Сделано в [Whisper AI](https://t.me/WhisperSummaryAI_bot)

Теория информации, Григорьев А.А., Лекция 04, 26.09.20

Чистая версия. Дата создания: ⁨04.10.2025⁩

И начали. Звук есть? Ну и хорошо. Это главное. Так. Чем мне рисовать-то? Вот этим будем рисовать. Что такое? Так. Ну ладно. Значит, мои излюбленные студенты. Так. Ну, стало быть... Занимаемся символьным кодированием. Кодируем источник. Значит, у нас имеется постановочка такая. У нас имеется какое-то множество. Х, как всегда, у меня. И мощность этого множества составляет m. И состоит оно из каких-то точек. И точек этих много. Это как бы входной алфавит. Он большой. Я хочу эти точки вам закодировать двоичными, ну, какими-то куичными или двоичными блоками, символами. Ну, то есть, к каждой точке хочу поставить в соответствии кодовое слово W1, там W, Enter. Где эти все W? Это либо 0, либо 1, если ку равно 2, либо какие-то куичные символы. То есть, у меня есть объем алфавита выходного, в котором я строю слова. Вот, у каждого слова будет некая характеристика, называемая его длиной, количество символов. L. Ну и стало быть, а у каждого этого символа, источника, будут вероятности P-житая. Ну вот у нас появляются P-житая и L-житая. Ну и, в общем, идея такая. Если у меня имеется M букв, вот здесь вот в этом алфавите всего множество из M-точек, то для передачи каждой точки мне нужно, грубо говоря, логарифм по основанию 2 от m битов. Ау! Так, логарифм по основанию 2 m битов. Ну, вот идея... Я хочу передавать покомпактнее, поменьше. То есть, пользоваться я собираюсь следующим. Буквы эти, ПЖТ, имеют неодинаковые вероятности. Но и стало быть, у этого источника имеется энтропия H от X. Ну, у которой вот есть сумма, ну, минус сумма. ПЖТ, маргарифм ПЖТ. Вот. Если все буквы равновероятны, то они интересны. Энтропия это составляет логарифм по основанию 2m. И сэкономить я ничего не смогу. Мне понадобится вот столько битов для передачи отдельного сообщения, для передачи одного символа. Но надежда вся на то, что буквы неравновероятны, и энтропия вот эта заметно меньше, чем логарифм по основанию 2m. Тогда есть надежда, что я смогу при кодировании сэкономить. Идея о экономии чрезвычайно проста. Я собираюсь тем буквам, которые сильно вероятны, обладают большими вероятностями, присваивать короткие кодовые слова с малыми длинными. Ну, а тем буквам, которые маловероятны, присваивать длинные кодовые слова, поскольку коротких мне не хватит. Ну, вот, в общем, в результате получается такая вот мера. Средняя длина кодового слова. Сумма по всем j. p, житая, на l, житая. Значит, у меня слова длины l, житая, встречаются с вероятностью p, житая. И вот в среднем кодовое слово будет иметь такую вот длину. Ну и вот я хочу заинтересоваться вопросом, какую вот эту среднюю длину я смогу получить. Мы с вами в прошлый раз... решили, что кодировать мы будем так, чтобы у нас было возможно однозначное декодирование. Чтобы при последовании конкатенациям большого количества слов, вот у меня слово 1, вот у меня слово 2, вот слово 3, вот я представил одно к другому, получил некое, как говорят, высказывание. Так вот, мы занялись проблемой однозначного декодирования. Вот этот кодирование, Код, который я буду использовать, значит, для каждой буквы-источника у меня имеется кодовое слово вот в этом ковичном алфавите. Это код. Ну и, стало быть, этот код обладает условию однозначного декодирования, если никакая конкатенация вот такая кодовых слов не может быть интерпретирована двояко. То есть вот эту последовательность букв. можно разбить на такую последовательность слов, а можно, скажем, на другую последовательность слов. Вот тогда это неоднозначное декодирование. А однозначное оно предполагает, что у нас любая последовательность кодовых слов однозначно интерпретируется только как именно эта последовательность кодовых слов. Ну и, стало быть, в теории формальных языков доказывается великая неравенна теорема Крафта о том, что Спектр длин однозначно декодируемого слова, говоря, удовлетворяет некому неравенство, неравенство крафта, крафта на квалити. Ну и вот мы его в прошлый раз с вами как бы доказали. Значит, составит это неравенство в следующем, что сумма по всем j, по всем кодовым словам, 2, ну вот q для куичного кода, в степени минус lj меньше или равно 1. Ну, то есть, длины кодовых слов, кода с однозначным декодированием, не могут быть любыми, а обязаны удовлетворять вот этой границе. Вот. Ну, чтобы быть, видите тут как? Значит, чем больше длина, тем меньше вклад вклада. вот этой компоненты в общую сумму. Поэтому это ограничение на длины снизу. Длины не могут быть слишком маленькими. Если я длину уменьшаю, то у меня вклад растет, а мне надо, чтобы сумма вкладов не превышала единица. То есть срова должны быть достаточно длинными. Вот это крафт и леквалити. Ну а теперь, значит, мы с вами систематически на ближайшее время переходим к у равному 2, простейший случай, И займемся обсуждением этого неравенства вот в каком контексте. Мы, значит, ограничим себя вполне конкретными кодами с однозначным декодированием. Это так называемые префиксные коды. Префиксные коды. Префиксные коды. Давай. Включил. Это у нас набор теперь двоичных слов. Значит, и мы хотим, чтобы код был префиксным. Это означает, что никакое кодовое слово w1, wm3 не является началом другого кодового слова. То есть, вот есть у меня длинное слово w1, wm3. У него есть множество суффиксов, если хотите это окончание, и множество префиксов. Вот это начало, всякое, W1, W2, я называю префиксом вот этого кодового слова. Так вот, код называется префиксным, если никакой префикс кодового слова не является кодовым словом сам. То есть не бывает так, чтобы вот у меня было такое короткое слово, а было слово, которое устроено вот так вот. W, Enter, а потом некое продолжение. W, M, плюс 1 и так далее. Вот тогда вот этот префикс является кодовым словом. Вот это нарушение префиксности. Итак, код обладает свойством префиксности, является префиксным кодом, если никакой префикс кодового слова не является кодовым словом сам по себе. Вот это свойство префиксности, оно гарантирует однозначную декодируемость. Понимаете, вот в этой общей постановке однозначная декодируемость была плохо формализована. В том смысле, что мы, глядя на словарь, не могли сказать явно, является он однозначно декодируемым или нет. А вот здесь мы можем это сказать. Вот смотрите, как это выглядит. У меня поступает на вход вон оттуда. Поток этих самых символов. Я начинаю грызть символ за символом до тех пор, пока не получу кодовое слово. Я рассматриваю префиксы все возможные. Сначала пришла буква W1, потом пришла буква W2 и так далее. Я то, что у меня есть, сравниваю со всеми словами в кодовом словаре. И как только я получу первое слово впервые, Я знаю, что никакое продолжение кодовым словом быть не может, поэтому вот это первое кодовое слово. Я обрываю все это, выкидываю это как результат и начинаю искать следующее кодовое слово в продолжении. Опять грызу символ за символом, пока кодовое слово не встретится. Как только оно встретится, я уверен, что продолжение уже не будет кодовым словом. Среди продолжений кодовые слова искать бессмысленно. Поэтому я вот в этом месте обрываю, и вот у меня следующие кодовые слова. То есть я, входной символный поток, бью на кодовые слова, руководствуясь вот этим условием префиксности. Так что свойство префиксности автоматически обеспечивает однозначное декодирование и обеспечивает выполнение неравенства крафта тем словом, который для двоичного алфавита выглядит вот так. 2 в степени минус льжитое меньше или равно единице. Замечательный факт состоит в том, что вот это свойство префиксности может быть хорошо очень понято и сформулировано в терминах двоичных деревьев поиска. То есть во всей этой тематике для нас будет большое значение иметь деревья поиска. Значит, смотрите. Любой кодовый словарь, вот у меня есть набор каких-то двоичных кодовых слов, W, с символами 0 либо 1. Я их свалил в кучу и стал изображать, представлять этот кодовый словарь, набор, код двоичным деревом. Делаю я это так. Вот у меня есть все множество кодовых слов, set. Оно естественным образом делится, ну вот так начну, вот это значит set. Оно делится естественным образом на слова, которые начинаются с нуля, ноль, а потом что-нибудь, и единица, а потом что-нибудь. То есть делится на два подмножества. Одно ноль, х, х, второе единица, х. Сюда я положил все слова, которые начинаются с нуля, а сюда я положил слова, которые начинаются с единицы. Вот этот путь я помечу символом 0, вот этот путь я помечу символом 1. У меня получились два подмножества S1, S0 и S1. Ну а дальше я эту процедуру могу продолжать. Смотрите, вот эти слова, они все начинают с 1. Но у них есть второй символ. Он может быть либо 0, либо 1. Поэтому я могу опять же эти слова поделить, тут написать нуля, тут написать единицу. И вот тут у меня появятся слова S11, а здесь у меня появятся слова S10. Первый символ единица, второй 0. Ну, соответственно, с нулем то же самое. Вот тут я нарисую два множества, set00 и set01. Вот так. И здесь я напишу 0, а здесь я напишу единицу. Ну и так я могу это вверх продолжать до посинения, пока множество не исчерпается, пока в нем не останется одно кодовое слово, и дальше уже делить будет нечего. В результате у меня появится то, что называется двоичное дерево поиска, binary search 3. Значит, начиная с корня, вот это у него корень, я иду по ветвям, либо по нулевому пути, либо по единичному пути, Ну и каждый раз попадаю в какое-то множество, и так продолжаю, и в конце концов прихожу куда-то. Любой код можно изобразить таким двоичным деревом. Тут никаких сомнений нет. Значит, смотрите, свойство префиксности выглядит в этой интерпретации так. Значит, вот у нас дерево. Оно состоит из путей, помеченных нулями и единицами. В общем виде это будет, оно будет куичное, помечено символами куичного алфавита. То есть у меня из одной вершины в куичном случае будет выходить от нуля до ку-1 путей. Соответственно, а тут только два пути, 0 либо 1. Вот оно дерево. Ну и, стало быть, идя по этому дереву от корня, я прихожу куда-то, где... Кончается кодовое слово. Вот это место я помечу листочком. Вот так он будет рисовать. Лиф. Смотрите, свойство префиксности выражается в терминах деревьев очень просто. Код является префиксным, если в дереве нет листьев в неконцевых узлах. Вот есть концевые узлы, из которых дальше идти некуда. Здесь имеют право расти листья, они могут быть кодовыми... Вот эти пути могут давать кодовые слова. А вот этот путь кодовое слово не может давать, потому что этот путь является префиксом для вот этого более длинного пути. То есть вот такие вещи отметаются. Итак, значит, у нас есть двоичное дерево поиска, и оно называется префиксным, оно дает... То есть всякий код изображается таким двоичным деревом. А префиксный код замечательен тем, что есть ограничения на то, где растут листья. Листья не могут расти в промежуточных узлах, из которых выходит хотя бы один путь. Листья могут быть только в концовых узлах. Вот это так формулируется свойство префиксности в терминах деревьев. Просто изящно. Ну то есть дерево красивое, оно в нем не растут листья бог знает где, а растут только на самых концах сучьев. В концевых узлах. Ну и, так как только это создано, становится понятно, что такое неравенство крафта. То есть, общее доказательство даже оказывается ненужным, оно становится просто очевидным. Смотрите, вот у меня есть уровень номер 0. Это длина кодового слова 0. Пустое кодовое слово. Вот есть уровень 1. На нем это вот L равно 1. На нем две вершины. Потом имеется уровень номер 2. На нем уже 4 вершины. Ну, то есть здесь 2. В общем, здесь 2 в первой степени, здесь 2 во второй степени. L равно 2. Где-то есть уровень номер L. На нем 2 в L-той степени вершин. Но каждый раз идет удвоение. Из каждой вершины выходят два нового пути. И каждый вот этот узик дает начало двум узгам. Поэтому общее число узгов удваивается каждый раз. Ну и вот если так идти, то на л-том уровне у меня будет 2 в л-той степени узгов. 2 в л-той степени узгов. Это я имею полное л-ярусное дерево. Значит, смотрите, оно кодирует успешно все слова длины L. Вот любой путь на этом дереве имеет длину из L ребер. Ну и, как бы быть, все слова длины L, коих имеется 2 в L степени штук, тут представлены. Ну а дальше я каждому... Все слова имеют одинаковую длину L. Значит, каждому слову припишу вес 2 в степени минус L. Ну вот, согласно нашему неравенству Крафта. То есть у нас сейчас полный код. Он состоит из 2 вольтой степени слов длины L каждая. Вот они все эти слова тут написаны. Это вот нулевое слово, вот сюда оно пришло, вот это единичное слово. Ну а по дороге вот отсюда до сюда все остальные слова в количестве 2 в степени L штук. Значит, у меня получается так, что все L житые равны L, и сумма по всем J, 2 в степени минус LGT. Это сумма 2 в степени минус L. А сумма эта включает ровно 2 в L-т степени слагаемых. Поэтому получается, что это 2 в степени минус L умножить на 2 в степени L аккуратно равно 1. То есть, вот смотрите, для полного дерева, у него, стало быть, листья растут только вот здесь, так что я имею префиксный код. Это очень частный случай префиксного кода, включающего все слова длины «ль». Без всякого сомнения, это код префиксный, потому что префикс никакого кодового слова не является кодовым словом сам. То есть листья вот здесь в промежуточных узгах не растут, все листья только в конце. А неравенство Крафта выполнено тривиальным образом с равенством. Сумма 2 в степени минус L по 2 в льтой степени узгам составляет как раз единицу. То есть для полного дерева вот такого неравенство Крафта более-менее тривиально выполняется. Кроме того, могут быть чертова тучи других префиксных кодов, не являющихся полными. Все они получаются путем обстригание этого дерева то есть беру я илья русное полное дерево и его начинаю подстригать встреч но она кой-то некрасивая треугольная я его хочу превратить во что-нибудь эдакая значит вот смотрите есть Процедуры правильной стрижки. Вот у вас есть L-3 уровень, а вот у вас предыдущий, L-1 уровень. И вот тут есть узел, из него вышли два пути, нулевой и единичный. Они дали начало двум путям на верхнем уровне. У каждого из этих листьев, которые тут растут, вес 2 в степени минус L. Идея стрижки состоит в следующем. Я вот эти две ветки обстригаю. А те листи, которые вот тут растут, на этой и этой ветве, я переношу вот сюда. Я вот тут вешаю лист с весом 2 в степени минус L без одного. То есть я вот эти два веса объединил в один. 2 в степени L минус одного. Вот это я называю правильная стрижка, регулярная стрижка дерева. То есть пока я выполняю только такие операции, значит это совершенно не обязательно делать... Обрезать именно вот так. Можно обрезать сразу на нот. Вот, скажем, четыре точки. Потом две и две. А потом еще две. Вы можете обстричь прям вот это все. Вот ровно так. Это тоже будет правильная стрижка. У вас тут четыре листа. Вы их все перевешиваете вот на эту вилку. Здесь был вес 2 в степени минус L у каждого листа. А этот вес, это суммарный вес. 4 раза по 2 в степени минус L. То есть 2 в степени минус L минус 2. Вот такой вес. И вот обратите внимание, пока я подстригаю дерево правильно, у меня ни один лист не падает на землю. А сумма весов листьев, она удовлетворяет неравенству крафта. Сумма... 2 в степени минус лжи, 3 равна единице. При правильной стрижке ни один лист не падает на землю, то есть я их просто перевешиваю из одного места в другое. Поэтому вот это равенство не нарушается. Так что до тех пор, пока я обстригаю дерево регулярно, полное дерево, пользуясь только правильными методами, правильная стрижка, у меня равенство крафта продолжает выполняться с равенством. сколько бы я ни стриг, я его могу обгрызть вот прямо вот до сих пор, повесив сюда все эти листья в одно место. У меня получится код нулевой длины с полным весом единицы вот здесь. Вот могу обстричь вот эту ветку целиком. Вот все это допустимо. Значит, вот такие операции регулярной, правильной стрижки, красивой стрижки дерева, они не нарушают неравенство крафта. Если дерево правильно подстриженное, ППД, правильно подстреженное дерево, то у него полный вес всех листьев равен единице. А вот когда нарушается неравенство крафта? Нарушается тогда, когда вы срезаете неправильно, то есть неизящно. Вот смотрите, у вас как бы есть вот четыре точки, вот дерево, и я, стало быть, вот эту ветку отрезал, а эту оставил. Вот это неправильная стрижка. Понимаете, после того, как я эту ветку отрезал, мне вот эти листья некуда повесить. Сюда я их повесить не могу, потому что тут лист быть не может. Это же промежуточный узел. В префиксном коде листья растут только на концевых узгах. Поэтому мне приходится просто вот эти листья бросить на землю. Так что всякая неправильная стрижка приводит к тому, что у вас в дереве Падают листья на землю, и неравенство крафта начинает пополняться с неравенством. То есть сумма 2 в степени лжитое, 2 в степени минуса лжитое становится меньше единицы. Так что вот, всякий префиксный код есть подстриженное дерево. Два тезиса. Дерево может быть подстрижено правильно и неправильно. Принципы стрижки правильные я вам указал. Надо, если обрезаешь ветку, то и обрезать другую. То есть обе ветки сразу. Тогда у тебя тут появляется концевой узел. Можно думать, что вилочка появляется. Куда я могу перевесить все вот эти яблоки с этого дерева. Вот их собрал и тут привесил в сетке. Вот это правильная стрижка. Она не нарушает неравенство крафта. То есть до тех пор, пока дерево правильно подстрижено, сумма 2 в степени... минус лжи ты аккуратно равна единице без всяких вариантов вот кроме того могут быть префиксные коды которым отвечает неправильно подстриженные деревья это когда вот какая-то ветка так обломана из этого узла идет продолжение вот тогда вы потеряли вот этот вес он исчез упал на землю и у вас два в степени минус лжи ты сумма стала меньше единицы вот То есть вот эта сумма стала отличаться на единицу ровно на то, что вы потеряли, вот этот суммарный вес. И так вот выясняется, что множество профиксных кодов и множество подстриженных деревьев – это одно и то же. Ну и стало быть, можно даже предложить процедуру построения этого самого дерева по заданному спектру длины. Вот я вам задаю... Кодовые слова... набор длин L1, L2 и так далее, LMT. Таких, что сумма 2 в степени минус LGT, ну вот, меньше или равна единице. Как построить дерево? Дерево строится таким образом. Из этих, например, можно дерево строить снизу вверх. Дерево строится так. Среди этих длин есть наименьшие. Вот у меня есть узел, корень этого дерева. Это L равно 0. Потом идет L равно 1. Это вот 2 узга и так далее. Среди этих L есть наименьший. Вот он, L минимум. L мин. Тут у меня 2 в степени L вершин. 2 в L степени L минимум узгов. Из каждого узла растет дерево в свою очередь. То есть я нахожусь в лесу. Вот это у меня деревья, их много. Два в степени или минимальная. Ну и, стало быть, каждое дерево неограниченно растет сюда. И вот смотрите, как я делаю. Я просто смотрю, сколько у меня сров имеется. Вот этой длины или минимальная. Ну и вот обстригаю эти деревья, нужное количество деревьев. Вот пусть этих сров минимальной длины имеется 5 штук. Тогда я 5 деревьев тут отрежу, и вот это у меня будут концевые вершины уже. То есть я уже обстриг дерево регулярно, правильно, вот на этом уровне. Все остальные срова имеют большую длину. То есть я иду теперь на верхнюю длину, и там у меня оставшийся лес. Ну и стало быть, где-то есть следующая длина по возрастанию. Сколько-то слов такой длины, я поднимаюсь на этот уровень и стригу там нужное количество деревьев. Вот и все. И действуя таким образом, вот если у меня тут равенство, смотрите, сумма 2 в степени минус лжитых не превышает единицы. Вот сумма этих весов... Когда я обстригаю дерево, я откладываю... сбрасываю вес вот с каждым деревом 2 в степени минус л который вот тут есть а или это высота этого дерева ну вот значит я тут просумевал сколько-то выбросил но эта сумма заведомо не превышает единицу так что деревья еще остались я могу двигаться дальше когда я не смогу двигаться дальше а тогда когда на очередном уровне у меня просто не останется деревьев нечего стричь Но тогда это означает, что у меня будет неравенство крафта нарушено вот в эту сторону. Ну, то есть в пределе я получу, что сумма 2 в степени минуса лежитых равна единице. Ну и стало быть... Если у меня изначально 2 в степени лежитые меньше единицы, то поднявшись куда-то на верхний уровень, я обнаружу, что вот я эти деревья обрезан. У меня тут получились концовые вершины и еще что-то осталось. Это вот за счет вот этого дефекта. Ну, то есть вот этот вес, это и есть разница между суммой 2 в степени лежитой и единицей. Вот этот вес. Ну, его просто надо отбросить, обрезать, не присваивая этим вершинам, не считая эти вершины концовыми. И вот вы получаете дерево с заданным спектром длин, удовлетворяющих неравенству крафта. Вот таким образом. Ну вот это всякие словеса про... То есть наука о кодировании источника неожиданно хорошо формулируется и уходит в теорию двоичных деревьев поиска. Ну теории это особенно не назовешь. В общем банальные вещи. Мы строим и обрезаем двоичные деревья. Ну а теперь стало быть... Теорема кодирования для источника. Значит, у нас с вами был источник х. Он состоял из букв количеством штук. Эти буквы обозначались х и имели вероятность p-житая. У этого источника была энтропия h от х, в которой есть сумма минус p-житая на гогарифм p-житая. Мы сейчас с вами сравним, научимся сравнивать среднюю длину кодового слова. Нашей мерой качества кодирования является средняя длина кодового слова, которая есть сумма P-житая на L-житая. Значит, мы житой буквой источника присвоили кодовое слово двоичной длины L-житая, и вот у нас средняя длина кодового слова. Охота ее сравнить с энтропией. h от x в обе стороны. То есть как-то понять, какова связь между средней длиной и энтропией. Ну вот, мы будем это ограничивать в две стороны. Значит, мы покажем, что эта самая средняя длина, L, она, во-первых, больше или равна энтропии источника, вот такое неравенство, и меньше или равна энтропии источника плюс единица. Вот это две теоремы кодирования, прямая и обращение. Обращение это вот это. Оно говорит следующее, это неприятная вещь. Оно говорит, что средняя длина не может быть сделана очень маленькой, она больше энтропии. Энтропия источника у вас составляет 128 битов, значит, средняя длина кодового слова обязана быть больше 128. У нас источник большой, поэтому битовая энтропия большая. То есть энтропия, верхняя граница для нее – это гогарифм по основанию 2m. А m – это вообще-то большое число, считается сейчас. То есть мы кодируем большой алфавит в кодовые слова над малым алфавитом. Итак, как будем доказывать это дело? Я начну вот с этого неравенства. Обращение теоремы кодирования источника, символьного кодирования. Значит, средняя длина больше или равна этой самой энтропии. тут интересно то что доказательствами уши очень простые они что называются в одну строчку значит смотрите вот неравенство крафта если у меня построена кодовая построен словарь со спектром длина лежит и то заведомо выполнено условие 2 в степени минус и джит меньше или равно единица обозначу это число буквой гамма допустим есть то самое вот так напишу равно гамма меньше или равно единиц вот ну то есть по неравенству крафта коль раз я собираюсь строить префиксный код в то вот для спектра длин должно выполняться неравенство крафта и сумма этих длин должна быть некой гаммы точнее сейчас сильно хочется вот эти длинны начать считать вероятностями и Но они на вероятности не тянут, по той причине, что их сумма отнюдь не равна единице, а равна вот этой гамме. Ну, как надо поступить? Надо взять вероятности кужиты, сделать равными 2 в степени минус лжито и делить на гамма. Ну, отнормировать это дело на вот эту гамму, которая порядка единицы. Ну и окажется, что сумма кужитов аккуратно равна единице. Сумма кужитых. Это сумма 2 в степени лжитых. Она равна гамма. Гамма сокращается. То есть я построил распределение вероятностей Q в виде по длинам кодовых слов. 2 минус лжитые на гамма. Ну а теперь я просто пользуюсь... У меня есть два распределения теперь. Есть распределение P, которое изначально вероятностей букв. И есть распределение Q. Вот я его только что построил. Посмотрю на информационную дивергенцию между P и Q, которая, как известно, вот дивергенция между Q и P, которая, как известно, положительна. Вот это и будет отсюда я сейчас и получу нашу границу. Значит, это самая дивергенция. Это есть сумма по всем J. P от X на логарифм Q от X. Ну, P житая лучше. P житая на гагарифм Q житая поделить на Q житая. Вот так вот. И эта штуковина положительная. Вот это просто равенство для дивергенции. Ну что, подставляю, начинаю это дело раскрывать и вычислять. Вот смотрите, тут уже видно, что вот получится энтропия. Сумма P житая на гагарифм P житая. То есть это равно минус H от X от X. Минус антропия моего источника. Сумма. И то, что останется, это плюс сумма ПЖТ на гогарифм КЖТ. Ну и тут надо поставить минус. Потому что К вздоминатели. Подставляю мое определение для КЖТ. Значит, гогарифм КЖТ, гогарифм К. Это есть минус L. У меня гогарифм по основанию 2, поэтому это минус L и минус гамма, минус гогарифм гамма. Вот и все. Значит, вот эта сумма, она оказывается равной. Плюс, потому что гогарифм Q, вот я, минус гогарифм Q равен плюс L плюс гогарифм гамма. Поэтому здесь минус гогарифм я заменяю на сумму pg на lg, ну и плюс сумма pg на гамма. Ну вот, собственно, и все. Вот это, ребятки, средняя длина, а это, ну, некая добавка. Значит, я получил какую оценку? Что у меня средняя длина, l от x, она равна вот это, l средняя. Значит, l средняя равно h от x Вот H от X, это же у меня больше... Так, тут больше или равно нуля. Вот это все положительно. Поэтому у меня энтропия, средняя длина, больше или равна. То есть средняя длина, вот она, средняя длина, больше или равна. Энтропия, перенес направо. И минус... Вот эта сумма, она просто равна гамма. По той причине, что гамма постоянная, а сумма P-житых равна единице. вот эта сумма просто равна гамма вот но только в горифм кома прожитая на гагарит гамма ну то есть она просто равна вы горифм гамма значит поэтому тут h от x минус вы горифм гагарит гамма ну вот значит смотрите средняя длина больше или равна энтропия за вычетом вот этой величины гамма эта величина меньше единицы и Гамма, вот оно что такое. Это величина порядка единицы, но чуть-чуть поменьше единицы. Больше единицы она не бывает. Значит, логарифм от нее, это величина отрицательная. Значит, тут добавлен плюс логарифм единицы на гамма. Лучше вот так вот написать. Вот. Вот эта величина погожительная. Единица на гамма чуть побольше единицы. Гамма около единицы, но меньше. А эта около единицы, но больше. Поэтому вот средняя длина больше или равна энтропии плюс некая добавка. Но эта добавка положительная, поэтому это больше или равно h от x. Вот, собственно, и все, что и требовалось доказать. Значит, средняя длина больше энтропии. Даже понятно, когда хорошо это неравенство выполняется, когда средняя длина приближается к энтропии. А ровно тогда, когда гамма равно единице. есть когда неравенство крафта выполнено с равенством тогда вот эта добавочка уходит в 0 а так средняя длина больше энтропии на вот эту поправку она не просто превышает энтропию превышает и вот на эту поправку который растет с уменьшением гамма то есть чем сильнее и хуже у вас выполнено неравенство крафта тем больше это добавка вот это обращение теоремы кодирования средняя длина не привыкла за заведомо выше энтропии источника и никуда не денешься лучше закодировать нельзя доказательство ну вот сами видите она совершенно простое стоит вводите распределение к вероятности и ищите информационную дивергенцию между дэйп между д между ку и п распределение ну вот она и получается вот так ну вот это прямаете обращение теоремы кодирования значит Кодировать источник можно, и можно хорошо, но средняя длина заведомо будет превышать энтропию. Вот это как бы еще одно оправдание нашего определения энтропии. У вас есть источник, вы измеряете энтропию этого источника, для этого нам надо... Померить распределение вероятности его букв. Вот простой пример. Скажем, обычный текст на английском языке или на русском языке. Есть буквы, они неравновероятные. Есть распределение вероятности этих букв, эмпирически подсчитанное. Можно посчитать энтропию этого источника. Энтропию печатного текста. Эта энтропия составляет что-то 2,7... Для английского алфавита 2,7 бита на символы. вот стала быть средняя длина кодового слова если кодировать буквы двоичными блоками но где-то 2,7 бита на букву приходится если это вот оптимальным образом закодировать в обратную сторону значит смотрите теперь мне предстоит доказывать Мы с вами доказали следующее, что средняя длина, L средняя, она больше или равна энтропии. Мы сейчас научимся строить код такой, что оно будет меньше, чем h плюс единица. Делать это мы будем так. Мы рассмотрим наш набор вероятностей, пожитая. Ну и слово с вероятностей пожитой я должен поставить в соответствие какое-то кодовое слово какой-то длины, L житая. Вот это для LJ я возьму просто равным логарифмом единица на PJ. То есть возьму я вероятность PJ, она меньше единицы. Ну или то, что то же самое, это я так пишу, чтобы минус написать. Минус логарифм PJ. Ну уже вот понятно, откуда уж и растут. Значит, если я захочу длинные иметь такими вот, то у меня не очень-то получится. Дело в том, что длинные они обязательно целые, а логарифмы вероятностей целые не обязательно. Короче, вот у вас есть ось длин, на ней есть целые числа, и куда-то вот сюда придется логарифм единиться на p. Он не будет целым числом. Ну и тут надо как-то выйти из положения. Длина должна быть целой обязательно. Длина не бывает не целой. Я добавлю к этому вот этот отрезок. Назову это μ-житое. То есть я это, выгодив МП, округлю сверху до ближайшего целого. Прибавлю к нему плюс μ-житое. Для μ это вот эта поправочка. Это μ-житое заведомо меньше единицы. Оно больше нуля, больше или равно нуля, и заведомо меньше единицы. Вот так. И вот я нашел некий спектр длин кодовых слов. Некий спектр длин кодовых слов. Ну и стало быть, то есть я взял вероятности пожитые и для каждого пожитого посчитал длину кодового слова вот по этой формуле. Значит, мне нужно проверить, что существует префиксный код с этим спектром длин. И тогда я его построю конструктивно, вот так, как я вам рассказывал. Значит, смотрите, проверяем, что вот этот спектр длин, Удовлетворяет неравенство Крафта. То есть сумма 2 в степени минус лжитое меньше или равна единице. Ну, просто тупо вычисляю. Подставляю сюда лжитое. Значит, это будет 2 в степени логарифм П житое. И плюс и минус мюм житое. Это я написал 2 в степени минус L житая. Вот L житая я поставил вот в этом виде. Минус гогарифм плюс мю и знак поменял. 2 в степени гогарифм P житая это ровно P житая. То есть это P житая на 2 в степени минус мю житая. Вот. Ну и стало быть имею для неравенства Крафта сумму 2 в степени минус L житая. Ну я вот просто тут продолжу. Равно, равно. Сумма П житых на 2 в степени минус мил житая. Вот так. Значит, все 2 в степени минус мил житая положительные, не превышают единицы даже. Ну вот, значит, 2 в степени минус х это вот такая функция. 2 в степени минус х, вот она вот так себя ведет. У меня 2 в степени минус мил житая, ну вот мил где-то Меньше единицы, но вот это единица. Значит, они все меньше единицы, поэтому это можно сверху ограничить. Суммой пожитых. Значит, 2 в степени минус мюжитое меньше единицы. Ну и стало быть, это меньше или равно суммы пожитых, а это равно одному. Вот неравенство Крафта. Вот так. Значит, таким образом код с таким спектром длин кодовых слов... Заведомо существует. Я его могу эффективно построить. Вот так, как я вам рассказывал. То есть я смотрю на свой лес. Ищу среди этих длин минимальная длина. И начинаю этот лес на эльтом уровне рубить. Вот у меня есть пять слов вот этой длины. Я вот пять деревьев тут погублю. Остальные останутся. И они будут губиться на верхних уровнях. Значит... Если неравенство крафта выполнено, то этот процесс закончится на каком-то уровне тем, что я вырубил все деревья, и, возможно, еще какие-то деревья остались. Их я просто проигнорирую. Я не буду им ставить соответствие кодовые слова никакие. Вот я получу кодовый словарь с суммой 2 в степени L житых меньше 1. Если сумма 2 в степени лжитых минус лжитая просто равна единице, то у меня получится полное дерево. Свободных деревьев их не останется. Я вырублю все. Таким образом, существование такого кода почти очевидно по этой процедуре. Осталось оценить его характеристики. Этот код будет хороший или плохой? В смысле средней длины. Давайте считать его на среднюю длину. Средняя длина – это сумма LJ на PJ. LJ – ну, вон оно. Это сумма логарифм. Ну, вот давай так напишу. Вот тут PJ, тут логарифм единица на PJ плюс μJ. Вот. PJ на логарифм PJ – это энтропия H от X. Вот она. Вот она прямо есть. И добавочка в виде суммы PJ – межитая вот эта добавочка заведомо не превышает единицы то есть меньше или равна меньше или аж от x плюс 1 ну и даже вот эту сумму я значит мы у меня не превышает единицы но сумма прожитых равно одному вот и все значит сумма значит таким образом у меня средняя длина того что получилось превышает энтропию как максимум на одну единицу, на один бит. И, значит, более-менее понятно, за счет чего это превышение. Оно, ребятки, за счет вот этих округлений. Вот если бы округлений не было, и у меня все вероятности имели бы целые логарифмы, то есть выражались бы степенями двойки, P-житое, чтобы было равно 2 в степени минус L-житое, Тогда вот эти межитые были бы нулями все, и у меня добавка была бы нулевая. Я имел бы меньше или равно h от x, то есть я имел бы качественный код. А если мне по дороге приходится округлять вероятности, а не не степени двойки, то возникают вот эти поправки. И вот эта единица, это, в общем, среднее значение этой поправки. Когда я округляю логарифм вероятности до ближайшего целого, я вношу вот такую поправку. Она не превышает единицы, но, собственно, вот эта избыточная длина, она и состоит из математического ожидания этих поправок по распределению P. Вот такие дела. Ну, стало быть, вот это прямая и обратная теорема кодирования источника. Ну, вот... Быть такие дела. Обе эти теоремы обнажают очень простой факт. Важность энтропии. Если у вас имеется источник с энтропией H, то вы можете построить префиксный код с такой средней длиной, которая заведомо превышает энтропию, но меньше, чем эта энтропия плюс единица. Хорошо это или плохо? Границы эти весьма хорошие при больших мощностях алфавита. Когда мощность алфавита велика, то энтропия большая. Она, как известно, порядка логарифма М. Если у меня М миллион, 10 в 6, то это, стало быть, 20 битов будет. Если М равно 2 в 20, это порядка миллиона, то у вас 20 битов вот этой энтропии. И вы, стало быть, добавляете границу всего один бит лишний избыточный бит мы поэтому эта граница над тем точнее чем больше у вас мощность алфавита и чем больше энтропия в количественном выражении вот ну и стала быть вот так но остается вопрос как строить эти самые оптимальные деревья на то есть был человек который назывался фамилии хаффман и он предложил оптимальную процедуру построения деревьев тот так и называется хаффмановское кодирование алгоритм хаффмана построение оптимального дерева двоичного по заданному набору вероятности ошибки вероятности слов иначе есть у вас вероятности p1 p mt ваша задача нисходя с места построить дерево минимальной длины и Такое, чтобы длина кодового слова, средняя, была минимально возможна. Ну вот, из каких соображений строит это дерево? Значит, смотрите, два наблюдения простых. Вообще, все это хафмановское кодирование, оно предельно простая наука. Значит, смотрите, более-менее легко понять следующее, что у этого дерева будет самый верхний ярус. на котором будет хотя бы одно кодовое слово. Ну, то есть кодовое слово самой большой длины. Значит, вот смотрите, кодовым словам самой большой длины должна отвечать самая маленькая... Значит, теорема такая. В оптимальном дереве, имеющем минимальную длину, среднюю, самой малой вероятности p отвечает самое длинное кодовое слово. Более-менее понятно. Вот смотрите... Оценка тут такая. Дело в том, что представим себе, что у меня имеется вот эта длина L, я ее условно обозначу L, чтобы отличать от L. А где-то вот тут обладается длина L. И вот у этого слова вероятность P маленькая, а у этого слова вероятность P большая. Значит, идея состоит в следующем. Если я вот эти кодовые точки поменяю местами, то есть сделаю вероятность маленькой вот здесь, а большой вот здесь, то какой проигрыш в длине я получу? Вот сейчас у меня длина такая. У меня длина L встречается с вероятностью P. И при длина L маленькая встречается с вероятностью P большая. Вот P большое отличается от маленького просто тем, что можно было бы писать Pmin, Pmax, но это просто вот трехконечное обозначение. Вот это вкрад в среднюю длину, которая имеет место сейчас, когда я длинному слову присвоил маленькую вероятность. А теперь я поступлю наоборот. А что будет иначе? Если я длинному слову присвоил большую вероятность... А короткому слову L маленькая присвою маленькую вероятность. То есть поменяю эти два символа местами просто-напросто. У меня вклад в среднюю длину будет вот такой. Ну и вот достаточно их вычесть, чтобы все понять. Вот я от этого отнимаю вот это. L на P на L на P. Дальше простая алгебра. Группирую слагаемые с L. Получается L на P минус P маленькая. Ну и плюс L. И, значит, вот, короче говоря, приводится к такому виду. L-L на P-L, чтобы долго не писать. Вот если раскрыть скобочки, оно так и выходит. Ну, и, стало быть, поскольку это положительное и это положительное, P-L больше, чем P-L, L больше, чем L-L, то от такой перестановки я проигрываю в средней длине. Только и всего. Вот такое простое наблюдение. самая малая вероятность отвечает слову на верхнем ярусе ну а дальше так второе наблюдение но такое итак вот первые теоремы среди этих вероятности есть самая меньшая вероятность и ей отвечает самое длинное кодовое слово лежащие вот на дереве на самом верхнем ярусе автор это из такой это слово не может быть одно у него есть сосед вот самом деле если это слово одно то у нас вниз уходит ветка вот такая у которой нет второй вид в там она одиночная у него нет второй ветви вот это обгрызана но тогда я могу сократить и вот сократить с кодовой словарь вот на этот путь не взять вот эту точку за кода вот сюда взять этого листа и перевесить его вот сюда тем самым я сокращу опять же среднюю длину значит поэтому так Самая минимальная вероятность — это самое длинное кодовое слово, а следующая вероятность — она тут же лежит. Ну вот это, стало быть, два тезиса. Итак, у оптимального кода есть самый верхний уровень, на котором лежит некое кодовое слово, и оно, это кодовое слово, самое наименьшее, наименее вероятное. Рядом с ним обязательно лежит другое кодовое слово. Вот, ну и, стало быть... Идея Хаффмана состоит в том, что эти коды войскова надо объединить в одно и начать строить оптимальный код далее. Только и всего индуктивно. То есть поступаем так. Вот в чем состоит процедура построения оптимального кода. Рассматривайте набор вероятностей. Ну и для конкретности их упорядочиваем так. p1, меньше p2, меньше p3. Ну меньше или равно. Меньше или равно ПМТ. Ну, то есть я их в каком-то порядке упорядочил, в порядке возрастания. Беру вот эти две вероятности, самые маленькие. И мы обязаны в оптимальном коде, в самом коротком, в среднем коде, обязаны отвечать две вершины на верхнем уровне. Я поступаю так. Объединяю эти листья в один... Вот это у меня P1, вот это у меня P2. У меня появляется виртуальный узел с вероятностью P1 плюс P2. То есть вот эти два листа объединяю в один с вероятностью P1 плюс P2. А потом идет P3 и так далее, Pm. Вот у меня получился... В двоичном дереве я построил два пути. Ну и, стало быть, у меня количество вершин... сократились на одну. Было М вершин, осталось М-1 вершина. Вот эта вершина виртуальная, она заменила собой две вершины. Ну а дальше я ту же самую процедуру применяю далее. Я сортирую это в порядке убывания, ищу две самые маленькие вероятности и снова объединяю их в одну. Тем самым, вот стало быть, у меня получились какие-то теперь точки, у них есть какие-то вероятности. Ну и стало быть, я ищу две самые маленькие вероятности, объединяю их в одну и еще на единицу сокращаю свой размер алфавита. Так продолжаю до тех пор, пока останется две точки с какими-то вероятностями. Я их объединю в одну. В результате у меня получится двоичное дерево. Ну, может быть, вот простой пример, совсем простой пример. Значит, возьмем четыре точки, скажем, вот четыре точки 1 2 3 4 с вероятностями такими допустим 1 2 учетом 1 4 1 8 1 8 прекрасно вот начинаю строить хаффмановский код для этой фиде для вот этого набора вероятностей значит одна вероятность половинка ну вот делаю как алгоритм значит Рассматриваю все эти вероятности, ищу среди них две, самую маленькую и следующую по возрастанию. Ну вот они, стало быть, вот они. Я их объединяю в один узел. У меня получается узел с вероятностью 1,8 плюс 1,8, 1,4. Рассматриваю то, что у меня осталось. Осталось у меня вот это. Ну, значит, тут есть два узла с вероятностями 1,4 каждой. Я их объединяю вот в один узел и получаю узел с вероятностью 1,2. И у меня осталось два узла с вероятностями 1,2. Их я объединяю вот так. Вот получилось двоичное дерево поиска. Корень 0,1, 0,1, 0,1. У меня возник префиксный код, состоящий из нулевого кодового слова. Кодовое слово 1,0 длины 2. 1,0. Кодовое слово 1,1,0 длины 3. 1,1,0. Ну и 1,1,1. Вот 4 кодовое слово. Вот это я присвоил символу с вероятностью 1,2. Вот это я присвоил символу с вероятностью 1,4. А два другие я присвоил вот этим символам с вероятностью 1,8. Вот я работаю таким префиксным кодом. Ну и вот смотрите, если я устрою любую конкатенацию этих кодовых слов, вот вслед за этим запишу этого, а потом запишу этого, а потом запишу нуля. То вот как я это буду интерпретировать на стороне приема? На стороне приема ко мне будет вот так сыпаться этот битовый поток. Я буду гулять от корня дерева. Дерево у меня перед глазами. Вижу три единицы. Я так и иду. Один, один, один. Встал в коре, в этот лист. Это я получил вот этот символ. поставил вот тут границу вернулся в корень и пошел дальше 110 это я пришел сюда значит вот я прошел этот путь раз я дошел до листа значит я снова возвращаюсь корень и начинаю двигаться дальше значит 10 это я пришел в эту точку и наконец на последнем этапе я сразу получу нуля и попаду вот сюда и буду вынужден вернуться обратно вот так это интерпретить вот вот это я вам как бы продемонстрировал свойства однозначной декодируемости. То есть, когда вам предъявляют поток из таких кодовых слов, то каждое кодовое слово, оно просто указывает путь на дереве к листу. Вы идете по этому пути, руководствуясь вот этими битами, то есть в каждом узле берете очередной бит и идете либо налево, либо направо. Приходите в концовой узел. Этот концовой узел размечен каким-то символом. Вы даете на выход этот символ, сами возвращаетесь в корень и начинаете идти по новому пути. Вот так декодируются эти коды. Вот это хаффмановское кодирование. Есть оптимальная процедура построения хаффмановского кода. Она дает код минимальной длины. Процедура, которую я вам рассказывал, путем стрижки лесов, Она очевидная, но она не дает кода минимально возможной длины. А Хаффмановская процедура замечательна тем, что она оптимальна. То есть она дает такой код двоичный, что средняя длина его не может быть уменьшена никакими ухищрениями. То есть код оптимален в том смысле, что среди всех префиксных кодов этот код обгадает минимальной длиной. Мы его так строили. Мы строили по индукции. сообразили что на верхнем ярусе у оптимального кода должно быть и длинное слово что у него должен быть сосед мы их объединили и у нас так все хорошо получилось вот то есть раз он оптимально в этом месте не улучшаем а но не оптимальный во всех следующих местах не улучшаем а поэтому значит вот это действительно оптимальный код. Ну и, стало быть, то, что он дает, это вот средняя длина, больше энтропии, но меньше энтропии плюс единицы. Значит так. Оптимальный Хаффмановский код, естественно, не всегда единственен. Ну вот, спрошу рядом. Возникать будут ситуации, когда у вас на каком-то уровне есть много слов с одинаковыми вероятностями P, P, P, P. Вот такая вот картинка. Тогда для объединения вы можете выбрать любую пару равным образом. Можете вот эту пару объединить, а можете вот эту. Ну, то есть у вас появляется варианность, вариативность. Вы можете продолжать строить дерево многими способами. Но это вот дает... Оказывается, что заданному спектру вероятностей, набору вероятности, отвечает вообще говоря не одно оптимальное дерево, их может быть несколько. Довольно очевидно. Единственная природа вариативности – это много вариантов выбора двух соседних вероятностей. У вас всякий раз для объединения выбирается самая маленькая вероятность, она, и она неоднозначна. Самых маленьких тоже может быть пять штук. и какая-то следующая вероятность вот тут проблем если самых маленьких несколько то возникает проблема ну вот как выбрать пару из множества из нескольких вариантов ответ такой произвольно дело в том что все варианты одинаковые они дадут то же самое длину среднюю все в те оптимальные варианты одинаковы значит Перед разработчиками всяких вот этих кодеков встает такая проблема. Дело в том, что код приходится строить по заданному набору длин кодовых слов. Ну, то есть длина кодовых слов и вероятность это примерно одно и то же, как мы понимаем. Ну и тут вот, если вам по каналу связи... Вот смотрите, как это бывает обычно. Обычно это бывает так. На той стороне закодировали источник, построили хаффмановский код для него... И мне должны сначала сообщить этот код, а потом уже начать им пользоваться для передачи информационного потока. Код они мне сообщают обычно, указывая спектр длин кодовых слов. Вот набор этих самых длин. В данном случае 1, 2, 3, 3. Вот такой спектр они мне сообщили. И я по этому спектру должен построить дерево Хаффмана. Не какое-нибудь, а такое же, как у них. Вот тут вариативность, она мешает по жизни. То есть существует несколько вариантов построения этого оптимального кода. Ну и, стало быть, нам надо с ними, с той стороной, как-то договориться, чтобы у нас данный набор цифр приводил к одному и тому же результату. Договоренность простая. Мы должны договориться о порядке выбора вероятностей. есть когда есть три вероятности одинаковые допустим pp и p то мы должны договориться что вот на множестве этих символов установить какой-то порядок и значит вот этот вот эту пару делать предпочтительнее чем вот до любую другую пару ну то есть мы должны договориться как-то о нумерации символов и стали быть в пару всегда выбирать Пару символов с наименьшими номерами из имеющихся возможностей. И тогда мы будем получать одинаковый результат. Ну вот, примерно так. Значит, это что касается Хаффмановского кодирования. Значит так, домашнее задание. Вот вы мне в порядке упражнения к следующему разу построите код Хаффмана вот какой. Возьмем мы двоичный поток с вероятностью единицы P. Ну вот, P от единицы, пусть будет 1 треть. п от нуля будет 2 3 и мы рассмотрим 3 битовики бит 0 бит 1 бит 2 бит это каждый бит это 0 либо единица вот с такими вот вероятностями у нас имеется входной у фабит x состоящий из всех возможных таков битовиков коих имеется ровно 8 штук и у них есть какие-то вероятности которые вот вероятности в единицы по вероятности в нуля посчитаете мне требует шапу не построили оптимальный хаффмановский код ну вот для восьмиточного фарита вот этого алфавит состоит из всех то и всех трехбитовых комбинаций и p от b ну вот считается исходя из того что вероятность единицы 1 3 вероятность 0 2 3 вот это дает вам п от блока 0 1 2 2 вероятности тем самым у вас имеется алфавит мощности 8 его надо перекодировать у двоичный алфавит но вот с тем чтобы знать так я вам это рассказываю для двоичного алфавита если переходить к уичному алфавиту то ребятки ну ровно ничего нового то есть вот теперь мы с вами будем кодировать тот же самый источник состоящий из букв с алфавитом большущей мощности м в какой-нибудь куичный алфавит то есть слова дублевые житые них символы слова из вас слов будут выбираться из алфавита мощности кух только и всего что нового принесется ответ такой а ничего просто вот эти энтропии надо считать по основанию кух только и всего ну то есть В определении энтропии я почему брал гогарифм по основанию 2? Ну, потому что я рассчитываю на битовый поток. Гогарифм по основанию 2p. Ну, то есть, вот такое определение энтропии, оно основано на том, что я за единицу измерения энтропии выбираю бит. А когда у вас куичный алфавит, то естественная мера энтропии является гогарифм q. Гогарифм по основанию 2q. Ну и поэтому здесь надо просто брать куичные логарифмы, и вся наука сохраняется дословно. Ну а если с двоичными логарифмами, то вот такие энтропии делятся на логарифм Q. На логарифм Q. Ну вот смотрите, все банально. Дело в том, что когда вы кодируете куичным алфавитом, у вас емкость алфавита в Q раз больше. У вас каждая буква переносит логарифм кубитов. Поэтому на гагарифм Q среднюю длину надо умножить. Короче, либо энтропию берете по основанию Q, либо ее делите на гагарифм Q. Это одно и то же. Ну и, стало быть, вот такие границы получаются. Доказательства один к одному повторяют то, что было ничего тут нового. Когда вы начинаете строить Q-ичный код, у вас возникнет проблема. Проблема с последним уровнем. То есть если алгоритм Хаффмана пытаться применять для построения куичного кода, то все то же самое, за исключением последнего уровня. Вот смотрите, давайте посмотрим на куичное дерево. Вот у него есть вершина. Из нее выходит обязательно Q путей. Вот путь номер один, вот путь номер Q. Так что на первом уровне у меня Q вершина. Записал это дело вот тут. я начинаю переходить на следующий уровень что происходит я вот этот уровень еще он у меня исчезает вот этот узел зато появляется узлов вот тут вот их у штук то есть каждый переход на верхний уровень добавляет кузов исключает один узел так что добавляется что-то типа q минус 1 И вот таких добавок может быть сколько угодно. Тут надо поставить произвольное число s. Вот это стало быть n от q. Значит, когда вы строите полное дерево, то есть используете все возможные пути, то количество концевых вершин в нем не может быть любым числом, а должно быть числом из этого ряда. Q плюс S на Q минус 1, где S любое число. Вот. Ну вот смотрите, получается так, что когда Q равно 2, то тогда это число как раз есть 2 плюс S. И может быть вообще любым. Поэтому в двоичном дереве на последнем уровне обязательно две вот этих вершины. В Q-ичном дереве это не так. Дело в том, что среди этих... Ну вот смотрите, как оно выглядит. Когда даже вот Q равно 3. Вот Q равно 3. Тогда вы имеете тут 3 плюс S умножить на 2. То есть у вас количество концевых вершин в полном дереве выражается 3 плюс 2S. То есть числа 5, 7, 9 и так далее. Не любые числа. Поэтому не при любом размере вовита вы можете построить полное дерево. У вас просто количество концевых вершин дереве не может быть любым выбирается вот из этой прогрессии вот ну вот создает некие проблемы с построением оптимального дерева на самом верхнем уровне значит рецепт тут такой если вам задали алфавит мощности м и эта мощность не представляется в виде q плюс с на q минус 1 Ну вот, скажем, в моем этом случае, скажем, алфавит мощности 6, и число 6 среди вот этих чисел не представлено. Здесь числа идут 5, потом 7 следующее. Шестерка выпала. Значит, что делать? Значит, надо просто тупо добавить в алфавит нужное число символов, вот до этого, до числа этого типа, и присвоить им нулевые вероятности. Только и всего. Это решает проблему. То есть вы расширяете алфавит, Добавляя туда символы, вплоть до числа такого типа, приходится добавить немного. У вас идут эти самые допустимые числа с шагом s, с шагом q-1. 1 на q-1, потом 2 на q-1. Ну вот, значит, попало у вас число М какое-то вот сюда, вот сюда. Ну, стало быть, вы вот столько добавите, чтобы попасть вот на эту границу. И у вас число станет нормальным, и у вас будет существовать приятное полное кодовое дерево. Ну, тогда на первом этапе вы просто смотрите, у вас есть... Вы сколько-то добавили символов с нулевыми вероятностями, вы их и используете на первом этапе. У вас на первом этапе, вот, вы упорядочите свои вероятности по... возрастанию p1, p2, pмт. Возьмете q самых маленьких вероятностей и объедините их, вот q узлов, в один узел. Вот так. Вот q узлов. При этом в этот список сразу попадут все нулевые вероятности, которые вы добавили. И у вас окажется, что на верхнем ярусе некие точки имеют нулевую вероятность. Их просто надо выбросить. И там останется вот этот... есть у вас все вершины будут полные содержать полное число литвей а вот это единственная вершина в кодовом деле будет на верхнем уровне неполное вообще вообще говоря в ней будут 2 2 пути с нулевыми вероятностями которых реально нет они виртуальные то есть сюда вы никогда не пойдете вот так вот что еще по поводу хаффановского кодирования сказать значит смотрите Вещь эта весьма полезная, но она предполагает, что у меня мощность этого источника достаточно велика. Тогда вот эти границы, что L больше H, но меньше H плюс 1, они становятся довольно тесными, хорошими. Вот я вам приводил пример. Когда мощность порядка 2 в 20-й, то энтропия будет порядка 20-ти. Вот эта добавка единицы, она уже не такая большая. 2 в 20-е — это порядка миллиона мощность алфавита. Чтобы хорошо пользоваться этими хаффмановскими методами сжатия, нужна обширная статистическая модель. Давайте займемся проблемой сжатия текста. Количество букв в алфавите оценим где-нибудь порядка 30-ти. Я могу построить одномерное распределение вероятностей букв. П от А, П от Б, П от С. Оно хорошо известно для русского, для английского алфавита. И строить оптимальный код. К сожалению, алфавит не слишком велик. Много символов, и код будет короткий. То есть тут... Энтропия английского языка на символах составляет 2,7 бита, а вот эта поправка 1 бит. Поэтому, видите, оно сопоставимо. Поэтому захочется улучшить результаты. Тогда вы можете как поступить? Вы можете в качестве букв фавита рассматривать двухбуквенные комбинации. А, Б, потом С, Д. Всевозможные пары букв. А, С, А, Д, Д, Е. И значит, у вас получается уже 30 в квадрате. Это уже приличное количество. Вот 30 с комбинацией на координату, 30 в квадрате. Это, между прочим, 9 900 уже вариантов. То есть вы можете эмпирически построить распределение двухбуквенных символов в книжке, распределение вероятностей, оценить его, ну и потом строить кодовый словарь уже на такое количество кодовых слов. уже будет поприличнее энтропия и стала быть степень сжатия который обеспечит будет лучше потом вы можете не ограничиваться этим опередить и кто как будет которых буквенника по и бисе и будет еще на 30 в 30 раз больше это будет уже 27 на 10 в клубе вот потом вы можете изучать 4 буквенные буквы комбинации это будет еще больше и строить хаффмановские коды для этого всего Значит, упрётесь вы в простую проблему. Значит, для того, чтобы код Хаффмана построить хороший, вам нужна нормальная статистика вот на этом множестве. Вам нужны оценки вероятностей П, Житра. Их вам кто-то должен сказать. Их вам никто не скажет. Вам придётся их мерить экспериментально. То есть вы должны взять книжку и, значит... рассчитать количество вхождения в нее всяких букв вот и на то и на б и поделить это на общее число букв в книжке вы получите эмпирические оценки вероятности этих букв ну вот для одной буквы довольно легко проделать оценки вероятности будут хорошие для двух букв уже тяжелее много вариантов а вот для слова еще трех букв уже будет некая проблема дело в том что количество четырех буквенных комбинаций Так велико, что очень может оказаться, что в книжке каких-то комбинаций не встретится. И тогда вы будете иметь плохие оценки вероятности этих комбинаций. Короче, ребятки, с увеличением, значит, Хафмановское кодирование хорошо себя ведет, идеально, при очень большом объеме алфавита вот этого. То есть, когда вы пытаетесь кодировать что-то объемное, так, чтобы у вас было много битов на символ, на один символ этого алфавита. Тогда получается хорошее сжатие. Но для того, чтобы этого сжатия достичь, вы должны иметь статистическую модель этого источника. То есть эмпирические экспериментальные оценки вероятностей, которые становятся получать все труднее с ростом мощности алфавита. Поэтому вот в этом слабое место этих методов кодирования. Дальше эмпирический факт такой. Вот смотрите. Если говорить о хаффмановском кодировании, то за счет чего достигается сжатие-то? А сжатие достигается вот за счет чего. Вот если понаблюдать реально за тем, что получается, то получается так, что у вас оптимальный код Хаффмана, он уравновешен почти везде. То есть, когда вы идете от корня, то у вас полная вероятность вот на этой ветке висящая, тут много чего висит. Вот какие-то пути, какие-то слова, и в них какие-то вероятности. Если просуммировать вот эти вероятности, висящие на этой ветке, и просуммировать вероятности, висящие на этой ветке, то в хорошем коде они почти одинаковы. То есть одна вторая сюда и одна вторая сюда. То есть хороший код уравновешен. А поэтому в выходном потоке битовом У вас вероятность нуля и вероятность единицы одинаковы почти. p от нуля равно p от единицы равно 1 и 2. Всякое... Вот тогда этот код оптимальный, он дает максимально возможную степень сжатия. Выходной поток у него — это поток равновероятных независимых битов. То есть вот код очень хорош, когда у него в любом месте... Идет дихотомия, деление пополам. Вероятность пойти сюда 1, 2, вероятность пойти сюда 1, 2. Вот тогда каждый новый бит в выходном потоке битовым переносит максимальную энтропию. Один бит на двоичный символ. Иначе, как только у вас дерево оказывается... Все отклонения от оптимальности, о которых идет речь, связаны с этим, что у вас средняя длина... меньше энтропии плюс что-то там она бы будет бывает побольше энтропии на единичку или около того это все эффекты неравновесности у вас будет какая-то вероятность побольше какая-то поменьше вот совсем плохо если это 1 4 вероятность а это 3 4 ну вот тогда стало быть энтропии в этом месте энтропии этого бита могла то есть он Принимает значение 0 с вероятностью такой, единица с вероятностью такой, переносит магоэнтропию. И вы теряете в средней длине за счет того, что выходной поток у вас не оптимален. Ну вот так. Так что, чем больше у вас объем алфавита, тем вот эта оптимальная процедура Хаффмана лучше его уравновешивает. Совсем уравновесить его не удается, поэтому какая-то избыточная длина остается. То есть средняя длина все-таки чуть побольше энтропии оказывается. Это исключительно за счет того, что у вас дерево не всегда уравновешено. Ну вот хорошие деревни, вот я вам все рисовал, вот хорошо строится дерево, скажем, для 1,4, 1,8. Ну чего там еще? Два раза по 1,16. И 1, сколько тут? 1,4. 2,4. Ну, еще 1,2 надо. Вот тут все вероятности степени двойки. И поэтому дерево будет очень хорошо уравновешено. Смотрите, я объединяю вот эти две, потом объединяю вот эти две, потом объединяю вот это, а потом объединяю вот это. Вот смотрите, у меня в каждом месте вероятность вот это 1,2, вероятность это 1,2. Тут вот эта вероятность 1,4 и эта вероятность 1,4. Вот в чем суть. То есть в каждой ветке у меня идет дихотомия, деление вероятности пополам. Поэтому каждый бит информационный, вот этот 0,1, 0,1, он переносит максимально возможную энтропию. Он является выборкой из последовательности случайных независимых битов. Собственно говоря, вот сжатие данных, оно и сводится к тому, что вы, источник, пытаетесь перекодировать Белый шум. Случайный битовый поток, в котором символы равновероятны и независимы. Тогда вот это хорошая предельная степень сжатия. Ну вот так. Ну ладно, все у меня тогда сегодня на этом. Задачку я вам дал на Хаффмановское кодирование. Она у меня решена.