Вероятностные алгоритмы
Разделы 12 программы
Литература: [Кормен 2, §5 и дополнение C]
[Кормен 1, §6], [GL], [ДПВ], [К-Ф], [К-Ш-В]

Краткий конспект

В этом задании мы рассмотрим процедуры, использующие *рандомизацию*. Но мы не будем делать *никаких априорных предположений о входном распределении*,

Мы уже обсуждали как минимум три подобные процедуры (быструю сортировку, поиск медианы и проверку простоты), использующие бросание монетки (вероятностный алгоритм) и перейдем теперь к их формальному описанию. К сожалению, из-за недостатка времени мы успеем только ознакомиться с определениями и решить несколько задач. Но можно без преувеличения сказать, что вероятностные подходы к алгоритмам являются стержнем многих современных исследований[1]. Представить без них мир алгоритмов совершенно невозможно, так что всем заинтересованым лицам будет нелишне продолжить изучение этого подхода самостоятельно.

И основной лозунг, под которым оформлено это задание звучит так.

К возможности использования случайных битов нужно относиться как к дополнительному вычислительному ресурсу, позволяющему иногда существенно понизить трудоемкость и концептуально упростить процедуру.

Основным источником таких возможностей является (гипотетическая) возможность алгоритмического порождения "случайных объектов" в конкретных "универсумах", так сказать, "псевдослучайные генераторы". Мы будем использовать простейшие варианты: "орлянку", выбор случайного натурального числа на отрезке $[1, N]$, выбор случайного ребра в графе, выбор случайной плоскости, проходящей через начало координат и т.д. Практически это рутинные программистские операции, используемые часто рефлекторно. Хотя даже на практическом уровне серьезно обсуждаются вопросы о качестве этих генераторов.

В соответствие с идеологией нашего курса, в идеале, в каждом случае, когда нам нужно породить "случайный объект", мы должны указывать конкретную процедуру порождения, уметь проверять (доказывать), что она действительно порождает нужные объекты, и уметь оценивать ее трудоемкость. Хочу отметить, что эти вопросы достаточно тонкие и, более того, многие из них пока не имеют удовлетворительных ответов.

Вероятностная Машина Тьюринга (ВМТ) представляет из себя обычную МТ, которой в некоторых состояния разрешено совершать переходы в зависимости от бросания монеты. В отличии от недетерминированной МТ легко представить себе практическую реализацию такой конструкции. Более того, как утверждают некоторые апологеты теории вероятностей, иных устройств в природе просто не существует. Будем считать, что используются стандартные монетки для игры в орлянку[2], так что каждое вычисление ВМТ на входе x полностью определено, если считать (по аналогии с определением класса $\mathcal{NP}$), что одновременно с x на вход детерминированной МТ подается (вообще говоря, бесконечное) $\{0,1\}$-слово.

Как определить, что слово или язык принимается ВМТ?

В соответствии с определением ВМТ любое вычисление имеет некоторую вероятность.

Будем говорить, что язык $L \subset \Sigma^*$ принимается ВМТ M в *СЛАБОМ смысле [по стандарту МОНТЕ-КАРЛО]*, если для любого слова $x \in \Sigma^*$ вероятность получения ошибочного ответа на вопрос: ($x \in L$?) не превосходит $\frac{1}{3}$. Иначе говоря, если $x \in L$, то M с вероятностью, не меньшей $\frac{2}{3}$ ПРИНИМАЕТ x; а если $x \notin L$, то M с вероятностью, не меньшей $\frac{2}{3}$ ОТВЕРГАЕТ x.

Будем говорить, что язык $L \subset \Sigma^*$ принимается ВМТ M в *СИЛЬНОМ смысле [по стандарту ЛАС-ВЕГАС]*, если она дает *с вероятностью* 1 правильный ответ для любого слова $x \in \Sigma^*$. Это, в частности, означает, что M не может за конечное число шагов принять какое-нибудь слово $x \notin L$.

[1]Достаточно посмотреть на изменения, внесенных во 2-е издание Кормена.

[2]Монетки у которых, скажем, вероятность выпадения герба является каким-нибудь *невычислимым* числом, в принципе можно использовать, для распознавания невычислимых языков.

Языки, принимаемые ВМТ в СЛАБОМ смысле за *полиномиальное в среднем число шагов*, образуют класс $\mathcal{BPP}$.

Языки, принимаемые ВМТ в СИЛЬНОМ смысле за *полиномиальное в среднем число шагов*, образуют класс $\mathcal{ZPP}$.

Наконец, языки, для которых удается построить ВМТ, которая за полиномиальное в среднем число шагов принимает каждое слово из языка с вероятностью $\geq \frac{1}{2}$ и отвергает любое слово, не входящее в язык, образуют промежуточный класс $\mathcal{RP}$.

По построению, классы $\mathcal{BPP}$ и $\mathcal{ZPP}$ замкнуты относительно дополнения и $\mathcal{BPP} \supseteq \mathcal{RP} \supseteq \mathcal{ZPP} \supseteq \mathcal{P}$. В основном, мы будем изучать $\mathcal{BPP}$. Рекомендую почитать книгу Кузюрина и Фомина (гл. 4–5, §6.2) или книгу "Классические и квантовые вычисления". В последней, в разделах 1.3-1.4 дано определение класса $\mathcal{BPP}$, приведен вероятностный тест простоты Миллера-Рабина, показано, что $\mathcal{BPP} \subset \Sigma_1^p \cap \Pi_1^p$ (т.е. принадежит второму этажу т.н. полиномиальной иерархии). В разделе 12.2 построена вероятностная полиномиальная сводимость (что бы это значило?) вычисления дискретного логарифма (а это, что такое?) к задаче разложения числа на множители (задаче факторизации). Кроме того, сами квантовые вычисления являются аналогом вероятностных вычислений со специально определенным правилом вычисления вероятности. При этом, однако, оказывается, что квантовые компьютеры позволяют решать, например, задачу факторизации за полиномиальное время. Как известно, никто пока не знает, есть ли полиномиальный (детерминированный или вероятностный) классический алгоритм для этой задачи. Кроме того, никто пока не умеет решать на квантовом компьютере какую-нибудь NP-полную задачу за полиномиальное время.

Задача 1. $(0.01 + 0.01)$ Покажите, что класс $\mathcal{BPP}$ не изменится, если
(i) константу стандарта Монте-Карло $\frac{1}{3}$ заменить на любое число, строго меньшее $\frac{1}{2}$, а
(ii) **полиномиальное в среднем число шагов** заменить на **полиномиальное число шагов**.

Последняя задача позволяет дать определение класса $\mathcal{BPP}$ по аналогии с классом $\mathcal{NP}$.

Предикат L принадлежит классу $\mathcal{BPP}$, если существуют такие полином $q(\cdot)$ и предикат $R(\cdot,\cdot) \in \mathcal{P}$, что

$L(x) = 1 \implies$ *доля слов r длины $q(|x|)$, для которых выполнено $R(x,r)$, больше $2/3$;*
$L(x) = 0 \implies$ *доля слов r длины $q(|x|)$, для которых выполнено $R(x,r)$, меньше $1/3$.*

Иначе говоря, на вход недетерминированной МТ подается слово-вход x и слово-подсказка r, и x принимается, если и только если при некоторой (не слишком длинной) подсказке принимается пара (x,r). Соответственно, ВМТ читает слово-вход x, а роль слова-подсказки r выполняют результаты бросания монетки в процессе вычисления, причем слово x принадлежит языку, если пары (x,r) принимаются для фиксированной доли C_1 подсказок (бросаний монеты) и отвергается, если число допустимых пар (x,r) меньше некоторой фиксированой доли C_2. По определению, константы C_1 и C_2 должны иметь "зазор" $C_1 - C_2 > \varepsilon$. Если последнее требование опустить (т. е. положить $C_1 = C_2 = \frac{1}{2}$), то вычислительные возможности ВМТ неизмеримо возрастают, а класс распознаваемых на таких ВМТ языков называется $\mathcal{PP}$. В частности, $\mathcal{NP} \subseteq \mathcal{PP}$.

Проверка (полиномиальных) тождеств является одним из наглядных и убедительных примеров нетривиального использования вероятностных алгоритмов. В основе подхода лежит т. н. **Лемма Шварца- Зиппеля**[3].

[3]Смысл леммы в том, что если полином не равен нулю тождественно, то он не может слишком часто обращаться в нуль, например, в точках целочисленной решетки. Это утверждение известно как Schwartz-Zippel Lemma. На самом деле, Шварцу, видимо, принадлежит вероятностная интерпретация, поскольку сам факт давно известен (см., например, главу "Сравнения" в книге Боревича и

Пусть $f(x_1,\ldots,x_n)$, не равный тождественно нулю полином степени не выше k по каждой переменной[4], и пусть принимающие целые значения случайные величины $\xi_1,\ldots,\xi_n$ независимо и равномерно распределены на отрезке $[0,N-1]$.

Тогда $Prob\{f(\xi_1,\ldots,\xi_n)=0\}\le\frac{kn}{N}$.

Это утверждение мы обсудим на лекции, и на эту тему будут задачи как в ближайших контрольных, так и в финальном тесте.

Доказательство леммы *проводится индукцией по числу переменных n. Утверждение верно при $n=1$, поскольку нетривиальный полином степени k имеет не более k корней. При $n>1$ разложим f по переменной x_1: $f=f_0+f_1x_1+\ldots+f_tx_1^t$, где полиномы $f_0,\ldots,f_t$ не зависят от x_1, а f_t не равен нулю тождественно. Тогда по формуле полной вероятности $Prob\{f=0\}=Prob\{f=0\mid f_t=0\}Prob\{f_t=0\}+Prob\{f=0\mid f_t\ne0\}Prob\{f_t\ne0\}\le Prob\{f_t=0\}+Prob\{f=0\mid f_t\ne0\}$. Первый член оценивается по индуктивному предположению, а второй — не больше, чем $\frac{k}{N}$, поскольку на каждом отрезке $[(0,\xi_2,\ldots,\xi_n),(N-1,\xi_2,\ldots,\xi_n)]$ полином $f=f_0+f_1x_1+\ldots+f_tx_1^t$ с ненулевым старшим коэффициентом f_t может иметь не более $t\le k$ корней, так что $Prob\{f=0\}\le\frac{k(n-1)}{N}+\frac{k}{N}$.*

Задача 2. (4×0.01) Проверьте матричное равенство $C=AB$, где A,B,C– $n\times n$ матрицы, имеющие целочисленные элементы, не превышающие по абсолютной величине h, используя рандомизацию.

Пусть x— случайный n-мерный вектор, компоненты которого независимые целые числа, равномерно выбранные из интервала $[0,1,\ldots,N-1]$. Проверка равенства состоит в вычислении $A(Bx)=Cx$: если это равенство справедливо, то вы предполагаете, что исходное равенство верное, иначе вы сигнализируете об ошибке. Заметим, что каждую такую проверку можно выполнить за $O(n^2)$ операций над $O(\log(nh^2))$-разрядными числами, а любой сигнал об ошибке говорит о том, что исходное равенство неверное. С другой стороны, если проверка прошла успешно, то возможно, что исходное равенство неверное, но мы неудачно подобрали тестовый вектор x.

(i) Каким нужно выбрать N, чтобы вероятность ошибки вашей процедуры была меньше заданной вероятности p?

(ii) Тот же вопрос, если разрешается проводить несколько независимых проверок, а минимизировать нужно общую *битовую сложность* вычислений.

(iii) Сравните битовую сложность вероятностных процедур с стандартным детерминированным алгоритмом перемножения матриц для $n=10000;h=2^{15};p=0.001$

(iv) Для дальнейшей экономии вы решили использовать проверку $(A(Bx)x)=(Cx)x$ или проверку $(A(Bx)y)=(Cx)y$, где n-вектор y выбирается независимо от x и имеет те же характеристики. Как изменится для этих случаев N?

Язык 2-ВЫПОЛНИМОСТЬ состоит из выполнимых КНФ, в которых каждый дизъюнкт содержит не более двух литералов. Вы знаете, что задачу можно решать за линейное время, используя линейный алгоритм выделения сильно связных компонент в орграфах (об этом мы говорили на последней лекции). В этой задаче мы построим быстрый вероятностный алгоритм для проверки выполнимости 2-КНФ. Пусть 2-КНФ имеет n литералов и m дизъюнктов.

Алгоритм случайного поиска для языка 2-КНФ. Сначала всем переменным присваивается значение TRUE. На каждой итерации, пока формула невыполнима, берется произвольный невыполненный дизъюнкт, в нем равновероятно выбирается произвольный литерал и его логическое значение обращается. Этот процесс напоминает случайное блуждание и в принципе может продолжаться бесконечно (например, если взять невыполнимую КНФ). Однако для выполнимой 2-КНФ можно получить полиномиальные оценки среднего числа итераций. Дело в том, что *число отличий* между текущим набором логических значений переменных и их значениями в некотором произвольном (но фиксированном) выполняющем наборе (существующем по предположению) изменяется на каждой итерации с вероятностью $\frac12$ на 1. Таким образом, наш алгоритм можно интерпретировать как случайное блуждание на отрезке [0,1,. . . ,n].

Задача 3. $(0.04+0.01)$ (i) Покажите, что для выполнимой 2-КНФ среднее число итераций алгоритма (математическое ожидание числа итераций) равно $O(n^2)$.

(ii) Пусть пункт (i) справедлив (а больше ничего о языке 2-КНФ неизвестно). В какой из вероятностных классов, определенных выше, попадает тогда язык 2-КНФ?

Задача о минимальном разрезе в неориентированном графе $G=(V,E)$ заключается в том, чтобы разбить вершины графа на два дизъюнктных подмножества $(S,\bar S)$, $S\ne V$, $S\ne\varnothing$ так, чтобы минимизировать число ребер с концами в разных долях. Конечно, для ее решения можно применить потоковый алгоритм, но мы рассмотрим простую вероятностную процедуру. При этом основной будет операция стягивания ребра (кратные ребра остаются, а петли удаляются). Граф, полученный стягиванием ребра $(x,y)\in E$, обозначим $G/(x,y)$. Первоначальная идея заключается в следующем: при стягивании ребер величина минимального разреза не убывает (пока в графе остается не менее двух вершин), так что если стянуть все ребра $\{e_1,\ldots,e_p\}$, не входящие в минимальный разрез, то останется пара вершин, соединенная k ребрами, где k — величина минимального разреза в G. Остается понять, как часто реализуется подобная ситуация, если ребра стягиваются случайно.

Задача 4. $(2\times0.02+0.01)$ (i) Покажите, что вероятность того, что случайно выбранное ребро в графе входит в минимальный разрез не превышает $\frac{2}{|V|}$.

Из предыдущей задачи вытекает следующий вероятностный алгоритм определения минимального разреза:

MINCUT $[G(V,E),|V|=n]$

$G_0\leftarrow G$; $i\leftarrow0$;

while $|V(G_i)|>2$ **do**

В G_i выбираем случайное ребро e_i (с равномерным распределением на ребрах G_i).

$G_{i+1}\leftarrow G_{/}e_i$; $i\leftarrow i+1$;

end while

Комментарий. На выходе из цикла получаем (мульти)граф $\tilde G$, имеющий две вершины, соединенные ребрами, иногда отвечающими разрезу в исходном графе.

return Разрез в исходном графе G, отвечающий разрезу в $\tilde G$.

Времы работы алгоритма $O(n^2)$.

(ii) Покажите что MINCUT выдает минимальный разрез с вероятностью $\ge\frac{2}{n(n-1)}$.

(iii) Покажите, что если независимо повторить процедуру MINCUT n^2 раз, то минимальный разрез будет найден с вероятностью >0.85.

В следующей задаче мы покажем, что если привлечь дополнительные соображения, то можно понизить трудоемкость до $O(n^2\log^{O(1)}n)$. При этом алгоритм столь же прост и допускает параллелизацию.

Шафаревича "Теория чисел"), но ему не придавали вероятностной интерпретации.

[4]Лемма остается верной, если считать, что k — это суммарная степень по совокупности переменных.

Описанная выше процедура поиска минимального разреза последовательно выбирает случайные ребра и стягивает их концы до тех пор, пока в графе не останутся две вершины, соединенные (кратными) ребрами. Если при выборе случайных ребер мы ни разу не выбирали ребра разреза, то мы получаем ответ. Выше мы показали, что вероятность на i-м шаге выбрать ребро, входящее в разрез, равна $p_i = \frac{2}{n-i}$, отсюда вероятность того, что за i шагов не будет выбрано ни одно ребро, входящее в минимальный разрез (мы будем говорить, что выбранные *ребра не задевают разрез*), равна $P_i = (1-p_1)(1-p_2)\dots(1-p_i)$. Мы хотим ускорить алгоритм. Заметим, что чем больше ребер мы стягиваем, тем больше вероятность, что следующее выбранное ребро заденет минимальный разрез. Поэтому новая идея, которую мы хотим исследовать, заключается в том, чтобы стягивать ребра до какого-то порога, пока вероятность попадания в разрез еще достаточно мала, а дальше использовать рекурсию. Из формулы для P_i видно, что если стянуть $n/2$ случайных ребер, то они с вероятностью $\geq \frac{1}{4}$ не заденут минимальный разрез.

Блок-схема нового алгоритма приведена ниже. Процедура использует подпрограмму СТЯГИВАНИЕ (G,k), которая стягивает ребра до тех пор пока число вершин не уменьшится ниже порога k (при стягивании произвольного ребра число вершин уменьшается на единицу).

СТЯГИВАНИЕ (G,k)
for $i := n$ **downto** k
В G выбираем случайное ребро e
(с равномерным распределением
на ребрах).
$G \leftarrow G/e$
endfor
return G

МИН-РАЗРЕЗ (G)
if в G больше восьми вершин **then**
Повторить 4 раза процедуру
$X_1 \leftarrow$ МИН-РАЗРЕЗ [СТЯГИВАНИЕ $(G, \frac{n}{2})$];
$X_2 \leftarrow$ МИН-РАЗРЕЗ [СТЯГИВАНИЕ $(G, \frac{n}{2})$];
$X_3 \leftarrow$ МИН-РАЗРЕЗ [СТЯГИВАНИЕ $(G, \frac{n}{2})$];
$X_4 \leftarrow$ МИН-РАЗРЕЗ [СТЯГИВАНИЕ $(G, \frac{n}{2})$];
return $\min\{X_1, X_2, X_3, X_4\}$
else находим минимальный разрез вручную.

Задача Д–1. $(0.01+0.01+0.04+0.01)$

(i) Запишите рекуррентную оценку сложности $T(n)$ вычисления функции МИН-РАЗРЕЗ (G) для графа с $|V| = n$ вершинами.

(ii) Найдите Θ-асимптотику $T(n)$ (для этого нужно сначала оценить трудоемкость процедуры СТЯГИВАНИЕ (G,k)).

Оценим теперь с какой вероятностью $\mathbb{P}(n)$ алгоритм МИН-РАЗРЕЗ(G) выдает минимальный разрез для графа G с n вершинами. Вероятность успеха равна вероятности того, что хотя бы один рекурсивный вызов дал корректный ответ. Таким образом, получаем: $\mathbb{P}(n) = 1 - \left(1 - \frac{1}{4}\mathbb{P}(\frac{n}{2})\right)^4$, откуда следует, что $\mathbb{P}(n) \geq \mathbb{P}(\frac{n}{2}) - \frac{3}{8}\mathbb{P}(\frac{n}{2})^2$. Считая, что n является степенью двойки, обозначим $p_k = \mathbb{P}(2^k)$. По определению, p_k равно вероятности успеха, если потребовалось k рекурсивных вызовов процедуры МИН-РАЗРЕЗ. В частности, $p_0 = 1$.

Получаем рекурсию $p_{k+1} = 1-(1-\frac{1}{4}p_k)^4$. Откуда, если раскрыть скобки, следует: $p_{k+1} \geq p_k - \frac{3}{8}(\mathcal{P}_k)^2$.

(iii) Приведите как можно более точную оценку снизу рекурсии: $p_0 = 1; p_{k+1} = p_k - \frac{3}{8}(p_k)^2$ вида $p_k = \Omega(f(k))$.

Подсказка. Предположите, что $p_{k+1} - p_k \approx \frac{dp}{dk}$, и оцените порядок роста функции $p(k)$, а потом обоснуйте вашу гипотезу.

(iv) Получите оценку числа итераций модифицированно вероятностного алгоритма поиска минимального разреза, для получения заданной точности ε и приведите оценку общего числа операций.

Задача Д–2. $(0.01+0.03+0.01)$ **[Вероятностный алгоритм для языка ВЫПОЛНИМОСТЬ и дерандомизация].** Предположим, КНФ содержит m дизъюнктов и в каждый дизъюнкт входит ровно k литералов. Пусть X — случайная величина, равная числу выполненных дизъюнктов, если независимо и равновероятно приписать каждому литералу значения 0 или 1. Поскольку каждый дизъюнкт ложен лишь при одном значении литералов и матожидания суммируются, то математическое ожидание числа выполненных дизъюнктов равно: $E(X) = m(1-2^{-k})$.

Это значит, что существует логический набор, на котором выполнено не менее $E(X)$ дизъюнктов. Далее, как и в случае с задачами из $\mathcal{NP}$, возникает та же проблема. Мы знаем, что такой набор существует, но не знаем, как его найти, не используя полный перебор. В нашем случае, для нахождения искомого набора можно, во-первых, построить эффективный вероятностный алгоритм. И более того, можно провести дерандомизацию, т. е. конвертировать вероятностную процедуру в детерминированную (не сильно увеличивая сложность). Последнее возможно далеко не всегда, и одним из центральных вопросов теории сложности является соотношение между классами $\mathcal{P}$ и $\mathcal{BPP}$, т. е. верно ли, что всякую процедуру из $\mathcal{BPP}$ можно дерандомизировать?

Алгоритм нахождения набора заключается в следующем. Оценим вероятность события $p = Prob[X \geq m(1-2^{-k})]$ (т. е. что в КНФ *выполнено не менее* $m(1-2^{-k})$ *дизъюнктов*). Для этого распишем математическое ожидание: $E(X) = \sum_{i=1}^{m} i * Prob[X=i] \leq [m(1-2^{-k})-1](1-p) + mp$. Отсюда $p \geq (m2^{-k}+1)^{-1}$.

Такая оценка позволяет построить следующий вероятностный алгоритм для определения набора, выполняющего не менее $m(1-2^{-k})$ дизъюнктов: достаточно независимо повторить процедуру $\geq m2^{-k}+1$ раз и тогда с вероятностью $> \frac{1}{2}$ одна из попыток даст искомый набор.

(i) Будет ли предложенный алгоритм 1) Лас-Вегас алгоритмом; 2) Монте-Карло алгоритмом; 3) ни тем, ни другим?

Поскольку порождение случайного набора требует $O(n)$ операций, а проверка числа выполненных дизъюнктов требует $O(mk)$ операций, то одна итерация алгоритма занимает $O(m2^{-k}(mk+n))$

Теперь попробуем дерандомизировать процедуру.

Свяжем с каждым литералом x_i случайную величину Y_i, принимающую равновероятно значения 0 или 1, и будем считать $\{Y_i\}$, $i = 1,\dots,n$ независимыми. Значение величины Y_i будем обозначать маленькой буквой y_i. Мы используем т.н. метод *условных вероятностей*, который заключается в последовательном приписывании логических значений литералам так, чтобы на каждом шаге выполнялось неравенство: $E(X|y_1,\dots,y_j) \leq E(X|y_1,\dots,y_{j+1})$. Осуществить такой подход удается не всегда. В нашем случае ключевым является тождество[5]: $E(X|y_1,\dots,y_j) = \frac{1}{2}[E(X|y_1,\dots,y_j,Y_{j+1}=1) + E(X|y_1,\dots,y_j,Y_{j+1}=0)]$. Таким образом, нужно научиться *детерминированно* приписывать литералу x_{j+1} значение, имеющее *большее условное математическое ожидание*. Для этого нужно разбить множество дизъюнктов на 4 непересекающиеся подмножества [1) уже выполненных; 2) не зависящих от x_{j+1}; 3) выполняющихся при $x_{j+1}=1$; 4) выполняющихся при $x_{j+1}=0$], вычислить условные мат. ожидания и присвоить литералу x_{j+1} значение, отвечающее большему мат. ожиданию.

(ii) Закончите вычисления, т. е. явно укажите, какие истинностные значения следует присваивать литералам. При вычислении условных вероятностей не забывайте о вкладе переменных, значения которым **еще не присвоены**!

(iii) Проведите вычисления для 2-КНФ[6] $(\bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee$

[5] На самом деле, достаточно потребовать *квазивогнутости*: $E(X|\dots) \leq \max[E(X|\dots,Y_{j+1}=1), E(X|\dots,Y_{j+1}=0)]$.

[6] Контрольный вопрос: является ли указанная процедура поли-

$\bar{x}_6) \wedge (x_2 \vee \bar{x}_4) \wedge (\vee \bar{x}_5 \vee x_7) \wedge (\bar{x}_7 \vee x_8 \vee) \wedge (x_1 \vee \bar{x}_7)$

В заключение отметим, что по ходу рассуждений мы показали следующее полезное утверждение (которое полезно доказать каким-то иным способом): *всякая k-КНФ, имеющая меньше 2^k дизъюнктов, выполнима.*

Кроме того, *для случая 3-КНФ, каждый дизъюнкт которой содержит ровно 3 литерала, мы построили $\frac{7}{8}$-приближенный (детерминированный и вероятностный) полиномиальный алгоритмы для NP-трудной задачи MAX-3-SAT, в которой требуется выполнить максимальное число дизъюнктов*[7]. И в этом бы не было ничего удивительного, если бы J.Håstad не показал (это довольно тяжело), что при $\mathcal{P} \neq \mathcal{NP}$ **никакая** эффективная процедура для задачи MAX-ВЫПОЛНИМОСТЬ не может давать большую точность.

номиальным алгоритмом для 2-КНФ?

[7]Это, значит, что алгоритм находит набор, на котором выполняется не менее $\frac{7}{8}$ от максимально возможного числа дизъюнктов, которые можно одновременно выполнить.