)$$

Иначе говоря, язык $L \in P$ т. и т.т. когда существует МТ M, распознающая L, и полином $T(n)$, такой что на любом входе длины n M делает не больше $C \cdot T(n)$ переходов.

Так определяется класс P – одно из ключевых понятий курса. Его определение, а так же некоторые философские рассуждения про то, почему именно полиномиальные языки, можно найти в любой из книг, которые я вам советовал.

Почему мы рассматриваем именно полиномиальные языки? Почему не полиномиально вычислимые функции? Потому что с языками связано другое ключевое понятие курса – сводимость по Карпу, о котором мы поговорим потом.

3. *Алгоритм Евклида*

   Про него можно прочитать где угодно, довольно лаконично написано в Гаче и Ловасе.

   Сам алгоритм, его полиномиальность.

   Пример полиномиального языка: $L = \{(a, b, D) \mid \gcd(a, b) \geqslant D\}$.

   Применения алгоритма: решение уравнений $ax = b \mod m$ и систем уравнений

$$\begin{cases} x = x_1 \mod m_1 \\ x = x_2 \mod m_2 \\ \dots \\ x = x_k \mod m_k \end{cases}$$