Конспект к лекции 9. (Санкт-Петербург, 16 апреля 2017 г.)

21 Генератор Нисана

Генераторами псевдослучайных битов называют отображения, которые преобразуют сравнительно короткую *затравку* (в английской терминологии *seed*), состоящую из по-настоящему случайных битов, в довольно длинные «псевдослучайные» последовательности. При этом требуется, чтобы полученные псевдослучайные последовательности были в каком-то смысле похожи на настоящие случайные биты, т.е., на результат серии независимых испытаний с равными вероятностями нуля и единицы. Чтобы дать определение генератора, нужно уточнить класс «тестов», которые эти генераторы должны выдерживать. Другими словами, в определении генератора следует указать, для каких именно «наблюдателей» псевдослучайные последовательности должны казаться похожими на истинно случайные. Для наиболее важных и естественных уточнений такого определения утверждение о существовании (быстро вычислимых) генерторов псевдослучайных битов оказывается сильнее гипотезы $\mathrm{P} \neq \mathrm{NP}$. Однако некоторые содержательные варианты определения позволяют построить псевдослучайные генераторы без каких-либо недоказанных предположений. В этом параграфе мы изучим один из таких генераторов — генератор Нисана.

Ноам Нисан предложил конструкцию генератора псевдослучайных битов, которые выглядят почти неотличимыми от «настоящих» случайных битов для любого «наблюдателя», располагающего сравнительно небольшой (скажем, логарифмической) памятью. Это означает, в частности, что для всех вероятностных алгоритмов с логарифмической памятью в качестве источников случайности можно использовать псевдослучайные последовательности битов из генератора Нисана. Ниже мы опишем конструкцию Нисана (она использует экстракторы и косвенно — спектральные экспандеры) и докажем основные свойства этого генератора.

Лемма 21.1 (recycling lemma) *Для произвольного отображения*

$$f \ : \ \{0,1\}^n \to \{0,1\}^s$$

и для всякого экстрактора

$$\mathrm{Ext} : \ \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^n$$

с параметрами $(n, n, k = n - s - \log \frac{1}{\varepsilon}, t, \varepsilon)$ *выполняется*

$$\mathrm{dist}(f(X) \cdot Y, f(X) \cdot \mathrm{Ext}(X, Z)) < 2\varepsilon,$$

где X *есть случайная величина, равномерно распределенная на* $\{0,1\}^n$, Y *и* Z *независимы от* X *и также равномерно распределены на* $\{0,1\}^n$ *и* $\{0,1\}^t$ *соответственно,* $\mathrm{dist}(*,*)$ *обозначает статистическое расстояние между двумя распределениями, а «*$\cdot$*» обозначает конкатенацию.*

Отметим, что хотя формально эта лемма верна для произвольных значений параметров n и s, мы будем её использовать для $s \ll n$.

Доказательство: Для каждого $v \in \{0,1\}^s$ обозначим X_v условное распределение вероятностей на $\{0,1\}^n$, соответствующее событию $f(X) = v$. Другими словами, X_v есть равномерное распределение на строках из n битов x, для которых $f(x) = v$. Мы можем переписать интересующее нас статистическое расстояние в виде

$$\mathrm{dist}(f(X) \cdot Y, f(X) \cdot \mathrm{Ext}(X,Z)) =$$

$$= \tfrac{1}{2} \sum_{v,w} \Big| \mathrm{Prob}[f(X) = v \ \&\ Y = w] - \mathrm{Prob}[f(X) = v \ \&\ \mathrm{Ext}(X,Z) = w] \Big| =$$

$$= \sum_v \mathrm{Prob}[f(X) = v] \cdot \mathrm{dist}(Y, \mathrm{Ext}(X_v, Z)). \quad (*)$$

Теперь разделим все занчения w на «толстые» и «тонкие»:

$$Fat := \{v \ : \ \mathrm{Prob}[f(X) = v] \geqslant \varepsilon/2\}.$$

Заметим, что если $v \in Fat$, то распредление X_v имеет min-энтропию не меньше $k = n - s - \log \frac{1}{\varepsilon}$. А это значит (определение экстрактора), что $\mathrm{Ext}(X_v, Z)$ будет ε-близко к равномерному распределению $\{0,1\}^n$.

Теперь мы можем оценить оценить (*) следующим образом:

$$\sum_v \mathrm{Prob}[f(X) = v] \cdot \mathrm{dist}(Y, \mathrm{Ext}(X_v, Z)) =$$

$$= \sum_{v \in Fat} \mathrm{Prob}[f(X) = v] \cdot \mathrm{dist}(Y, \mathrm{Ext}(X_v, Z)) + \sum_{v \notin Fat} \mathrm{Prob}[f(X) = v] \cdot \mathrm{dist}(Y, \mathrm{Ext}(X_v, Z)) \leqslant$$

$$\leqslant \mathrm{Prob}[f(X) \in Fat] \cdot \varepsilon + \mathrm{Prob}[f(X) \notin Fat] \cdot 1.$$

Остаётся заметить, что суммарная мера прообразов «тонких» v невелика — не больше, чем

$$\mathrm{Prob}[f(X) \notin Fat] \leqslant \sum_{v \notin Fat} \frac{\varepsilon}{2^s} \leqslant 2^s \cdot \frac{\varepsilon}{2^s} = \varepsilon.$$

Таким образом, $\mathrm{dist}(f(X) \cdot Y, f(X) \cdot \mathrm{Ext}(X,Z))$ не превосходит 2ε.

Доказанную выше лемму можно понимать следующим образом. Если мы использовали n случайных битов X для каких-то промежуточных вычислений и смогли запомнить только s-битный результат этих вычислений, то теперь мы можем использовать те же случайные биты повторно; нужно лишь «обновить» значение X с помощью экстрактора Ext и небольшого числа свежих случайных битов Z. Например, если мы один раз использовали n случайных битов X для некоторых вычислений и получили результат из $s = O(\log n)$ битов, то мы можем пустить те же биты в дело повторно,

«восстановив» их случайность с помощью экстрактора из лекции 8, который потребует лишь $t = O(\log n)$ битов свежей случайности. Таким образом, вместо двух наборов по n независимых случайных битов мы можем обойтись почти вдвое более коротким набором из $n + O(\log n)$ случайных битов. При этом вероятность получить правильный ответ могла измениться лишь незначительно (на некоторое небольшое ε), так что замена «истинно случайных битов» на «псевдослучайные» корректна. Ниже мы покажим, как итерация описанного трюка позволяет заменить $\mathrm{poly}(n)$ случайных битов на затравку генератора длиной $O(\log^2 n)$.

Мы определим *генератор Нисана*

$$G_m \ : \ \{0,1\}^{mt} \to \{0,1\}^{2^m}$$

рекурсивно. Будем представлять себе вход генератора (строку из mt битов) в виде конкатенации $a \cdot b$, где длина a равна $(m-1)t$, а длина b, соответственно, t. Положим

$$G_m(a \cdot b) := \begin{cases} [\text{первые два бита входа}], & \text{если } m = 1, \\ G_{m-1}(a) \cdot G_{m-1}(\mathrm{Ext}_m(a,b)), & \text{если } m > 1 \end{cases}$$

(выбор экстрактора Ext_m мы уточним ниже).

Рассмотрим произвольный вероятностный алгоритм, который на входах длины n использует память размера $O(\log n)$. Для определенности можно считать, что *алгоритм* — это некоторая машина Тьюринга $\mathcal{M}$. В данном случае у машины есть входная лента (доступная только для чтения) и рабочая лента ограниченного размера (доступная для чтения и записи). Без ограничения общности можно считать, что в конце вычисления машина записывает ответ на рабочей ленте. Также мы предполагаем, что на каждом шаге машина получает один бит от датчика случайных чисел. В каждый момент времени конфигурация машины описывается следующим набором данных:

- содержание рабочей ленты,
- положение пишушей головки на рабочей ленте,
- положение читающей головки на входной ленте.

Если размер рабочей ленты ограничен $O(\log n)$ для входов длины n, то для каждого входа имеется $2^{O(\log n)}$ потенциально возможных конфигураций, и продолжительность вычисления ограничена некоторым числом

$$N = 2^{O(\log n)} = \mathrm{poly}(n)$$

(мы требуем, чтобы вычисление никогда не зацикливалось, так что число шагов не может быть больше числа конфигураций). Процесс вычисления

можно представлять себе как цепочку конфигураций машины. Если зафиксировать вход алгоритма, то соответствующая цепочка конфигруаций полностью определяется значениями, выдаваемыми датчиком случайных битов — на каждом шаге вычислений очередной случайный бит опредляет, в какую новую конфигурацию мы переходим из текущей.

Обозначим $\mathcal{M}_{v,T}(r_1 \dots r_T)$ конфигурацию, в которую мы попадем, начав вычисления в конфигурации v и сделав T шагов, получая от датчика случайных битов значения $r_1 \dots r_T$ (вход алгоритма мы по-прежнему считаем зафиксированным, так что в обозначение он не входит явно). Нас интересует распределение вероятностей

$$\mathcal{M}_{v_0,N}(r_1 \dots r_N)$$

для выделенной начальной конфигурации машины v_0 (состояние машины в начале работы) и случайных (или псевдослучайных) битов r_i. Следующая лемма показывает, что поведение машина на случайных и псевдослучайных (по Нисану) битах не очень сильно различается.

Лемма 21.2 (главная) *Пусть для некоторого t и для всех $m = 1, \dots, O(\log n)$ существует экстрактор*

$$\mathrm{Ext}_{m+1} \ : \ \{0,1\}^{mt} \times \{0,1\}^t \to \{0,1\}^{mt}$$

с параметрами $(mt, mt, k = mt - O(\log N) - O(\log \frac{1}{\varepsilon}), t, \varepsilon)$. Тогда для всех v

$$\mathrm{dist}(\mathcal{M}_{v,2^m}(U_{2^m}), \mathcal{M}_{v,2^m}(G_m(U_{mt}))) \leqslant 4^m\varepsilon,$$

где U_k обозначает равномерное распределение на булевом кубе $\{0,1\}^k$.

Замечание: в лекции 8 мы доказали существование экстракторов с требуемыми параметрами и $t = O(\log N + \log \frac{1}{\varepsilon})$.

Применение леммы: Пусть v_0 есть начальная конфигцрация нашей схемы вычислений и $m = \log N$. Если $4^m\varepsilon < 1/100$, то главная лемма гарантирует, что распределение вероятностей $\mathcal{M}_{v_0,2^m}(G_m(U_{mt}))$ отличается от распределения $\mathcal{M}_{v_0,2^m}(U_{2^m})$ не более, чем на $1/100$. Это значит, что вероятности получения данным алгоритмом правильного ответа на истинно случайных битах и на псевдо-случайных битах из генератора Нисана отличаются не больше, чем на 1%. Чтобы выполнялось неравенство $4^m\varepsilon < 1/100$, мы должны взять $\varepsilon = 1/(100 \cdot 4^m)$, так что $\log \frac{1}{\varepsilon} = O(\log n)$. Таким образом, длина входа («затравки») генератора Нисана оказывается равна

$$mt = O\left(\log N \cdot \left(\log N + \log \frac{1}{\varepsilon}\right)\right) = O(\log^2 n).$$

В качестве следствия мы получаем теорему о дерандомиации вычислений на логарифмической памяти:

Теорема 21.1 *Всякий вероятностный алгоритм, использующий для входов длины n память размера $O(\log n)$ и ошибающийся с вероятностью $\delta < 1/2$ можно переделать в детерминированный алгоритм, использующий память $O(\log^2 n)$.*

Доказательство теоремы: заменим случайные биты на псевдо-случайные выходы генереатора Нисана и вычислим вероятность ошибки алгоритма, перебрав все возможные значения «затравки» генератора (длины $O(\log^2 n)$). Отметим, что данное вычисление требует квазиполиномиального времени $2^{O(\log^2 n)}$.

Остается неизвестным, можно ли преобразовать всякий вероятностный алгоритм с логарифмической памятью в детерминированный *полиномиальный* алгоритм. В следующей лекции мы покажем, что для некоторых важных алгоритмов такая дерандомизация возможна (теорема Рейнголда).

Доказательство главной леммы: Мы докажем лемму по индукции. (Для шага индукции будет важно, что лемма сформулирована не только для начальной конфиграции v_0, но для произвольного состояния машины v).

База индукции тривиальная. Далее, для каждого $m > 1$ обозначим ε_m максимальное возможно занчение статистического расстояния

$$\mathrm{dist}(\mathcal{M}_{v,2^m}(U_{2^m}), \mathcal{M}_{v,2^m}(G_m(U_{mt})))$$

и покажем, что $\varepsilon_m \leqslant 2\varepsilon_{m-1} + 2\varepsilon$. Для этого мы рассмотрим четыре распределения вероятностей:

$\mathcal{D}_1$: U_{2^m}

$\mathcal{D}_2$: $U_{2^{m-1}} \cdot G_{m-1}(U_{(m-1)t})$

$\mathcal{D}_3$: $G_{m-1}(U_{(m-1)t}) \cdot G_{m-1}(U'_{(m-1)t})$ (штрих означает, что мы берём две независимые копии равномерного распределения на $\{0,1\}^{(m-1)t}$)

$\mathcal{D}_4$: $G_m(U_{mt})$

Нас интересует разница в поведении алгоритма на случайных битах, распределённых согласно $\mathcal{D}_1$ и $\mathcal{D}_4$. Для этого мы оценим различие между $\mathcal{D}_1$ и $\mathcal{D}_2$, затем между $\mathcal{D}_2$ и $\mathcal{D}_3$, и, наконец, между $\mathcal{D}_3$ и $\mathcal{D}_4$.

Шаг 1: по предположению индукции $\mathrm{dist}(\mathcal{M}_{v,2^m}(\mathcal{D}_1), \mathcal{M}_{v,2^m}(\mathcal{D}_2)) \leqslant \varepsilon_{m-1}$.

Шаг 2: аналогично получаем $\mathrm{dist}(\mathcal{M}_{v,2^m}(\mathcal{D}_2), \mathcal{M}_{v,2^m}(\mathcal{D}_3)) \leqslant \varepsilon_{m-1}$.

Шаг 3: неравенство $\mathrm{dist}(\mathcal{M}_{v,2^m}(\mathcal{D}_3), \mathcal{M}_{v,2^m}(\mathcal{D}_4)) \leqslant 2\varepsilon$ вытекает из recycling lemma. В самом деле, эта лемма гарантирует, что статистическое расстояние между

$$G_{m-1}(X) \cdot Y \text{ и } G_{m-1}(X) \cdot \mathrm{Ext}_m(X, Z)$$

(где X и Z равномено распределены на наборах нулей и единиц соответствующей длины) не превосходит 2ε. Остаётся заметить, что применяя к двум наборам (псевдо)случайных битов детерминированное отображение (в данном случае это функция $\mathcal{M}_{u,2^{m-1}}(\cdot)$), мы не увеличиваем статистическое расточние между распределениями. Лемма доказана.