Содержание

1 Введение

Обзор курса: понятие информации, энтропия Шеннона, колмогоровская сложность, коды, исправляющие ошибки, коммуникационная сложность.

Примерный адрес страницы курса: `/shad/base/Spring2017`.

2 Тривиум

Информация по Хартли (1928): текст из n символов из алфавита Σ кодируется $\log_2 |\Sigma|^n$ битами (далее логарифмы по умолчанию двоичные). Определение незамысловатое, но уже полезное.

Пример 1. Известно, что $x \in A$, сказано, что $x \in B$. Сколько информации передано? Ясно, что было $\log |A|$ информации, стало $\log |A \cap B|$. Значит передано $\log \frac{|A|}{|A \cap B|}$ бит.

Пример 2. Имеем n монет, одна из них фальшивая, легче остальных. Сколько нужно взвешиваний, чтобы её найти? Исходно не хватает $\log n$ информации, каждое взвешивание имеет 3 исхода, стало быть меньше, чем за $\frac{\log n}{\log 3}$ взвешиваний найти не получится.

Пример 3. $x \in S_n$ — перестановка. Можно сравнивать два элемента. Сколько нужно сравнений, чтобы найти перестановку?

$\log n! = \log \sqrt{2\pi n} \left(\frac{n}{e}\right)^n (1 + o(1)) = n \log n - n \log e + \frac{1}{2} \log n + O(1)$. Можно ли асимптотически приблизиться к этой границе?

Естественный алгоритм: сортировка вставками (выглядит довольно оптимально по сравнениям, не учитываем сдвиги). Используется $\lceil \log 1 \rceil + \lceil \log 2 \rceil + \ldots + \lceil \log n - 1 \rceil \leqslant (n-1) + \log(n-1)! = OPT(n) + n - 1 - \log n$.

3 Бинарный поиск

$A = [1, \ldots, m]$, нужно найти в нём $y \in \{1, \ldots, m\}$ с помощью сравнения $x \overset{?}{<} y$. Ясно, что нужно $\lceil \log_2 m \rceil$ вопросов. А что будет, если оппонент может соврать 1 раз? Легко придумать алгоритм, который даёт $3 \log n$ сравнений и $2 \log n$ (можно и лучше). Нас будет интересовать постановка, когда Responder (R) может соврать Questioner'у (Q) в доле вопросов не более ε.

Более формально, игра проходит с объявлением числа раундов n в самом начале игры и не более $n\varepsilon$ неверных ответов. Вопрос ставится так: при каких n существует стратегия у Q, которая гарантированно угадывает число? Утверждается, что можно предъявить алгоритм, работающий за $c(\varepsilon) \log n$ сравнений, чем мы и займёмся.

Ясно, что состояние бинарного поиска — это вершина бинарного дерева. Устроим алгоритм не в виде спуска по дереву, а в виде блуждания. Находясь в вершине, соответствующей числам $\{l, \ldots, r\}$, зададим вопросы $l \leqslant x, x \leqslant r$? Если получен хотя бы один отрицательный ответ, пойдём вверх. Далее, кроме случая, когда мы стоим в листе, задаём вопрос $m \leqslant x$ и идём в нужную сторону.

Утверждение 1. Лист, в который мы попадали чаще всего, есть ответ (при достаточно большой длине блуждания).

Доказательство. Подвесим дерево за лист x, тогда, если ориентировать рёбра к этому листу, то против этого направления можно идти только если среди ответов на данном шаге была ложь. В самом деле, разбор случаев помогает в этом убедиться.

Разделим все наши шаги на шаги вперёд f, назад b, l_x — шаги в листе x, l_{other} — шаги в других листах. Тогда можем утверждать, что $f \leqslant b + \log m, b + l_{other} \leqslant \varepsilon n, f + b + l_x + l_{other} = cn, c \geqslant \frac{1}{3}$. Итого $l_x \geqslant cn - (b + \log n) - \varepsilon n \leqslant cn - \log m - 2\varepsilon n$, а $l_{other} \leqslant \varepsilon n$.

$cn - \log m - 2\varepsilon n \geqslant \varepsilon n \Rightarrow cn - \log m \geqslant 3\varepsilon n \Rightarrow n \geqslant \frac{\log m}{c - 3\varepsilon}$. □

Константы, ясное дело, оценены грубо. Также про задачу известно, что при $\varepsilon > \frac{1}{2}$ всё плохо (ответ найти нельзя), при достаточно малых $\varepsilon < \frac{1}{10}$ всё совсем хорошо, при промежуточных можно получит вариации (например, экспоненциальный рост). Известны точные ответы для небольшого константного числа ошибок, и для некоторых вариаций (например, оффлайн поиск). Задача имеет связи с кодами, исправляющими ошибки.