

SPb HSE, 1 курс, зима 2024/25

Практика по алгоритмам #18

Невероятные истории

11 февраля 2025

Собрано 11 февраля 2025 г. в 16:24

Содержание

1. Невероятные истории	1
2. Разбор задач практики	3
3. Домашнее задание	7
3.1. Обязательная часть	7
3.2. Дополнительная часть	7

Невероятные истории

1. Приближённый MAX-CUT

Найдите $\frac{1}{2}$ -приближение MAX-CUT.

2. L-задачи

Лежит ли в классе L алгоритм Полларда? А чекер для НАМ-РАТН?

3. Случайный элемент на миллион

Есть длинный файл из n элементов и $\mathcal{O}(1)$ памяти.

Прочтите файл и в конце верните один из этих n элементов с равной вероятностью.

А если n заранее неизвестно?

Т.е. запросы двух типов: «add element», «eof and give me random element».

4. k -сочетание

Дан массив из n различных чисел. Равновероятно выбрать случайные k различных из них.

а) Время $\mathcal{O}(n)$, исходный массив можно менять.

б) Время $\mathcal{O}(k)$, исходный массив можно менять.

с) Время $\mathcal{O}(k)$, допамять $\mathcal{O}(k)$, исходный массив менять нельзя.

5. Простое в отрезке

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

6. Игра на дереве (вспомните решение с лекции)

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 $(2 - \varepsilon)^n$. Показать, что решение работает за $\Omega(\sqrt{2}^n)$.

7. Игра на дереве 2

Есть полное бинарное дерево глубины n . В листьях написаны \mathbb{R} числа. Двое играют в игру «спуск по дереву», ходят по очереди, первый хочет число в листе побольше, второй поменьше. Каков результат игры, если оба играют оптимально? Решить (а) за 2^n ; (б) за $(2 - \varepsilon)^n$.

8. Монте-Карло

В квадратике $[0, 1] \times [0, 1]$ живут k кругов.

Посчитайте площади их объединения и пересечения с ошибкой не более 10^{-2} .

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

9. Случайные тесты

Возможно, вы сталкивались с ситуацией «я написал алгоритм, он работает на всех моих тестах, а в тестирующей системе WA9». Что же делать? Общая практика — написать стресс-тест. Запустить на случайном небольшом тесте, сравнить ответ с более медленным решением.

Осталось научиться генерить случайные тесты. Как сгенерировать случайный граф из n вершин и m рёбер? Как сгенерировать связный случайный граф из n вершин и m рёбер?

10. Гамильтонов путь в случайном графе

Дан случайный большой достаточно плотный граф ($n \approx 1000$, $m = 20n$).

Найдите в нём гамильтонов путь (доказывать корректность не нужно).

(*) Преобразование вселенной, которое позволяет алгоритму реально быстро работать.

11. Экзамен за $\mathcal{O}(1)$

Преподаватель хочет принять у студента экзамен. Опытный преподаватель знает: 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.

12. Равенство мультимножеств

Проверить равенство двух мультимножеств за $\mathcal{O}(n)$ времени и $\mathcal{O}(1)$ памяти.

13. 4-SUM

Петя решил задачу

A «даны 4 массива, выбрать из каждого по одному числу, чтобы сумма была S », написал код, заоптимизировал, порадовался, прочёл условие и понял: нужно было решать задачу B «дан массив, выбрать из него 4 элемента так, чтобы сумма была S ».

Помогите Пете решить B , используя уже готовое решение A .

14. 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.

(a) Решение бинарными поисками. (b) (*) Продвинутое крутое решение.

15. 4-SAT random walk

Адаптируйте уже известный вам алгоритм. Оцените время работы.

16. (*) Пересечение полуплоскостей

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

17. (*) 3-Coloring

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

18. (*) Дерандомизация схем

Дана схема $C(x, y)$, решающая задачу с односторонней ошибкой:

$x \notin L \Rightarrow \forall y C(x, y) = 0$, $x \in L \Rightarrow \Pr_y[C(x, y) = 1] \geq \frac{1}{2}$. Постройте схему $C'(x)$ полиномиального от C размера, которая не использует случайность и верно работает на всех входах.

Разбор задач практики

1. Приближённый MAX-CUT

Каждую вершину кладём в случайное из двух множеств, для каждого ребра вероятность, что концы попали в разные половины $\frac{1}{2} \Rightarrow \mathbf{E}[W(\text{cut})] = \sum_e \frac{1}{2} w_e \geq \frac{1}{2} W(\text{MAXCUT})$.

2. L-задачи

Поллард не лежит. Когда раскладываем число A , размер входа $n = \log A \Rightarrow$ мы можем использовать $\mathcal{O}(\log \log A)$ памяти, а для Полларда нужно $\Theta(\log A)$ памяти.

Чекер для НАМ-РАТН является: дают граф в любом виде, нам нужно для каждого i проверить, \exists ли ребро в графе \Rightarrow достаточно $\mathcal{O}(\log n)$ бит, где n – размер графа, размер входа.

3. Случайный элемент

n заранее известно? Сгенерировали заранее случайное $0 \leq i < n$, сохранили нужный элемент.

Иначе идём по файлу, храним случайный элемент из прочитанных.

Когда считали i -й по счёту элемент, заменяем сохранённый с вероятностью $\frac{1}{i}$.

4. k -сочетание

- Сделаем `random_shuffle`, возьмём первые k элементов.
- Запустим второе решение из предыдущей задачи, но только на k итераций.
- k раз дёргаем случайный из всех элементов; уже использованные храним в хеш-таблице размера k ; если попали в уже использованный элемент, выберем другой.

5. Простое в отрезке

Ткнём 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.

$\mathcal{O}(\log B \log \log B \log B)$ (повторить Миллера-Рабина, повторить внутри М.Р., сам М.Р.).

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 $(1 - \frac{1}{\log B})^2 \frac{1}{4^k}$ 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 ткнём в составное, затем снова в составное и вернем его.

$\sum (1 - \frac{1}{\log B})^{s+1} \frac{1}{4^k} = \frac{(1 - \frac{1}{\log B})^{\frac{1}{4^k}}}{1/\log B} = (\log B - 1) \frac{1}{4^k}$.

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 $\frac{B - A}{\log B - \log A}$.

Если $B - A$ мало, можно перебрать все $B - A$. Если $A = \text{const} \cdot B$, вероятность = $\Theta(\frac{1}{\log B})$.

6. 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. $L(n)$ – E [времени работы из проигрышной вершины], $W(n)$ – выигрышной.

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

$$W(n) \leq \frac{1}{2}L(n - 1) + \frac{1}{2}(W(n - 1) + L(n - 1)) = \frac{1}{2}W(n - 1) + L(n - 1) = \frac{1}{2}W(n - 1) + 2W(n - 2)$$

Рекуррента $W(n)$ даёт $\mathcal{O}(1.69^n)$. Итого время $\max(L(n), W(n)) \leq 2W(n) = \mathcal{O}(1.69^n)$.

$$W(n) \leq \frac{1}{2}W(n - 1) + 2W(n - 2) = \mathcal{O}(1.69^n)$$

7. Игра на дереве 2

Как и предыдущую задачу можно решать динамикой по дереву за 2^n . Можно сделать бин-поиск по ответу, тогда первый хочет $\geq f$, второй хочет $< f$, получили 01-игру, которую умеем решать за $\mathcal{O}(1.69^n)$.

8. Монте-Карло

Алгоритм. Ткнём n случайных точек, для каждой p проверим, лежит ли она в объединении/пересечении кругов C_i ($\forall i$ проверить $p \in C_i$). Пусть попало n_{in} , отвечаем $\frac{n_{\text{in}}}{n}$.

Анализ. Искомую площадь обозначим p . $E[n_{\text{in}}] = pn$.

$n_{\text{in}} = pn \pm \mathcal{O}(\sqrt{n}) \Rightarrow$ мы возвращаем $p \pm \mathcal{O}(\frac{1}{\sqrt{n}})$.

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

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 $\mathcal{O}(1/\varepsilon^d)$.

9. Случайные тесты

Уточнение по условию: равномерное распределение по всем тестам.

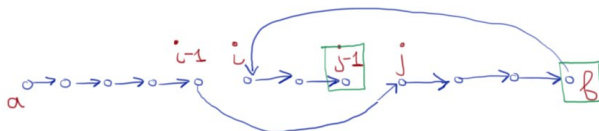
Генерим случайный граф, пока `dfs` не скажет, что он связный.

Если n и m маленькие или $m \geq \frac{1}{2}n \log n$ [Random Graphs], ждать недолго.

10. Гамильтонов путь в случайном графе

Angluin'Valiant'1979 ([AV79]) – random walk работает на плотных графах $m \geq C \cdot n \log n$.

В 2021-м эту идею довели до $\mathcal{O}(n)$ [Nenadov'Steger'Su], там же есть версия [AV79].



11. Экзамен за $\mathcal{O}(1)$

Легенда. В прошлый раз мы так же проверяли дз одного студента, и поняли «алгоритм не работает». А вот чтобы отличить два типа студентов, алгоритм таки работает.

Алгоритм: зададим k случайных вопросов.

Осталось подобрать барьер b ($0 \leq b \leq k$): студент ответил $\leq b \Leftrightarrow$ студент не готов.

Интуитивно $b = \frac{1}{2}(0.9k + 0.5k) = 0.7k$ (при больших k).

При $k = 2$ имеем $\text{Pr}_1 = 0.9 \cdot 0.1 = 0.09$, $\text{Pr}_2 = 0.5 \cdot 0.5 = 0.25 \Rightarrow b = 1$.

Точные вычисления: $b = ck \Rightarrow 0.5^k = 0.9^{ck} 0.1^{(1-c)k} \Rightarrow c = \frac{\log 5}{\log 9} \approx 0.734$.

12. Равенство мультимножеств

$A = \{a_i\}$, $f_A(x) = \prod_i (x - a_i)$. $A = B \Leftrightarrow f_A \equiv f_B \Rightarrow$ подставим случайный x , посчитаем по модулю $P = 10^9 + 7$, лемма Шварца-Зипшеля даёт вероятность ошибки $\frac{n}{P}$.

Реализация алгоритма:

```

1 z = random, ra = 1, rb = 1
2 for i do ra *= z - a[i], ra %= MOD
3 for i do rb *= z - b[i], rb %= MOD
4 return ra == rb

```

13. 4-SUM

Покрасим элементы в 4 цвета случайным образом.

С вероятностью $\frac{4!}{4^4} = \frac{24}{64} > \frac{1}{3}$ искомые элементы попадут в 4 разных множества.

14. Ковид

Население обозначим n . Пусть заболело x человек. Смешаем случайных m человек $\Rightarrow p = \text{Pr}[\text{тест не нашёл больного}] = (1 - \frac{x}{n})^m$. При $m = \frac{n}{x}$ имеем $p \approx e^{-1}$.

Алгоритм. Нужно ошибиться ≤ 2 раза \Rightarrow ищем x в форме $x = 2^k$. Делаем бинпоиск по k , внутри делаем $q=5$ проверок при $m = \frac{n}{x}$ и проверяем $E[\text{тестов без больных}] < q \cdot e^{-1}$.

Улучшение: на первых фазах бинпоиска делаем 1 проверку; с места где бинпоиск остановился, ищем вправо-влево вторым мини-бинпоиском по 3 проверки; в найденной точке x' сделаем $q = 100 - 1 \cdot 23 - 3 \cdot 5$ проверок. Решим уравнение:

$$y = E[\text{тестов без больных}] = q \cdot (1 - \frac{x}{n})^{\frac{n}{x'}} = q \cdot e^{-\frac{x}{x'}} \Rightarrow x = x' \cdot -\ln \frac{y}{q}.$$

15. 4-SAT random walk

Давайте сделаем то же, что и для 3-sat. Но теперь мы приближаемся с вероятностью $\frac{1}{4}$.

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

n шагов: вероятность $(\frac{5}{8})^n$, запусков 1.6^n .

На расстоянии k от ответа с вероятностью $\binom{n}{k} (\frac{1}{2})^n \cdot \sum (\frac{1}{2})^n \binom{n}{k} (\frac{1}{4})^k = (\frac{1}{2})^n (1 + \frac{1}{4})^n = (\frac{5}{8})^n$.

$2n$ шагов: вероятность $\approx (\frac{2}{3})^n$, запусков $\approx 1.5^n$. С вероятностью $\geq (\frac{2k}{k/2}) (\frac{1}{4})^{3k/2} (\frac{3}{4})^{k/2} = (\frac{2k}{k/2}) \frac{3^{k/2}}{4^{2k}}$ среди первых $2k$ шагов $\leq k/2$ будут не туда \Rightarrow придем в ответ.

$\text{Pr}[\text{успеха}] \geq \sum \binom{n}{k} (\frac{1}{2})^n \binom{2k}{k/2} \frac{3^{k/2}}{4^{2k}} \geq 2^{-n} \sum \binom{n}{k} \text{poly}(\frac{1}{k}) (\frac{1}{3})^k \geq \text{poly}(\frac{1}{n}) 2^{-n} (1 + \frac{1}{3})^n = \text{poly}(\frac{1}{n}) (\frac{4}{6})^n \approx (\frac{2}{3})^n$.

$\binom{2k}{k/2} \frac{3^{k/2}}{4^{2k}} \geq \text{poly}\left(\frac{1}{k}\right) \left(\frac{1}{3}\right)^k$ берется из формулы Стирлинга ($n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$).

16. (*) Пересечение полуплоскостей

Наблюдение. Ответ — либо пересечение двух границ полуплоскостей, либо «точка на бесконечности». Второй случай можно отсеять, добавив bounding box (обрамляющий квадрат).

Алгоритм. Изначально ответ — один из углов bounding box. Добавляем полуплоскости по одной в порядке randomShuffle. Пусть текущий ответ p , мы добавляем полуплоскость A_i . Если $p \in A_i \Rightarrow p$ остаётся ответом. Иначе новый ответ должен лежать на границе $A_i \Rightarrow$ пересечём границу A_i с предыдущими $i-1$ и выберем новый ответ. Тратим на это $\mathcal{O}(i)$.

В худшем случае $\mathcal{O}(n^2)$. Но. Пусть после добавления A_i (i шагов) ответ — пересечение прямых l и $m \Rightarrow$ на i -м шаге мы потратили $\mathcal{O}(i)$ только если или l , или m последняя из i , вероятность этого $\frac{2}{i}$. Итого $\mathbb{E}[\text{time}] = \sum \mathcal{O}(i) \frac{2}{i} = \mathcal{O}(n)$.

17. (*) 3-Coloring

Алгоритм: random walk. 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, $\geq \frac{1}{4}$: 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 4-sat-random-walk.

При n шагах получится 2^n запусков, но за $\mathcal{O}^*(2^n)$ и так умели.

При $2n$ шагах получится вероятность $\text{poly}\left(\frac{1}{n}\right) 3^{-n} \left(1 + \frac{2}{3}\right)^n \approx \left(\frac{5}{9}\right)^n$, запусков $\boxed{= \mathcal{O}(1.8^n)}$

18. (*) Дерандомизация схем

Привет от Пети Смирнова.

Домашнее задание

3.1. Обязательная часть

1. (3) Числа Кармайкла

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

Не забудьте разобрать все 4 случая.

Само решение даёт (1). Оценка каждого случая даёт максимум (0.5).

2. (3) Устранение ошибок

Есть линейная функция $f: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$. Линейность $f: \forall x, y f(x + y) = f(x) + f(y)$.

Есть g , отличающаяся от f на ε -доле входов: $\Pr_x[f(x) \neq g(x)] \leq \varepsilon$.

Дан x . Найти $f(x)$. Можно вызывать только функцию g . Какой тип алгоритма получился?

3. (2) Случайный недвудольный граф

Придумайте алгоритм, позволяющий для $3 \leq n \leq 8$, $3 \leq m \leq \frac{n(n-1)}{2}$ возвращать случайный из всех недвудольных графов из n вершин и m рёбер. Каждый недвудольный граф должен возвращаться с одинаковой вероятностью.

Покажите, что ожидаемое время работы не велико, 10^5 графов/сек. – точно полный балл.

4. (2) Случайное подмножество

Есть длинный файл из n элементов, его можно последовательно читать, но не хранить целиком. Требуется вывести одно из k элементных подмножеств. Каждое подмножество должно быть выбрано с одинаковой вероятностью $1/\binom{n}{k}$. k дано. n заранее не дано, будет известно только к концу чтения файла. Можно пользоваться только целыми случайными числами.

3.2. Дополнительная часть

1. (3) ВРР+

Пусть есть типа-ВРР алгоритм M :

$\forall x \in L \Pr[M(x)=1] \geq p+\varepsilon \wedge \forall x \notin L \Pr[M(x)=1] \leq p-\varepsilon$ ($p, \varepsilon > 0$).

Получите из него ВРР алгоритм с ошибкой 2^{-100} .

2. (1+1) Approximated median

Дан массив из различных чисел. За $\mathcal{O}(n^{1/2})$ оцените максимально точно медиану.

Ошибка – расстояние по индексам между настоящей медианой и найденной нами.

(+1) Оцените ошибку, можно программно.

3. (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 .