Случайные числа

Борис Золотов (Б09, Б11)
Михаил Иванов (Б07, Б08)
Иван Казменко (Б01, Б05)
Владислав Макаров (Б02, Б03)
Мария Радионова (Б06)
Арина Филимонова (Б10)

Санкт-Петербургский Государственный Университет
Факультет математики и компьютерных наук

Среда, 11 февраля 2026 года

Содержание

Содержание

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Вариации:

- Что, если Боб обманывает?

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Вариации:

- Что, если Боб обманывает?
- Например, если он как-то узнал всю секретную строку.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Вариации:

- Что, если Боб обманывает?
- Например, если он как-то узнал всю секретную строку.
- Тогда он может победить за 1 вопрос.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Вариации:

- Что, если Алиса обманывает?

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Вариации:

- Что, если Алиса обманывает?
- Например, если она подменяет секретную строку после вопроса.
- Но так, чтобы предыдущие ответы оставались верными.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Вариации:

- Что, если Алиса обманывает?
- Например, если она подменяет секретную строку после вопроса.
- Но так, чтобы предыдущие ответы оставались верными.
- Тогда она может заставить Боба задать $n + 1$ вопрос.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Вариации:

- Как Алисе и Бобу играть честно?

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Вариации:

- Как Алисе и Бобу играть честно?
- Алиса загадывает строку, записывает её и отдаёт запись Карлу.
- После игры Алиса предъявляет строку.
- Боб убеждается, что все ответы верные.
- Карл показывает запись, и Боб убеждается, что строка та же.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Вариации:

- Как Алисе и Бобу играть честно?
- Алиса загадывает строку, вычисляет *трудно обратимую функцию* от неё и показывает ответ Бобу.
- После игры Алиса предъявляет строку.
- Боб убеждается, что все ответы верные.
- Боб вычисляет функцию от строки Алисы и убеждается, что строка та же.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Как решать?

- При любом решении при $n > 100$ может оказаться, что 100 вопросов недостаточно.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Как решать?

- При любом решении при $n > 100$ может оказаться, что 100 вопросов недостаточно.
- Нас устроит *вероятностное* решение задачи.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Как решать?

- Выберем очередную позицию *случайно*: пусть это равномерно распределённое целое число от 1 до $2n$.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Как решать?

- Выберем очередную позицию *случайно*: пусть это равномерно распределённое целое число от 1 до $2n$.
- Вероятность неудачи равна $1/2$.

Мотивирующая задача

Задача:

- У Алисы есть секретная строка из $2n$ букв.
- Известно, что в ней n букв «a» и n букв «b».
- Боб может спросить: «какая буква на позиции i?» и получить ответ.
- Цель Боба — получить ответ «a», задав $\leqslant 100$ вопросов.

Как решать?

- Выберем очередную позицию *случайно*: пусть это равномерно распределённое целое число от 1 до $2n$.
- Вероятность неудачи равна $1/2$.
- Вероятность неудачи 100 раз подряд равна $1/2^{100}$.

Нужные свойства

Что нам может быть нужно от *случайных* чисел?

- Непредсказуемость: по предыдущим не узнать следующее.
- Достоверность: математическое ожидание, дисперсия...
- Повторяемость: тот же результат при повторе.
- Скорость: сотни миллионов в секунду?

Нужные свойства

Что нам может быть нужно от *случайных* чисел?

- Непредсказуемость: да.
- Достоверность: всё хорошо.
- Повторяемость: нет.
- Скорость: мала.

Как получать *случайные* числа:

- «Настоящие» случайные числа: аппаратные датчики.
- Например, младшие цифры датчика температуры на процессоре.

Нужные свойства

Что нам может быть нужно от *случайных* чисел?

- Непредсказуемость: нет.
- Достоверность: только самые базовые свойства.
- Повторяемость: да.
- Скорость: да.

Как получать *случайные* числа:

- Псевдослучайные числа: сгенерированные алгоритмом.
- Например, линейный конгруэнтный генератор:
- $s \leftarrow (s \cdot a + c) \bmod m$

Нужные свойства

Что нам может быть нужно от *случайных* чисел?

- Непредсказуемость: да.
- Достоверность: скорее да.
- Повторяемость: да.
- Скорость: меньше.

Как получать *случайные* числа:

- Псевдослучайные числа: сгенерированные алгоритмом.
- Например, какой-нибудь криптографически стойкий генератор.

Псевдослучайные числа

Как устроен генератор псевдослучайных чисел:

- Есть состояние s.
- Есть команда «инициализировать состояние» с параметром *seed*.
- Есть команда «выдать следующее случайное число»:
  - состояние меняется известной функцией, $s \leftarrow f(s)$;
  - выдаётся значение другой известной функции от нового состояния, $answer \leftarrow g(s)$.

Линейный конгруэнтный генератор: устройство

Пример генератора псевдослучайных чисел: линейный конгруэнтный генератор.

- $s \leftarrow (s \cdot a + c) \bmod m$
- s — состояние
- a, c, m — константы

Линейный конгруэнтный генератор: устройство

Пример генератора псевдослучайных чисел: линейный конгруэнтный генератор.

- $s \leftarrow (s \cdot a + c) \bmod m$
- s — состояние
- a, c, m — константы
- `init (`$seed$`)`: $s \leftarrow seed$
- `next ()`: $s \leftarrow (s \cdot a + c) \bmod m$
- `random (`k`)`:

```
    next ();
    return s mod k;
```

Линейный конгруэнтный генератор: устройство

Пример генератора псевдослучайных чисел: линейный конгруэнтный генератор.

- $s \leftarrow (s \cdot a + c) \bmod m$
- s — состояние
- a, c, m — константы
- `init (`$seed$`)`: $s \leftarrow seed$
- `next ()`: $s \leftarrow (s \cdot a + c) \bmod m$
- `random (`k`)`:

```
next ();
return (s · k) div m;
```

Линейный конгруэнтный генератор: устройство

Пример генератора псевдослучайных чисел: линейный конгруэнтный генератор.

- $s \leftarrow (s \cdot a + c) \bmod m$
- s — состояние
- a, c, m — константы
- `init (`$seed$`)`: $s \leftarrow seed$
- `next ()`: $s \leftarrow (s \cdot a + c) \bmod m$
- `random (`k`)`:

  $m' = m - m \bmod k$;
  `do next (); while (`$s \geqslant m'$`);`
  `return` $(s \cdot k) \operatorname{div} m'$;
- Выборка с отклонением медленнее, но обеспечивает равномерность распределения.

Линейный конгруэнтный генератор: период

Какой период у генератора?

- $s \leftarrow (s \cdot a + c) \bmod m$
- Понятно, что состояний не больше m.
- И, если мы попали в одно и то же состояние, — дальше будет одна и та же последовательность.
- Когда состояний в периоде ровно m?

Линейный конгруэнтный генератор: период

Какой период у генератора?

- $s \leftarrow (s \cdot a + c) \bmod m$
- Понятно, что состояний не больше m.
- И, если мы попали в одно и то же состояние, — дальше будет одна и та же последовательность.
- Когда состояний в периоде ровно m?
  - m и c взаимно просты,
  - $a - 1$ делится на все простые делители m,
  - если m делится на 4, то и $a - 1$ делится на 4.

Код

```cpp
#include <iostream>

using namespace std;

const unsigned a = 1664525, c = 1013904223;
unsigned s;
void init (int seed) {s = seed;}
void next () {s = s * a + c;}
int random (int k) {

    next ();
    return s % k;
}

int main () {
    unsigned seed;
    int k;
    cin >> seed >> k;
    init (seed);
    for (int step = 0; step < 10; step++)
        cout << random (k) << endl;
    return 0;
}
```

Ввод:

```
123456
1000000000
```

Вывод:

```
351072415
870155634
704390697
406627700
759071299
62768838
939533677
945261032
903911975
50228058
```

Код

```cpp
#include <iostream>

using namespace std;

const unsigned a = 1664525, c = 1013904223;
unsigned s;
void init (int seed) {s = seed;}
void next () {s = s * a + c;}
int random (int k) {

    next ();
    return (s * 1LL * k) >> 32;
}

int main () {
    unsigned seed;
    int k;
    cin >> seed >> k;
    init (seed);
    for (int step = 0; step < 10; step++)
        cout << random (k) << endl;
    return 0;
}
```

Ввод:

```
123456
1000000000
```

Вывод:

```
81740416
202598896
164003739
560336676
642396346
14614508
451582874
220085734
443289050
943017205
```

Код

```cpp
#include <iostream>

using namespace std;

const unsigned a = 1664525, c = 1013904223;
unsigned s;
void init (int seed) {s = seed;}
void next () {s = s * a + c;}
int random (int k) {
    long long z = (1LL << 32) / k * k;
    do next (); while (s >= z);
    return (s * 1LL * k) / z;
}

int main () {
    unsigned seed;
    int k;
    cin >> seed >> k;
    init (seed);
    for (int step = 0; step < 10; step++)
        cout << random (k) << endl;
    return 0;
}
```

Ввод:

```
123456
1000000000
```

Вывод:

```
87768103
217538908
176097674
601656925
689767824
15692209
484883419
236315258
475977993
975811047
```

Код

```
1   #include <iostream>
2   #include <cstdlib>
3   using namespace std;
4
5
6
7
8
9   int random (int k) {
10
11
12      return rand () % k;
13  }
14
15  int main () {
16      unsigned seed;
17      int k;
18      cin >> seed >> k;
19      srand (seed);
20      for (int step = 0; step < 10; step++)
21          cout << random (k) << endl;
22      return 0;
23  }
```

Ввод:

```
123456
1000000000
```

Вывод:

```
9977
22818
10150
16017
7706
20368
21548
8141
828
19946
```

Код

```cpp
#include <iostream>
#include <random>
using namespace std;









int main () {
    unsigned seed;
    int k;
    cin >> seed >> k;
    mt19937 rng (seed); // MT19937 is not an LCG!
    uniform_int_distribution <int> random (0, k - 1);
    for (int step = 0; step < 10; step++)
        cout << random (rng) << endl;
    return 0;
}
```

Ввод:

```
123456
1000000000
```

Вывод:

```
136332816
552883898
964354828
279683982
757868586
963400379
836262026
404531926
361015349
675133609
```

Содержание

Задача: бесконечный граф

- Рассмотрим бесконечный случайный граф.
- Вершины пронумерованы целыми числами.
- Между каждыми двумя вершинами есть ребро с вероятностью p.

Как попасть из вершины u в вершину v?

Задача: бесконечный граф

Как попасть из вершины u в вершину v?

Задача: бесконечный граф

Как попасть из вершины u в вершину v?

1. Если ребро $u \to v$ есть, идём по нему, и задача решена.
2. Если его нет, будем пробовать перейти в случайную вершину w, пока не получится.
3. И там вернёмся к пункту 1.

Задача: бесконечный граф

Как попасть из вершины u в вершину v?

1. Если ребро $u \to v$ есть, идём по нему, и задача решена.
2. Если его нет, будем пробовать перейти в случайную вершину w, пока не получится.
3. И там вернёмся к пункту 1.

Как найти математическое ожидание количества рассмотренных рёбер?

Задача: бесконечный граф

Как попасть из вершины u в вершину v?

1. Если ребро $u \to v$ есть, идём по нему, и задача решена.
2. Если его нет, будем пробовать перейти в случайную вершину w, пока не получится.
3. И там вернёмся к пункту 1.

Как найти математическое ожидание количества рассмотренных рёбер?

- С вероятностью p это 1.
- С вероятностью $1 - p$ это 1, затем $1/p$ рассмотренных рёбер из u, а затем решение такой же задачи.
- $E = p \cdot 1 + (1 - p) \cdot (1 + 1/p + E)$
- $p \cdot E = p + (1 - p)(1 + p)/p$
- $E = 1 + (1 - p)(1 + p)/p^2 = 1 + (1 - p^2)/p^2 = 1/p^2$

Задача: многочлен в «чёрном ящике»

- Рассмотрим многочлен P над полем $\mathbb{Z}/m\mathbb{Z}$ (m простое), заданный как «чёрный ящик»: мы можем только узнавать значение в точке.
- Нужно узнать этот многочлен.
- И при этом сделать не слишком много запросов.

Задача: многочлен в «чёрном ящике»

- Рассмотрим многочлен P над полем $\mathbb{Z}/m\mathbb{Z}$ (m простое), заданный как «чёрный ящик»: мы можем только узнавать значение в точке.
- Нужно узнать этот многочлен.
- И при этом сделать не слишком много запросов.

Как решать:

1. Будем поддерживать интерполяционный многочлен Q — догадку о том, каким мог бы быть многочлен P.
2. Выберем случайную точку x_i, спросим значение $P(x_i) = y_i$ и вычислим $Q(x_i) = y'_i$.
3. Если $y_i \neq y'_i$, интерполируем Q заново ($\deg Q$ увеличится на единицу) и вернёмся к шагу 1.
4. Если же $y_i = y'_i$, с большой вероятностью Q тождественно равен P.

Задача: многочлен в «чёрном ящике»

Оценим вероятность.

- Пусть Q не равен тождественно P.
- Тогда многочлен $R = Q - P$ имеет степень $\deg R \leqslant \max(\deg Q, \deg P)$.
- У многочлена R не больше $\deg R$ различных корней.
- Вероятность того, что мы случайно выбранной точкой x_i попали в корень, не больше $\deg R/m$.
- Итак, мы научились восстанавливать многочлен за $\deg P + 2$ запроса с вероятностью ошибки $\deg^2 P/m$.

Задача: многочлен в «чёрном ящике»

Оценим вероятность.

- Пусть Q не равен тождественно P.
- Тогда многочлен $R = Q - P$ имеет степень $\deg R \leqslant \max(\deg Q, \deg P)$.
- У многочлена R не больше $\deg R$ различных корней.
- Вероятность того, что мы случайно выбранной точкой x_i попали в корень, не больше $\deg R/m$.
- Итак, мы научились восстанавливать многочлен за $\deg P + 2$ запроса с вероятностью ошибки $\deg^2 P/m$.

Чуть более общий факт называется леммой Шварца–Зиппеля и работает также для многочленов от нескольких переменных.

Условие

Задача L. Кольцевой маршрут

- Это задача с двойным запуском.
- Дано $n \leqslant 50\,000$. У жюри есть секретный ориентированный цикл на числах $1, 2, \ldots, n$. Цикл зафиксирован заранее.
- Решение может задать числа $k \leqslant 1000$ и $l \leqslant 1000$, а также k начальных позиций.
- После этого жюри от каждой начальной позиции выдаст l следующих вершин цикла.
- Наконец, решение должно восстановить весь цикл.
- Пример: $n = 5$, цикл $1 \to 3 \to 5 \to 4 \to 2 \to 1$.
  - Участник задаёт $k = 2$, $l = 3$ и позиции 1 и 2.
  - Жюри выдаёт части цикла: $1 \to 3 \to 5$ и $2 \to 1 \to 3$.
  - Участник по ним восстанавливает весь цикл.

Нерешаемая задача

Очевидно, задача не имеет решения.

- Пусть $n = 50\,000$, а решение спросит какие-то 1000 позиций.
- Может оказаться, что они все расположены подряд в цикле.
- Тогда решение узнает порядок не более чем для 2000 вершин, и не узнает, в каком порядке расположены остальные 48 000.
- Что же делать?

Вероятностные решения

Чтобы справиться с задачей, чуть понизим требования к тому, что такое «решение».

- Пусть нашему решению можно работать не всегда, а почти всегда — с какой-то вероятностью, близкой к единице.
- Обычно в задаче десятки тестов, максимум — сотни.
- Если вероятность успеха велика — неважно, что существуют тесты, на которых решение не работает. Важно, чтобы на любом зафиксированном заранее тесте оно работало с хорошей вероятностью.

Решение

Зафиксируем $k = 1000$ и $l = 1000$. Выберем k случайных позиций. Будем надеяться, что k полученных путей содержат каждое ребро цикла хотя бы по разу.

- Как восстанавливать цикл?
- Начнём с какой-нибудь выбранной позиции p.
- Будем идти по полученному пути из p, пока не придём в другую выбранную позицию q.
- После этого будем идти по пути из q, пока не придём в очередную выбранную позицию r.
- И так далее, пока не вернёмся в p.

Решение

Зафиксируем $k = 1000$ и $l = 1000$. Выберем k случайных позиций. Будем надеяться, что k полученных путей содержат каждое ребро цикла хотя бы по разу.

- Почему это вообще может сработать?
- Представим себе цикл как окружность с n точками.
- Мы выбрали на этой окружности k случайных точек.
- Мы не знаем порядок точек, но знаем, что выбирали каждую точку C равномерно: вероятность того, что C равна какой-то конкретной точке A, не зависит от A и равна $1/n$.
- Чтобы успешно восстановить цикл, нужно, чтобы все расстояния по циклу между соседними выбранными точками были меньше l.

Решение

Зафиксируем $k = 1000$ и $l = 1000$. Выберем k случайных позиций. Будем надеяться, что k полученных путей содержат каждое ребро цикла хотя бы по разу.

- Предположим, что наше решение не работает: на окружности есть «плешь» размера хотя бы l, в которую не попала ни одна из наших точек.
- Значит, каждая из наших точек попала в одну из $(n - l)/n$ оставшихся позиций. Если мы выбирали точки случайно и равномерно — вероятность этого равна $(n - l)/n$ для каждой из k выбранных точек.
- Вероятность того, что все точки одновременно не попали в «плешь», равна произведению отдельных вероятностей: $((n - l)/n)^k$.

Решение

Зафиксируем $k = 1000$ и $l = 1000$. Выберем k случайных позиций. Будем надеяться, что k полученных путей содержат каждое ребро цикла хотя бы по разу.

- В выражение $((n - l)/n)^k$ подставим конкретные числа: $((50\,000 - 1000)/50\,000)^{1000} = 0.98^{1000} \approx 1.683 \cdot 10^{-9}$.
- Даже если, грубо оценивая, считать вероятность каждой из n возможных «плешей» отдельно, получаем $\approx 1.683 \cdot 10^{-9} \cdot 50\,000 = 8.415 \cdot 10^{-5}$.
- Итак, вероятность того, что наше решение не работает, на каждом тесте точно меньше 10^{-4}. Для задачи, по которой есть 99 попыток, это вполне приемлемо.

Условие

Задача M. Прямоугольник из хаоса

- Рассмотрим точки с целыми координатами от 0 до 10^5 включительно.
- Даны n (от 1 до 300 000) таких точек.
- Точки выбраны случайно, равномерно и независимо.
- Найдите прямоугольник максимальной площади, стороны которого параллельны осям координат, а все четыре вершины лежат в выбранных точках.

Наивные решения

Важное свойство задачи: заданные точки выбраны случайно.

- Наивное решение:
  - переберём четвёрки точек,
  - проверим на прямоугольность,
  - найдём максимум.
- Время работы: $\mathcal{O}(n^4)$.

Наивные решения

Важное свойство задачи: заданные точки выбраны случайно.

- Ещё более наивное решение: выведем 0.
- Почему при маленьких n это, скорее всего, правда?
- Чтобы прикинуть вероятность, разберём другое решение.

Полное решение

Важное свойство задачи: заданные точки выбраны случайно.

- Полное решение:
- • Для каждого x заведём список точек с таким x.
- • Для каждого y заведём список точек с таким y.
- • Положим все точки в `set` S.
- • Для каждой точки (x, y) — первой вершины прямоугольника:
- • • Для каждой второй вершины вида (x, y'):
- • • • Для каждой третьей вершины вида (x', y):
- • • • • Если четвёртая вершина (x', y') есть в S:
- • • • • • Вычислим площадь $|x' - x| \cdot |y' - y|$, обновим максимум.

Полное решение

Важное свойство задачи: заданные точки выбраны случайно.

- Анализ. Пусть $C = 100\,001$ — диапазон возможных координат.
- • Для каждой точки (x, y) — первой вершины прямоугольника:
  - (n вариантов)
- • • Для каждой второй вершины вида (x, y'):
  - • (в среднем будет $(n-1)/C$ таких точек)
- • • • Для каждой третьей вершины вида (x', y):
  - • • (в среднем будет $(n-1)/C$ таких точек)
- • • • • Если четвёртая вершина (x', y') есть в S:
  - • • • (проверка за $\mathcal{O}(\log n)$ или $\mathcal{O}(1)$)
- • • • • • Вычислим площадь $|x' - x| \cdot |y' - y|$, обновим максимум.
  - • • • • (вычисление за константу)
- Порядок можно грубо оценить, перемножив эти числа (хотя для точной оценки это некорректно). Получаем, что мы проверим существование $\approx n^3/C^2$ четвёртых вершин прямоугольника.

Полное решение

Важное свойство задачи: заданные точки выбраны случайно.

- Итак, грубая оценка: мы проверим существование $\approx n^3/C^2$ четвёртых вершин прямоугольника.
- Когда $n = 100$ и $C = 100\,001$, получаем $n^3/C^2 \approx 10^{-6}$.
  И каждая такая вершина есть с вероятностью $n/C^2 \approx 10^{-8}$.
  Значит, ответ будет больше нуля с вероятностью 10^{-14}.
  Действительно, ответ почти всегда равен нулю.
- Когда $n = 300\,000$ и $C = 100\,001$, получаем $n^3/C^2 \approx 27 \cdot 10^5$.
  И каждая такая вершина есть с вероятностью $n/C^2 \approx 3 \cdot 10^{-5}$.
  Значит, в среднем (по нашей грубой оценке) найдётся 81 прямоугольник.
- Время работы: когда $n = 300\,000$ и $C = 100\,001$, получаем $n^3/C^2 \approx 27 \cdot 10^5$ обращений к `set`.

Содержание

Хеширование массива

Идея:

- В массиве $S = S_1 S_2 \dots S_n$ важен порядок элементов.
- Хеш массива: $h(S) = (h(S_1 \dots S_{n-1}) \cdot p + S_n) \bmod q$.
- Подробнее разбирались на прошлом занятии.

Хеширование множеств

Идея:

- Во множестве порядок элементов не важен.
- Отсортируем последовательность.
- Теперь можно использовать полиномиальный хеш.
- Анализ: видно, что вероятность коллизии точно такая же, как у полиномиального хеша для массива.

Хеширование множеств

Идея:

- Во множестве порядок элементов не важен.
- Отсортируем последовательность.
- Теперь можно использовать полиномиальный хеш.
- Анализ: видно, что вероятность коллизии точно такая же, как у полиномиального хеша для массива.

Как теперь менять множество?

Хеширование множеств

Идея:

- Во множестве порядок элементов не важен.
- Отсортируем последовательность.
- Теперь можно использовать полиномиальный хеш.
- Анализ: видно, что вероятность коллизии точно такая же, как у полиномиального хеша для массива.

Как теперь менять множество?

- Хранить его в структуре данных, поддерживающей сортировку.
- Например, в декартовом дереве.
- В поддереве будем хранить размер и хеш поддерева.

Хеширование множеств: альтернатива

Идея (tabulation hashing):

- Во множестве порядок элементов не важен.
- Пусть возможных элементов немного.
- Заранее сопоставим каждому возможному элементу случайное целое число — например, равномерно распределённое от 0 до $2^{63} - 1$.
- Хеш множества — это xor всех чисел, соответствующих его элементам.
- Заметим, что xor-сумма нескольких (для начала двух) независимых случайных чисел с таким распределением — тоже случайное число с таким распределением.

Хеширование множеств: альтернатива

Идея (tabulation hashing):

- Во множестве порядок элементов не важен.
- Пусть возможных элементов немного.
- Заранее сопоставим каждому возможному элементу случайное целое число — например, равномерно распределённое от 0 до $2^{63} - 1$.
- Хеш множества — это xor всех чисел, соответствующих его элементам.
- Заметим, что xor-сумма нескольких (для начала двух) независимых случайных чисел с таким распределением — тоже случайное число с таким распределением.

Какая операция с двумя множествами получается легко и удобно?

Хеширование множеств: альтернатива

Идея (tabulation hashing):

- Во множестве порядок элементов не важен.
- Пусть возможных элементов немного.
- Заранее сопоставим каждому возможному элементу случайное целое число — например, равномерно распределённое от 0 до $2^{63} - 1$.
- Хеш множества — это xor всех чисел, соответствующих его элементам.
- Заметим, что xor-сумма нескольких (для начала двух) независимых случайных чисел с таким распределением — тоже случайное число с таким распределением.

Какая операция с двумя множествами получается легко и удобно?

- Симметрическая разность: $h(A \triangle B) = h(A)$ xor $h(B)$.

Хеширование множеств: альтернатива

Как хранить эти случайные числа?

- Можно использовать `map`, `unordered_map`, массив.
- Можно использовать какой-нибудь другой хеш.
- Но при плохом выборе хеша может испортиться вероятность коллизии.
- Если использовать полиномиальные хеши и сумму по модулю вместо xor-а, то вероятность коллизии можно будет оценить аналогично предыдущему способу хеширования.

Хеширование массива: альтернатива

Та же идея (tabulation hashing), но для массива:

- Представим массив как множество пар из индекса элемента и его значения.
- Пусть возможных значений немного. Тогда возможных пар (индекс, значение) тоже не очень много.
- Заранее сопоставим каждой возможной паре случайное целое число.
- Преимущества:
  1. Очень быстро работает, если массивы короткие и можно сгенерировать случайные числа заранее.
  2. Легко поддерживать хеш при изменениях массива.
  3. Легко оценить вероятность коллизии, если в массиве хранятся случайные числа.
- Недостаток — следующая формула выполняется всегда: $h([0,0]) \oplus h([0,1]) \oplus h([1,0]) \oplus h([1,1]) = 0$.

Хеши и двоичный поиск

Задача:

- Дана строка длины до 10^5.
- Приходят запросы: найти наибольший общий префикс двух суффиксов.

Хеши и двоичный поиск

Задача:

- Дана строка длины до 10^5.
- Приходят запросы: найти наибольший общий префикс двух суффиксов.
- Построим суффиксное дерево?

Хеши и двоичный поиск

Задача:

- Дана строка длины до 10^5.
- Приходят запросы: найти наибольший общий префикс двух суффиксов.
- Построим суффиксное дерево?
- С хешами подстрок эту задачу можно решать двоичным поиском!

Хеши и двоичный поиск

Задача:

- Дана строка длины до 10^5.
- Приходят запросы: найти наибольший общий префикс двух суффиксов.
- Построим суффиксное дерево?
- С хешами подстрок эту задачу можно решать двоичным поиском!
- С одной стороны, такая задача может встретиться при использовании алгоритмов сжатия, похожих на LZ77 и LZ78.

Хеши и двоичный поиск

Задача:

- Дана строка длины до 10^5.
- Приходят запросы: найти наибольший общий префикс двух суффиксов.
- Построим суффиксное дерево?
- С хешами подстрок эту задачу можно решать двоичным поиском!
- С одной стороны, такая задача может встретиться при использовании алгоритмов сжатия, похожих на LZ77 и LZ78.
- С другой стороны, при помощи такой техники можно, например, отсортировать все суффиксы строки за $O(n \log^2 n)$. Это не самый эффективный способ — задача может быть решена за $O(n)$ — но один из самых простых.

Содержание

Задача: угадывание числа

Задача:

- Загадано целое число x от 1 до n.
- Можно сказать целое число y и получить ответ: $x < y$ или $x = y$ или $x > y$.
- Как добиться равенства за небольшое количество запросов?

Задача: угадывание числа

Задача:

- Загадано целое число x от 1 до n.
- Можно сказать целое число y и получить ответ: $x < y$ или $x = y$ или $x > y$.
- Как добиться равенства за небольшое количество запросов?

Решение: двоичный поиск.

- Спросим про середину отрезка: число $y = [n/2]$.
- Если равенство не достигнуто, в зависимости от ответа перейдём в одну из половин.
- Всего запросов будет не более $\log_2 n$.

Задача: одинокое число

Задача:

- Дана последовательность из нескольких чисел: число x встречается в ней один раз, а все остальные её элементы — по два раза.
- Найдите x.

Задача: одинокое число

Задача:

- Дана последовательность из нескольких чисел: число x встречается в ней один раз, а все остальные её элементы — по два раза.
- Найдите x.

Решение:

- Будем поддерживать для встреченных чисел количество вхождений.
- Рассматривая очередное число, увеличим счётчик на единицу.
- В конце найдём то число, у которого счётчик равен единице.

Задача: одинокое число

Задача:

- Дана последовательность из нескольких чисел: число x встречается в ней один раз, а все остальные её элементы — по два раза.
- Найдите x.

А если нельзя завести $O(n)$ памяти на счётчики?

Задача: одинокое число

Задача:

- Дана последовательность из нескольких чисел: число x встречается в ней один раз, а все остальные её элементы — по два раза.
- Найдите x.

А если нельзя завести $O(n)$ памяти на счётчики?

- Будем поддерживать xor всех встреченных чисел.
- Операция xor коммутативна, и a xor $a = 0$.
- Значит, в конце у нас получится в точности число x.

Задача: одинокое число

Задача:

- Дана последовательность из нескольких чисел: число x встречается в ней один раз, а все остальные её элементы — по два раза.
- Найдите x.

Что, если одиноких числа два, а не одно?

Задача: частичная сортировка

Задача:

- Дан массив из n чисел.
- Операция — сравнить два числа.
- Нужно найти наименьшие k чисел (k гораздо меньше n).

Задача: частичная сортировка

Задача:

- Дан массив из n чисел.
- Операция — сравнить два числа.
- Нужно найти наименьшие k чисел (k гораздо меньше n).

Решение:

- Отсортируем массив из n чисел за $O(n \log n)$ операций.
- Ответ — первые k чисел.

Задача: частичная сортировка

Задача:

- Дан массив из n чисел.
- Операция — сравнить два числа.
- Нужно найти наименьшие k чисел (k гораздо меньше n).

Теперь хотим сделать меньше $O(n \log n)$ операций.

Задача: частичная сортировка

Задача:

- Дан массив из n чисел.
- Операция — сравнить два числа.
- Нужно найти наименьшие k чисел (k гораздо меньше n).

Теперь хотим сделать меньше $O(n \log n)$ операций.

- На первых k числах построим и будем поддерживать двоичную кучу с максимумом.
- Эта куча хранит k наименьших встреченных элементов.
- Очередной элемент добавляется в кучу, только когда он строго меньше её максимума.
- Добавление в кучу занимает $O(\log k)$ операций.
- Значит, всего будет $O(n \log k)$ операций.

Задача: частичная сортировка

Задача:

- Дан массив из n чисел.
- Операция — сравнить два числа.
- Нужно найти наименьшие k чисел (k гораздо меньше n).

Теперь хотим сделать даже меньше $O(n \log k)$ операций.

Задача: частичная сортировка

Задача:

- Дан массив из n чисел.
- Операция — сравнить два числа.
- Нужно найти наименьшие k чисел (k гораздо меньше n).

Теперь хотим сделать даже меньше $O(n \log k)$ операций.

- Будем рассматривать элементы исходного массива в случайном порядке!
- Будем так же поддерживать кучу с максимумом размера k.
- Каково математическое ожидание количества элементов, которые попадут в кучу?

Задача: частичная сортировка

Задача:

- Дан массив из n чисел.
- Операция — сравнить два числа.
- Нужно найти наименьшие k чисел (k гораздо меньше n).

Теперь хотим сделать даже меньше $O(n \log k)$ операций.

- Будем рассматривать элементы исходного массива в случайном порядке!
- Будем так же поддерживать кучу с максимумом размера k.
- Каково математическое ожидание количества элементов, которые попадут в кучу?
  - Вероятность того, что в случайной перестановке длины i последний элемент меньше всех предыдущих, равна $1/i$.
  - Вероятность того, что в случайной перестановке длины i последний элемент среди k наименьших, равна k/i.
- $k/(k+1)+k/(k+2)+\ldots+k/n \leqslant k\cdot(1/1+1/2+\ldots+1/n) \approx k \log n$.

Задача: частичная сортировка

Задача:

- Дан массив из n чисел.
- Операция — сравнить два числа.
- Нужно найти наименьшие k чисел (k гораздо меньше n).

Теперь хотим сделать даже меньше $O(n \log k)$ операций.

- Будем рассматривать элементы исходного массива в случайном порядке!
- Будем так же поддерживать кучу с максимумом размера k.
- Каково математическое ожидание количества элементов, которые попадут в кучу?
- $k/(k+1)+k/(k+2)+\ldots+k/n \leqslant k\cdot(1/1+1/2+\ldots+1/n) \approx k \log n$.
- Добавление в кучу занимает $O(\log k)$ операций.
- Значит, всего будет $n + O(k \log k \log n)$ операций.

Задача: частичная сортировка

Задача:

- Дан массив из n чисел.
- Операция — сравнить два числа.
- Нужно найти наименьшие k чисел (k гораздо меньше n).

Можно ли сделать меньше $n + O(k \log k \log n)$ операций?

Вопросы?

Вопросы?