Лекция 11

Приближенный подсчет количества наборов, выполняющих формулу в ДНФ

(Конспект: У. Тюлякова)

Будем искать количество наборов, выполняющих ДНФ. Ранее приводился вероятностный алгоритм. Хотелось бы уменьшить количество случайных битов, а в идеале от них избавиться.

Случайные переменные, независимые по k штук. Нам понадобятся случайные переменные независимые по несколько штук. Это объясняется тем, что если много случайных переменных совсем независимых, то получается большое вероятностное пространство, а если переменные независимы по несколько штук, то вероятностное пространство можно сделать гораздо меньше.

Общая конструкция: Сконструируем случайные переменные x_i ($i \in [0..2^q-1]$), независимые по k штук, из $F = [0..2^q-1]$. В качестве вероятностного пространства P возьмем множество всех полиномов степени не более k с коэффициентами из F. Для того, чтобы найти значение случайной переменной x_i, берем случайный полином из этого множества и вычисляем его значение (по модулю 2^q) в точке i: $x_i = p(i)$ (где $p \leftarrow random(P)$).

Лемма 11.1. *Существуют эффективно конструируемые вероятностные пространства размера 2^{kq} для 2^q переменных, равновероятно принимающих значения $0..2^q-1$ и независимых по k штук.*

Упражнение 11.1. Доказать лемму 11.1, используя построенное выше вероятностное пространство.
Указание. Чтобы вычислить вероятность события $\{p(i_1) = c_1, \dots, p(i_{k'}) = c_{k'}\}$, достаточно посчитать количество наших полиномов, удовлетворяющих этим уравнениям. Можно воспользоваться следующим: если $p(i_1) = c_1$, то i_1 — корень многочлена $p(x)-c_1$. Значит, многочлен $p(x)-c_1$ можно нацело поделить на $x - i_1$ итд.

Субэкспоненциальный алгоритм для общего случая. Нужно выяснить количество наборов, выполняющих формулу $\Phi = (a \wedge b \wedge c) \vee \dots$ в ДНФ. Пусть будет n переменных, m клозов. Достаточно найти вероятность $P\{\Phi(x) = 1\}$, где x — вектор, равномерно распределенный на $[0,1]^n$ (т.е. n независимых случайных битов). Детерминированно это сделать трудно, поскольку придется перебирать экспоненциально много x.

Возьмем x из другого вероятностного пространства и покажем, что вероятность от этого не слишком изменится. Именно, будем искать $P\{\Phi(G(y)) = 1\}$, где G — некий генератор случайных чисел, который переводит короткие случайные вектора в длинные псевдослучайные,

$$G \;:\; \{0,1\}^l \longrightarrow \{0,1\}^n.$$

Покажем, что

$$|Pr\{\Phi(x) = 1\} - Pr\{\Phi(G(y)) = 1\}| \leq \epsilon.$$

G будет выглядеть следующим образом: $G(y) = (s_1, \dots, s_n)$, где

$$s_i = \bigoplus_{j\in S_i}^{(r)} y_j, \qquad S_i = y_j | T_{ij} = 1.$$

Здесь T — матрица $n \times l$, которую мы построим позже, y — короткая случайная строка, требуемая нашему генератору, значок $^{(r)}$ означает, что суммируются только первые r битов множества S_i, а если $|S_i| < r$, то выдается 0.

Тем самым, мы берем для суммирования некоторые биты строки y, свои для каждого s_i; какие – определяется матрицей T: если $T_{ij} = 1$, "берем y_i для s_i"; если $T_{ij} = 0$, "не берем y_i для s_i".

Матрицу T строим так: $T_{ij} = 1$ с вероятностью $2r/l$, и $= 0$ с оставшейся с вероятностью $1 - 2r/l$ (т.е. в среднем в S_i входит $l \cdot 2r/l = 2r$ битов строки y из l возможных). При этом обеспечим, чтобы T_{ij} были независимы по $2d$ штук.

Как построить необходимое для этого вероятностное пространство? В лемме 11.1 мы научились брать случайные переменные независимые по k штук, но они равномерно распределены от 0 до $2^l - 1$. Надо перейти к 0 и 1 и не совсем равномерному распределению. Как получить 1 с вероятностью $1/c$? Для простоты, пусть c — степень двойки. Можно просто считать, что если выпал 0, то выдать 1, а если выпало 1 .. 2^l-1, то 0. (Если c — не степень двойки, то исходы $(c-1)$ .. (2^l-1) будем игнорировать.) Новые переменные будут также независимы по k штук. Возьмем $k = 2d$, $q = \log(ln)$; этого достаточно для построения матрицы T. Тогда размер полученного вероятностного пространства $2^{kq} = 2^{2d\log(ln)} = (nl)^{O(d)}$. Запомним, что тем самым вероятностное пространство для полученных T_{ij} имеет мощность $(nl)^{O(d)}$, а для y_i — можность 2^l.

Зафиксируем параметры

$$\begin{aligned} k &= d = \log(3nm/\epsilon), \\ r &= d^2, \\ l &= 24kd^3, \end{aligned}$$

где ϵ — точность, с которой хотим найти приближение. Подставим эти параметры; мощность используемого нами вероятностного пространства тем самым оказывается $2^{O(\log^4(3nm/\epsilon))}$. Т.е., нам понадобится $O(\log^4(3nm/\epsilon))$ случайных битов или время $2^{O(\log^4(3nm/\epsilon))}$ для их детерминированного перебора. Остается показать, что приведенный алгоритм действительно находит приближение с аддитивной погрешностью не более ϵ.

Доказательство оценки погрешности в общем случае. На самом деле слегка модифицируем алгоритм. Прежде чем его применить, выкинем клозы размера $> \log(n/\epsilon)$. Такой клоз могут выполнять не более $2^{n-\log(n/\epsilon)}$ наборов. Следовательно, вероятность того, что набор выполняет формулу, мы увеличим не более, чем на $2^{n-\log(n/\epsilon)}/2^n = 2^{-\log(n/\epsilon)} = \epsilon/n$, а всего клозов не более чем n, следовательно, всего погрешность $\leq \epsilon$ (конечно, это другой ϵ, поменьше ...).

Лемма 11.2. *С вероятностью не менее $1-\epsilon$ (снова другой ϵ), для всех i и j*

1. $|S_i| \geq r$,

2. $|S_i \cap \bigcup_{t\in C_j,\ t\neq i} S_t| < d$, *где $C_1, \ldots, C_m$ — все конъюнкции нашей формулы.*

Доказательство.

$$P\{|S_i \cap \bigcup S_j| \geq d\}$$

$$\leq \binom{l}{d}$$

$$\cdot P\{\text{некоторое множество } A \text{ мощности } d \text{ содержится в } S_i \text{ и в } \bigcup S_j\}$$

$$\leq \binom{l}{d} \left(\frac{2r}{l} \cdot \frac{2r}{l} \log(m/l)\right)^d.$$

(В последнем неравенстве использовано, что мы выбросили конъюнкции длиннее $\log(m/l)$.) Теперь это надо домножить на mn (столько у нас j и i) и, подставив значения для l, r, d, убедиться, что получилось $<< \epsilon$.

Далее,

$$nP\{|S_i| < r\} \leq nP\{||S_i| - 2r| > r\} \leq n \cdot \frac{\mathbf{E}||S_i| - 2r|^{2d}}{r^{2d}}$$

по неравенству Чебышева для $2d$-го момента. Величины назависимы по $2d$, так что все равно, как подсчитать момент порядка $2d$.

Задача 11.1. Посчитать этот момент и убедиться, что это мало.

□

Убедимся, что s_i практически неотличимы от случайных: предположим обратное (что вероятность у нас получилась иная, чем с "совсем случайными" битами) и покажем, как вычислить функцию четности. Это будет противоречить следующему факту:

Факт 11.1 (Boppana–Håstad). *Никакая формула F в КНФ (или ДНФ) с клозами длины $\leq t$ и n переменными не может вычислить функцию четности лучше, чем $2^{-n/t}$; более точно,*

$$|P\{F = \bigoplus\} - P\{F \neq \bigoplus\}| \leq 2^{-n/t}. \tag{11.1}$$

Замечание 11.1. Точно вычислить – значит, получить (11.1) не $\leq 2^{-n/t}$, а, напротив, единицу. Если F отношения к четности не имеет, то левая часть (11.1) равна нулю. Факт утверждает, что если дизъюнкции не слишком длинные, то F совершенно не похожа на четность.

Итак, предположим, что при использовании псевдослучайных чисел формула стала выдавать другой результат, т.е. (НУО)

$$P\{\Phi(x_1, \dots, x_n)\} - P\{\Phi(s_1, \dots, s_n)\} > \epsilon \tag{11.2}$$

(скорректировав ϵ, можем считать, что это на тех матрицах T, для которых пункты в лемме 11.2 выполняются). Будем потихоньку менять один набор на другой: $(x_1, \dots, x_n) \to (s_1, x_2, \dots, x_n) \to (s_1, s_2, x_3, \dots, x_n) \to \dots \to (s_1, \dots, s_n)$ и перепишем (11.2) как сумму разностей на соседних наборах. Рассмотрим шаг k, где соответствующая разность вероятность больше всего. Из соображений среднего это означает, что

$$P\{\Phi(s_1, \dots, s_{k-1}, x_k, \dots, x_n)\} - P\{\Phi(s_1, \dots, s_k, x_{k+1}, \dots, x_n)\} \geq \epsilon/n.$$

Научимся вычислять s_k (а тем самым, функцию четности от r переменных). Начнем с того, что вычислим его вероятностно как функцию от предыдущих s_i. Возьмем формулу Φ и подставим в нее $s_1, \dots, s_{k-1}$, а остальные переменные — случайные биты. Если

$$\Phi(s_1, \dots, s_{k-1}, x_k, \dots, x_n) = 1,$$

выдаем $1 - x_k$; если $\ldots = 0$, выдаем x_k.

Лемма 11.3. *Тем самым s_k вычислено верно с вероятностью $\geq \epsilon/n$. (Вероятность здесь не только по выбору случайных битов $x_k, \dots, x_n$, но и по выбору y_i.)*

Доказательство. Доказать лемму 11.3. □

Чтобы получить противоречие с фактом 11.1, нам надо не просто вычислить s_k (тем более вычислить вероятностно), а построить формулу в ДНФ (или КНФ) от его аргументов (это r штук y_i), значение которой достаточно часто совпадает с s_k.

Во-первых, избавимся от использования случайных битов $x_k, \dots, x_n$, а также тех y_i, от которых s_k не зависит, — просто зафиксируем их все наилучшим образом (нам ведь не надо эту формулу вычислять; достаточно показать, что она существует), вероятность, с которой мы вычисляем s_k правильно, не уменьшится. Далее, построим формулу в ДНФ или КНФ от $s_1, \dots, s_{k-1}$: это будет просто формула Φ или $\neg\Phi$ (в зависимости от значения x_k, которое мы зафиксировали). Теперь подставим вместо $s_1, \dots, s_{k-1}$ их значения на y. Заметим, что по лемме 11.2 они зависят от небольшого количества y_j, от которых зависит s_k, так что их можно вычислить по таблице истинности и записать в ДНФ (или КНФ) с небольшими клозами (длины не больше, чем количество y_j, от которых они еще зависят). Раскрыв скобки (чтобы формула вновь была в ДНФ или КНФ), получим клозы длины не более d (по лемме 11.2 биты s_t, входившие в один клоз C_j исходной формулы, зависели все вместе не более, чем от d битов строки y, от которых зависел s_k).

Получили противоречие с фактом 11.1: клозы у нас длины $\leq d$; значит, построенная нами формула могла бы вычислить четность не лучше, чем
$$2^{-n_{\text{факта}}/t_{\text{факта}}} = 2^{-r/d} = 2^{-d} = 2^{-\log(3mn/\epsilon)} = \frac{\epsilon}{3mn}.$$
А у нас получилось не менее ϵ/n.

Резюмируем. В формулу подставляли псевдослучайные числа. В какой-то момент от добавления s_k все портится. Тогда вычисляем s_k. Простроили алгоритм, зависящий от $s_1, \ldots, s_{k-1}$. Соответствующую формулу записали. Но эта формула от s_i. Чтобы заменить s_i на исходные y_k, каждое s_i записываем через y_i по таблице истинности, после чего дизъюнкции окажутся короткими по лемме 11.2.

Полиномиальный алгоритм для формул с “короткими” конъюнкциями. Если конъюнкции не слишком длинные, то это все можно сделать за полиномиальное время. Именно, уменьшим количество конъюнкций, а потом применим предыдущий алгоритм.

Определение 11.1. Назовем m-ромашкой m множеств F_i $(i \in I)$ с общим центром, т.е. $\forall i, j \in I\ F_i \cap F_j = \bigcap\limits_{i \in I} F_i$.

Лемма 11.4 (Erdös–Rado). *Пусть $F = \{F_i | i \in J\}$, $\forall i \in J\ |F_i| \leq t$, $|F| > t!(m-1)^t$. Тогда среди множеств F_i имеется m-ромашка.*

индукция по t. База $(t = 1)$ тривиальна. Если есть m попарно не пересекающихся множеств F_i, то это m-ромашка с пустым центром. В противном случае возьмем максимальное количество попарно дизъюнктных $\{F_i | i \in I,\ |I| = m'\}$; очевидно, $m' \leq m - 1$. Каждое размером $\leq t$. Следовательно, их объединение U содержит $\leq t(m-1)$ элементов и пересекается со всеми множествами F_i $(i \in J)$. Согласно принципу Дирихле, в U имеется элемент x, который принадлежит очень многим множествам F_i, а именно, $\geq t(m-1)^t/(t(m-1)) = (t-1)!(m-1)^{t-1}$ множествам. Выкинем его из этих множеств. Получили $(t-1)!(m-1)^{t-1}$ множеств мощности $\leq t - 1$. По предположению индукции, среди них имеется m-ромашка. Тогда вернем x в каждое из множеств этой ромашки и получим m-ромашку в исходном F. □

Имеется m дизъюнкций (соответствующие им множества переменных будут нашими множествами), n переменных, размер дизъюнкций $\le t$. Предположим, что в формуле много конъюнкций, а именно $> t!\lceil 2^{2t}\ln(m/\epsilon)\rceil^t$ конъюнкций, т.е. у нас имеется $> t!\lceil 2^t\ln(m/\epsilon)\rceil^t$ соответствующих им множеств переменных. Следовательно, по лемме 11.4 существует ромашка из $\ge \lceil 2^t\ln(m/\epsilon)\rceil$ множеств. Центр C ромашки — это некое множество переменных. Надо расставить отрицания так, чтобы получился клоз, имеющий достаточное число продолжений в исходной формуле (продолжение для $(x_1\wedge\neg x_2)$ — это, например, $(x_1\wedge\neg x_2\wedge\neg x_3\wedge x_4\ldots))$. Пусть $|C|=i$. Имеется 2^i способов расставить отрицания; значит, из соображений среднего имеется $\ge 2^{t-i}\ln(m/e)$ продолжений. Заменим их все на наш C с расставленными отрицаниями.

Заметим, что если на каком-то наборе значений переменных C ложный, то и все "лепестки" ложны (которые удалили). Если же C истинен, то хотя бы один из "лепестков" истинен с хорошей вероятностью. Именно, $P\{\text{все ложны}\}\le(1-2^{i-t})^{2^{(t-i)\ln(m/\epsilon)}}\le\epsilon/m$. Будем делать пока конъюнкций $> t!\lceil 2^{2t}\ln(m/\epsilon)\rceil^t$. Каждый раз удаляем хотя бы 1 конъюнкцию, так что будет не более m итераций.

Теорема 11.1. *Если конъюнкции не длиннее* $t=O(\log^{1/8}(mn))$, *а* $1/\epsilon=2^{O(\log^{1/4}(mn))}$, *то алгоритм заканчивает работу за полиномиальное время.*

Доказательство. После "обрывания лепестков" остается не более

$$m''=t!(2^{2t\ln(m/\epsilon)})^t\le\log^{O(\log^{1/8})}\cdot 2^{O(\log^{1/4})}\cdot\ln(m/\epsilon)^{O(\log^{1/8})}$$

конъюнкций (можно считать — равно как и переменных), где все недописанные log — это $\log(3mn/\epsilon)$. Самый большой из сомножителей $2^{O(\log^{1/4})}$; время работы общего алгоритма на такой формуле

$$2^{O(\log^4(m''n''/\epsilon))}=2^{O(\log^4(2^{O(\log^{1/4})}))}=2^{O(\log(mn))}=(mn)^{O(1)}.$$

$\square$