1. *Организационные вопросы.* (кратко)

2. *Асимптотики.*

   В курсе изучаются алгоритмы и их сложность. У алгоритма должен быть конечный вход (и конечный выход). На практике приходится решать задачи для больших входов, поэтому нужно уметь оценивать (асимптотически) количество затрачиваемых ресурсов (например времени, памяти) в зависимости от размера входа. Объясняю, что такое время $T(n)$ работы алгоритма в худшем случае. Пару слов про разные модели вычислений: MT, RAM, другие модельные. Нам надо научиться работать с асимптотическими обозначениями и освоить инструменты для оценивания асимптотического поведения функций, заданных сложным образом.

   Дальше ввожу асимптотические обозначения.

   Примеры: $\sum_{k=1}^{n} k$ (явное выражение и оценки сверху-снизу) и $\sum_{k=1}^{n} \sqrt{k}$ (сверху-снизу). $\sum_{k=1}^{n} \frac{1}{k \log k}$ на оценки интегралами. $\sum_{k=1}^{n} k \cdot k!$ на телескопию.

   Часто требуется оценить асимптотическое поведение функции, заданной рекурсивно. Если вы каким-нибудь образом угадали ответ, то его можно доказать по индукции.

   **Пример:** $T(n) = 2T(\lfloor n/2 \rfloor) + \Theta\left(\frac{n}{\log n}\right)$ (имеется в виду, что это выполнено для $n > N$ для некоторого $N$, а значения $T(1), T(2), \ldots, T(N)$ произвольные).

   Сначала постараемся догадаться какая асимптотика у заданной последовательности. Можно нарисовать дерево рекурсии (иногда с его помощью можно заметить нетривиальные вещи, как например при анализе линейного алгоритма поиска медианы). Но обычно хватает просто развернуть эту рекурренту в сумму. Так как мы пока ничего не доказываем, а делаем *прикидку*, то можно считать $n$ степенью двойки: $n = 2^m$, взять в качестве $\Theta\left(\frac{n}{\log n}\right)$ просто $\frac{n}{\log_2 n}$ и считать, что рекуррента верна, начиная с $n = 2$.

$$T(n) = 2T(n/2) + \frac{n}{\log_2 n} = 2\left(2T(n/4) + \frac{n/2}{\log_2(n/2)}\right) + \frac{n}{\log_2 n} = \ldots =$$
$$= \sum_{k=0}^{\log_2 n - 1} \frac{n}{\log_2(n/2^k)} + 2^{\log_2 n} T(n/2^{\log_2 n}) = 2^m T(1) + \sum_{k=0}^{m-1} \frac{2^m}{(m-k)} =$$
$$= 2^m \left(T(1) + \sum_{l=1}^{m} \frac{1}{l}\right) \sim 2^m \log m \sim n \log\log n$$

   Теперь строго докажем, что любая функция $T(n)$, которая удовлетворяет данной рекурренте, начиная с некоторого $n$, равна $\Theta(n \log\log n)$. Будем доказывать по индукции два утверждения.

- Докажем, что $\exists c_1$, т. ч. $T(n) \leqslant c_1 n \log_2 \log_2 n$ при $n \geqslant 4$. Обращаю внимание на то, что мы один раз выбираем $c_1$ (достаточно большой) и затем для этой $c_1$ не зависящей от $n$ доказываем и базу и любой переход. Очевидно, что мы можем выбрать ее, т. ч. $T(k) \leqslant c_1 k \log_2 \log_2 k$ для всех $k = 4, 5, \ldots, N$. После этого мы должны доказать переход. Пусть $T(k) \leqslant c_1 k \log_2 \log_2 k$ для всех $k = 4, 5, \ldots, N, N+1, \ldots, n-1$. Нам нужно доказать, что ту же константу $c_1$ можно использовать для оценки $T(n) < c_1 n \log_2 \log_2 n$.

$$T(n) \leqslant 2T(\lfloor n/2 \rfloor) + a_1 \frac{n}{\log_2 n} \leqslant 2c_1 \lfloor n/2 \rfloor \log_2 \log_2 \lfloor n/2 \rfloor + a_1 \frac{n}{\log_2 n} \leqslant$$
$$\leqslant c_1 n \log_2 \log_2 (n/2) + a_1 \frac{n}{\log_2 n} = c_1 n \log_2 (\log_2 n - 1) + a_1 \frac{n}{\log_2 n} =$$
$$= c_1 n \log_2 \log_2 n + c_1 n \log_2 \left(1 - \frac{1}{\log_2 n}\right) + a_1 \frac{n}{\log_2 n} \leqslant$$
$$\leqslant c_1 n \log_2 \log_2 n - c_1 \log_2 e \frac{n}{\log_2 n} + a_1 \frac{n}{\log_2 n} \leqslant c_1 n \log_2 \log_2 n,$$

  если мы выберем $c_1 \log_2 e > a_1$ (изначально мы могли это сделать). В предпоследнем переходе использовалось неравенство $\ln(1-x) < -x$.
- Доказательство того, что $\exists c_2 > 0$, т. ч. $T(n) \geqslant c_2 n \log_2 \log_2 n$ при $n \geqslant 4$ можно проделать почти так же, только в конце нужно чуть более аккуратно воспользоваться утверждением, что вообще говоря $\ln(1-x) \sim -x$ при $x \to 0$.

Если рекуррента имеет некоторый стандартный вид, то можно воспользоваться следующей теоремой и не проводить выкладки и доказательство каждый раз.

**Теорема.** *Для решения рекурренты вида*

$$T(n) = aT(\frac{n}{b}) + f(n),$$

*где $f(n)$ – известная положительная функция, $a \geqslant 1$, $b > 1$, верно следующее.*

*(a) Если $f(n) = O(n^{\log_b a - \varepsilon})$ для некоторого $\varepsilon > 0$, то $T(n) = \Theta(n^{\log_b a})$.*

*(b) Если $f(n) = \Theta(n^{\log_b a})$, то $T(n) = \Theta(n^{\log_b a} \log n)$.*

*(c) Пусть $f(n) = \Omega(n^{\log_b a + \varepsilon})$ для некоторого $\varepsilon > 0$. Пусть также выполнено дополнительное условие регулярности функции $f$: $af(\frac{n}{b}) \leqslant cf(n)$ при некотором $c < 1$ и всех достаточно больших $n$. Тогда $T(n) = \Theta(f(n))$.*

Примеры использования: алгоритм Карацубы, быстрое возведение в степень, mergesort.

Алгоритм Карацубы: задача перемножить два числа $a$, $b$ в некоторой системе счисления. Поразрядные операции выполняются за $O(1)$. Разделяй и властвуй.

Быстрое возведение в степень: задача возвести число $a$ в степень $n$. Все умножения выполняются за $O(1)$. Поэтому размер $a$ на сложность не влияет. Считаем длиной входа $\log n$.

- справа налево – возводим $a$ в квадрат $m$ раз и перемножаем нужные степени
- слева направо – разделяй и властвуй: $a^{2(\dots)+n_0} = a^{(\dots)^2} \cdot a^{n_0}$

Нерандомизированный алгоритм поиска порядковой статистики, доказательство его линейности.