Алгоритмы и структуры данных

Элементы теории вероятностей, линейность математического ожидания, время работы быстрой сортировки в среднем

Мини-задача #11 (1 балл)

Задача о флаге Нидерландов.

Мини-задача #11 (1 балл)

Задача о флаге Нидерландов.

Пусть есть массив, состоящий только из элементов 0, 1 и 2.

Отсортируйте его за один проход по массиву.

Проверить решение нужно здесь:

https://leetcode.com/problems/sort-colors

Переходим в высшую лигу сортировок!

Быстрая сортировка

Быстрая сортировка

Задача: Преобразовать массив `arr` таким образом, чтобы для выбранного элемента `k` было верно:

$$\exists p : arr[p] = k;$$

$$\forall i : 0 \leq i < p \Rightarrow arr[i] < arr[p];$$

$$\forall j : p < j < N \Rightarrow arr[j] > arr[p]$$

Быстрая сортировка

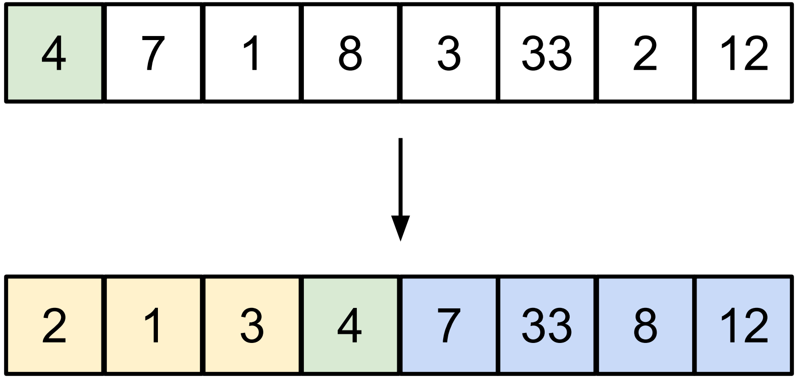
4	7	1	8	3	33	2	12

$$\exists p : arr[p] = k;$$

$$\forall i : 0 \leq i < p \Rightarrow arr[i] < arr[p];$$

$$\forall j : p < j < N \Rightarrow arr[j] > arr[p]$$

Быстрая сортировка

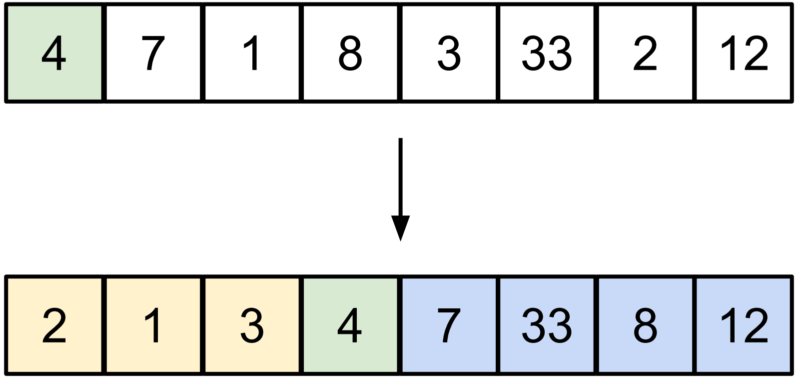


$$\exists p : arr[p] = k;$$

$$\forall i : 0 \leq i < p \Rightarrow arr[i] < arr[p];$$

$$\forall j : p < j < N \Rightarrow arr[j] > arr[p]$$

Быстрая сортировка



$$\exists p : arr[p] = k;$$

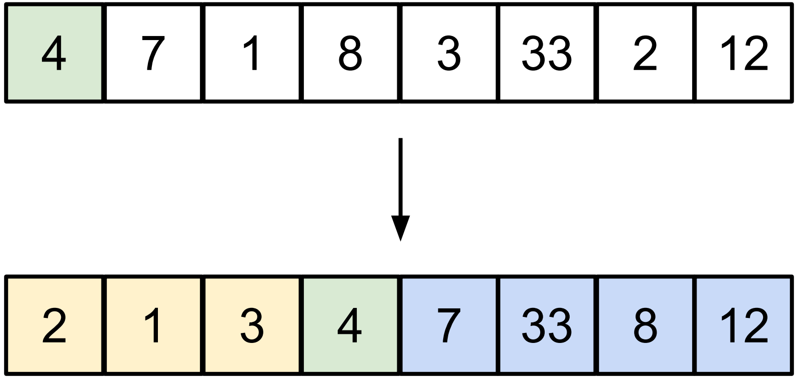
$$\forall i : 0 \leq i < p \Rightarrow arr[i] < arr[p];$$

$$\forall j : p < j < N \Rightarrow arr[j] > arr[p]$$

k — опорный элемент (pivot)

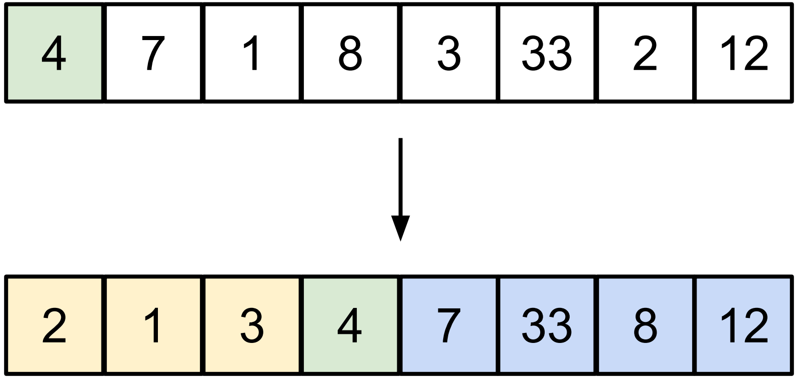
Операция — разбиение относительно опорного элемента

Быстрая сортировка



После разбиения опорный элемент стоит на своем месте ⇒ выполнили часть сортировки

Быстрая сортировка



После разбиения опорный элемент стоит на своем месте ⇒ выполнили часть сортировки

Тогда что делать дальше?

Быстрая сортировка

4	7	1	8	3	33	2	12

2	1	3	4	7	33	8	12

После разбиения опорный элемент стоит на своем месте ⇒ выполнили часть сортировки

Тогда что делать дальше?

Рекурсия на для сортировки левой и правой частей!

Быстрая сортировка

4	7	1	8	3	33	2	12

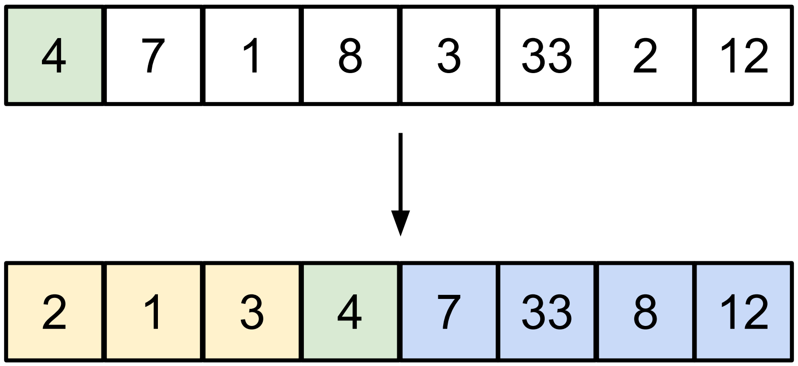
Алгоритм:

1. Выбрать опорный элемент

Быстрая сортировка

Алгоритм:

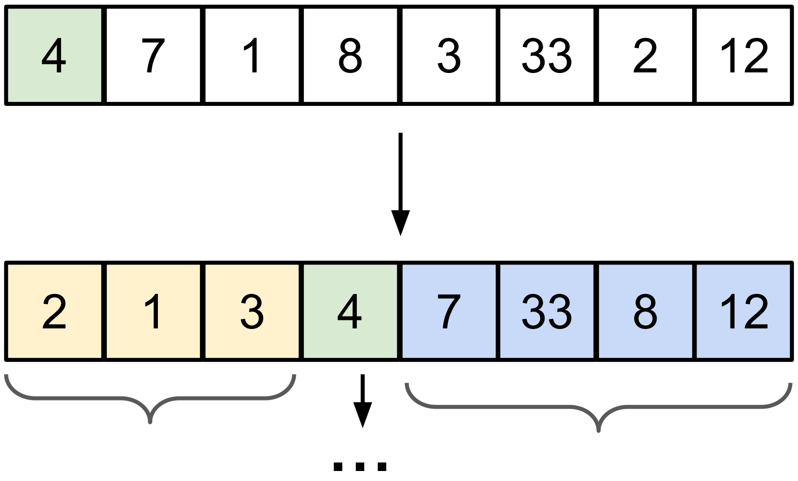
1. Выбрать опорный элемент
2. Выполнить разбиение



Быстрая сортировка

Алгоритм:

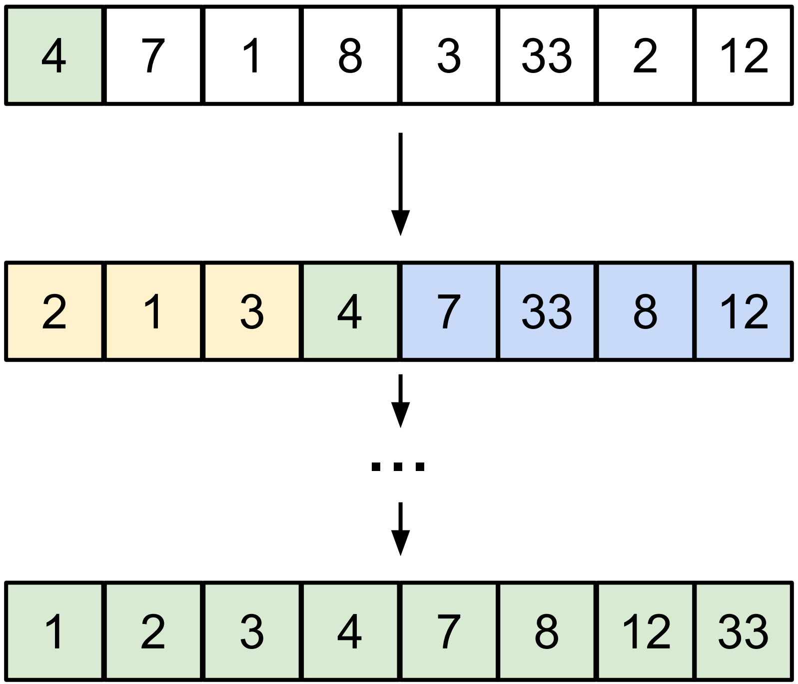
1. Выбрать опорный элемент
2. Выполнить разбиение
3. Рекурсия на левую и правые части разбиения



Быстрая сортировка

Алгоритм:

1. Выбрать опорный элемент
2. Выполнить разбиение
3. Рекурсия на левую и правые части разбиения



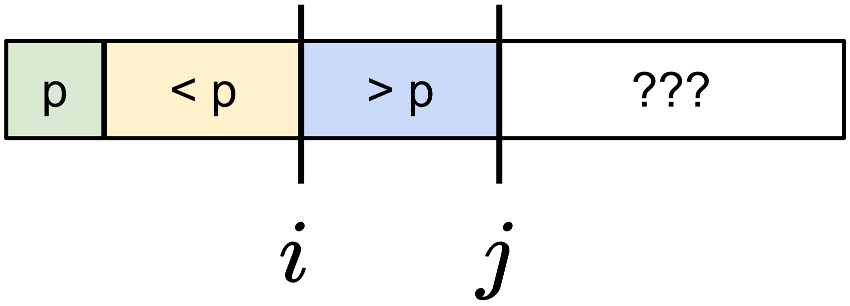
Быстрая сортировка: разбиение

```
Ставим опорный элемент первым
(просто меняем их местами, если он не там)
```

Быстрая сортировка: разбиение

Ставим опорный элемент первым
(просто меняем их местами, если он не там)

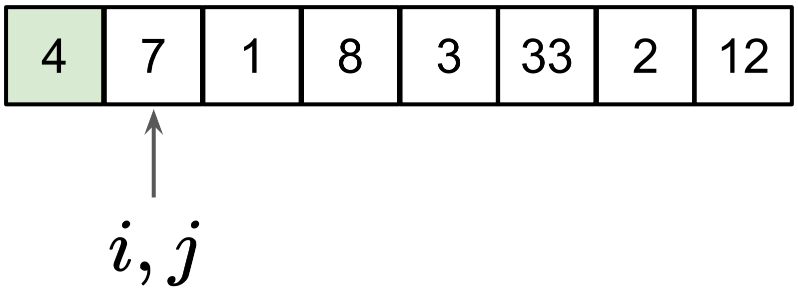
Будем поддерживать инвариант:



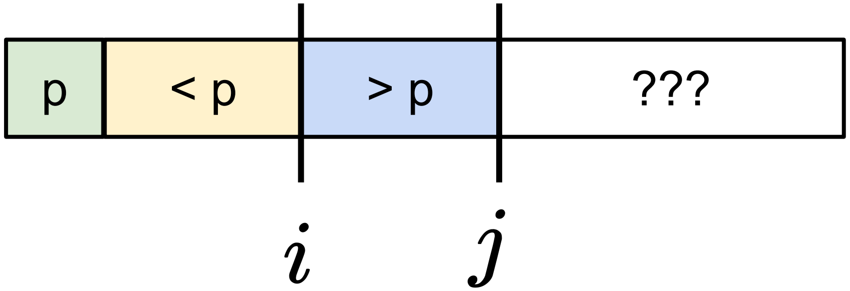
Быстрая сортировка: разбиение

4	7	1	8	3	33	2	12

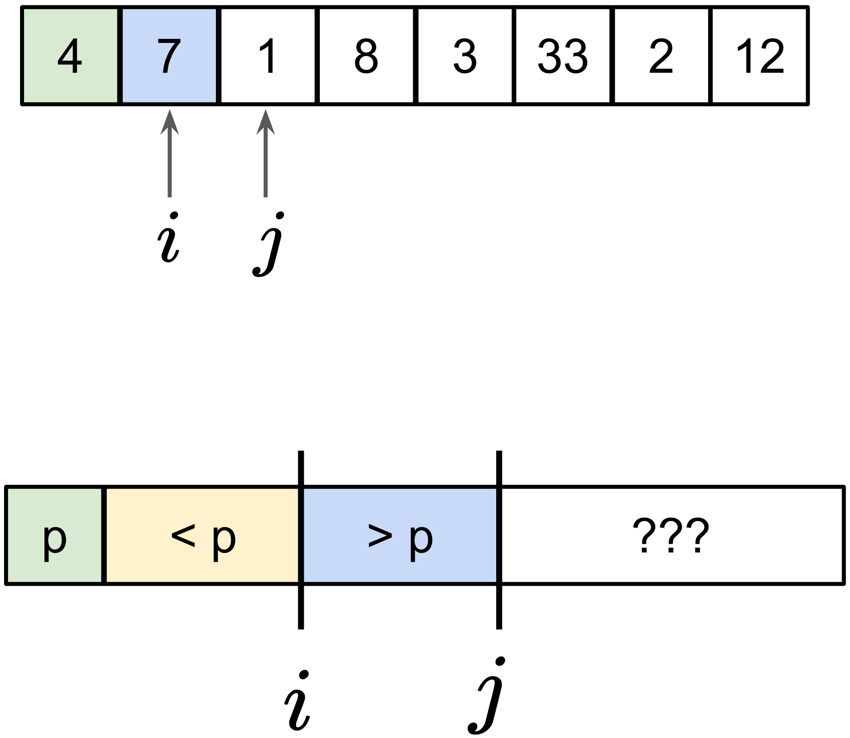
Быстрая сортировка: разбиение



На каждом шаге алгоритма двигаем j вперед

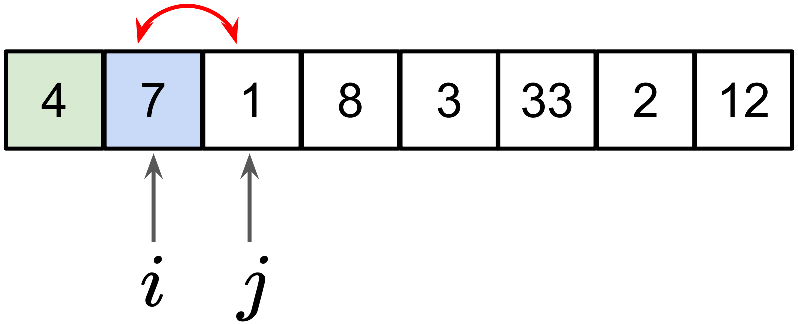


Быстрая сортировка: разбиение



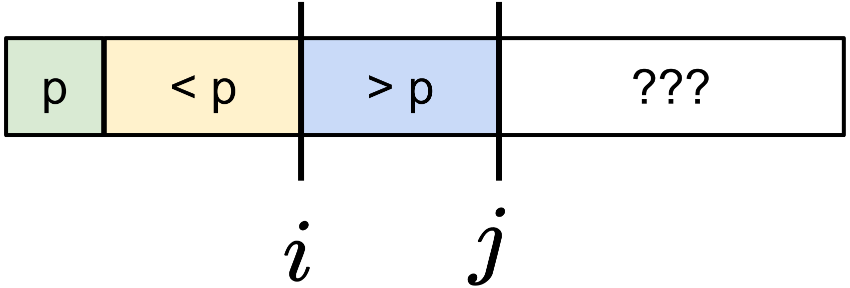
На каждом шаге алгоритма двигаем j вперед

Быстрая сортировка: разбиение

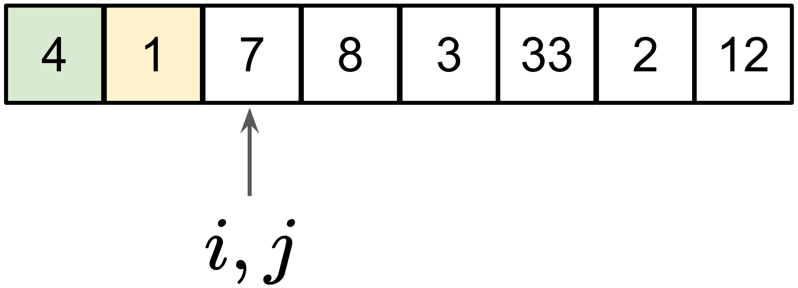


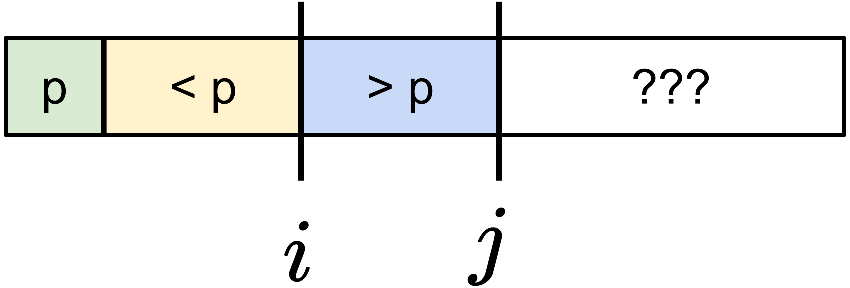
На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j]



Быстрая сортировка: разбиение

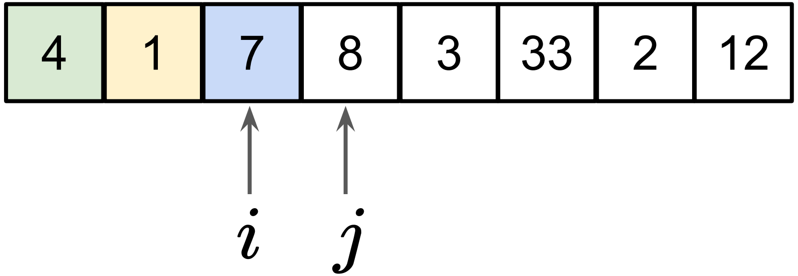




На каждом шаге алгоритма двигаем j вперед

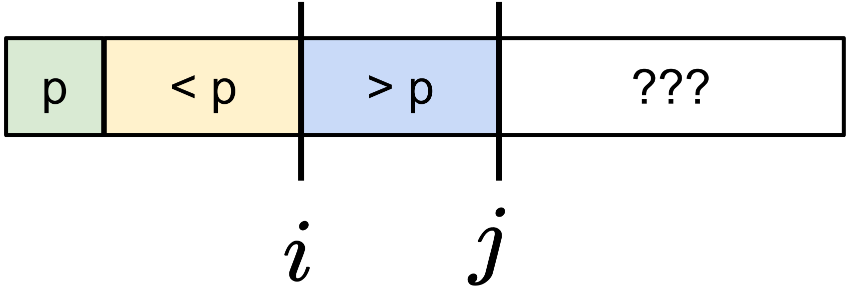
Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

Быстрая сортировка: разбиение

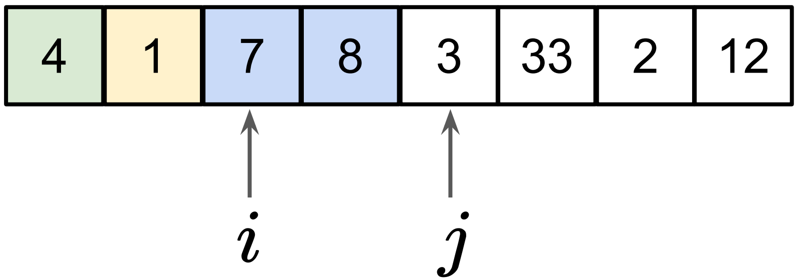


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

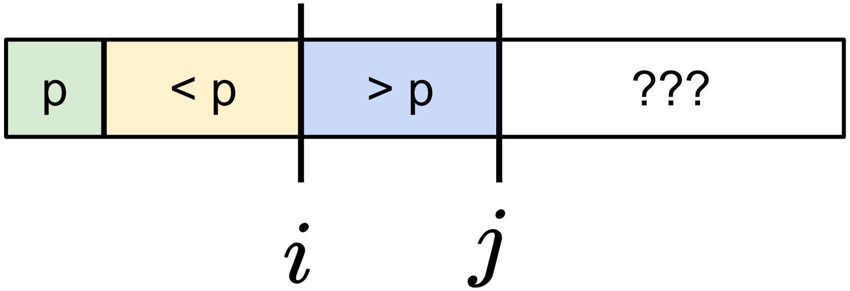


Быстрая сортировка: разбиение

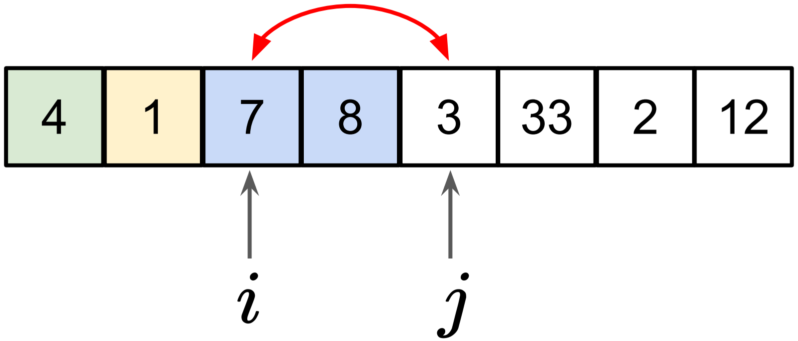


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

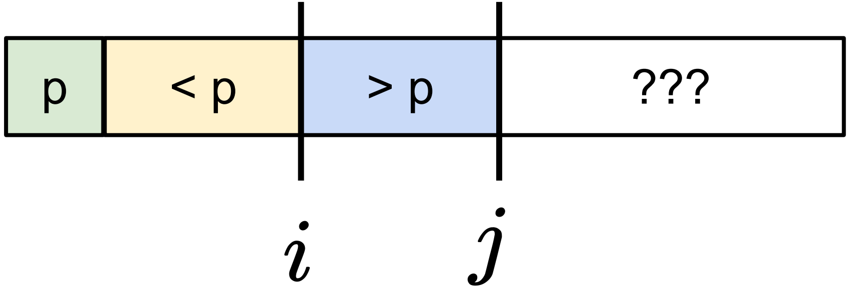


Быстрая сортировка: разбиение

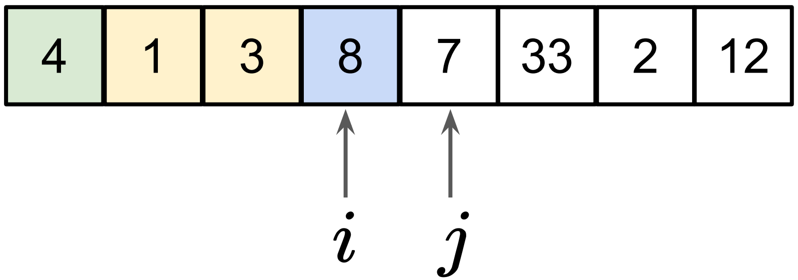


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

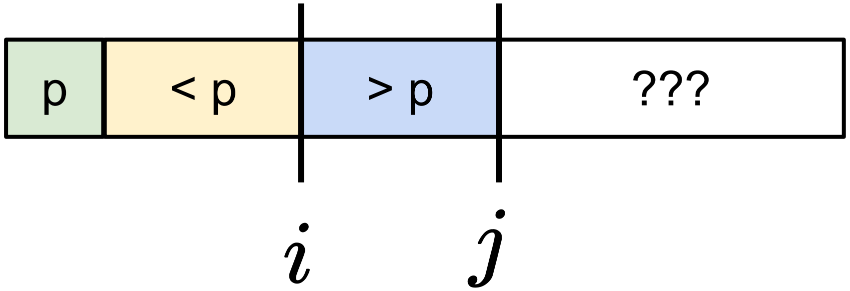


Быстрая сортировка: разбиение

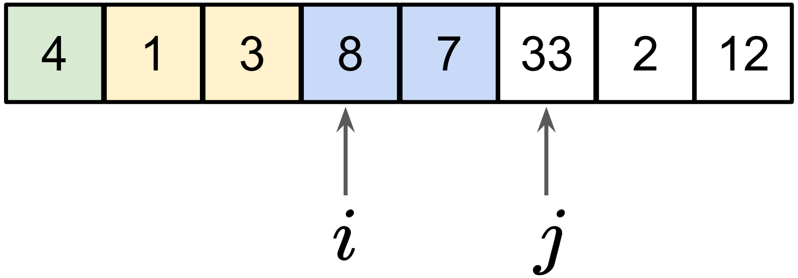


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

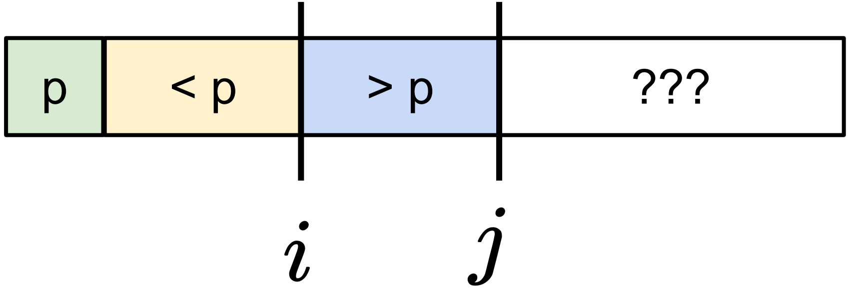


Быстрая сортировка: разбиение

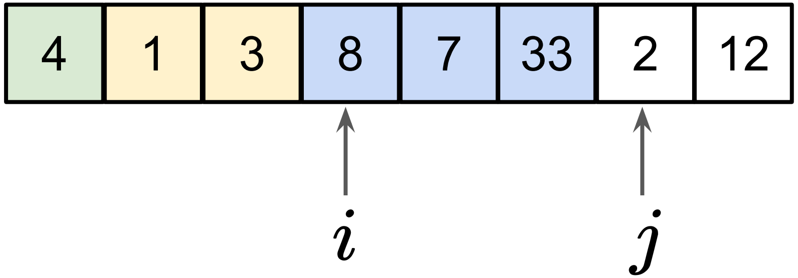


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

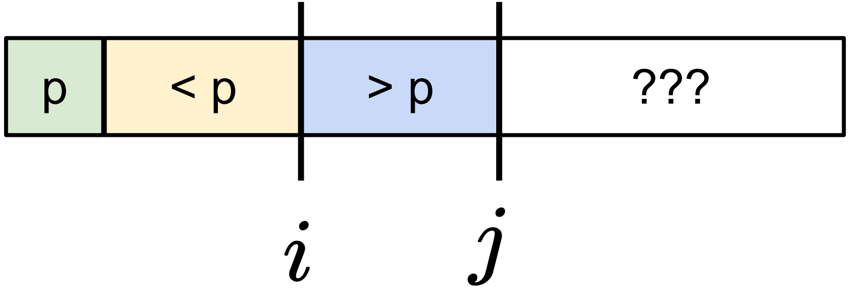


Быстрая сортировка: разбиение

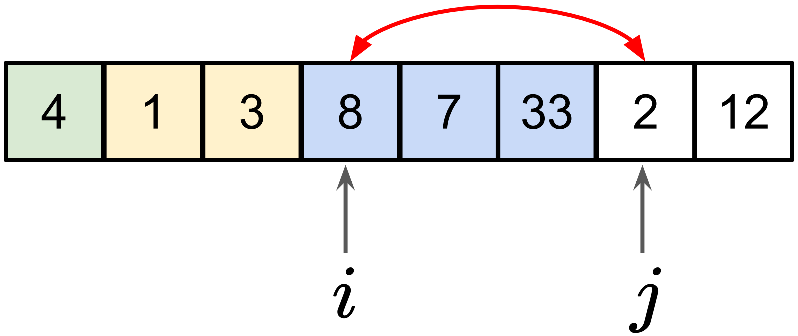


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

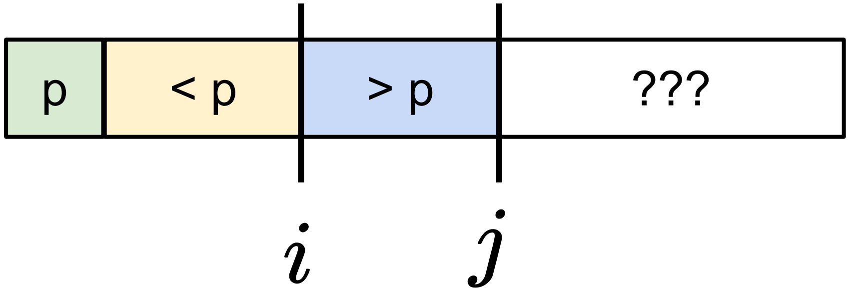


Быстрая сортировка: разбиение

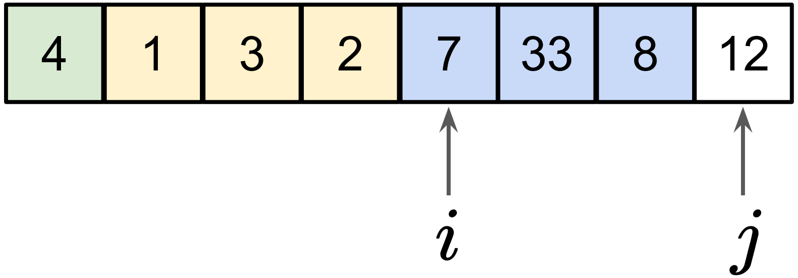


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

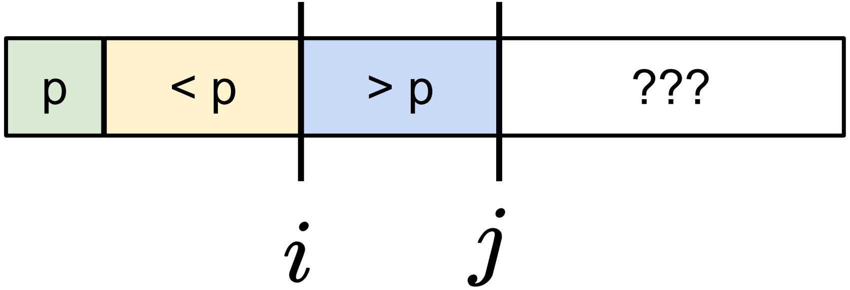


Быстрая сортировка: разбиение

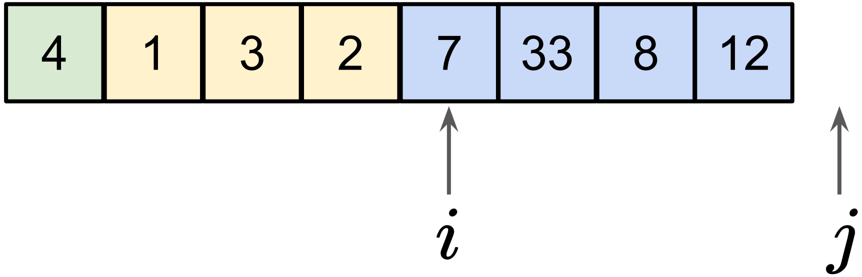


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

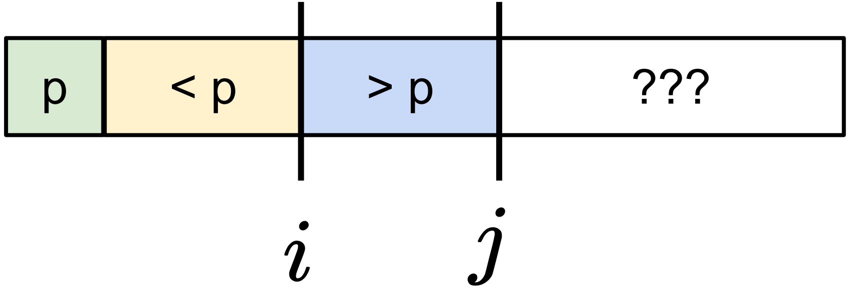


Быстрая сортировка: разбиение

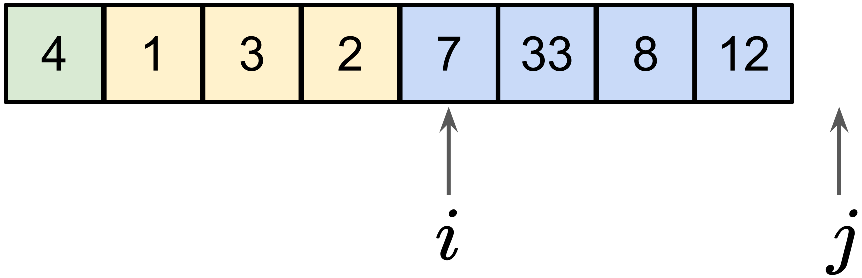


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

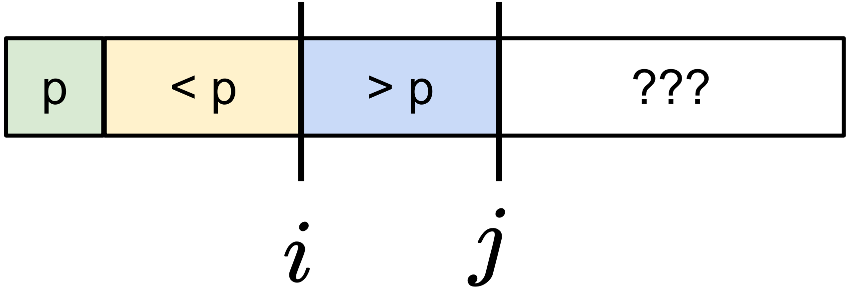


Быстрая сортировка: разбиение



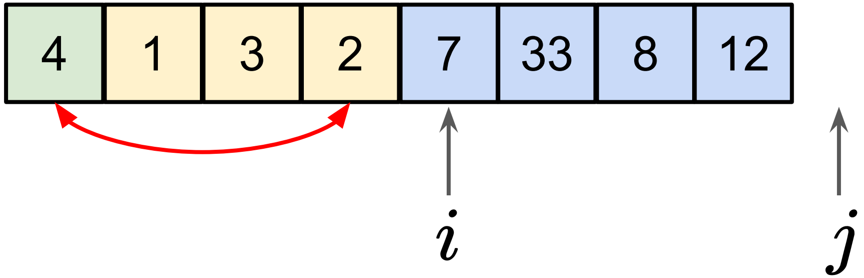
На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i



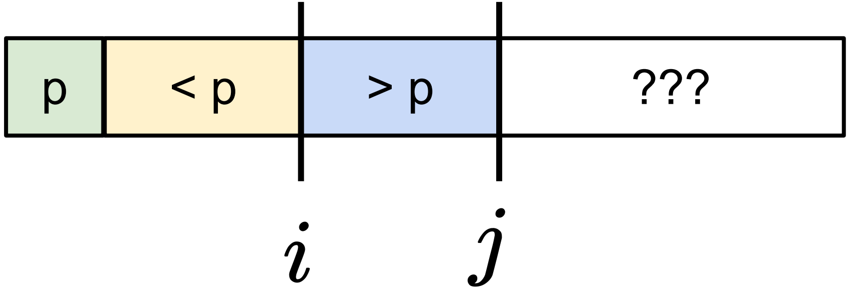
Осталось поставить опорный элемент на его место

Быстрая сортировка: разбиение



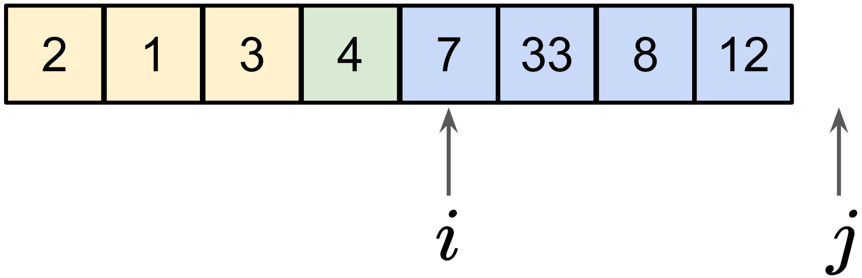
На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i



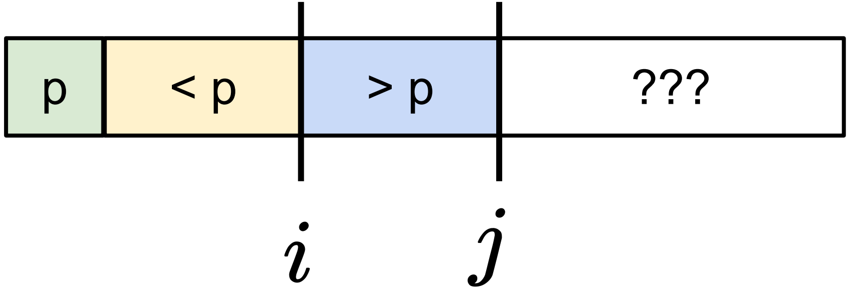
Осталось поставить опорный элемент на его место

Быстрая сортировка: разбиение



На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i



Осталось поставить опорный элемент на его место

Быстрая сортировка

Алгоритм:

1. Выбрать опорный элемент
2. ✓ Выполнить разбиение
3. Рекурсия на левую и правые части разбиения

Быстрая сортировка

Алгоритм:

1. Выбрать опорный элемент
2. Выполнить разбиение
3. Рекурсия на левую и правые части разбиения

Сложность разбиения?

Быстрая сортировка

Алгоритм:

1. Выбрать опорный элемент
2. ✓ Выполнить разбиение
3. Рекурсия на левую и правые части разбиения

Сложность разбиения?

O(N) - отлично!

Быстрая сортировка

Алгоритм:

1. Выбрать опорный элемент
2. Выполнить разбиение
3. Рекурсия на левую и правые части разбиения

Сложность разбиения?

O(N) - отлично!

А как выбрать опорный элемент?

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый

4	7	1	8	3	33	2	12

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

4	7	1	8	3	33	2	12

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

1	2	3	4	12	33	55	68

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

   Сколько будет
   рекурсивных вызовов?

1	2	3	4	12	33	55	68

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

   Сколько будет
   рекурсивных вызовов?

1	2	3	4	12	33	55	68

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

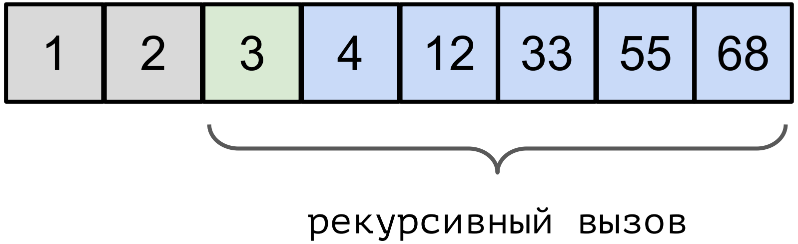
   Сколько будет
   рекурсивных вызовов?

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

   Сколько будет
   рекурсивных вызовов?



Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

   Сколько будет
   рекурсивных вызовов?

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

   Сколько будет
   рекурсивных вызовов?

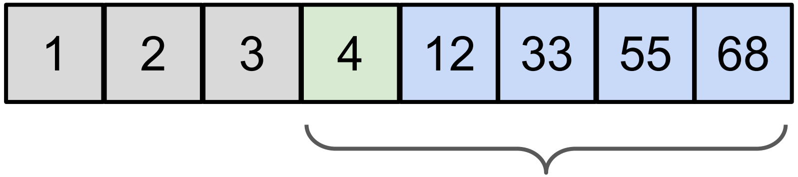
Всего будет n вызовов, каждый раз размер массива уменьшается на 1

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый. Худший случай?

   Сколько будет рекурсивных вызовов?



рекурсивный вызов

Всего будет n вызовов, каждый раз размер массива уменьшается на 1

$$n + (n-1) + \ldots + 1$$

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

   Сколько будет
   рекурсивных вызовов?

Всего будет n вызовов, каждый раз размер массива уменьшается на 1

$$n + (n-1) + \ldots + 1 = \frac{n^2}{2} - \frac{n}{2}$$

$$O(n^2)$$

Быстрая сортировка: выбор опорного элемента

Варианты:

1. Всегда брать первый.
   Худший случай?

   Сколько будет
   рекурсивных вызовов?

1	2	3	4	12	33	55	68

рекурсивный вызов

Всего будет n вызовов, каждый раз размер массива уменьшается на 1

$$n + (n-1) + \ldots + 1 = \frac{n^2}{2} - \frac{n}{2}$$

$$O(n^2)$$

Что-то она не очень то быстрая!

Быстрая сортировка: выбор опорного элемента

Варианты:

1. ~~Всегда брать первый~~

Быстрая сортировка: выбор опорного элемента

Варианты:

1. ~~Всегда брать первый~~

2. Всегда брать элемент, который поделит массив ровно пополам!

2	1	3	4	7	33	8	12

Быстрая сортировка: выбор опорного элемента

Варианты:

1. ~~Всегда брать первый~~

2. Всегда брать элемент, который поделит массив ровно пополам!

2	1	3	4	7	33	8	12

Это бы дало сложность O(N*logN)

Быстрая сортировка: выбор опорного элемента

Варианты:

1. ~~Всегда брать первый~~

2. Всегда брать элемент, который поделит массив ровно пополам!

2	1	3	4	7	33	8	12

Это бы дало сложность O(N*logN)

Но как такой элемент найти?

Быстрая сортировка: выбор опорного элемента

Варианты:

1. ~~Всегда брать первый~~
2. ~~Всегда брать элемент, который поделит массив ровно пополам!~~

Быстрая сортировка: выбор опорного элемента

Варианты:

1. ~~Всегда брать первый~~
2. ~~Всегда брать элемент, который поделит массив ровно пополам!~~
3. Каждый раз брать случайный элемент в качестве опорного!

Быстрая сортировка: выбор опорного элемента

Каждый раз брать случайный элемент в качестве опорного.

Теперь время работы алгоритма зависит не только от входных данных, но и ~~от вашей удачи~~ от того, как повезет с опорными элементами

Быстрая сортировка: выбор опорного элемента

Каждый раз брать случайный элемент в качестве опорного.

Теперь время работы алгоритма зависит не только от входных данных, но и ~~от вашей удачи~~ от того, как повезет с опорными элементами

Сложность в худшем случае?

Быстрая сортировка: выбор опорного элемента

Каждый раз брать случайный элемент в качестве опорного.

Теперь время работы алгоритма зависит не только от входных данных, но и ~~от вашей удачи~~ от того, как повезет с опорными элементами

Сложность в худшем случае? $O(N^2)$

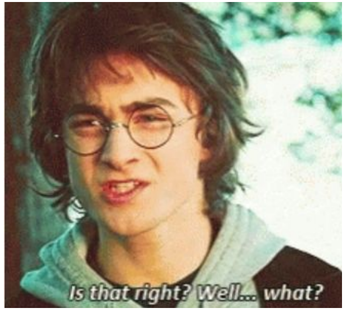
Быстрая сортировка: выбор опорного элемента

Каждый раз брать случайный элемент в качестве опорного.

Теперь время работы алгоритма зависит не только от входных данных, но и ~~от вашей удачи~~ от того, как повезет с опорными элементами

Сложность в худшем случае? $O(N^2)$

Но время работы в среднем $O(N * logN)$



Важные замечания

Сложность в среднем - термин связанный с анализом средней по всем возможным входным данным сложности.

Важные замечания

Сложность в среднем - термин связанный с анализом средней по всем возможным входным данным сложности.

Т.е. ищется ответ на вопрос: чего стоит ожидать чаще всего, если попробовать запустить алгоритм на всех возможных входных данных.

Важные замечания

Сложность в среднем - термин связанный с анализом средней по всем возможным входным данным сложности.

Т.е. ищется ответ на вопрос: чего стоит ожидать чаще всего, если попробовать запустить алгоритм на всех возможных входных данных (включая, например, и худший случай).

Важные замечания

Сложность в среднем - термин связанный с анализом средней по всем возможным входным данным сложности.

Т.е. ищется ответ на вопрос: чего стоит ожидать чаще всего, если попробовать запустить алгоритм на всех возможных входных данных (включая, например, и худший случай).

Мы в этом курсе сосредоточимся на другом - на среднем времени работы рандомизированного алгоритма на любом фиксированном наборе входных данных.

Важные замечания

Т.е. в случае QuickSort фиксируем любой массив и анализируем среднее время работы на множестве всех возможных выборов случайных опорных элементов во время работы алгоритма.

Важные замечания

Т.е. в случае QuickSort фиксируем любой массив и анализируем среднее время работы на множестве всех возможных выборов случайных опорных элементов во время работы алгоритма.

Такой подход:

1. Защищает алгоритм от внешней атаки (никто не сможет подобрать плохой массив для сортировки)

Важные замечания

Т.е. в случае QuickSort фиксируем любой массив и анализируем среднее время работы на множестве всех возможных выборов случайных опорных элементов во время работы алгоритма.

Такой подход:

1. Защищает алгоритм от внешней атаки (никто не сможет подобрать плохой массив для сортировки)

2. Позволяет применить алгоритм k раз, чтобы получить результат с нужной вероятностью (увидим это позже)

Быстрая сортировка

Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

Быстрая сортировка

Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

Замечание: сначала для простоты будем считать, что все элементы массива различны

Немного теории вероятностей #1

Немного теории вероятностей #1

Ω — множество элементарных событий

Немного теории вероятностей #1

Ω – множество элементарных событий (все, что может произойти)

Немного теории вероятностей #1

Ω – множество элементарных событий (все, что может произойти)

$p : \Omega \to [0, 1]$ – отображает элементарные события на их вероятности

$$\sum_{i \in \Omega} p(i) = 1$$

Немного теории вероятностей #1

Ω – множество элементарных событий
(все, что может произойти)

$p : \Omega \to [0, 1]$ – отображает элементарные события на их вероятности

$$\sum_{i \in \Omega} p(i) = 1$$

Пример #1: бросаем монетку, есть всего два исхода:

$$\Omega = \{O, P\}, \forall i \in \Omega : p(i) = \frac{1}{2}$$

Немного теории вероятностей #1

Ω – множество элементарных событий (все, что может произойти)

$p : \Omega \to [0, 1]$ – отображает элементарные события на их вероятности

$$\sum_{i \in \Omega} p(i) = 1$$

Пример #2: бросаем две кости (6 граней).

Какие будут элементарные события и вероятности?

Немного теории вероятностей #1

Ω – множество элементарных событий (все, что может произойти)

$p : \Omega \to [0, 1]$ – отображает элементарные события на их вероятности

$$\sum_{i \in \Omega} p(i) = 1$$

Пример #2: бросаем две кости (6 граней).

$$\Omega = \{(1,1), (1,2), (1,3), \ldots, (6,6)\}$$

$$\forall i \in \Omega : p(i) = \frac{1}{36}$$

Немного теории вероятностей #1

Ω – множество элементарных событий
(все, что может произойти)

$p : \Omega \to [0, 1]$ – отображает элементарные события
на их вероятности

$$\sum_{i \in \Omega} p(i) = 1$$

Пример #3: случайным образом выбираем опорный элемент
в быстрой сортировке массива длины **n**?

Немного теории вероятностей #1

Ω – множество элементарных событий (все, что может произойти)

$p : \Omega \to [0, 1]$ – отображает элементарные события на их вероятности

$$\sum_{i \in \Omega} p(i) = 1$$

Пример #3: случайным образом выбираем опорный элемент в быстрой сортировке массива длины **n**?

$$\Omega = \{0, 1, 2, \ldots, n-1\} \quad \forall i \in \Omega : p(i) = \frac{1}{n}$$

Немного теории вероятностей #2

Событие – подмножество $S \subseteq \Omega$

Немного теории вероятностей #2

Событие – подмножество $S \subseteq \Omega$

Вероятность события S: $Pr[S] = \sum_{i \in S} p(i)$

Немного теории вероятностей #2

Событие – подмножество $S \subseteq \Omega$

Вероятность события S: $Pr[S] = \sum_{i \in S} p(i)$

Пример #4: Событие – при броске двух игральных костей выпала сумма 7

Назовите вероятность такого события?

Немного теории вероятностей #2

Событие – подмножество $S \subseteq \Omega$

Вероятность события S: $Pr[S] = \sum_{i \in S} p(i)$

Пример #4: Событие – при броске двух игральных костей выпала сумма 7

Назовите вероятность такого события?

$$S = \{(1,6), (2,5), (3,4), (4,3), (5,2), (6,1)\}$$

Немного теории вероятностей #2

Событие – подмножество $S \subseteq \Omega$

Вероятность события S: $Pr[S] = \sum_{i \in S} p(i)$

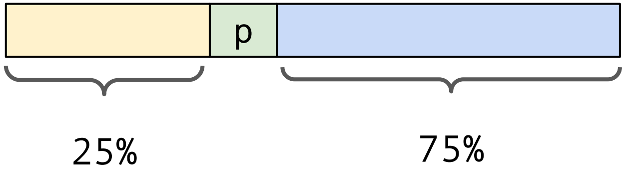
Пример #4: Событие – при броске двух игральных костей выпала сумма 7

Назовите вероятность такого события?

$$S = \{(1,6),(2,5),(3,4),(4,3),(5,2),(6,1)\}$$

$$Pr[S] = \sum_{i \in S} p(i) = \frac{1}{6}$$

Немного теории вероятностей #2



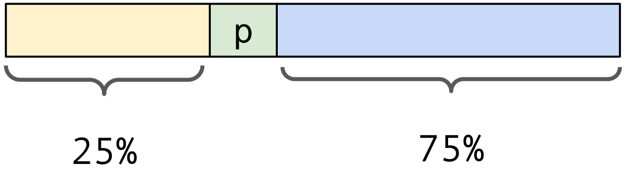
Событие — подмножество $S \subseteq \Omega$

Вероятность события S: $Pr[S] = \sum_{i \in S} p(i)$

Пример #5: Событие — выбор случайного опорного элемента дал разбиение 25% к 75% или лучше (ближе к 50:50)

Назовите вероятность такого события?

Немного теории вероятностей #2



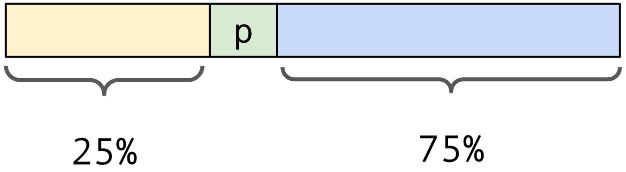
Событие – подмножество $S \subseteq \Omega$

Вероятность события S: $Pr[S] = \sum_{i \in S} p(i)$

Пример #5: Событие – выбор случайного опорного элемента дал разбиение 25% к 75% или лучше (ближе к 50:50)

1	2	3	4	5	6	7	8	9	10	11	12

Немного теории вероятностей #2



Событие – подмножество $S \subseteq \Omega$

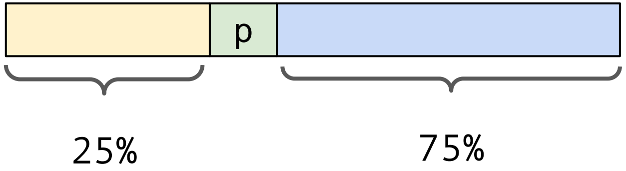
Вероятность события S: $Pr[S] = \sum_{i \in S} p(i)$

Пример #5: Событие – выбор случайного опорного элемента дал разбиение 25% к 75% или лучше (ближе к 50:50)

1	2	3	4	5	6	7	8	9	10	11	12

Попадаем в один из зеленых элементов => разбиение 25:75 или лучше

Немного теории вероятностей #2



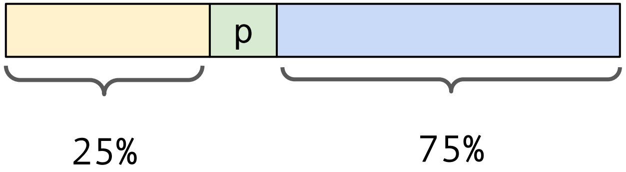
Событие – подмножество $S \subseteq \Omega$

Вероятность события S: $Pr[S] = \sum_{i \in S} p(i)$

Пример #5: Событие – выбор случайного опорного элемента дал разбиение 25% к 75% или лучше (ближе к 50:50)

$S = \{$ выбрали (n/4 + 1)-ый элемент,… , выбрали (3*n/4 - 1)-ый элемент $\}$

Немного теории вероятностей #2



Событие – подмножество $S \subseteq \Omega$

Вероятность события S: $Pr[S] = \sum_{i \in S} p(i)$

Пример #5: Событие – выбор случайного опорного элемента дал разбиение 25% к 75% или лучше (ближе к 50:50)

$$S = \left\{ \begin{array}{l} \text{выбрали (n/4 + 1)-ый элемент,… ,} \\ \text{выбрали (3*n/4 - 1)-ый элемент} \end{array} \right\}$$

$$Pr[S] = \sum_{i \in S} p(i) = \frac{n}{2} * \frac{1}{n} = \frac{1}{2}$$

Немного теории вероятностей #3

Случайная величина – $X : \Omega \to R$

Немного теории вероятностей #3

Случайная величина – $X : \Omega \to R$

Обозначим: $X = a \Leftrightarrow \{s \in \Omega \mid X(s) = a\}$

А его вероятность: $Pr[X = a]$

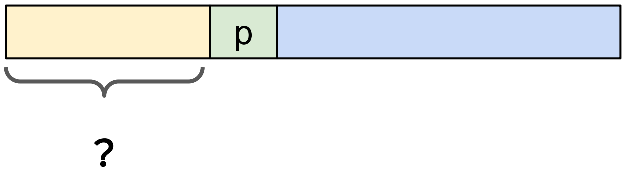
Немного теории вероятностей #3

Случайная величина – $X : \Omega \to R$

Обозначим: $X = a \Leftrightarrow \{s \in \Omega \mid X(s) = a\}$

А его вероятность: $Pr[X = a]$

Пример #6: случайная величина - сумма очков при броске двух костей



Немного теории вероятностей #3

Случайная величина – $X : \Omega \to R$

Обозначим: $X = a \Leftrightarrow \{s \in \Omega \mid X(s) = a\}$

А его вероятность: $Pr[X = a]$

Пример #6: случайная величина - сумма очков при броске двух костей

Пример #7: случайная величина - размер подмассива, который будет передан в первый рекурсивный вызов QuickSort

Немного теории вероятностей #4

Пусть $X : \Omega \to R$ – случайная величина

Тогда: математическое ожидание случайной величины X –

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Немного теории вероятностей #4

Пусть $X : \Omega \to R$ – случайная величина

Тогда: математическое ожидание случайной величины X —

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Физический смысл - среднее значение случайной величины; то, что мы "скорее всего" получим

Немного теории вероятностей #4

Пусть $X : \Omega \to R$ – случайная величина

Тогда: математическое ожидание случайной величины X –

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Пример #8: чему равно мат. ожидание суммы очков при броске двух игральных костей?

Немного теории вероятностей #4

Пусть $X : \Omega \to R$ – случайная величина

Тогда: математическое ожидание случайной величины X –

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Пример #8: чему равно мат. ожидание суммы очков при броске двух игральных костей?

Можно решить честным брутфорсом:
вероятность каждого события p(i) = 1/36

Немного теории вероятностей #4

Пусть $X : \Omega \to R$ – случайная величина

Тогда: математическое ожидание случайной величины X –

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Пример #8: чему равно мат. ожидание суммы очков при броске двух игральных костей?

Можно решить честным брутфорсом:
вероятность каждого события p(i) = 1/36
А дальше считаем все варианты

Немного теории вероятностей #4

Пусть $X : \Omega \to R$ – случайная величина

Тогда: математическое ожидание случайной величины X –

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Пример #8: чему равно мат. ожидание суммы очков при броске двух игральных костей?

$$\mathbb{E}[X] = \frac{1}{36}(1 * 2 + 2 * 3 + \ldots + 1 * 12) = \frac{252}{36} = 7$$

Немного теории вероятностей #4

Пусть $X : \Omega \to R$ – случайная величина

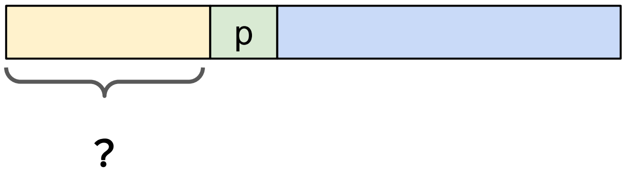
Тогда: математическое ожидание случайной величины X –

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Пример #8: чему равно мат. ожидание суммы очков при броске двух игральных костей?

$$\mathbb{E}[X] = \frac{1}{36}(1 * 2 + 2 * 3 + \ldots + 1 * 12) = \frac{252}{36} = 7$$

но можно это посчитать и умнее (чуть позже)



Немного теории вероятностей #4

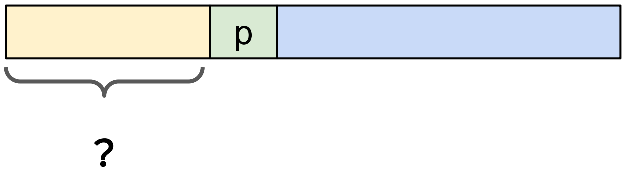
Пусть $X : \Omega \to R$ – случайная величина

Тогда: математическое ожидание случайной величины X –

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Пример #9: чему равно мат. ожидание длины первого подмассива в QuickSort?

Немного теории вероятностей #4



Пусть $X : \Omega \to R$ – случайная величина

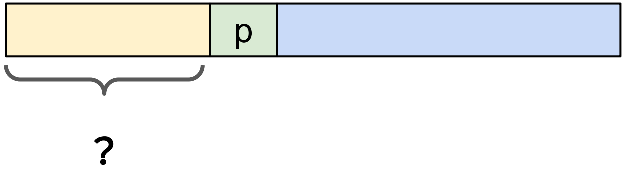
Тогда: математическое ожидание случайной величины X –

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Пример #9: чему равно мат. ожидание длины первого подмассива в QuickSort?

$$\mathbb{E}[X] = \frac{1}{n} * 0 + \frac{1}{n} * 1 + \cdots + \frac{1}{n} * (n-1)$$

Немного теории вероятностей #4



Пусть $X : \Omega \to R$ – случайная величина

Тогда: математическое ожидание случайной величины X —

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Пример #9: чему равно мат. ожидание длины первого подмассива в QuickSort?

$$\mathbb{E}[X] = \frac{1}{n} * 0 + \frac{1}{n} * 1 + \cdots + \frac{1}{n} * (n-1)$$

$$= \frac{1}{n} * (\frac{n^2 - n}{2}) = \frac{n-1}{2}$$

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum\limits_{j=1}^{n} X_j] = \sum\limits_{j=1}^{n} \mathbb{E}[X_j]$

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum_{j=1}^{n} X_j] = \sum_{j=1}^{n} \mathbb{E}[X_j]$

Пример #11: чему равно мат. ожидание суммы очков при броске двух игральных костей? (revisited)

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum\limits_{j=1}^{n} X_j] = \sum\limits_{j=1}^{n} \mathbb{E}[X_j]$

Пример #11: чему равно мат. ожидание суммы очков при броске двух игральных костей? (revisited)

Вводим две новые случайные величины X_1, X_2

Каждая из них - значение, которое выпало на соответствующей кости

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum\limits_{j=1}^{n} X_j] = \sum\limits_{j=1}^{n} \mathbb{E}[X_j]$

Пример #11: чему равно мат. ожидание суммы очков при броске двух игральных костей? (revisited)

Вводим две новые случайные величины X_1, X_2

$$\mathbb{E}[X_1] = \frac{1}{36}(6 * 1 + 6 * 2 + \cdots + 6 * 6)$$

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum\limits_{j=1}^{n} X_j] = \sum\limits_{j=1}^{n} \mathbb{E}[X_j]$

Пример #11: чему равно мат. ожидание суммы очков при броске двух игральных костей? (revisited)

Вводим две новые случайные величины X_1, X_2

$$\mathbb{E}[X_1] = \frac{1}{36}(6 * 1 + 6 * 2 + \cdots + 6 * 6)$$

$$= \frac{1}{6}(1 + 2 + \cdots + 6) = \frac{21}{6} = 3.5$$

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum\limits_{j=1}^{n} X_j] = \sum\limits_{j=1}^{n} \mathbb{E}[X_j]$

Пример #11: чему равно мат. ожидание суммы очков при броске двух игральных костей? (revisited)

Вводим две новые случайные величины X_1, X_2

$$\mathbb{E}[X1] = 3.5; E[X2] = 3.5$$

$$\mathbb{E}[X1 + X2] = \mathbb{E}[X1] + \mathbb{E}[X2] = 7$$

Немного теории вероятностей #4

Пусть $X : \Omega \to R$ – случайная величина

Тогда: математическое ожидание случайной величины X –

$$\mathbb{E}[X] = \sum_{i \in \Omega} X(i) * p(i)$$

Пример #8: чему равно мат. ожидание суммы очков при броске двух игральных костей?

$$\mathbb{E}[X] = \frac{1}{36}(1 * 2 + 2 * 3 + \ldots + 1 * 12) = \frac{252}{36} = 7$$

но можно это посчитать и умнее (чуть позже)

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum_{j=1}^{n} X_j] = \sum_{j=1}^{n} \mathbb{E}[X_j]$

Пример #11: чему равно мат. ожидание суммы очков при броске двух игральных костей? (revisited)

Вводим две новые случайные величины X_1, X_2

$\mathbb{E}[X1] = 3.5; E[X2] = 3.5$

$\mathbb{E}[X1 + X2] = \mathbb{E}[X1] + \mathbb{E}[X2] = 7$

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum\limits_{j=1}^{n} X_j] = \sum\limits_{j=1}^{n} \mathbb{E}[X_j]$

Доказательство:

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum\limits_{j=1}^{n} X_j] = \sum\limits_{j=1}^{n} \mathbb{E}[X_j]$

Доказательство: $\sum\limits_{j=1}^{n} \mathbb{E}[X_j] = \sum\limits_{j=1}^{n} \sum\limits_{i \in \Omega} X_j(i) * p(i)$

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum_{j=1}^{n} X_j] = \sum_{j=1}^{n} \mathbb{E}[X_j]$

Доказательство: $$\sum_{j=1}^{n} \mathbb{E}[X_j] = \sum_{j=1}^{n} \sum_{i \in \Omega} X_j(i) * p(i)$$

$$= \sum_{i \in \Omega} \sum_{j=1}^{n} X_j(i) * p(i)$$

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum\limits_{j=1}^{n} X_j] = \sum\limits_{j=1}^{n} \mathbb{E}[X_j]$

Доказательство: $\sum\limits_{j=1}^{n} \mathbb{E}[X_j] = \sum\limits_{j=1}^{n} \sum\limits_{i \in \Omega} X_j(i) * p(i)$

$$= \sum_{i \in \Omega} \sum_{j=1}^{n} X_j(i) * p(i) = \sum_{i \in \Omega} p(i) \sum_{j=1}^{n} X_j(i)$$

Линейность мат. ожидания

Пусть $X_1, X_2, \ldots, X_n$ – случайные величины на Ω

Тогда: $\mathbb{E}[\sum\limits_{j=1}^{n} X_j] = \sum\limits_{j=1}^{n} \mathbb{E}[X_j]$

Доказательство: $\sum\limits_{j=1}^{n} \mathbb{E}[X_j] = \sum\limits_{j=1}^{n} \sum\limits_{i \in \Omega} X_j(i) * p(i)$

$$= \sum_{i \in \Omega} \sum_{j=1}^{n} X_j(i) * p(i) = \sum_{i \in \Omega} p(i) \sum_{j=1}^{n} X_j(i)$$

$$= \mathbb{E}[\sum_{j=1}^{n} X_j] \quad \square$$

Линейность мат. ожидания

Пример #12: (балансировка нагрузки) пусть есть N серверов и N процессов. Нужно как-то распределить процессы по серверам.

Линейность мат. ожидания

Пример #12: (балансировка нагрузки) пусть есть N серверов и N процессов. Нужно как-то распределить процессы по серверам.

Ленивое решение: каждый процесс будем отправлять на случайный сервер

Линейность мат. ожидания

Пример #12: (балансировка нагрузки) пусть есть N серверов и N процессов. Нужно как-то распределить процессы по серверам.

Ленивое решение: каждый процесс будем отправлять на случайный сервер

Насколько это будет плохо?

Линейность мат. ожидания

Пример #12: (балансировка нагрузки) пусть есть N серверов и N процессов. Нужно как-то распределить процессы по серверам.

Ленивое решение: каждый процесс будем отправлять на случайный сервер

Насколько это будет плохо? Какое ожидаемое количество процессов на каждом сервере?

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

$$\Omega = \{\text{варианты назначений}\}$$

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

$$\Omega = \{\text{варианты назначений}\};\ |\Omega| = N^N; \forall i \in \Omega : p(i) = \frac{1}{N^N}$$

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

$$\Omega = \{\text{варианты назначений}\};\ |\Omega| = N^N; \forall i \in \Omega : p(i) = \frac{1}{N^N}$$

Пусть $Y-$ количество процессов на первом* сервере

т.к. рассуждение для всех серверов одинаковые, то можем взять любой

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

$$\Omega = \{\text{варианты назначений}\};\ |\Omega| = N^N; \forall i \in \Omega : p(i) = \frac{1}{N^N}$$

Пусть $Y-$ количество процессов на первом* сервере

$\mathbb{E}[Y] = ?$

т.к. рассуждение для всех серверов одинаковые, то можем взять любой

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

$$\Omega = \{\text{варианты назначений}\};\ |\Omega| = N^N; \forall i \in \Omega : p(i) = \frac{1}{N^N}$$

Пусть $Y-$ количество процессов на первом* сервере

$\mathbb{E}[Y] =?$

Пусть $X_j = \begin{cases} 1, \text{если процесс } j \text{ на первом сервере} \\ 0, \ \text{иначе} \end{cases}$

т.к. рассуждение для всех серверов одинаковые, то можем взять любой

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

$$\Omega = \{\text{варианты назначений}\};\ |\Omega| = N^N; \forall i \in \Omega : p(i) = \frac{1}{N^N}$$

Пусть $Y-$ количество процессов на первом* сервере

$$\mathbb{E}[Y] = ?$$

Пусть $X_j = \begin{cases} 1, \text{если процесс } j \text{ на первом сервере} \\ 0, \ \text{иначе} \end{cases}$

такие случайные величины называют индикаторными и не редко используют в доказательствах

т.к. рассуждение для всех серверов одинаковые, то можем взять любой

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

Пусть Y – количество процессов на первом* сервере

$\mathbb{E}[Y] = ?$

Пусть $X_j = \begin{cases} 1, \text{если процесс } j \text{ на первом сервере} \\ 0, \ \text{иначе} \end{cases}$

Заметим, что $Y = \sum_{j=1}^{N} X_j$.

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

Пусть Y – количество процессов на первом* сервере

$\mathbb{E}[Y] = ?$

Пусть $X_j = \begin{cases} 1, \text{если процесс } j \text{ на первом сервере} \\ 0, \text{ иначе} \end{cases}$

Заметим, что $Y = \sum\limits_{j=1}^{N} X_j$. Тогда $\mathbb{E}[Y] = \mathbb{E}[\sum\limits_{j=1}^{N} X_j] = \sum\limits_{j=1}^{N} \mathbb{E}[X_j]$

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

Пусть Y – количество процессов на первом* сервере

$\mathbb{E}[Y] = ?$

Пусть $X_j = \begin{cases} 1, \text{если процесс } j \text{ на первом сервере} \\ 0, \text{ иначе} \end{cases}$

Заметим, что $Y = \sum_{j=1}^{N} X_j$. Тогда $\mathbb{E}[Y] = \mathbb{E}[\sum_{j=1}^{N} X_j] = \sum_{j=1}^{N} \mathbb{E}[X_j]$

$$= \sum_{j=1}^{N} (Pr[X_j = 0] * 0 + Pr[X_j = 1] * 1)$$

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

Пусть Y – количество процессов на первом* сервере

$\mathbb{E}[Y] = ?$

Пусть $X_j = \begin{cases} 1, \text{если процесс } j \text{ на первом сервере} \\ 0, \text{ иначе} \end{cases}$

Заметим, что $Y = \sum_{j=1}^{N} X_j$. Тогда $\mathbb{E}[Y] = \mathbb{E}[\sum_{j=1}^{N} X_j] = \sum_{j=1}^{N} \mathbb{E}[X_j]$

$$= \sum_{j=1}^{N} (\cancel{Pr[X_j = 0] * 0} + \boxed{Pr[X_j = 1]} * 1)$$

$\frac{1}{N}$

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

Пусть $Y-$ количество процессов на первом* сервере

$\mathbb{E}[Y] = ?$

Пусть $X_j = \begin{cases} 1, \text{если процесс } j \text{ на первом сервере} \\ 0, \text{ иначе} \end{cases}$

Заметим, что $Y = \sum_{j=1}^{N} X_j$. Тогда

$$\mathbb{E}[Y] = \mathbb{E}[\sum_{j=1}^{N} X_j] = \sum_{j=1}^{N} \mathbb{E}[X_j]$$

$$= \sum_{j=1}^{N} (\cancel{Pr[X_j = 0] * 0} + \overset{\frac{1}{N}}{\boxed{Pr[X_j = 1]}} * 1)$$

$$= \sum_{j=1}^{N} \frac{1}{N} = 1$$

Линейность мат. ожидания

Ленивое решение: каждый процесс будем отправлять на случайный сервер

Пусть $Y-$ количество процессов на первом* сервере

$\mathbb{E}[Y] = ?$

Пусть $X_j = \begin{cases} 1, \text{если процесс } j \text{ на первом сервере} \\ 0, \text{ иначе} \end{cases}$

Заметим, что $Y = \sum_{j=1}^{N} X_j$. Тогда $\mathbb{E}[Y] = \mathbb{E}[\sum_{j=1}^{N} X_j] = \sum_{j=1}^{N} \mathbb{E}[X_j]$

$$= \sum_{j=1}^{N} (\cancel{Pr[X_j = 0] * 0} + \boxed{Pr[X_j = 1]} * 1)$$

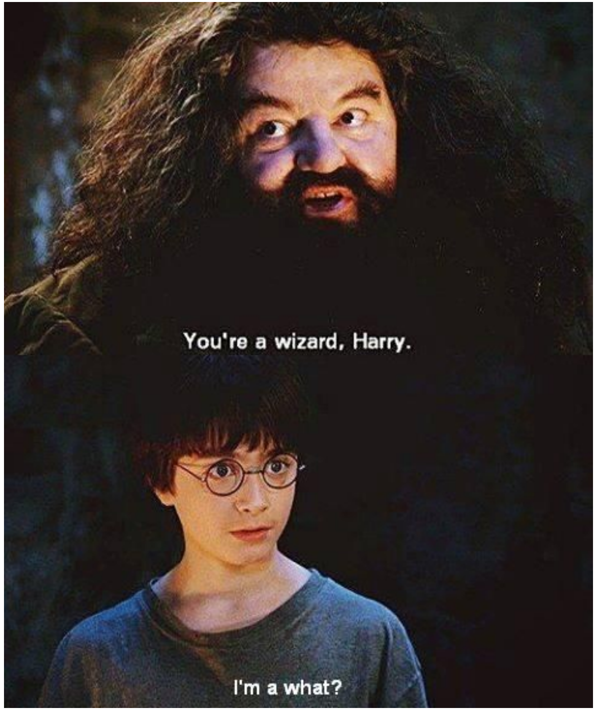
$$\frac{1}{N}$$

$$= \sum_{j=1}^{N} \frac{1}{N} = 1$$

Получается, что не такой уж и плохой алгоритм! В среднем все сбалансированно 😊

Немного теории вероятностей

Вот и все, что нам нужно для доказательства теоремы о времени работы быстрой сортировки!



Быстрая сортировка

Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

Быстрая сортировка

Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

Замечание: сначала для простоты будем считать, что все элементы массива различны

Быстрая сортировка

Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

Лемма: пусть X - количество сравнений двух элементов массива, которое выполняется при полной обработке массива длины **n** за всю работу быстрой сортировки.

Быстрая сортировка

Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

Лемма: пусть X - количество сравнений двух элементов массива, которое выполняется при полной обработке массива длины **n** за всю работу быстрой сортировки.

тогда время работы быстрой сортировки O(n + X)

Быстрая сортировка

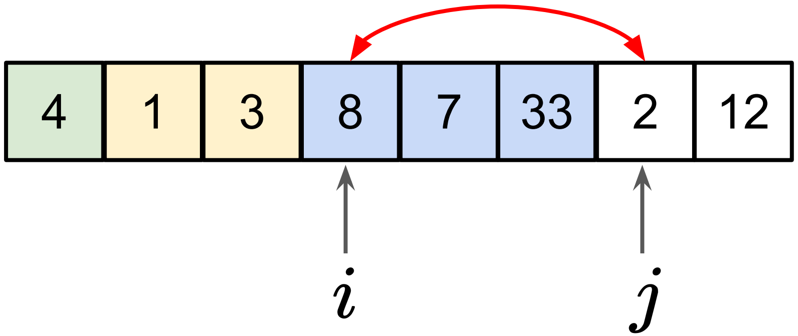
Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

Лемма: пусть X - количество сравнений двух элементов массива, которое выполняется при полной обработке массива длины **n** за всю работу быстрой сортировки.

тогда время работы быстрой сортировки O(n + X)

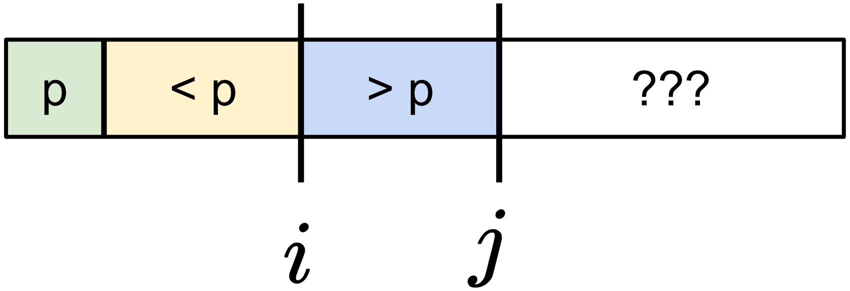
Доказательство: упражнение; идея: аккуратно оценить операции в процедуре разбиения

Быстрая сортировка: разбиение

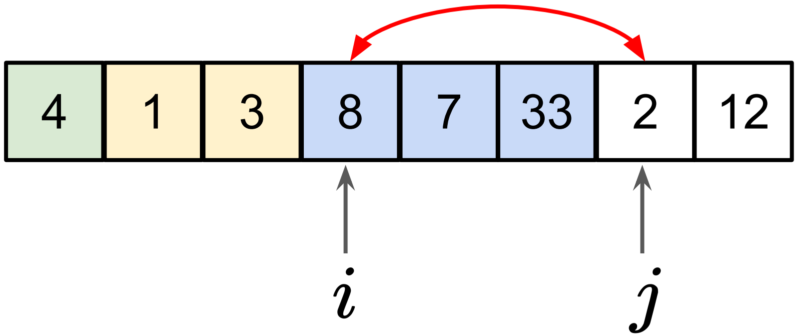


На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

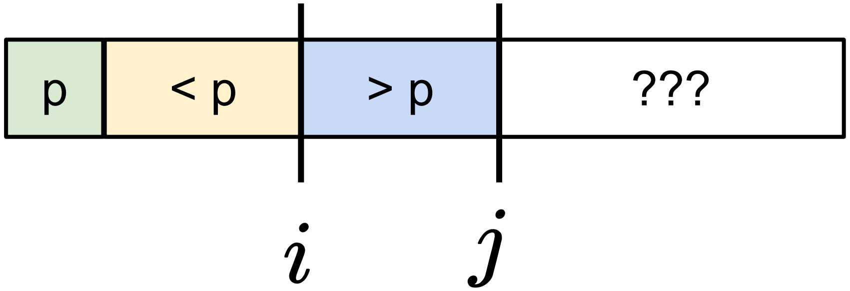


Быстрая сортировка: разбиение



На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i



Проверка на нарушение - это и есть сравнение элементов с опорным, чаще всего этим и занимаемся

Быстрая сортировка

Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

Лемма: пусть X - количество сравнений двух элементов массива, которое выполняется при полной обработке массива длины **n** за всю работу быстрой сортировки.

тогда время работы быстрой сортировки O(n + X)

Так что для доказательства теоремы будем оценивать количество сравнений элементов массива.

Быстрая сортировка

Фиксируем входной некоторый массив A длины n.

Быстрая сортировка

Фиксируем входной некоторый массив A длины n.

$$\Omega = \{\text{последовательности выбранных опорных элементов}\}$$

Быстрая сортировка

Фиксируем входной некоторый массив A длины n.

$$\Omega = \{\text{последовательности выбранных опорных элементов}\}$$

Для $\sigma \in \Omega : C(\sigma) =$ количество сравнений элементов массива A за все время работы быстрой сортировки при условии, что были выбраны опорные элементы из σ

Быстрая сортировка

Фиксируем входной некоторый массив A длины n.

$$\Omega = \{\text{последовательности выбранных опорных элементов}\}$$

Для $\sigma \in \Omega : C(\sigma) =$ количество сравнений элементов массива A за все время работы быстрой сортировки при условии, что были выбраны опорные элементы из σ

$$\mathbb{E}[C] = \ ?$$

Быстрая сортировка

Введём обозначение: z_i — i-ый по величине элемент в массиве

2	1	3	4	7	33	8	12
z_1	z_0	z_2	z_3	z_4	z_7	z_5	z_6

Быстрая сортировка

Введём обозначение: z_i — i-ый по величине элемент в массиве

2	1	3	4	7	33	8	12
z_1	z_0	z_2	z_3	z_4	z_7	z_5	z_6

Введём новые случайные величины:

для $i < j, \sigma \in \Omega : X_{ij}(\sigma) =$ количество сравнений элементов z_i и z_j за всю работу быстрой сортировки с последовательностью опорных элементов σ

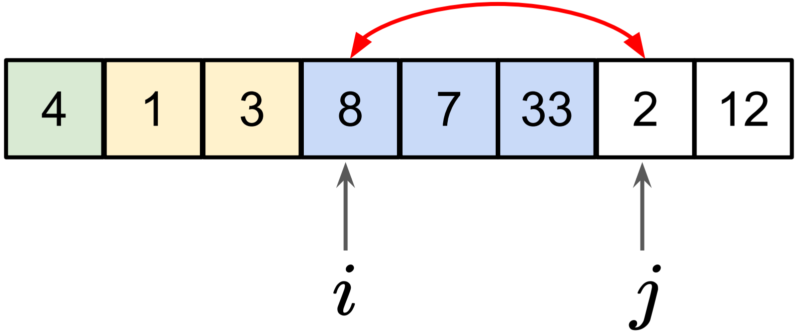
Быстрая сортировка

Введём новые случайные величины:

для $i < j, \sigma \in \Omega : X_{ij}(\sigma) =$ количество сравнений элементов z_i и z_j за всю работу быстрой сортировки с последовательностью опорных элементов σ

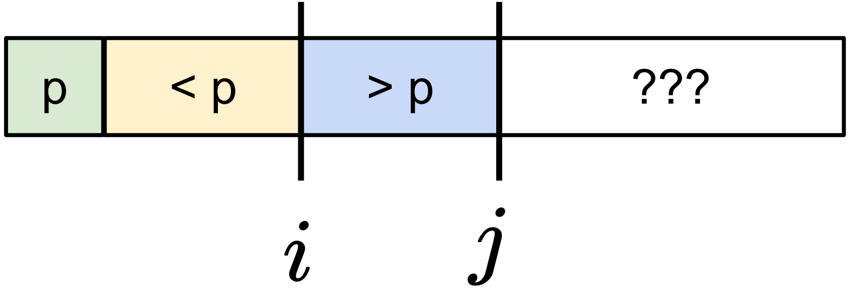
А чему вообще может быть равно $X_{ij}(\sigma)$?

Быстрая сортировка: разбиение



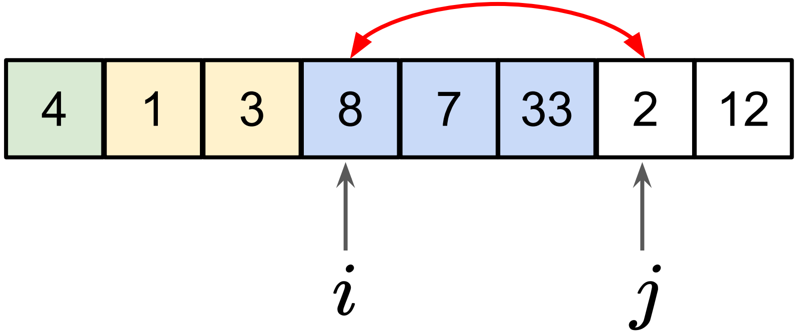
На каждом шаге алгоритма двигаем j вперед

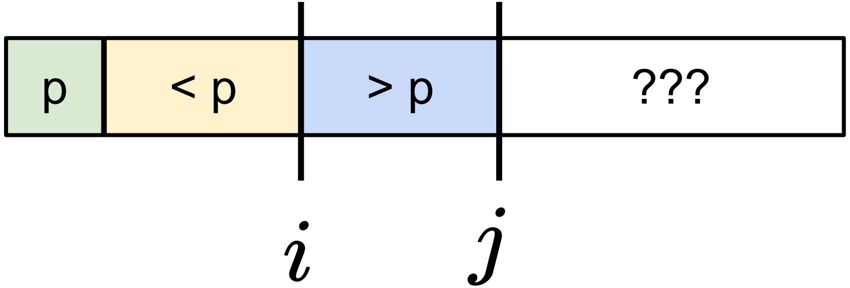
Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i



Проверка на нарушение - это и есть сравнение элементов с опорным, чаще всего этим и занимаемся

Быстрая сортировка: разбиение

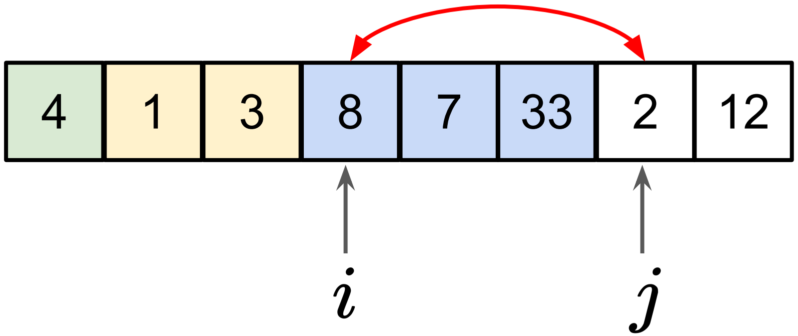




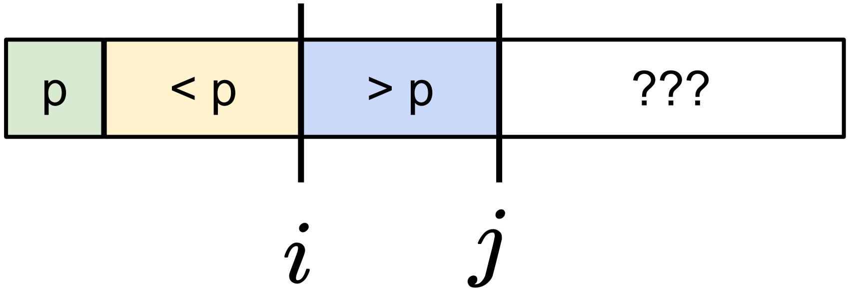
Проверка на нарушение - это и есть сравнение элементов с опорным, чаще всего этим и занимаемся

Т.е. если один из z_i и z_j выбрали в качестве опорного, а второй был в подмассиве, то сравнение случилось.

Быстрая сортировка: разбиение



Проверка на нарушение - это и есть сравнение элементов с опорным, чаще всего этим и занимаемся



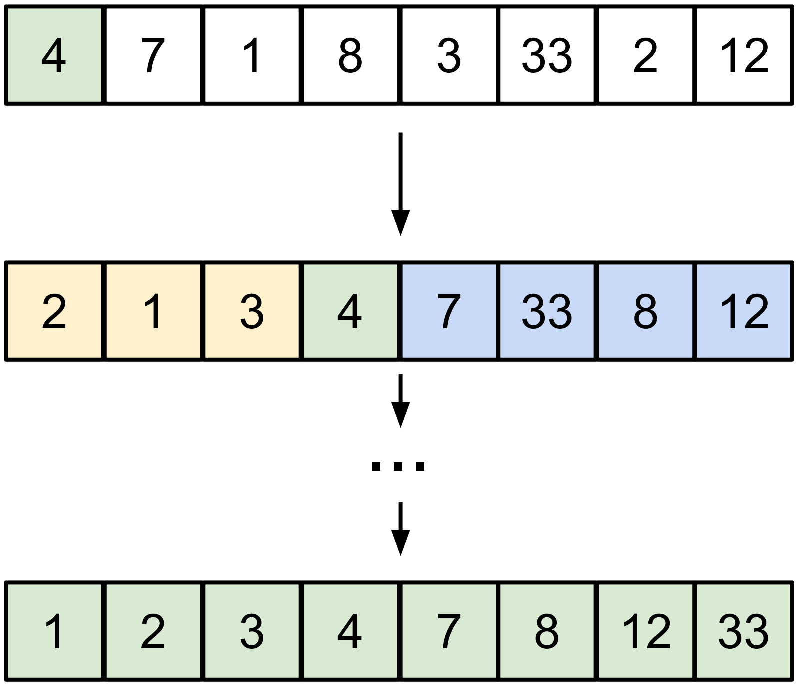
Т.е. если один из z_i и z_j выбрали в качестве опорного, а второй был в подмассиве, то сравнение случилось.

А как еще может быть?

Быстрая сортировка

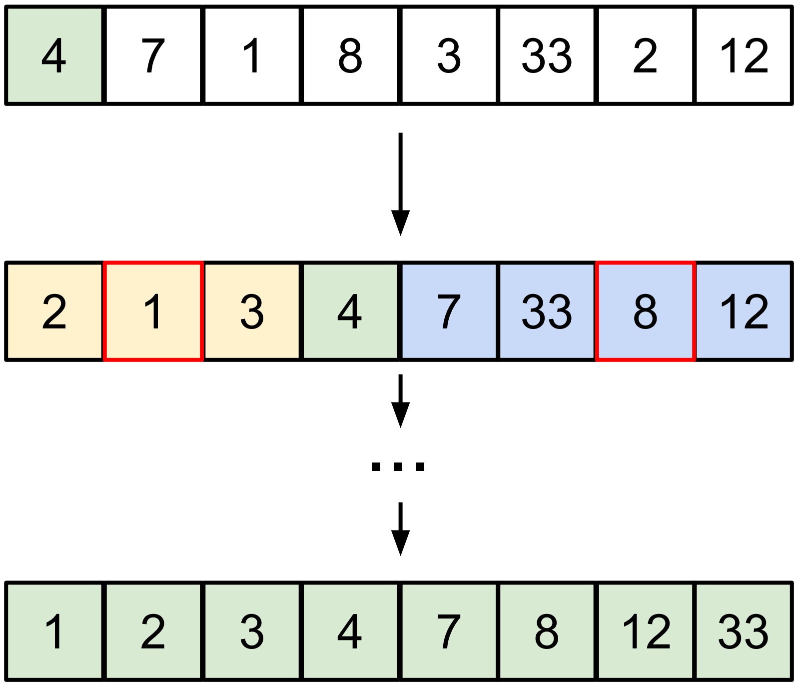
Алгоритм:

1. Выбрать опорный элемент
2. Выполнить разбиение
3. Рекурсия на левую и правые части разбиения



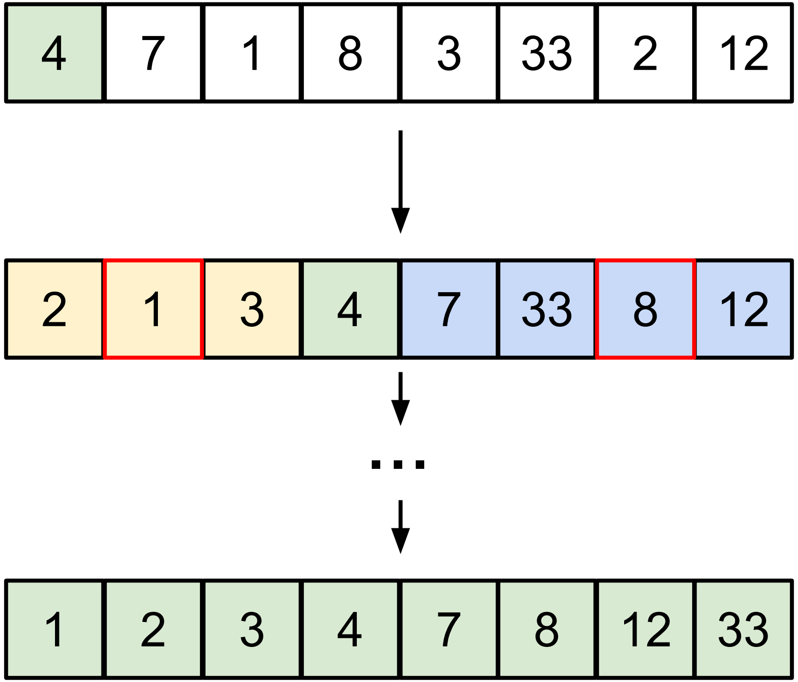
Быстрая сортировка

Элементы 1 и 8 остались по разные стороны от опорного, а значит никогда больше не будут сравниваться.



Быстрая сортировка

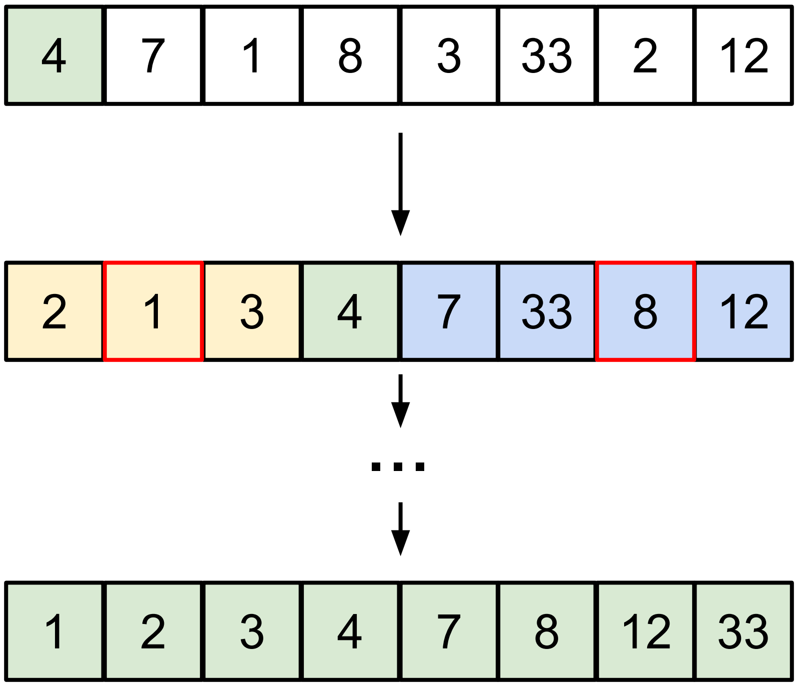
Элементы 1 и 8 остались по разные стороны от опорного, а значит **никогда** больше не будут сравниваться.



Быстрая сортировка

Элементы 1 и 8 остались по разные стороны от опорного, а значит никогда больше не будут сравниваться.

До этого момента они тоже не сравнивались, т.к. никто из них не был опорным элементом.



Быстрая сортировка

Введём новые случайные величины:

для $i < j, \sigma \in \Omega : X_{ij}(\sigma) =$ количество сравнений элементов z_i и z_j за всю работу быстрой сортировки с последовательностью опорных элементов σ

А чему вообще может быть равно $X_{ij}(\sigma)$?

Либо 0, либо 1. Значит что это за случайная величина?

Быстрая сортировка

Введём новые случайные величины:

для $i < j, \sigma \in \Omega : X_{ij}(\sigma) =$ количество сравнений элементов z_i и z_j за всю работу быстрой сортировки с последовательностью опорных элементов σ

А чему вообще может быть равно $X_{ij}(\sigma)$?

Либо 0, либо 1. Значит что это за случайная величина? Индикаторная!

Теперь:

$$X_{ij}(\sigma) = \text{количество сравнений элементов } z_i \text{ и } z_j$$

$$C(\sigma) = \text{количество сравнений всех элементов}$$

Как они связаны?

Теперь:

$X_{ij}(\sigma) =$ количество сравнений элементов z_i и z_j

$C(\sigma) =$ количество сравнений всех элементов

Как они связаны?

$$\forall \sigma \in \Omega : C(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}(\sigma)$$

Теперь:

$X_{ij}(\sigma) =$ количество сравнений элементов z_i и z_j

$C(\sigma) =$ количество сравнений всех элементов

Как они связаны?

$$\forall \sigma \in \Omega : C(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}(\sigma)$$

Тогда: $\mathbb{E}[C] = \mathbb{E}[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}]$

(по свойству линейности мат. ожидания)

Теперь:

$X_{ij}(\sigma) =$ количество сравнений элементов z_i и z_j

$C(\sigma) =$ количество сравнений всех элементов

Как они связаны?

$$\forall \sigma \in \Omega : C(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}(\sigma)$$

Тогда: $\mathbb{E}[C] = \mathbb{E}[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}]$ (по свойству линейности мат. ожидания)

При этом: $\mathbb{E}[X_{ij}] = \cancel{0 * Pr[X_{ij} = 0]} + 1 * Pr[X_{ij} = 1]$

Теперь:

$X_{ij}(\sigma) =$ количество сравнений элементов z_i и z_j

$C(\sigma) =$ количество сравнений всех элементов

Как они связаны?

$$\forall \sigma \in \Omega : C(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}(\sigma)$$

Тогда: $\mathbb{E}[C] = \mathbb{E}[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}]$ (по свойству линейности мат. ожидания)

При этом: $\mathbb{E}[X_{ij}] = \cancel{0 * Pr[X_{ij} = 0]} + 1 * Pr[X_{ij} = 1]$

Поэтому: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

Теперь:

$$X_{ij}(\sigma) = \text{количество сравнений элементов } z_i \text{ и } z_j$$

$$C(\sigma) = \text{количество сравнений всех элементов}$$

Как они связаны?

$$\forall \sigma \in \Omega : C(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}(\sigma)$$

Тогда: $\mathbb{E}[C] = \mathbb{E}[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}]$ (по свойству линейности мат. ожидания)

При этом: $\mathbb{E}[X_{ij}] = \cancel{0 * Pr[X_{ij} = 0]} + 1 * Pr[X_{ij} = 1]$

Поэтому: $\boxed{\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]}$

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

2	1	3	4	7	33	8	12
z_1	z_0	z_2	z_3	z_4	z_7	z_5	z_6

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

2	1	3	4	7	33	8	12
z_1	z_0	z_2	z_3	z_4	z_7	z_5	z_6

Зафиксируем z_i и $z_j : i < j$

И рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

2	1	3	4	7	33	8	12
z_1	z_0	z_2	z_3	z_4	z_7	z_5	z_6

Зафиксируем z_i и $z_j : i < j$

И рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i \text{ или } k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

2	1	3	4	7	33	8	12
z_1	z_0	z_2	z_3	z_4	z_7	z_5	z_6

Зафиксируем z_i и $z_j : i < j$

И рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i$ или $k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

впервые это изменится, когда в качестве опорного будет выбран кто-то из самих этих элементов $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i$ или $k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

впервые это изменится, когда в кач-ве опорного будет выбран кто-то из $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i \text{ или } k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

впервые это изменится, когда в кач-ве опорного будет выбран кто-то из $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Варианты: 1) впервые выбрали опорным z_i или z_j

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i$ или $k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

впервые это изменится, когда в кач-ве опорного будет выбран кто-то из $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Варианты: 1) впервые выбрали опорным z_i или z_j
тогда эти элементы сравниваются (с опорным элементом сравниваются все остальные)

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i$ или $k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

впервые это изменится, когда в кач-ве опорного будет выбран кто-то из $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Варианты: 1) впервые выбрали опорным z_i или z_j тогда эти элементы сравниваются (с опорным элементом сравниваются все остальные)

2) впервые выбрали опорным элемент между z_i и z_j

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i$ или $k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

впервые это изменится, когда в кач-ве опорного будет выбран кто-то из $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Варианты: 1) впервые выбрали опорным z_i или z_j
тогда эти элементы сравниваются (с опорным элементом сравниваются все остальные)

2) впервые выбрали опорным элемент между z_i и z_j
тогда эти элементы попадут в разные подмассивы и никогда больше не сравнятся

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Вопрос: чему равна вероятность того, что в множестве $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ впервые опорным элементом выберут z_i или z_j?

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Вопрос: чему равна вероятность того, что в множестве $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ впервые опорным элементом выберут z_i или z_j?

Всего элементов $j - i + 1$

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Вопрос: чему равна вероятность того, что в множестве $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ впервые опорным элементом выберут z_i или z_j?

Всего элементов $j - i + 1$

Все элементы выбираются в качестве опорных с равной вероятностью.

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Вопрос: чему равна вероятность того, что в множестве $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ впервые опорным элементом выберут z_i или z_j?

Всего элементов $j - i + 1$

Все элементы выбираются в качестве опорных с равной вероятностью.

Вероятность выбора любого из элементов: $\frac{1}{j-i+1}$

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Вопрос: чему равна вероятность того, что в множестве $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ впервые опорным элементом выберут z_i или z_j?

Всего элементов $j - i + 1$

Все элементы выбираются в качестве опорных с равной вероятностью.

Вероятность выбора любого из элементов: $\frac{1}{j-i+1}$

Тогда впервые выбрали z_i или z_j с вероятностью $\frac{2}{j-i+1}$

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Вопрос: чему равна вероятность того, что в множестве $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ впервые опорным элементом выберут z_i или z_j?

Всего элементов $j - i + 1$

Все элементы выбираются в качестве опорных с равной вероятностью.

Вероятность выбора любого из элементов: $\frac{1}{j-i+1}$

Тогда впервые выбрали z_i или z_j $\boxed{\text{с вероятностью } \frac{2}{j-i+1}}$

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i$ или $k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

впервые это изменится, когда в кач-ве опорного будет выбран кто-то из них $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Варианты:

1) впервые выбрали опорным z_i или z_j тогда эти элементы сравниваются (с опорным элементом сравниваются все остальные)

2) впервые выбрали опорным элемент между z_i и z_j тогда эти элементы попадут в разные подмассивы и никогда больше не сравнятся

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Теперь:

$X_{ij}(\sigma) =$ количество сравнений элементов z_i и z_j

$C(\sigma) =$ количество сравнений всех элементов

Как они связаны?

$$\forall \sigma \in \Omega : C(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}(\sigma)$$

Тогда: $\mathbb{E}[C] = \mathbb{E}[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}]$ (по свойству линейности мат. ожидания)

При этом: $\mathbb{E}[X_{ij}] = \cancel{0 * Pr[X_{ij} = 0]} + 1 * Pr[X_{ij} = 1]$

Поэтому: $\boxed{\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]}$

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

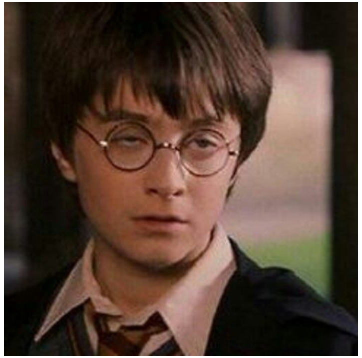
А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Осталось совсем чуть-чуть!



А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$ Сделаем замену: $k = j - i$

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$ Сделаем замену: $k = j - i$

Тогда: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} <$

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Сделаем замену: $k = j - i$

Тогда: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} < 2 * \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k}$

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Сделаем замену: $k = j - i$

Тогда: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} < 2 * \sum_{i=1}^{n-1} \boxed{\sum_{k=1}^{n} \frac{1}{k}}$

Что можно сказать
про такую сумму?

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$= \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} \frac{2}{j-i+1}$ Сделаем замену: $k = j - i$

Тогда: $\mathbb{E}[C] = \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} \frac{2}{j-i+1} = \sum\limits_{i=1}^{n-1} \sum\limits_{k=1}^{n-i} \frac{2}{k+1} < 2 * \sum\limits_{i=1}^{n-1} \boxed{\sum\limits_{k=1}^{n} \frac{1}{k}}$

Что можно сказать про такую сумму?

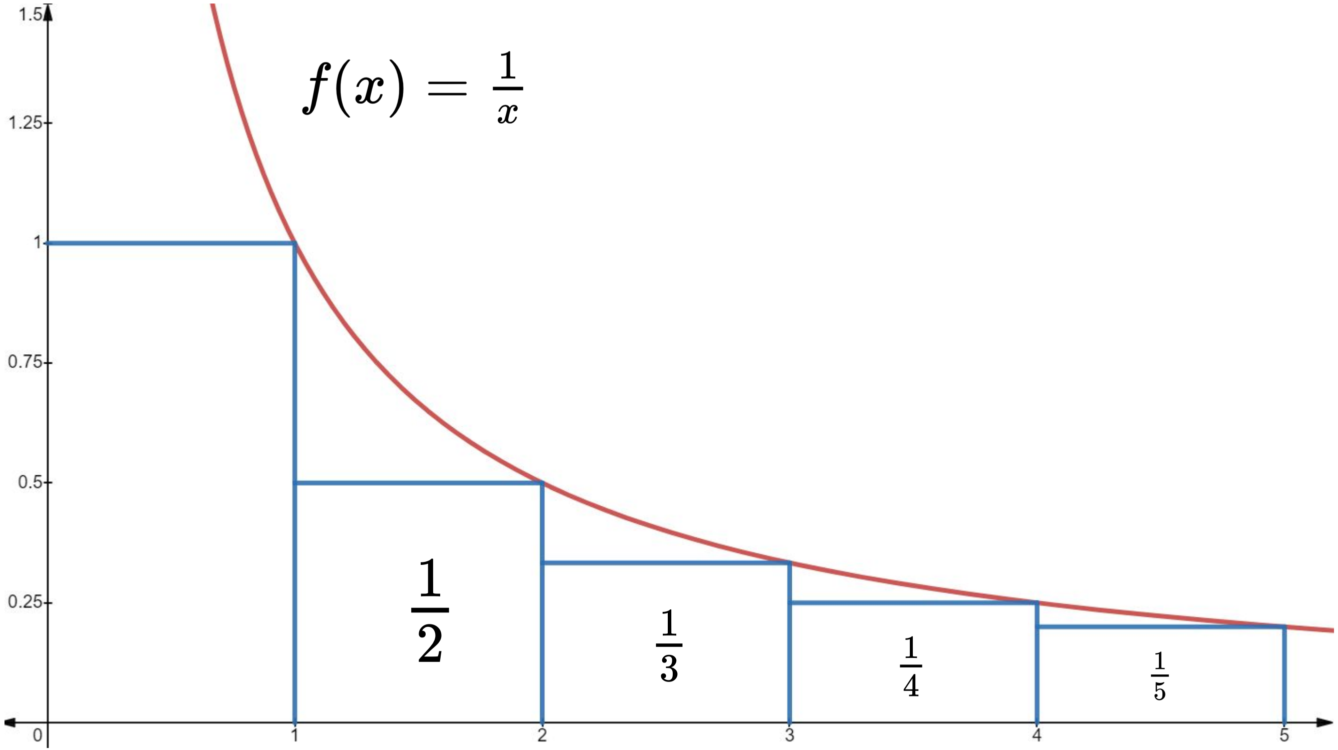
Да это же n-ое гармоническое число!

$$\sum_{k=1}^{n} \frac{1}{k}$$

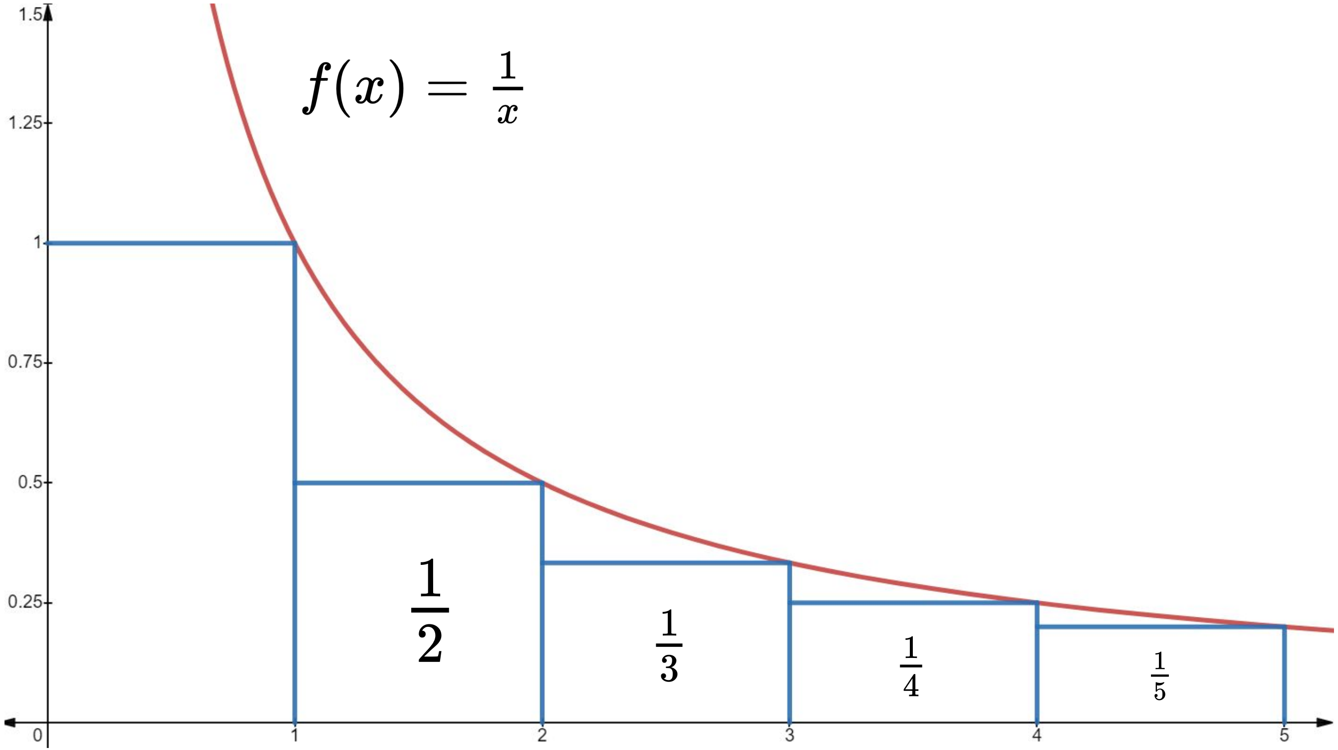
$$\sum_{k=1}^{n} \frac{1}{k} = 1 + (\sum_{k=2}^{n} \frac{1}{k})$$

$$\sum_{k=1}^{n} \frac{1}{k} = 1 + \boxed{(\sum_{k=2}^{n} \frac{1}{k})}$$

$$\sum_{k=1}^{n} \frac{1}{k} = 1 + \boxed{\left(\sum_{k=2}^{n} \frac{1}{k}\right)}$$



$$\sum_{k=1}^{n} \frac{1}{k} = 1 + (\sum_{k=2}^{n} \frac{1}{k}) \leq 1 + \int_1^n \frac{1}{x} dx \ = 1 + ln(n) - ln(1) = 1 + ln(n)$$



А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$= \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} \frac{2}{j-i+1}$ Сделаем замену: $k = j - i$

Тогда: $\mathbb{E}[C] = \sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} \frac{2}{j-i+1} = \sum\limits_{i=1}^{n-1} \sum\limits_{k=1}^{n-i} \frac{2}{k+1} < 2 * \sum\limits_{i=1}^{n-1} \boxed{\sum\limits_{k=1}^{n} \frac{1}{k}}$

Что можно сказать
про такую сумму?

Да это же n-ое гармоническое число!

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Сделаем замену: $k = j - i$

Тогда: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} < 2 * \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k}$

$$\mathbb{E}[C] < 2 * \sum_{i=1}^{n-1} *(1 + ln(n)) < 2 * n * (1 + ln(n))$$

А чему равна вероятность: $Pr[z_i, z_j \text{ сравнивались}]$?

Из соображений выше: $Pr[z_i, z_j \text{ сравнивались}] = \frac{2}{j-i+1}$

Поэтому: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr[z_i, z_j \text{ сравнивались}]$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Сделаем замену: $k = j - i$

Тогда: $\mathbb{E}[C] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} < 2 * \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k}$

$$\mathbb{E}[C] < 2 * \sum_{i=1}^{n-1} * (1 + ln(n)) < 2 * n * (1 + ln(n)) = O(n * ln(n))$$

Быстрая сортировка

Фиксируем входной некоторый массив A длины n.

$$\Omega = \{\text{последовательности выбранных опорных элементов}\}$$

Для $\sigma \in \Omega : C(\sigma) =$ количество сравнений элементов массива A за все время работы быстрой сортировки при условии, что были выбраны опорные элементы из σ

$$\mathbb{E}[C] = \ ?$$

Быстрая сортировка

Фиксируем входной некоторый массив A длины n.

$$\Omega = \{\text{последовательности выбранных опорных элементов}\}$$

Для $\sigma \in \Omega : C(\sigma) =$ количество сравнений элементов массива A за все время работы быстрой сортировки при условии, что были выбраны опорные элементы из σ

$$\mathbb{E}[C] = O(n * ln(n)) \quad \square$$

Быстрая сортировка

Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

Практическое значение: в среднем рандомизированная быстрая сортировка работает хорошо на любом примере

Быстрая сортировка

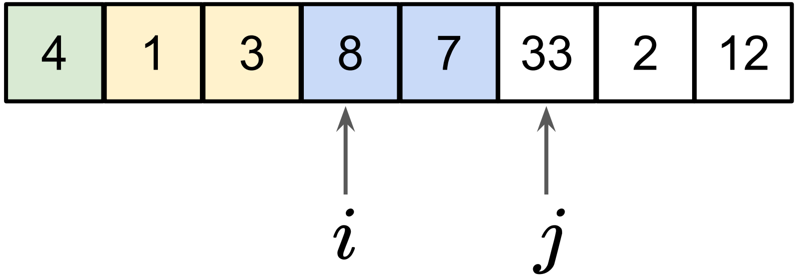
Теорема: для любого входного массива длины N среднее время работы алгоритма быстрой сортировки (со случайным выбором опорных элементов) равно O(N*logN)

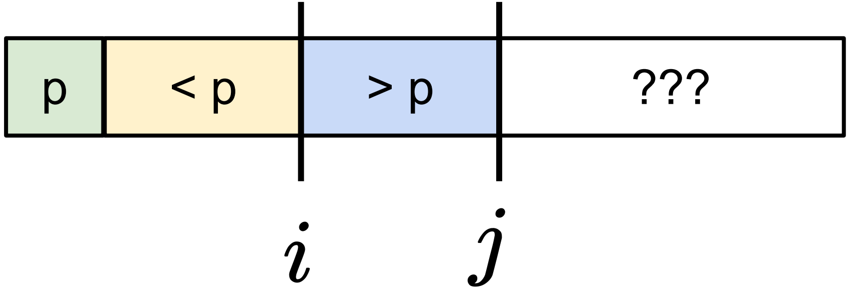
Практическое значение: в среднем рандомизированная быстрая сортировка работает хорошо на любом примере

Плохие выборки случайных опорных элементов встречаются, поэтому быстрая сортировка не подойдет для задач, требующих гарантированного времени работы (кардиостимуляторы, полеты в космос и т.д.)

Быстрая сортировка: практические замечания

Быстрая сортировка: разбиение

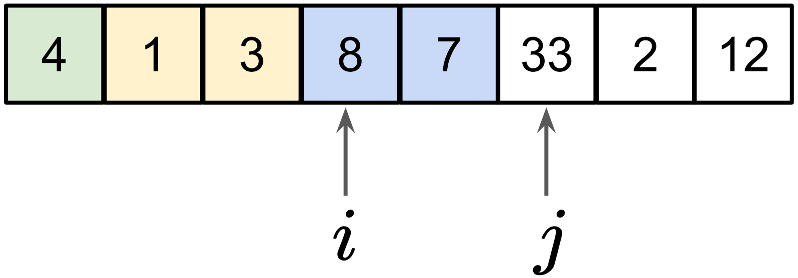




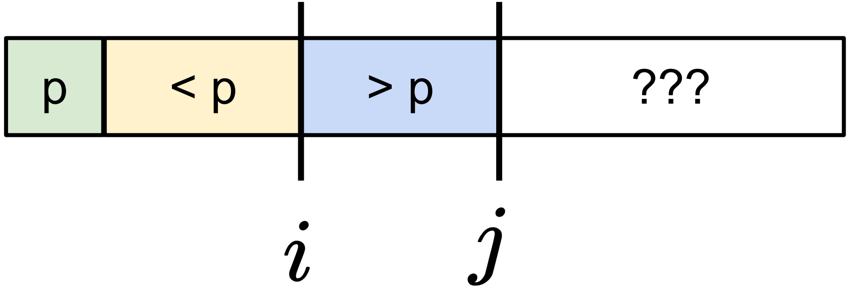
На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i

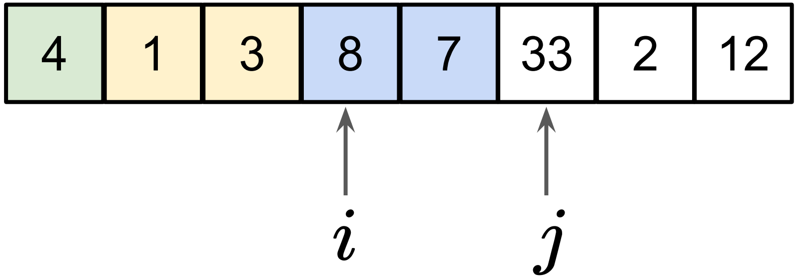
Быстрая сортировка: разбиение

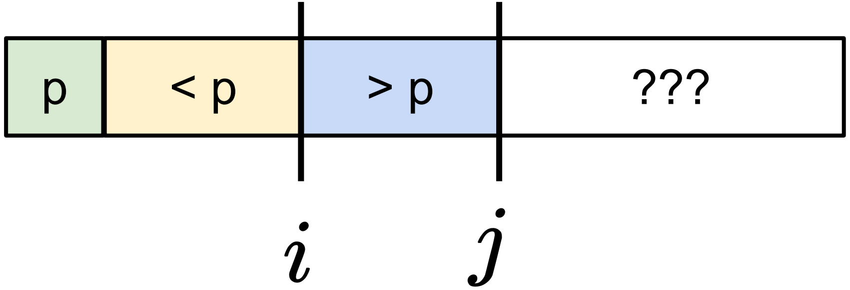


Такая схема называется разбиением Ломуто



Быстрая сортировка: разбиение





Такая схема называется разбиением Ломуто

Пусть теперь в массиве могут быть совпадающие элементы.

Приведите пример массива, где разбиение Ломуто приведёт к гарантированной деградации до O(n^2)

Быстрая сортировка: разбиение

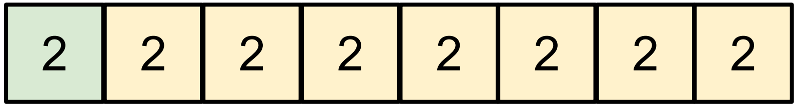
2	2	2	2	2	2	2	2

Такая схема называется разбиением Ломуто

Пусть теперь в массиве могут быть совпадающие элементы.

Приведите пример массива, где разбиение Ломуто приведёт к гарантированной деградации до O(n^2)

Быстрая сортировка: разбиение



Такая схема называется разбиением Ломуто

Пусть теперь в массиве могут быть совпадающие элементы.

Приведите пример массива, где разбиение Ломуто приведёт к гарантированной деградации до O(n^2)

Быстрая сортировка: разбиение

2	2	2	2	2	2	2	2

2	2	2	2	2	2	2

Такая схема называется разбиением Ломуто

Пусть теперь в массиве могут быть совпадающие элементы.

Приведите пример массива, где разбиение Ломуто приведёт к гарантированной деградации до O(n^2)

Быстрая сортировка: разбиение

2	2	2	2	2	2	2	2

2	2	2	2	2	2	2

2	2	2	2	2	2

Такая схема называется разбиением Ломуто

Пусть теперь в массиве могут быть совпадающие элементы.

Приведите пример массива, где разбиение Ломуто приведёт к гарантированной деградации до O(n^2)

Быстрая сортировка: разбиение

2	2	2	2	2	2	2	2

2	2	2	2	2	2	2

2	2	2	2	2	2

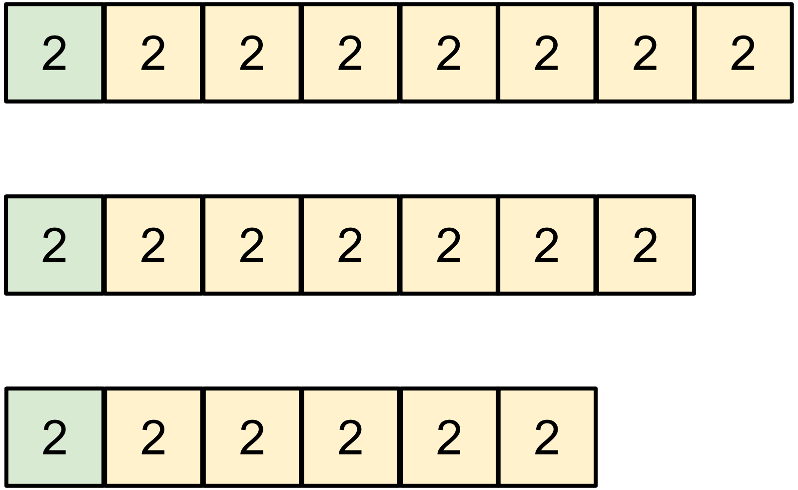
Очевидно деградирует в квадратичную сложность

Такая схема называется разбиением Ломуто

Пусть теперь в массиве могут быть совпадающие элементы.

Приведите пример массива, где разбиение Ломуто приведёт к гарантированной деградации до O(n^2)

Быстрая сортировка: разбиение



Такая схема называется разбиением Ломуто

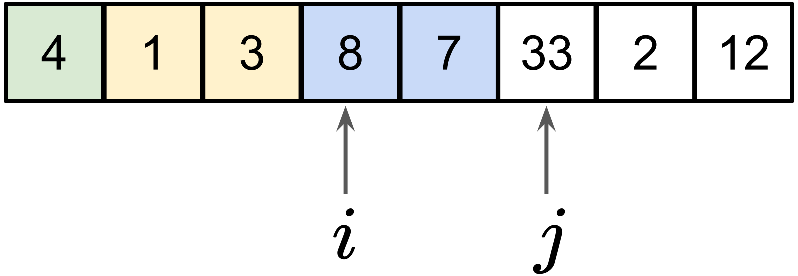
Пусть теперь в массиве могут быть совпадающие элементы.

Что же пошло не так, мы же доказали теорему?

Где мы опирались на различие элементов?

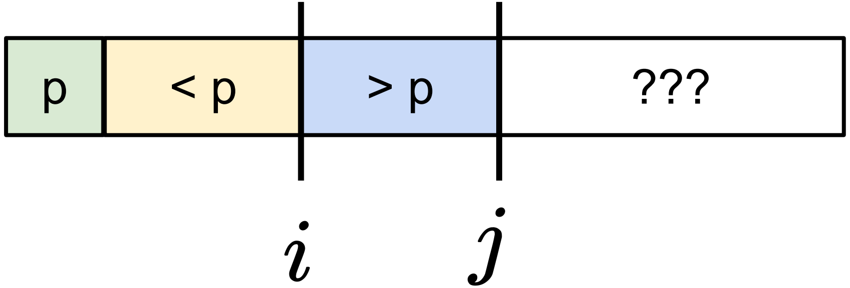
Очевидно деградирует в квадратичную сложность

Быстрая сортировка: разбиение



На каждом шаге алгоритма двигаем j вперед

Видим нарушение инварианта => меняем a[i] и a[j] и двигаем i



Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i$ или $k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

впервые это изменится, когда в кач-ве опорного будет выбран кто-то из $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Варианты: 1) впервые выбрали опорным z_i или z_j тогда эти элементы сравниваются (с опорным элементом сравниваются все остальные)

2) впервые выбрали опорным элемент между z_i и z_j тогда эти элементы попадут в разные подмассивы и никогда больше не сравнятся

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Наблюдение: до тех пор, пока опорным выбирается элемент $z_k : k < i$ или $k > j$ элементы $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$ остаются в одном подмассиве

впервые это изменится, когда в кач-ве опорного будет выбран кто-то из $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Варианты: 1) впервые выбрали опорным z_i или z_j тогда эти элементы сравниваются (с опорным элементом сравниваются все остальные)

2) впервые выбрали опорным элемент между z_i и z_j тогда эти элементы попадут в разные подмассивы и никогда больше не сравнятся

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Допустим, что $z_i = z_{i+1} = \cdots = z_j$

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Допустим, что $z_i = z_{i+1} = \cdots = z_j$

Тогда из выбора элемента $z_k : i < k < j$ совершенно не следует, что z_i и z_j больше никогда не будут сравниваться!

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Допустим, что $z_i = z_{i+1} = \cdots = z_j$

Тогда из выбора элемента $z_k : i < k < j$ совершенно не следует, что z_i и z_j больше никогда не будут сравниваться!

Совсем наоборот: они попадут в один подмассив, а значит рассуждения на этом заканчивать нельзя.

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

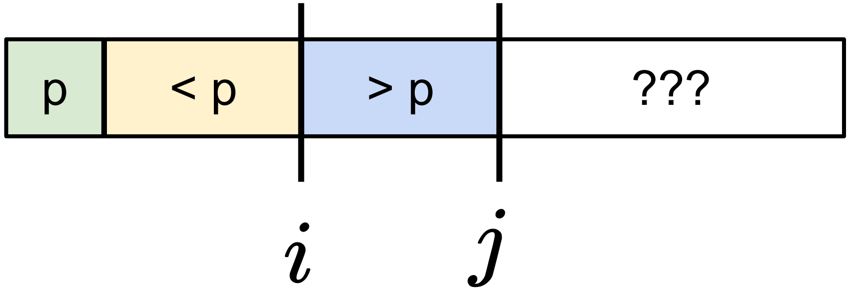
Допустим, что $z_i = z_{i+1} = \cdots = z_j$

Тогда из выбора элемента $z_k : i < k < j$ совершенно не следует, что z_i и z_j больше никогда не будут сравниваться!

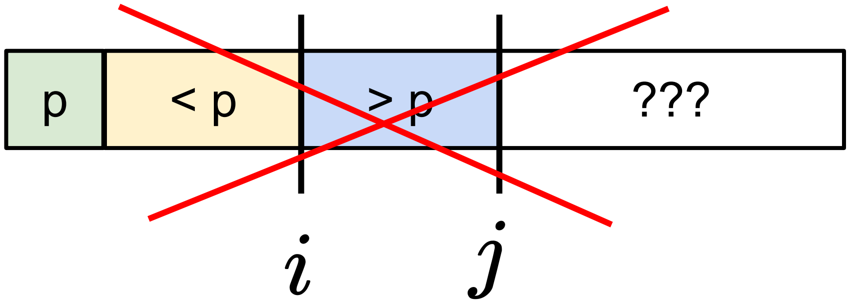
Совсем наоборот: они попадут в один подмассив, а значит рассуждения на этом заканчивать нельзя.

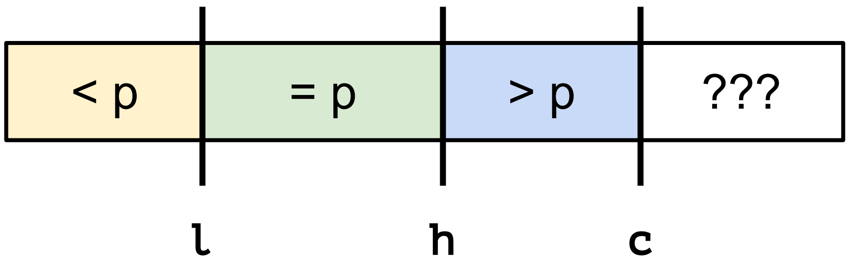
А что делать то?

Модернизируем процедуру разбиения:

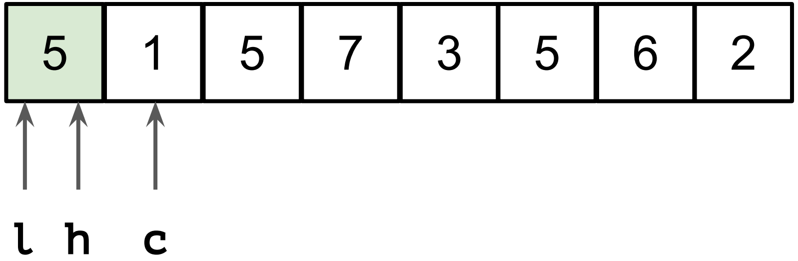


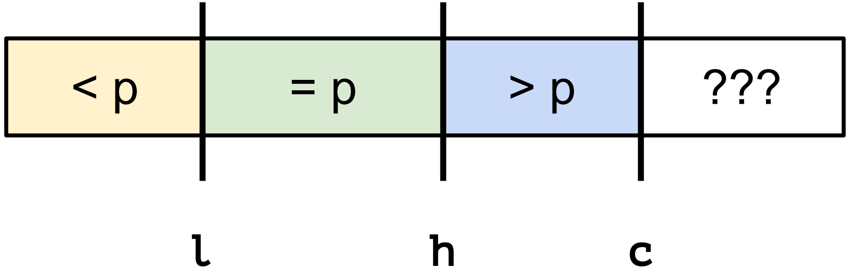
Модернизируем процедуру разбиения:



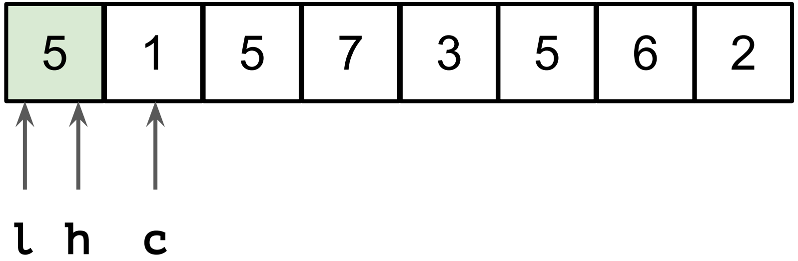


Модернизируем процедуру разбиения:



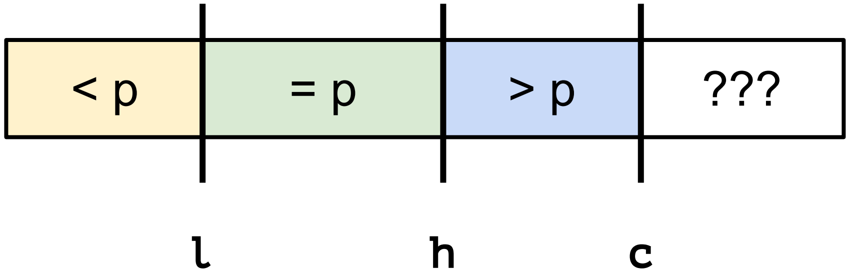


Модернизируем процедуру разбиения:

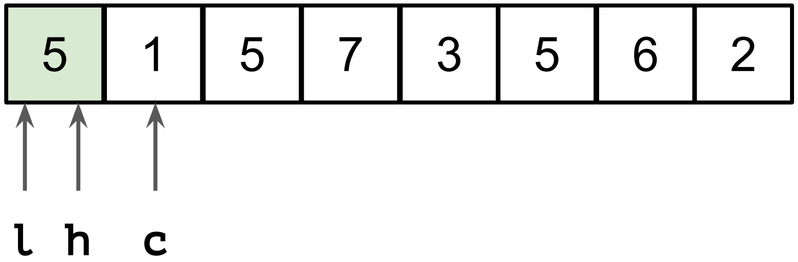


Если A[c] < pivot, то:

1. tmp = A[c]

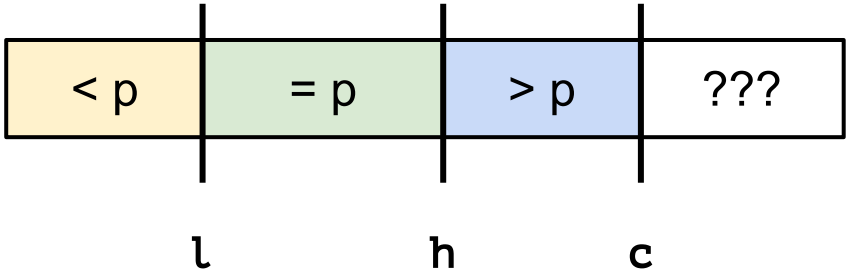


Модернизируем процедуру разбиения:

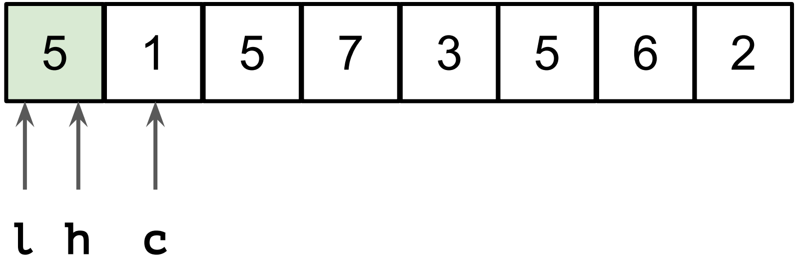


Если A[c] < pivot, то:

1. tmp = A[c]
2. A[c] = A[h + 1]

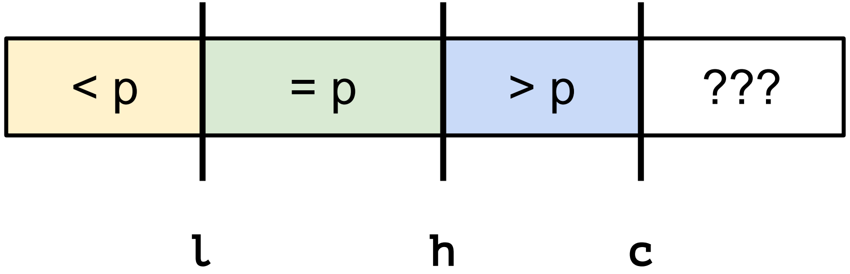


Модернизируем процедуру разбиения:

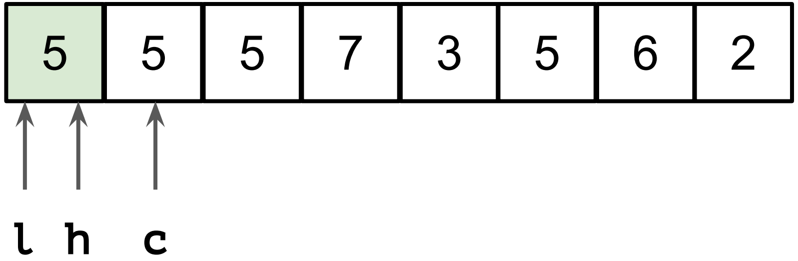


Если A[c] < pivot, то:

1. tmp = A[c]
2. A[c] = A[h + 1]

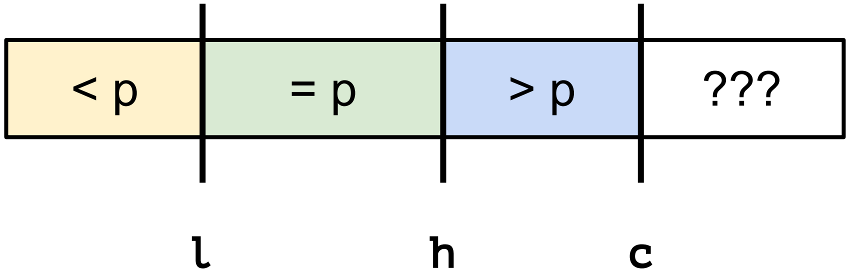


Модернизируем процедуру разбиения:

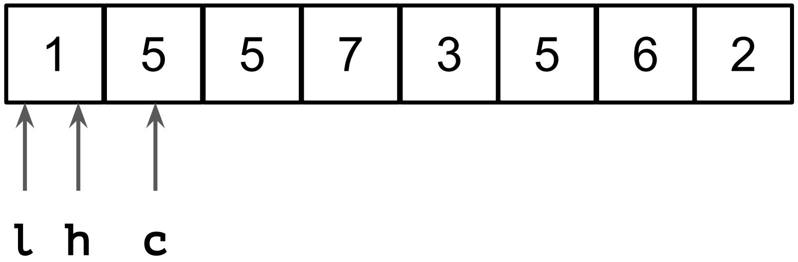


Если A[c] < pivot, то:

1. tmp = A[c]
2. A[c] = A[h + 1]
3. A[h + 1] = A[l]

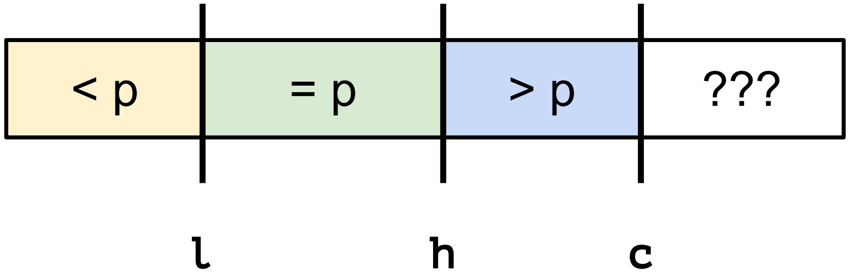


Модернизируем процедуру разбиения:

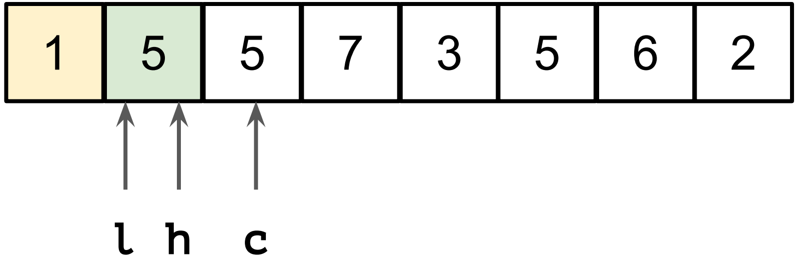


Если A[c] < pivot, то:

1. tmp = A[c]
2. A[c] = A[h + 1]
3. A[h + 1] = A[l]
4. A[l] = tmp

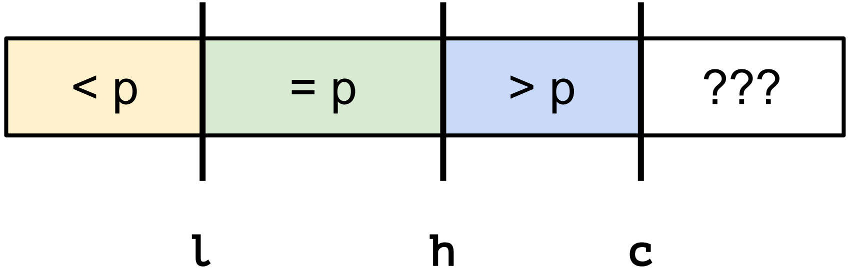


Модернизируем процедуру разбиения:

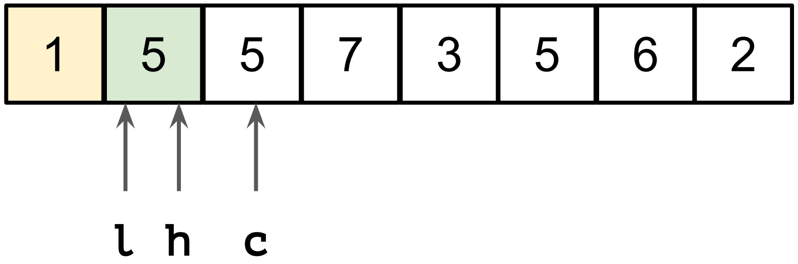


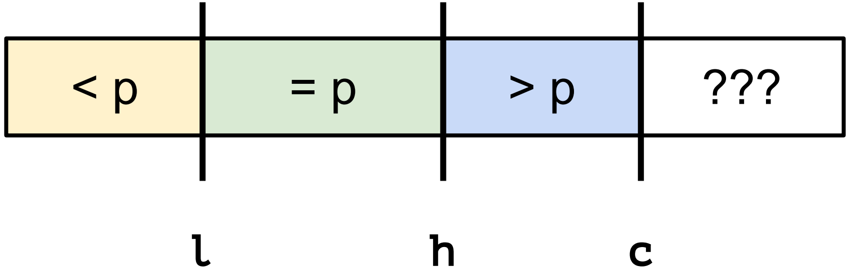
Если A[c] < pivot, то:

```
1.  tmp = A[c]
2.  A[c] = A[h + 1]
3.  A[h + 1] = A[l]
4.  A[l] = tmp
5.  l += 1, h += 1
6.  c += 1
```



Модернизируем процедуру разбиения:





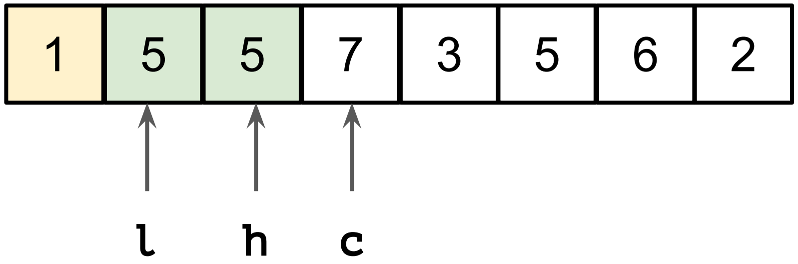
Если A[c] < pivot, то:

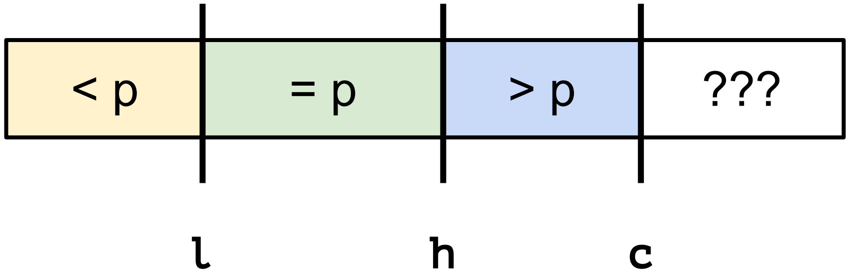
1. tmp = A[c]
2. A[c] = A[h + 1]
3. A[h + 1] = A[l]
4. A[l] = tmp
5. l += 1, h += 1
6. c += 1

Если A[c] == pivot, то:

1. swap(A[h + 1], A[c])
2. h += 1
3. c += 1

Модернизируем процедуру разбиения:





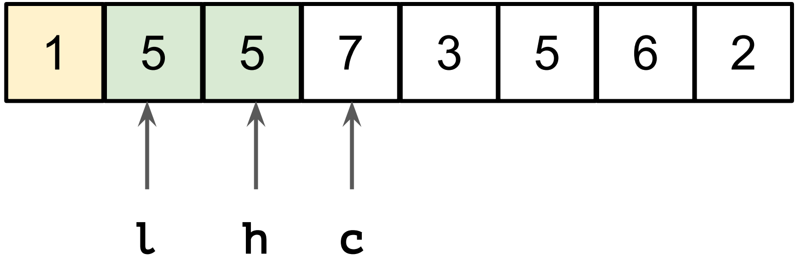
Если A[c] < pivot, то:

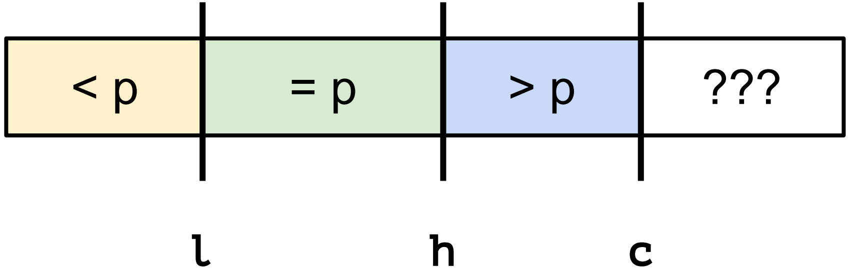
1. tmp = A[c]
2. A[c] = A[h + 1]
3. A[h + 1] = A[l]
4. A[l] = tmp
5. l += 1, h += 1
6. c += 1

Если A[c] == pivot, то:

1. swap(A[h + 1], A[c])
2. h += 1
3. c += 1

Модернизируем процедуру разбиения:





Если A[c] < pivot, то:

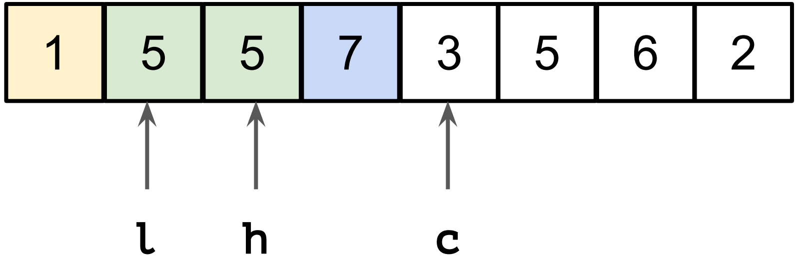
1. tmp = A[c]
2. A[c] = A[h + 1]
3. A[h + 1] = A[l]
4. A[l] = tmp
5. l += 1, h += 1
6. c += 1

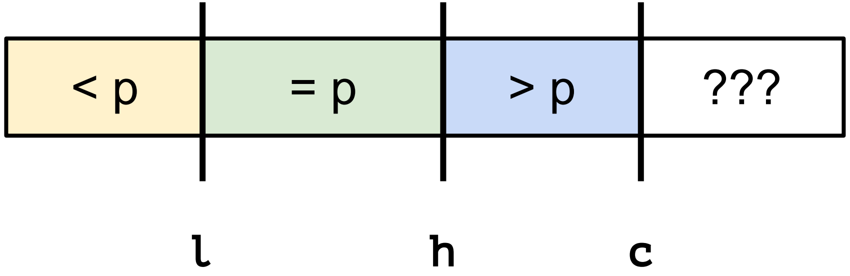
Если A[c] == pivot, то:

1. swap(A[h + 1], A[c])
2. h += 1
3. c += 1

Иначе: c += 1

Модернизируем процедуру разбиения:





Если A[c] < pivot, то:

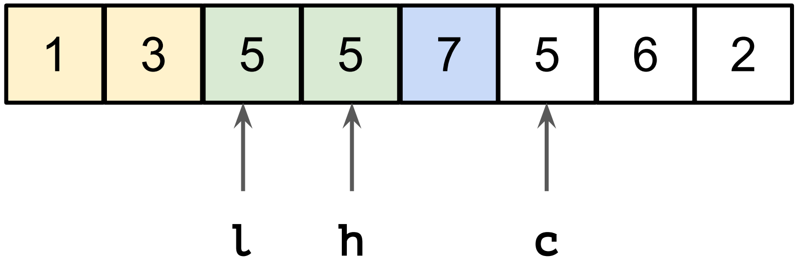
1. tmp = A[c]
2. A[c] = A[h + 1]
3. A[h + 1] = A[l]
4. A[l] = tmp
5. l += 1, h += 1
6. c += 1

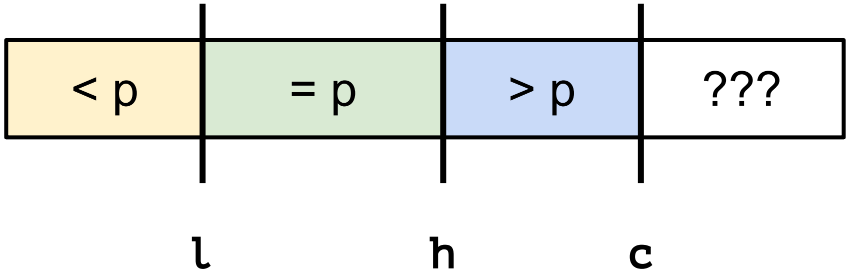
Если A[c] == pivot, то:

1. swap(A[h + 1], A[c])
2. h += 1
3. c += 1

Иначе: c += 1

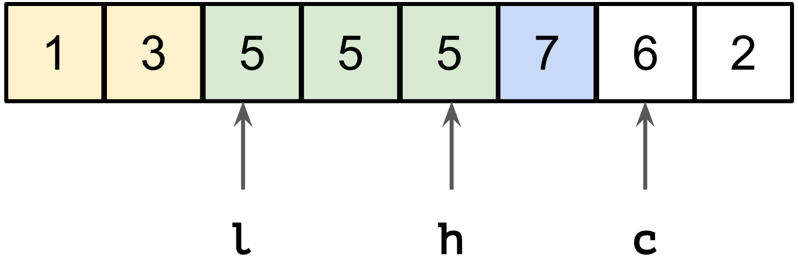
Модернизируем процедуру разбиения:





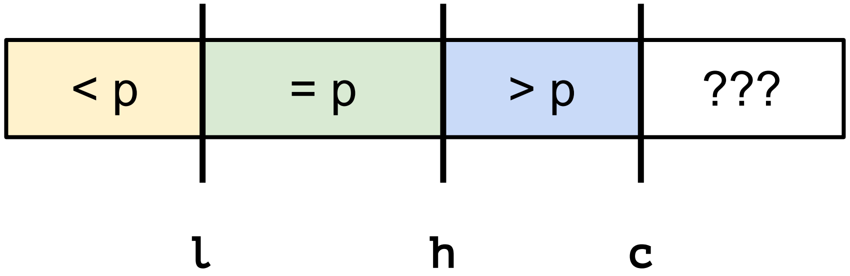

```
Если A[c] < pivot, то:

    1.  tmp = A[c]
    2.  A[c] = A[h + 1]
    3.  A[h + 1] = A[l]
    4.  A[l] = tmp
    5.  l += 1, h += 1
    6.  c += 1

Если A[c] == pivot, то:

    1.  swap(A[h + 1], A[c])
    2.  h += 1
    3.  c += 1

Иначе: c += 1
```

Модернизируем процедуру разбиения:



Если A[c] < pivot, то:

1. tmp = A[c]
2. A[c] = A[h + 1]
3. A[h + 1] = A[l]
4. A[l] = tmp
5. l += 1, h += 1
6. c += 1

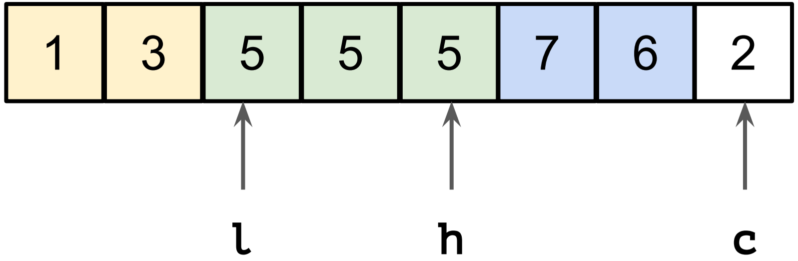


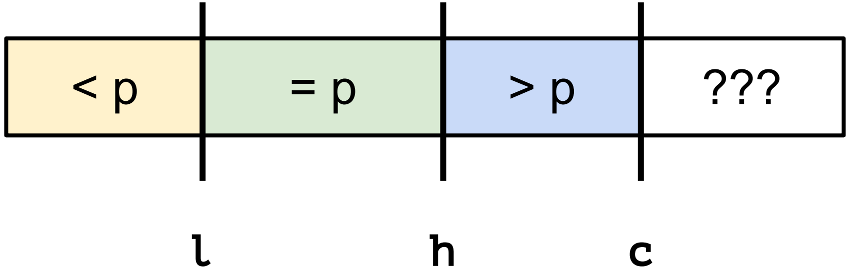
Если A[c] == pivot, то:

1. swap(A[h + 1], A[c])
2. h += 1
3. c += 1

Иначе: c += 1

Модернизируем процедуру разбиения:





Если A[c] < pivot, то:

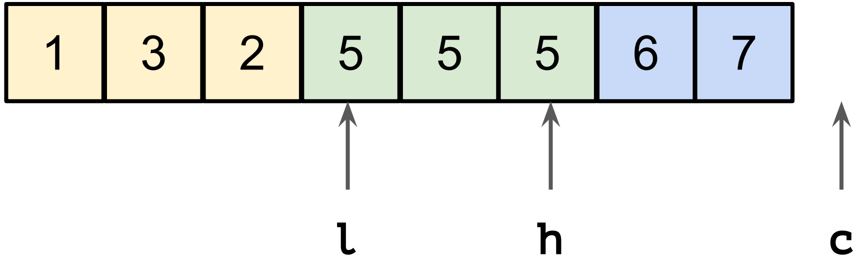
```
1.  tmp = A[c]
2.  A[c] = A[h + 1]
3.  A[h + 1] = A[l]
4.  A[l] = tmp
5.  l += 1, h += 1
6.  c += 1
```

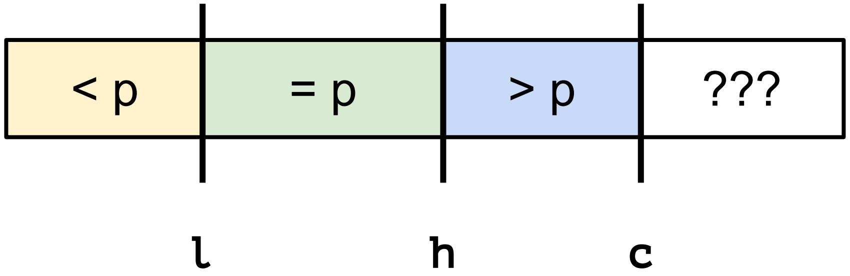
Если A[c] == pivot, то:

```
1.  swap(A[h + 1], A[c])
2.  h += 1
3.  c += 1
```

Иначе: c += 1

Модернизируем процедуру разбиения:





Если A[c] < pivot, то:

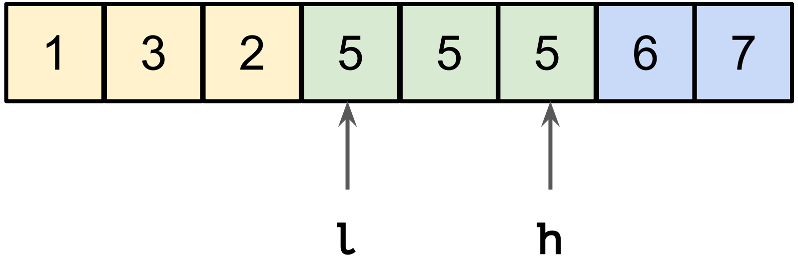
1. tmp = A[c]
2. A[c] = A[h + 1]
3. A[h + 1] = A[l]
4. A[l] = tmp
5. l += 1, h += 1
6. c += 1

Если A[c] == pivot, то:

1. swap(A[h + 1], A[c])
2. h += 1
3. c += 1

Иначе: c += 1

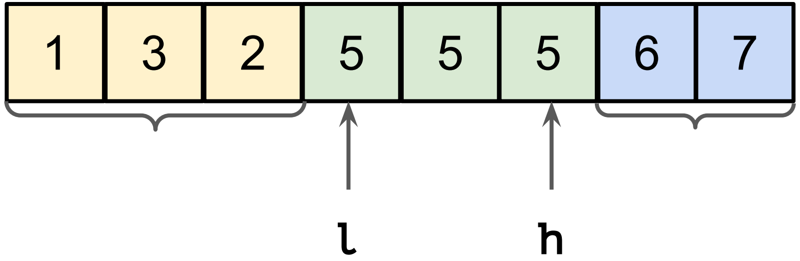
Модернизируем процедуру разбиения:



Похоже на задачу про флаг Нидерландов!

Сложность разбиения все еще линейная.

Модернизируем процедуру разбиения:



Дальше запускаем рекурсию только для желтой и синей частей, опорные элементы в подмассивы не попадают

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Допустим, что $z_i = z_{i+1} = \cdots = z_j$

Тогда из выбора элемента $z_k : i < k < j$ совершенно не следует, что z_i и z_j больше никогда не будут сравниваться!

Совсем наоборот: они попадут в один подмассив, а значит рассуждения на этом заканчивать нельзя.

А что делать то?

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

Допустим, что $z_i = z_{i+1} = \cdots = z_j$

Тогда из выбора элемента $z_k : i < k < j$ действительно следует, что z_i и z_j больше не будут сравниваться.

Они вообще выпадают из рассмотрения, они больше не попадают в подмассивы.

Рассмотрим множество: $z_i, z_{i+1}, \ldots, z_{j-1}, z_j$

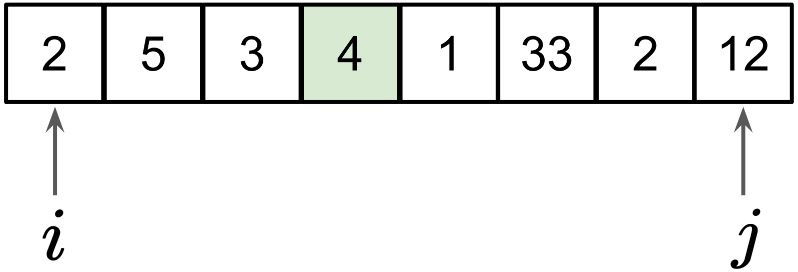
Допустим, что $z_i = z_{i+1} = \cdots = z_j$

Тогда из выбора элемента $z_k : i < k < j$ действительно следует, что z_i и z_j больше не будут сравниваться.

Они вообще выпадают из рассмотрения, они больше не попадают в подмассивы.

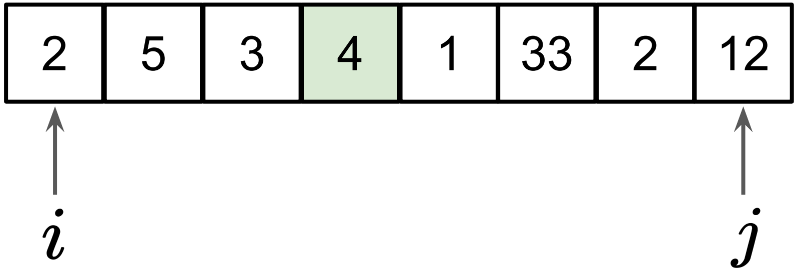
Из этого следует, что теорема о среднем времени работы рандомизированной быстрой сортировки остается O(n*logN) и в случае совпадающих элементов.

Быстрая сортировка: разбиение Хоара



Альтернативная схема: идем с двух сторон массива, пока i < j

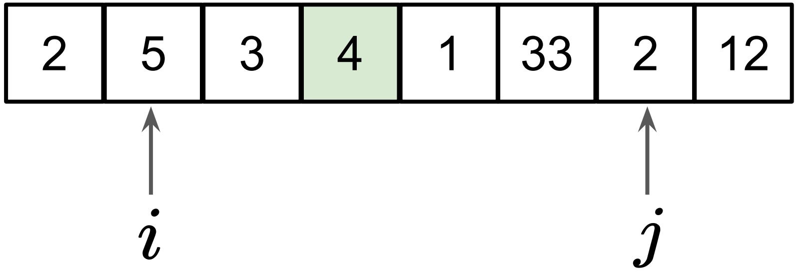
Быстрая сортировка: разбиение Хоара



Альтернативная схема: идем с двух сторон массива, пока i < j

Если находим инверсию (A[i] >= pivot, A[j] <= pivot), делаем swap

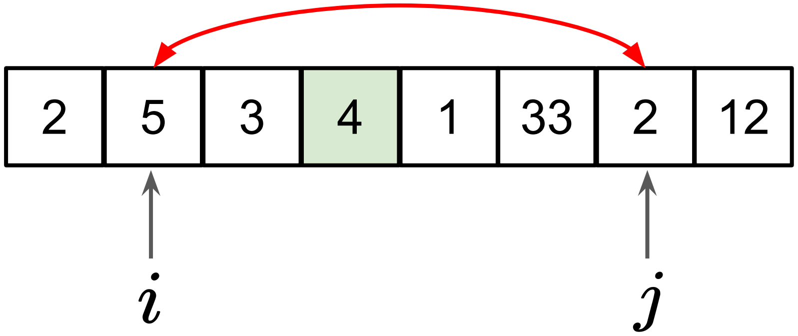
Быстрая сортировка: разбиение Хоара



Альтернативная схема: идем с двух сторон массива, пока i < j

Если находим инверсию (A[i] >= pivot, A[j] <= pivot), делаем swap

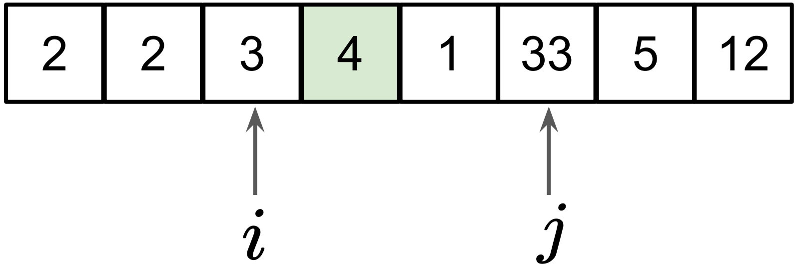
Быстрая сортировка: разбиение Хоара



Альтернативная схема: идем с двух сторон массива, пока i < j

Если находим инверсию (A[i] >= pivot, A[j] <= pivot), делаем swap

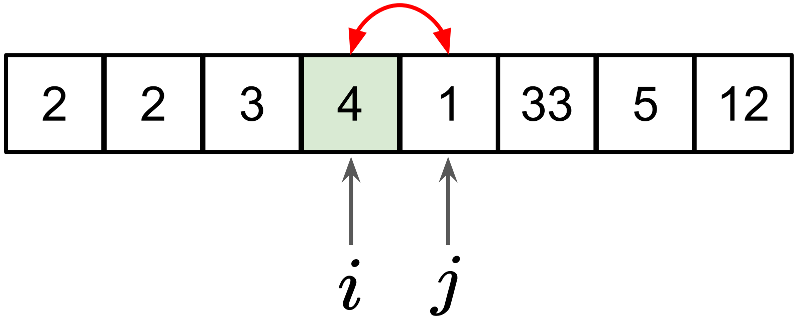
Быстрая сортировка: разбиение Хоара



Альтернативная схема: идем с двух сторон массива, пока i < j

Если находим инверсию (A[i] >= pivot, A[j] <= pivot), делаем swap

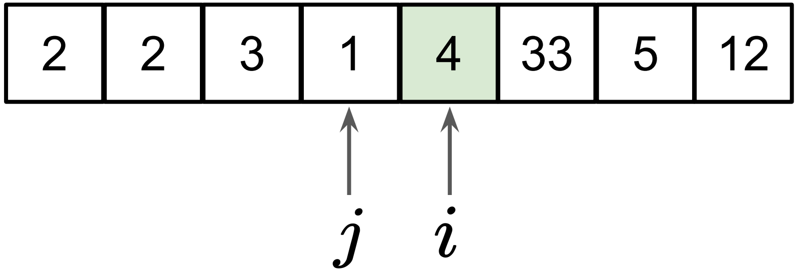
Быстрая сортировка: разбиение Хоара



Альтернативная схема: идем с двух сторон массива, пока i < j

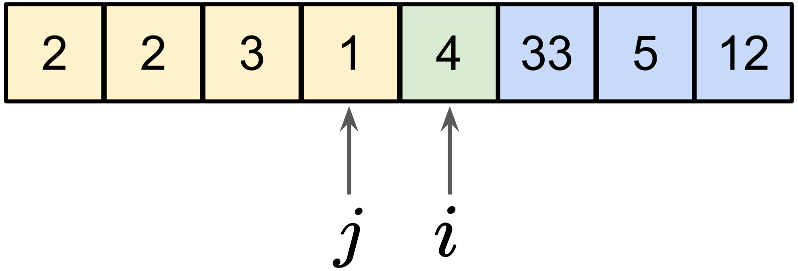
Если находим инверсию (A[i] >= pivot, A[j] <= pivot), делаем swap

Быстрая сортировка: разбиение Хоара




```
Альтернативная схема:
идем с двух сторон
массива, пока i < j

Если находим инверсию
(A[i] >= pivot, A[j] <=
pivot), делаем swap
```

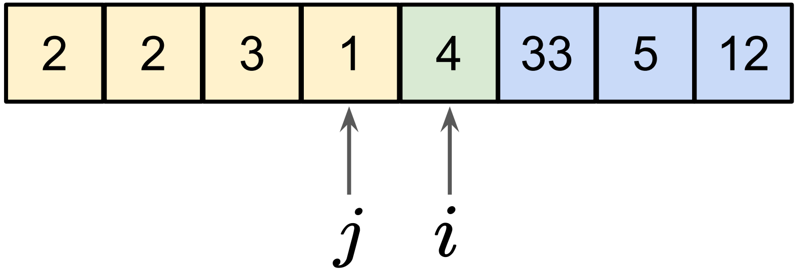
Быстрая сортировка: разбиение Хоара



Альтернативная схема: идем с двух сторон массива, пока i < j

Если находим инверсию (A[i] >= pivot, A[j] <= pivot), делаем swap

Быстрая сортировка: разбиение Хоара

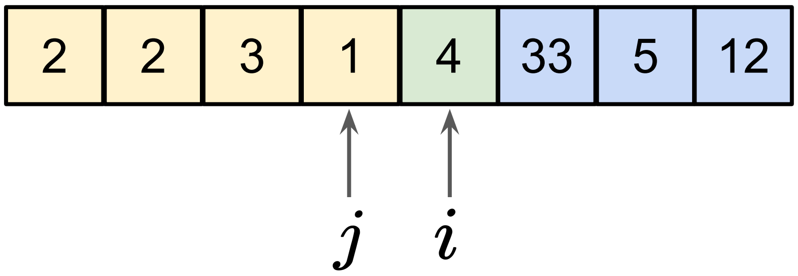


Замечание: такое разбиение совершает меньше сравнений (в 3 раза) и сразу, без модификаций справляется с повторяющимися элементами

Альтернативная схема: идем с двух сторон массива, пока i < j

Если находим инверсию (A[i] >= pivot, A[j] <= pivot), делаем swap

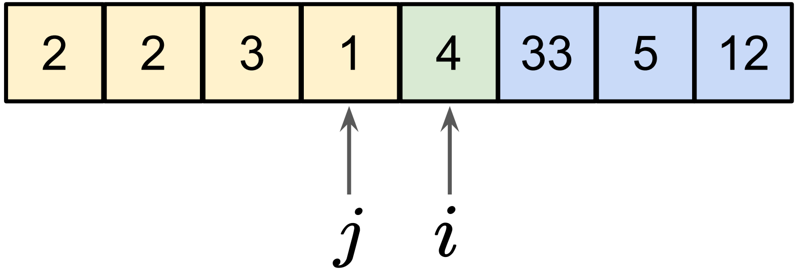
Быстрая сортировка: разбиение Хоара



Замечание: такое разбиение совершает меньше сравнений (в 3 раза) и сразу, без модификаций справляется с повторяющимися элементами

Базовая реализация при этом не гарантирует попадание опорного элемента на свое место, он может оказаться в любом из подмассивов (и снова обрабатываться в рекурсивных вызовах)

Быстрая сортировка: разбиение Хоара



Замечание: такое разбиение совершает меньше сравнений (в 3 раза) и сразу, без модификаций справляется с повторяющимися элементами

Базовая реализация при этом не гарантирует попадание опорного элемента на свое место, он может оказаться в любом из подмассивов (и снова обрабатываться в рекурсивных вызовах)

Однако, это легко исправляется.

Мини-задача #12 (1 + 1 балл)

Реализовать быструю сортировку с выбором случайного элемента в качестве опорного.

Реализовать разбиения Ломуто и Хоара и проверить на литкоде: https://leetcode.com/problems/sort-an-array

За каждое из разбиений, проходящих тесты, получите 1 балл.

Мини-задача #13 (1 балла)

Прочитать статью Андрея Александреску:
"Триумфальное возвращение Ломуто"

Оригинал:
https://dlang.org/blog/2020/05/14/lomutos-comeback/

Перевод:
https://habr.com/ru/post/512106/

Мини-задача #13 (1 балла)

Прочитать статью Андрея Александреску:
"Триумфальное возвращение Ломуто"

Оригинал:
https://dlang.org/blog/2020/05/14/lomutos-comeback/

Перевод:
https://habr.com/ru/post/512106/

Разобраться в версии разбиения Ломуто без ветвлений и провести собственный эксперимент, сравнив эту версию, обычное разбиение Ломуто и разбиение Хоара. (use C)

Быстрая сортировка: финальные замечания

- Стабильна ли быстрая сортировка?

Быстрая сортировка: финальные замечания

- Стабильна ли быстрая сортировка? Конечно нет!

Быстрая сортировка: финальные замечания

- Стабильна ли быстрая сортировка? Конечно нет!
- Не всегда используется рандомизированная версия QuickSort. Выбор опорного элемента может быть и детерминированным.

Быстрая сортировка: финальные замечания

- Стабильна ли быстрая сортировка? Конечно нет!
- Не всегда используется рандомизированная версия QuickSort. Выбор опорного элемента может быть и детерминированным.
- Для таких алгоритмов доказывается, что сложность в среднем (в этот раз по входным данным) тоже будет O(N*logN). Но при этом они уязвимы для атак конкретными последовательностями элементом.

Takeaways

- Впервые познакомились с понятием рандомизированного алгоритма

Takeaways

- Впервые познакомились с понятием рандомизированного алгоритма
- Узнали самые простые вещи из теории вероятностей, которые уже можно использовать для оценки среднего времени работы алгоритмов

Takeaways

- Впервые познакомились с понятием рандомизированного алгоритма

- Узнали самые простые вещи из теории вероятностей, которые уже можно использовать для оценки среднего времени работы алгоритмов

- Быстрая сортировка - глубокая тема, включающая в себя разные разбиения, способы выбора опорного элемента и вероятностный анализ.