Первый курс, осенний семестр 2020/21
Практика по алгоритмам #4

Невероятные истории

5 февраля

Собрано 5 февраля 2021 г. в 16:37

Содержание

Невероятные истории

1. **Понижение вероятности**

   Алгоритм работает за $\mathcal{O}(n^2 \log n)$, вероятность успеха $1/\log n$.
   За какое время можно добиться вероятности успеха $1 - 1/n$?

2. **Абсолютно надежный шифр**

   Шифруем n-битное сообщение `XOR`-ом со случайным равномерно распределенным n-битным ключом. Покажите, что результат – случайная равномерно распределенная n-битная строка.

3. **Монте-Карло**

   В квадратике $[0, 1] \times [0, 1]$ живут k кругов. Посчитайте площади их объединения ипересечения с ошибкой не более 10^{-2}.

   А если есть и круги, и прямоугольники? А в 3D (кубик и сферы)?

4. **Случайный элемент на миллион**

   Есть длинный файл из n элементов, его можно последовательно читать, но не хранить целиком. Требуется в конце вывести один из этих n элементов с равной вероятностью.

5. **Игра на дереве**

   Есть полное бинарное дерево глубины n. В листьях написаны числа 0 и 1. Двое играют в игру «спуск по дереву», ходят по очереди, первый хочет в 0, второй в 1. Кто выиграет? Решить за $(2 - \varepsilon)^n$.

6. **Random в задачах оптимизации**

   Пусть у нас есть алгоритм, который ищет VERTEX-COVER.
   На любом графе матожидание размера покрытия, которое выдаст алгоритм, равно $C \cdot \texttt{OPT}$.
   Как получить покрытие, которое с высокой вероятностью будет размера не более $C(1+\varepsilon)\texttt{OPT}$?

7. **Приближение** MAX-SAT

   Для MAX-SAT построить полиномиальный вероятностный алгоритм, ищущий набор переменных, обращающий хотя бы $\frac{1}{2}\texttt{OPT}$ клозов в истину.

8. **Числа в окрестности**

   Дан массив и число ε. Известно, что $\frac{2n}{3}$ элементов массива лежат в $[x, x + \varepsilon]$ для некоторого **неизвестного** x. За линейное время выбрать $\frac{n}{3}$ элементов из $[y, y + \varepsilon]$ для некоторого y.

9. **Простое в отрезке**

   Найдите простое число в диапазоне $[A, B]$.

10. **(*) Первообразный корень**

    Первообразный корень по модулю p – такое x: $\langle 1, x, x^2, \ldots, x^{p-2} \rangle$ различны. Дано p, найти x.

11. **(*) 4-SAT random walk**

    Адаптируйте уже известный вам алгоритм. Оцените время работы.

12. **(*) 3-Coloring**

    Предложите Random walk алгоритм для задачи 3-Coloring. Оцените время работы.

13. **(*) Пересечение полуплоскостей**

    Найдите за $\mathcal{O}(n)$ точку x в пересечении n полуплоскостей: $\langle c, x \rangle \to \max$.

14. **(*) Проверка ДЗ**

    Алгоритм проверки дз: проверить случайную половину присланных решений. Пусть обнаружено 0 ошибок, какова вероятность наличия ошибок/матожидание количества ошибок?

15. **(*) Экзамен за $\mathcal{O}(1)$**

    Преподаватель хочет принять у студента экзамен. Опытный преподаватель знает, что, если студент плохо готов, он плавает как минимум в половине билетов, а если студент хорошо готов, он знает не менее 90% билетов. У преподавателя есть время спросить k билетов. Придумайте алгоритм, позволяющий максимально точно различить два типа студентов студентов.

16. **(*) Ковид**

    Каждый человек или болеет ковидом, или нет. Взять биоматериал на анализы у человека – не проблема. А вот собственно сделать сам анализ – дорого и долго. Есть идея: за один раз можно взять биоматериал случайных x человек, смешать, и сделать 1 анализ, который покажет «болеет ли хотя бы кто-то из данных x человек». В Санкт-Петербурге $\approx 5\,000\,000$ человек. Предложите алгоритм, делающий не более 100 анализов, и определяющий число больных с ошибкой не более чем в два раза. Минимизировать количество взятого биоматериала не нужно.

Разбор задач практики

1. **Понижение вероятности**

   После $\log n$ повторов вероятность ошибки $(1 - \frac{1}{\log n})^{\log n} \leqslant e^{-1}$.
   Всё это вместе $\ln n$ раз, тогда $\frac{1}{n}$. Итого $\log n \cdot \ln n$ повторов, время работы $\mathcal{O}(n^2 \log^3 n)$.

2. **Абсолютно надежный шифр**

   Из m делаем $m \hat{} k$, k – равномерный рандом.
   $c = m \hat{} k \Leftrightarrow c \hat{} m = k \hat{} m \hat{} m = k$.
   $\Rightarrow$ вероятность получить c равна вероятности, что ключ $k = c \hat{} m$.
   $\Rightarrow$ сообщения распределены так же, как ключи, каждое с вероятностью $\frac{1}{2^n}$.

3. **Монте-Карло**

   Алгоритм.
   Кинем n случайных точек, для каждой проверим, лежит ли она в объединении/пересечении кругов. Пусть попало $n_{\texttt{in}}$, отвечаем $\frac{n_{\texttt{in}}}{n}$.

   Анализ.
   Искомую площадь обозначим p. $\mathsf{E}[n_{\texttt{in}}] = pn$.
   $n_{\texttt{in}} = pn \pm \mathcal{O}(\sqrt{n}) \Rightarrow$ мы возвращаем $p \pm \mathcal{O}(\frac{1}{\sqrt{n}})$.

   Более точная оценка: $\mathsf{D}[n_{\texttt{in}}] = p(1-p)n \Rightarrow Pr[n_{\texttt{in}} \in [pn \pm 3\sqrt{p(1-p)n}]] \geqslant 99.7\%$.

   Есть и другой подход.
   Вообще можно без рандома: просто накидать сетку с шагом $1/\varepsilon$. В сетке будет $1/\varepsilon^2$ клеток, проверим центр каждой клетки.
   На плоскости новый подход работает примерно также, как Монте-Карло.
   В d-мерном этот подход работает хуже Монте-Карло – получается сетка размера $\mathcal{O}(1/\varepsilon^d)$.

4. **Случайный элемент**

   Идём по файлу храним один элемент из прочитанных. Если считали i-й по счёту элемент, то заменяем сохранённый с вероятностью $1/i$.

5. **Игра на дереве**

   Из вершины на глубине d всегда ходит игрок, который хочет попасть в число $d \bmod 2$.

   Алгоритм.
   Рассмотрим вершину v. Найдем рекурсивно ответ для ее случайного ребенка. Если ребенок проигрышный, то v сразу выигрышная, и второго ребенка смотреть не надо. Если выигрышный, надо пойти и во второго.

   Анализ.
   $L(n)$ – матожидание времени работы из проигрышной вершины, $W(n)$ – выигрышной.

   $$L(n) = 2W(n-1)$$

   $$W(n) \leqslant \tfrac{1}{2}L(n-1) + \tfrac{1}{2}(W(n-1) + L(n-1)) = \tfrac{1}{2}W(n-1) + L(n-1) = \tfrac{1}{2}W(n-1) + 2W(n-2)$$

   Рекуррента для $W(n)$ дает примерно $\mathcal{O}(1.69^n)$.
   Итого время $\max(L(n), W(n)) \leqslant 2W(n) = \mathcal{O}(1.69^n)$.

6. **Random в задачах оптимизации**

   Алгоритм: запустить много раз, выбрать наименьший ответ.

   По неравенству Маркова на одном запуске ответ будет $> C(1+\varepsilon)\texttt{OPT}$ с вероятностью $< \frac{1}{1+\varepsilon} = 1 - \frac{\varepsilon}{1+\varepsilon}$ (это вероятность ошибки).

   Обозначим $p = \frac{\varepsilon}{1+\varepsilon}$, после $\frac{1}{p} = \frac{1+\varepsilon}{\varepsilon}$ запусков $\Pr[\text{ошибки}] = (1-p)^{\frac{1}{p}} \leqslant e^{-1}$.

7. **Приближение** MAX-SAT

   Детерминированный способ. Возьмём произвольные значения переменных, если выполнено меньше половины клозов, инвертируем все переменные.

   Рандомизированный способ. Подставим все переменные случайно.
   В каждом клозе с вероятностью $\geqslant \frac{1}{2}$ выполнился первый литерал.
   Матожидание числа выполненных клозов равно сумме матожиданий по всем клозам того, что они выполнены $\Rightarrow$ матожидание $\geqslant \frac{1}{2}m \geqslant \frac{1}{2}\texttt{OPT}$.
   Но нам нужно не матожидание, а гарантия. Тут поможет предыдущая задача.
   Максимизировать число выполненных клозов $\Leftrightarrow$ минимизировать число нарушенных.
   Если отклонились от X менее, чем на $\frac{1}{2}$, то из-за целочисленности попали в X.
   $\frac{1}{2}m + \frac{1}{2} = \frac{1}{2}m(1 + \frac{1}{m})$. Пользуемся предыдущей задачей для $\varepsilon = \frac{1}{m}$, нужно

$$\frac{1+\varepsilon}{\varepsilon} = \frac{1+\frac{1}{m}}{1/m} = m + 1$$

   запусков.

8. **Числа в окрестности**

   Простой способ.
   Ткнем в случайное число a_i, выведем в ответ все числа из $[a_i..a_i + \varepsilon]$.
   Обозначим за B минимальные $\frac{n}{3}$ чисел из $[x..x+\varepsilon]$. Если $a_i \in B$, то на $[a_i..a_i+\varepsilon]$ лежит хотя бы $\frac{n}{3}$ чисел. Вероятность попасть в B равна $\frac{n/3}{n} = \frac{1}{3}$.

   Надежный способ.
   Найдем статистики с номерами $\frac{n}{3}$ и $\frac{2n}{3}$, берем всё между ними.

9. **Простое в отрезке**

   Ткнем в случайное число из отрезка, проверим на простоту. Проверяем Миллером-Рабином, запускаем его k раз, вероятность ошибки $\frac{1}{4^k}$.
   Вероятность попасть в простое равна $\frac{1}{\log B}$, в среднем $\log B$ раз попадем в составное до успеха.
   Вероятность успеха $(1 - \frac{1}{4^k})^{\log B}$, берем $k = \log\log B$ и побеждаем.

   Строгий анализ.
   С вероятностью $\frac{1}{\log B}$ попали в простое, распознали его и закончили.
   С вероятностью $(1 - \frac{1}{\log B})$ попали в составное.
   С вероятностью $(1 - \frac{1}{\log B})(1 - \frac{1}{4^k})$ распознаем составное и продолжаем.
   С вероятностью $(1 - \frac{1}{\log B})\frac{1}{4^k}$ ошибочно заканчиваем.
   Среднее время работы $\frac{1}{\frac{1}{\log B} + (1 - \frac{1}{\log B})\frac{1}{4^k}} = \frac{4^k \log B}{4^k + \log B - 1}$.
   Ошибка, если s раз ткнем в составное, затем снова в составное и вернем его.

$\sum(1-\frac{1}{\log B})^{s+1}\frac{1}{4^k} = \frac{(1-\frac{1}{\log B})\frac{1}{4^k}}{\frac{1}{\log B}} = (\log B - 1)\frac{1}{4^k}$.
Если взять $k = \log\log B$, то вероятность ошибки мала, $\approx \frac{1}{\log B}$, и число шагов $\approx \log B$.

На самом деле вероятность попасть в простое $\frac{\frac{B}{\log B}-\frac{A}{\log A}}{B}$.
Но если A близко к B, можно и перебрать. Если же $A = \text{const}\cdot B$, то настоящая вероятность выходит $\Theta(\frac{1}{\log B})$.

10. **(*) Первообразный корень**

Ткнем в случайное число из $[2, p-1]$ и проверим. Будем тыкать, пока не найдем.
g – первообразный $\Rightarrow \forall k\colon (k, p-1) = 1$ g^k тоже первообразный.
Итого первообразных корней $\varphi(p-1) \Rightarrow$ вероятность попасть $\frac{\varphi(p-1)}{p-1} \geqslant \frac{1}{\log\log p}$.

Проверка числа g: достаточно убедиться, что $x^{\frac{p-1}{d}} \neq 1$ для всех простых $d \mid (p-1)$.
Для этого нужно факторизовать p, а далее за $\mathcal{O}(\log^2 p)$.
Факторизовать мы сейчас умеем за $\mathcal{O}(p^{1/4}\log p)$.

11. **(*) 4-sat random walk**

Давайте сделаем то же самое. Но теперь мы приближаемся с вероятностью $\frac{1}{4}$.

$n/2$ **шагов:** вероятность $(\frac{1}{4})^n$, запусков 2^n.
С вероятностью $\geqslant \frac{1}{2}$ расстояние до ответа $\leqslant \frac{n}{2}$.

n **шагов:** вероятность $(\frac{5}{8})^n$, запусков 1.6^n.
На расстоянии k от ответа с вероятностью $\binom{n}{k}(\frac{1}{2})^n$.
$\sum(\frac{1}{2})^n\binom{n}{k}(\frac{1}{4})^k = (\frac{1}{2})^n(1+\frac{1}{4})^n = (\frac{5}{8})^n$.

$2n$ **шагов:** вероятность $\approx (\frac{2}{3})^n$, запусков $\approx 1.5^n$.
С вероятностью $\geqslant \binom{2k}{k/2}(\frac{1}{4})^{3k/2}(\frac{3}{4})^{k/2} = \binom{2k}{k/2}\frac{3^{k/2}}{4^{2k}}$ среди первых $2k$ шагов $\leqslant k/2$ будут не туда ($\Rightarrow$ придем в ответ).
Вероятность успеха $\geqslant \sum\binom{n}{k}(\frac{1}{2})^n\binom{2k}{k/2}\frac{3^{k/2}}{4^{2k}}$
$\geqslant 2^{-n}\sum\binom{n}{k}\text{poly}(\frac{1}{k})(\frac{1}{3})^k$
$\geqslant \text{poly}(\frac{1}{n})2^{-n}(1+\frac{1}{3})^n = \text{poly}(\frac{1}{n})(\frac{4}{6})^n \approx (\frac{2}{3})^n$.
$\binom{2k}{k/2}\frac{3^{k/2}}{4^{2k}} \geqslant \text{poly}(\frac{1}{k})(\frac{1}{3})^k$ берется из формулы Стирлинга ($n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$).

12. **(*) 3-Coloring**

Шаг walk'a: найти нарушенное ребро, случайный конец перекрасить в случайный цвет (причем выбор из двух цветов, перекрашивать в тот же нет смысла).
Вероятность того, что мы приблизились к ответу, $\geqslant \frac{1}{4}$: угадали вершину и цвет.
На расстоянии k от ответа с вероятностью $\binom{n}{k}\frac{2^k}{3^n}$.
Дальше те же вычисления, что в прошлой задаче.
При n шагах получится 2^n запусков, но за $\mathcal{O}^*(2^n)$ и так умели.
При $2n$ шагах получится вероятность $\text{poly}(\frac{1}{n})3^{-n}(1+\frac{2}{3})^n \approx (\frac{5}{9})^n$, запусков $\approx 1.8^n$.

13. **(*) Пересечение полуплоскостей**

Наблюдение: ответ всегда либо пересечение двух границ полуплоскостей, либо «бесконечная точка». Второй случай можно отсечь, добавив большой обрамляющий квадрат.
Пусть есть текущая точка x, и мы добавляем i-ю полуплоскость.

Если x лежит в ней, то x остается ответом.
Иначе x должна лежать на границе i-й полуплоскости, пересечем эту границу с предыдущими $i-1$ и выберем новый ответ. Тратим на это $\mathcal{O}(i)$.
В худшем случае $\mathcal{O}(n^2)$.

А теперь перед началом сделаем `shuffle` полуплоскостей.
Пусть после i шагов ответ – пересечение прямых l и m.
Потратим $\mathcal{O}(i)$ только если l или m была последней, вероятность $\frac{2}{i}$.
Итого матожидание $\sum \mathcal{O}(i)\frac{2}{i} = \mathcal{O}(n)$.

Домашнее задание

3.1. Обязательная часть

1. **(2) Понижение вероятности**

   Алгоритм работает за $\mathcal{O}(n^3)$, вероятность **успеха** $1/n^3$ (ошибка односторонняя).
   За какое время можно добиться вероятности успеха $1 - 1/2^n$?

2. **(2) Числа Кармайкла**

   Проверьте, что данное число является числом Кармайкла.
   Оцените время работы и ошибку алгоритма.

   Не забудьте разобрать все 4 случая.

3. **(3) Устранение ошибок**

   Есть линейная функция $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$. Линейность $f\colon \forall x, y\ f(x+y) = f(x) + f(y)$.
   Есть g, которая отличается от f на ε-доле входов.
   Дан x. Найти $f(x)$. Можно вызывать только функцию g. Какой тип алгоритма получился?

4. **(2) Приближённый** MAX-3-SAT

   Придумайте приближенный ZPP алгоритм с константой лучше $\frac{3}{4}$ для MAX-3-SAT, про который дополнительно известно что в каждом клозе *ровно три различные переменные*.

5. **(2) Randomized NP**

   Что будет, если в NP проверяющий подсказку алгоритм будет из ZPP, а не из P? Как полученный класс соотносится с уже известными?

   Подсказка: покажите, что этот класс содержит NP.
   Эту задачу обязательно сдать до мягкого дедлайна так как, опыт прошлых поколений показывает вероятность ошибок в решении $\approx 100\%$.

6. **(2) Сдать Полларда**

   В системе $C_1 = 100$ тестов. Поллард работает с вероятностью $\frac{1}{2}$.
   Сколько раз нужно запустить Полларда, чтобы с вероятностью $1 - C_2 = 1 - \frac{1}{10}$ получить `OK`?
   Больше баллов получат простые вычисления, которые можно сделать в уме для $\forall\ C_1, C_2$.

3.2. Дополнительная часть

1. **(3) Zero-knowledge-GI**

   Алиса и Боб знают два графа. Алиса утверждает, что знает перестановку π, которая задаёт изоморфизм этих графов. Придумайте полиномиальный по времени протокол обмена сообщениями, который позволит Алисе убедить Боба, что она знает π, и при этом не сообщить никакой полезной информации про π.

2. **(1+2) Approximated median**

   Дан массив. За $\mathcal{O}(n^{1/2})$ оцените максимально точно медиану.
   Ошибка – расстояние по индексам между настоящей медианой и найденной нами.
   (+2) Оцените ошибку, можно программно.

3. **(3) BPP+**

   Пусть есть типа-BPP алгоритм M:
   $\forall\, x \in L\ \Pr[M(x) = 1] \geqslant p \wedge \forall\, x \notin L\ \Pr[M(x) = 0] \geqslant 1 - p + \varepsilon$ $(p, \varepsilon > 0)$.
   Получите из него BPP алгоритм с ошибкой 2^{-100}.

4. **(3) k-путь**

   k-путь – простой путь длины ровно k **вершин** из a в b. Придумайте алгоритм для нахождения k-пути за $\mathcal{O}^*(\alpha^k)$, где α – константа.
   Подсказка: покрасьте вершины в случайные цвета...

5. **(3) Квадродерево**

   Методом Монте-Карло на практике мы научились считать площадь пересечения-объединения кругов в области $[0, 1] \times [0, 1]$ за $\Theta(n)$ с ошибкой $n^{-1/2}$. Есть более мощное решение с временем $\Theta(n)$ и ошибкой всего n^{-1}.

   Корень квадродерева – квадрат $[0, 1]\times[0, 1]$. У каждой вершины ровно 4 сына – делим квадрат на 4 равных квадрата. Поиск пересечения кругов – спуск по дереву. Если какая-то вершина-квадрат целиком снаружи или целиком внутри всех данных кругов, мы можем остановить рекурсию и не идти вглубь. Если мы спустились до квадрата со стороной ε можно вернуть половину $\frac{1}{2}\varepsilon^2$ и не идти вглубь.

   Задача: выбор ε, чтобы получились заявленные время работы и ошибка, анализ времени работы и ошибки.