Алгоритмы и структуры данных

Слияния

Порядковые статистики

CS Center, Новосибирск

Ещё немного про merge

Как слить две последовательности разной длины?

- merge(A, B)
  - A длины N, B длины M

Сложность merge

Как слить две последовательности разной длины?

- merge(A, B)
  - A длины N, B длины M
- T = O(N + M)

Сложность merge

Как слить две последовательности разной длины?

- merge(A, B)
  - A длины N, B длины M
- T = O(N + M)

Предположим сравнение гораздо дольше перемещения

- Что тогда?

Сложность merge

- merge(A, B); |A| = N, |B| = M

Сравнение гораздо дольше перемещения

Сложность merge

- merge(A, B); |A| = N, |B| = M

Сравнение гораздо дольше перемещения

- Неизбежно N+M перемещений

Сложность merge

- merge(A, B); |A| = N, |B| = M

Сравнение гораздо дольше перемещения

- Неизбежно N+M перемещений
- Минимизируем число сравнений

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев
  - вычислим количество возможных исходов

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев
  - вычислим количество возможных исходов

  A = ***, B = *****

  merge(A, B) = ********

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев
  - вычислим количество возможных исходов

  A = ***, B = *****

  merge(A, B) = ******** или ********

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев
  - вычислим количество возможных исходов

  A = ***, B = *****

  merge(A, B) = ******** или ******** или ********

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев
  - вычислим количество возможных исходов

  A = ***, B = *****

  merge(A, B) = ******** или ******** или ******** или ...

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев
  - вычислим количество возможных исходов

  A = ***, B = *****

  merge(A, B) = ******** или ******** или ******** или ...

  Сколько таких вариантов?

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев
  - вычислим количество возможных исходов

  A = ***, B = *****

  merge(A, B) = ******** или ******** или ******** или ...

  Сколько таких вариантов?
    - C(N+M, N)

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев
  - число исходов = C(N+M, N)

Число сравнений merge

- merge(A, B); |A| = N, |B| = M
- Минимизируем число сравнений

Получим теоретико-информационную нижнюю оценку

- используем модель разрешающих деревьев
  - число исходов = C(N+M, N)
  - минимальное число сравнений = log(C(N+M, N))

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = log((N+M)! / (N! M!))

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = log((N+M)! / (N! M!)) =

= log((N+M)!) – log(N!) – log(M!)

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = log((N+M)! / (N! M!)) =

= log((N+M)!) – log(N!) – log(M!) =

= (N+M) log(N+M) – (N+M) – (N logN – N) - (M logM – M)

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = log((N+M)! / (N! M!)) =

= log((N+M)!) – log(N!) – log(M!) =

= (N+M) log(N+M) ~~– (N+M)~~ – (N logN ~~– N~~) - (M logM ~~– M~~)

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = log((N+M)! / (N! M!)) =

= log((N+M)!) – log(N!) – log(M!) =

= (N+M) log(N+M) ~~– (N+M)~~ – (N logN ~~– N~~) - (M logM ~~– M~~) =

= (N+M) log(N+M) – N logN – M logM

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = log((N+M)! / (N! M!)) =

= log((N+M)!) – log(N!) – log(M!) =

= (N+M) log(N+M) ~~– (N+M)~~ – (N logN ~~– N~~) - (M logM ~~– M~~) =

= (N+M) log(N+M) – N logN – M logM =

= N (log(N+M) – logN) + M (log(N+M) – logM)

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = log((N+M)! / (N! M!)) =

= log((N+M)!) – log(N!) – log(M!) =

= (N+M) log(N+M) ~~– (N+M)~~ – (N logN ~~– N~~) - (M logM ~~– M~~) =

= (N+M) log(N+M) – N logN – M logM =

= N (log(N+M) – logN) + M (log(N+M) – logM) =

= N log((N + M) / N) + M log((N + M) / M)

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = N log((N + M) / N) + M log((N + M) / M)

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = N log((N + M) / N) + M log((N + M) / M)

- Пусть N >> M

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = N log((N + M) / N) + M log((N + M) / M)

- Пусть N >> M

  Тогда log(N+M / N) -> 1, смотрим на втрое слагаемое

Число сравнений merge: оценка

Минимальное число сравнений = log(C(N+M, N))

log(N!) = N log N – N

log(C(N + M, N)) = N log((N + M) / N) + M log((N + M) / M)

- Пусть N >> M

  Тогда log(N+M / N) -> 1, смотрим на втрое слагаемое

log(C(N + M, N)) = M log(N / M)

Число сравнений merge: оценка

Оценка снизу на число сравнений = M log(N / M)

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)
Пусть M = 1

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Пусть M = 1

- Тогда, чтобы слить последовательности достаточно найти куда вставить один элемент

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Пусть M = 1

- Тогда, чтобы слить последовательности достаточно найти куда вставить один элемент
  - Выполняем бинпоиск, T = log(N)

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Пусть M = 1

- Тогда, чтобы слить последовательности достаточно найти куда вставить один элемент
  - Выполняем бинпоиск, T = log(N) – оценка точна

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = $M \log(N / M)$

Пусть $M > 1$

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Пусть M > 1

- M бинпоисков?

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Пусть M > 1

- M бинпоисков?
  - T = M log (N)

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Пусть M > 1

- M бинпоисков?
  - T = M log (N) > T = M log (N / M)

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Пусть M > 1

- M бинпоисков?
  - T = M log (N) > T = M log (N / M)
- Проблема: каждый бинпоиск может выполняться на отрезке длиной = O(N)

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Пусть M > 1

- M бинпоисков?
  - T = M log (N) > T = M log (N / M)
- Проблема: каждый бинпоиск может выполняться на отрезке длиной = O(N)
  - Как уменьшить суммарную длину отрезков поиска?

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Идея: начинать бинпоиск от края

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Идея: начинать бинпоиск от края
  - Делаем первый поиск не на отрезке [0, N), а на [0, K)

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Идея: начинать бинпоиск от края
  - Делаем первый поиск не на отрезке [0, N), а на [0, K)
  - Подбираем K = 1, 2, 4 …, так чтобы искомое значение попадало в отрезок

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Идея: начинать бинпоиск от края
  - Делаем первый поиск не на отрезке [0, N), а на [0, K)
  - Подбираем K = 1, 2, 4 …, так чтобы искомое значение попадало в отрезок
  - Пусть X – найденная позиция

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Идея: начинать бинпоиск от края
  - Делаем первый поиск не на отрезке [0, N), а на [0, K)
  - Подбираем K = 1, 2, 4 …, так чтобы искомое значение попадало в отрезок
  - Пусть X – найденная позиция
  - Следующий поиск на отрезке [X, X + K), K снова подбираем

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Идея: начинать бинпоиск от края
  - Делаем первый поиск не на отрезке [0, N), а на [0, K)
  - Подбираем K = 1, 2, 4 …, так чтобы искомое значение попадало в отрезок
  - Пусть X – найденная позиция
  - Следующий поиск на отрезке [X, X + K), K снова подбираем
  - Продолжаем так для всех M элементов

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Бинпоиск от края – galloping

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = M log(N / M)

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Бинпоиск от края – galloping
  - Суммарная длина отрезков поиска составит N

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = $M \log(N / M)$

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Бинпоиск от края – galloping
  - Суммарная длина отрезков $\Sigma N_i = N$

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = $M \log(N / M)$

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Бинпоиск от края – galloping
  - Суммарная длина отрезков $\Sigma N_i = N$
  - Какова сложность?

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = $M \log(N / M)$

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Бинпоиск от края – galloping
  - Суммарная длина отрезков $\Sigma N_i = N$
  - $T = \Sigma \log(N_i)$

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = $M \log(N / M)$

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Бинпоиск от края – galloping
  - Суммарная длина отрезков $\Sigma N_i = N$
  - $T = \Sigma \log(N_i)$

  нер-во Йенсена: $\Sigma \log(N_i) / M \le \log(\Sigma N_i / M)$

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = $M \log(N / M)$

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Бинпоиск от края – galloping
  - Суммарная длина отрезков $\Sigma N_i = N$
  - $T = \Sigma \log(N_i)$

  нер-во Йенсена: $\Sigma \log(N_i) / M \le \log(\Sigma N_i / M)$
  - $T = O(M \log(N / M))$

Число сравнений merge: достижимость оценки

Оценка снизу на число сравнений = $M \log(N / M)$

Выполним бинпоиск для каждого из M элементов

Требуется уменьшить суммарную длину отрезков поиска

- Бинпоиск от края – galloping
  - Суммарная длина отрезков $\Sigma N_i = N$
  - $T = \Sigma \log(N_i)$

  нер-во Йенсена: $\Sigma \log(N_i) / M \le \log(\Sigma N_i / M)$
  - $T = O(M \log(N / M))$ – оценка точна

Порядковые статистики

Дан набор ключей (k_1, k_2 … k_N), $k_i \in K$
Какой из них будет на позиции M, если их отсортировать?

Порядковые статистики

Дан набор ключей (k_1, k_2 … k_N), $k_i \in K$
Какой из них будет на позиции M, если их отсортировать?
«M-ая порядковая статистика»

Порядковые статистики

Дан набор ключей (k_1, k_2 … k_N), $k_i \in K$
Какой из них будет на позиции M, если их отсортировать?
«M-ая порядковая статистика»

Примеры:
- min
- max
- медиана

Порядковые статистики

Дан набор ключей (k_1, k_2 … k_N), $k_i \in K$
Какой из них будет на позиции M, если их отсортировать?
«M-ая порядковая статистика»

Примеры:
- min, M=0
- max, M=N-1
- медиана, M=N/2

Порядковые статистики

Дан набор ключей (k_1, k_2 … k_N), $k_i \in K$
Какой из них будет на позиции M, если их отсортировать?
«M-ая порядковая статистика»

Примеры:

- min, M=0, можно найти за T=O(N)
- max, M=N-1, можно найти за T=O(N)
- медиана, M=N/2, можно найти за T=?

Порядковые статистики

Дан набор ключей (k_1, k_2 … k_N), $k_i \in K$
Какой из них будет на позиции M, если их отсортировать?
«M-ая порядковая статистика»

- M – произвольное, T=?

Порядковые статистики

Дан набор ключей (k_1, k_2 … k_N), $k_i \in K$
Какой из них будет на позиции M, если их отсортировать?
«M-ая порядковая статистика»

- M – произвольное
  - за T=O(NlogN)

Порядковые статистики

Дан набор ключей (k_1, k_2 … k_N), $k_i \in K$
Какой из них будет на позиции M, если их отсортировать?
«M-ая порядковая статистика»

- M – произвольное
  - за T=O(NlogN) – слишком много

Порядковые статистики за O(N)

Вспомним QuickSort

```
S(A) {
    if (|A| == 1) {
        return
    }
    pivot = pick_pivot(A)
    (B, C) = partition(A, pivot)
    S(B)
    S(C)
}
```

Порядковые статистики за O(N)

Вспомним QuickSort

```
S(A) {
    if (|A| == 1) {
        return
    }
    pivot = pick_pivot(A)
    (B, C) = partition(A, pivot)
    S(B)
    S(C)
}
```

Из этих двух вызовов нужен только один

Порядковые статистики за O(N)

Вспомним QuickSort

```
S(A) {
    if (|A| == 1) {
        return
    }
    pivot = pick_pivot(A)
    (B, C) = partition(A, pivot)
    S(B)
    S(C)
}
```

Из этих двух вызовов нужен только один, содержащий позицию M

Порядковые статистики за O(N)

«Randomized Select»

```
Select(A, M) {
    if (|A| == 1) {
        return A[0]
    }
    pivot = pick_pivot(A)
    (B, C) = partition(A, pivot)
    return M < |B| ?
        Select(B, M) :
        Select(C, M - |B|)
}
```

Порядковые статистики за O(N)

«Randomized Select»

```
Select(A, M) {
    if (|A| == 1) {
        return A[0]
    }
    pivot = pick_pivot(A)
    (B, C) = partition(A, pivot)
    return M < |B| ?
        Select(B, M) :
        Select(C, M - |B|)
}
```

Можно убрать хвостовую рекурсию

Порядковые статистики за O(N)

«Randomized Select»

```
Select(A, M) {
    while (|A| > 1) {
        pivot = pick_pivot(A)
        (B, C) = partition(A, pivot)
        if (M < |B|)
            A = B
        else {
            A = C
            M -= |B|
        }
    }
    return A[0]
}
```

Порядковые статистики за O(N)

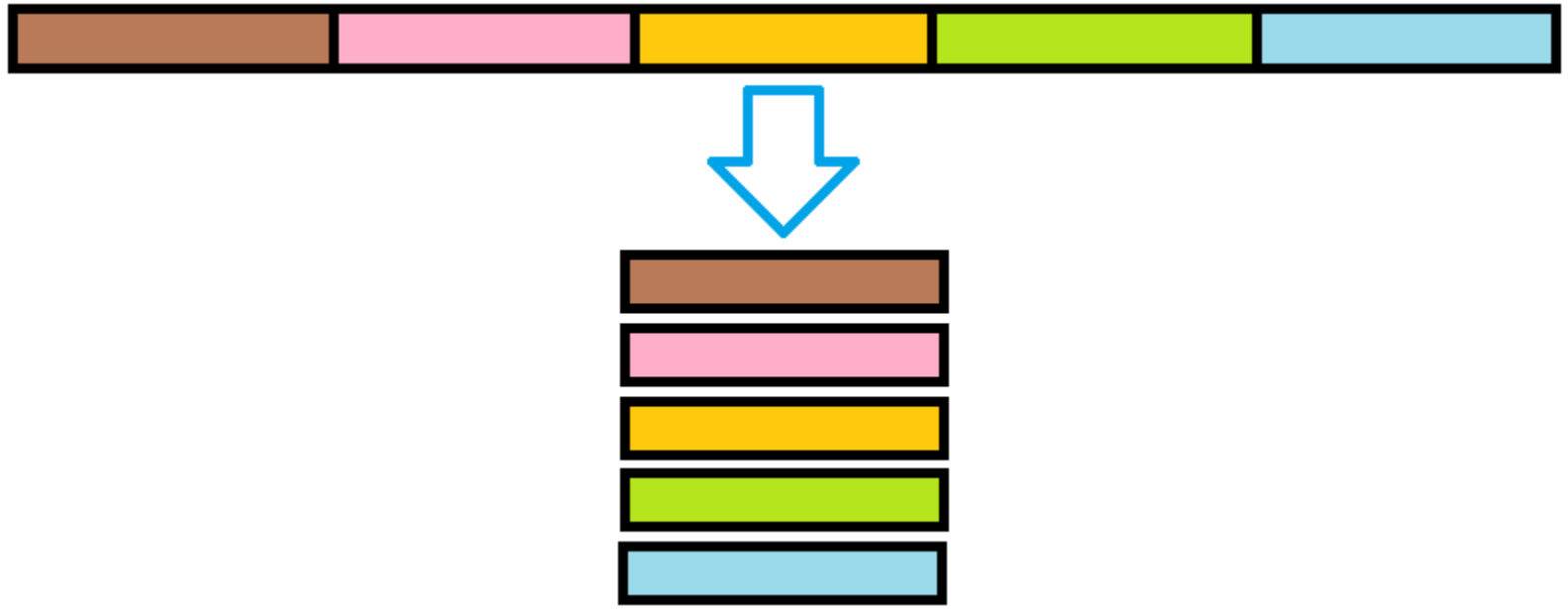
Алгоритм без рандомизации?

Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»

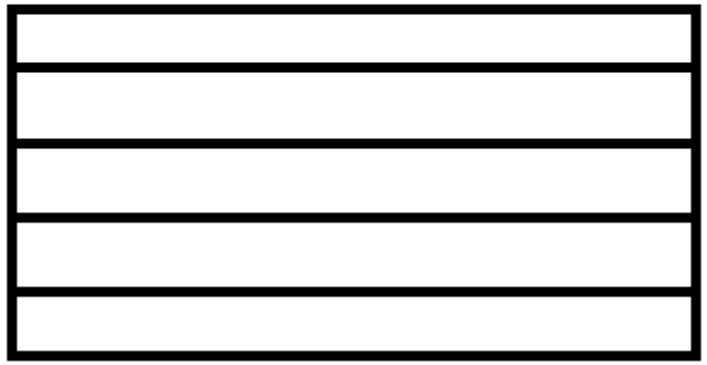
Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»



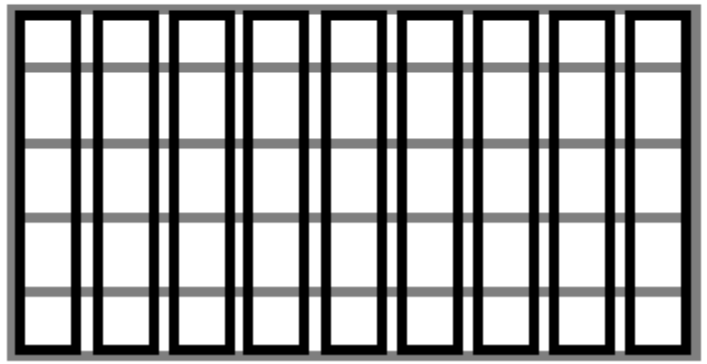
Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»
Таблица «(N/5) * 5»



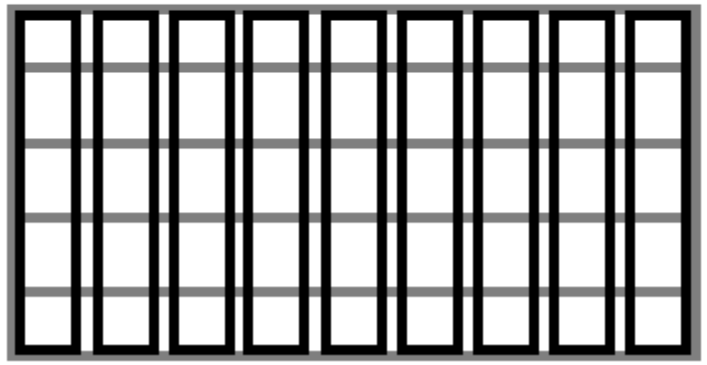
Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»
Рассмотрим столбцы



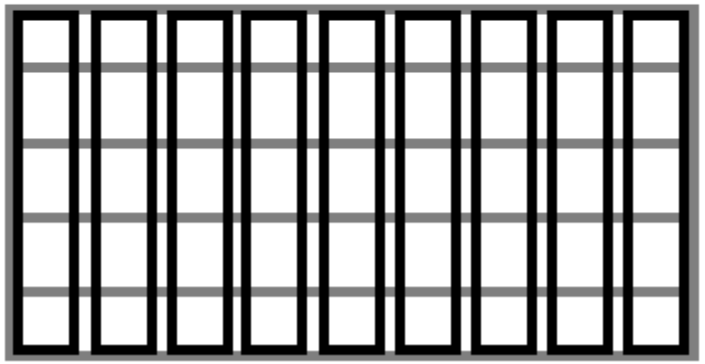
Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»
Каждый столбец отсортируем независимо



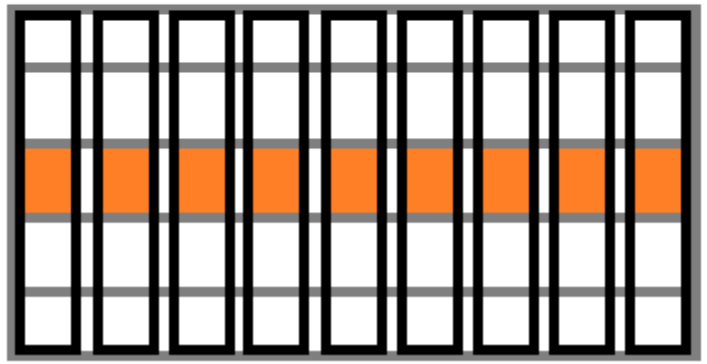
Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»
Каждый столбец отсортируем независимо – T = O(N)



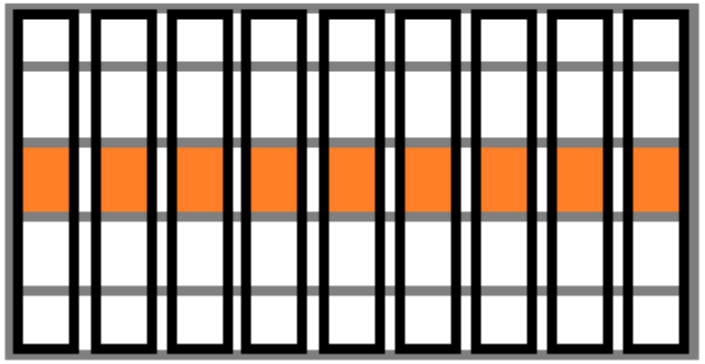
Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»
Средняя строка – медианы столбцов



Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»
Найдём медиану медиан рекурсивно



Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»
Медиана медиан – pivot

Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»
Медиана медиан – pivot
Далее – partition и рекурсивный вызов

Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»

Итак:

- сортировка пятёрок
- поиск медины их медиан
- partition
- рекурсивный вызов для части массива

Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»
T(N) - ?

- сортировка пятёрок
- поиск медины их медиан
- partition
- рекурсивный вызов для части массива

Порядковые статистики за O(N)

Алгоритм без рандомизации – «медиана медиан»

T(N) - ?

- сортировка пятёрок, T = O(N)
- поиск медины их медиан, T = T(N/5)
- partition, T = O(N)
- рекурсивный вызов для части массива, T = T(?)

Порядковые статистики за O(N)

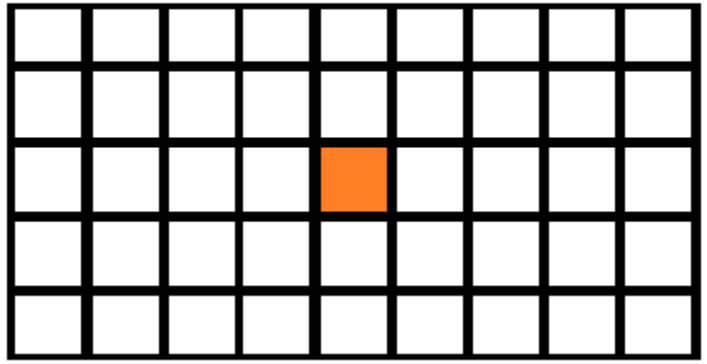
Алгоритм без рандомизации – «медиана медиан»

$T(N) = O(N) + T(N / 5) + T(?)$

- сортировка пятёрок, $T = O(N)$
- поиск медины их медиан, $T = T(N/5)$
- partition, $T = O(N)$
- рекурсивный вызов для части массива, $T = T(?)$

Порядковые статистики за O(N)

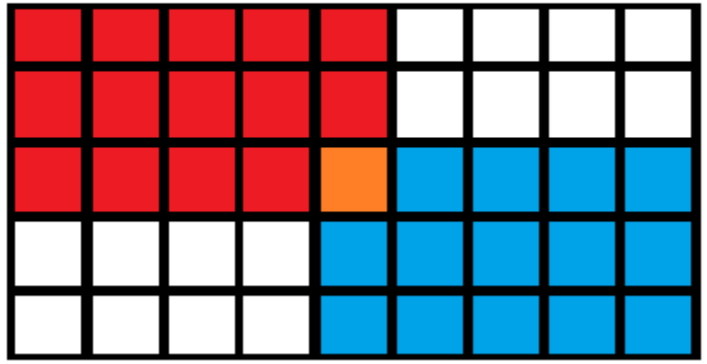
Насколько далека медиана медиан от медианы?



Порядковые статистики за O(N)

Насколько далека медиана медиан от медианы?

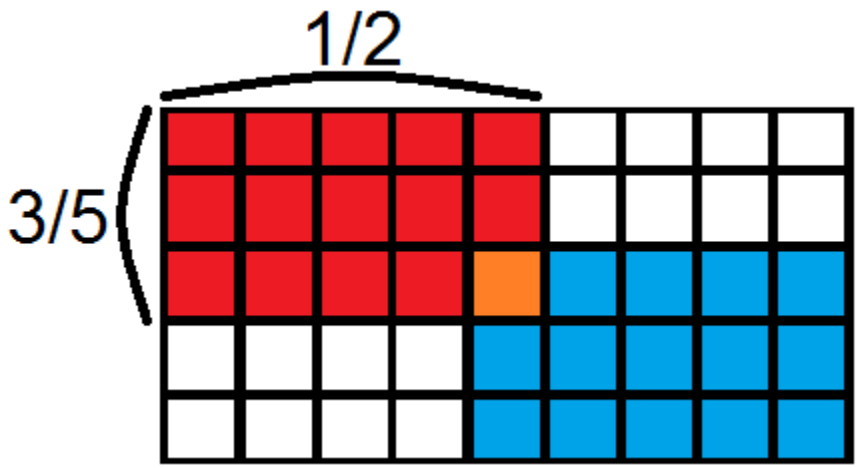
- Ключи меньше
- Ключи больше



Порядковые статистики за O(N)

Насколько далека медиана медиан от медианы?

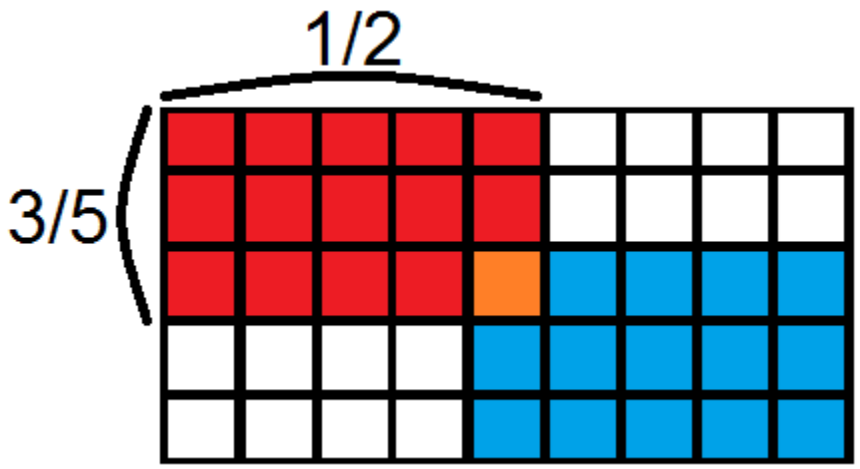
- Ключи меньше
- Ключи больше



Порядковые статистики за O(N)

Насколько далека медиана медиан от медианы?

- Ключи меньше, (3/10)N
- Ключи больше, (3/10)N



Порядковые статистики за O(N)

Медиана медиан отстоит от края не менее, чем на (3/10)N

Порядковые статистики за O(N)

Медиана медиан отстоит от края не менее, чем на (3/10)N

В худшем случае:

- partition разобьёт на (3/10) и (7/10)

Порядковые статистики за O(N)

Медиана медиан отстоит от края не менее, чем на (3/10)N

В худшем случае:

- partition разобьёт на (3/10) и (7/10)
- рекурсивно пойдём в (7/10)N

Порядковые статистики за O(N)

Медиана медиан отстоит от края не менее, чем на (3/10)N

В худшем случае:

- partition разобьёт на (3/10) и (7/10)
- рекурсивно пойдём в (7/10)N

$T(N) = O(N) + T(N/5) + T(7/10*N)$

Порядковые статистики за O(N)

Медиана медиан отстоит от края не менее, чем на (3/10)N

В худшем случае:

- partition разобьёт на (3/10) и (7/10)
- рекурсивно пойдём в (7/10)N

$T(N) = O(N) + T(N/5) + T(7/10*N)$ – оценка линейна

Порядковые статистики за O(N)

Медиана медиан отстоит от края не менее, чем на (3/10)N

В худшем случае:

- partition разобьёт на (3/10) и (7/10)
- рекурсивно пойдём в (7/10)N

$T(N) = O(N) + T(N/5) + T(7/10*N)$ – оценка линейна, $T = O(N)$

Частичная сортровка

Задача: из N ключей найти M наименьших

Частичная сортровка

Задача: из N ключей найти M наименьших
$T = O(N + M\log M)$

Частичная сортровка

Задача: из N ключей найти M наименьших
$T = O(N + M\log M)$

- За N находим M-ую порядковую статистику

Частичная сортровка

Задача: из N ключей найти M наименьших
T = O(N + MlogM)

- За N находим M-ую порядковую статистику
- Partition

Частичная сортровка

Задача: из N ключей найти M наименьших
$T = O(N + M\log M)$

- За N находим M-ую порядковую статистику
- Partition
- Сортируем M элементов

Частичная сортровка

Задача: из N ключей найти M наименьших
Что, если не можем запомнить N ключей? Память = O(M)

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Используем кучу на максимум

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Используем кучу на максимум
- Добавляем в неё ключи один за другим

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Используем кучу на максимум
- Добавляем в неё ключи один за другим
- Если в куче стало больше M элементов, выкидываем максимум

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Используем кучу на максимум
- Добавляем в неё ключи один за другим
- Если в куче стало больше M элементов, выкидываем максимум
- В конце сортируем элементы кучи

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Используем кучу на максимум
- Добавляем в неё ключи один за другим
- Если в куче стало больше M элементов, выкидываем максимум
- В конце сортируем элементы кучи

T = O(NlogM)

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Используем кучу на максимум
- Добавляем в неё ключи один за другим
- Если в куче стало больше M элементов, выкидываем максимум
- В конце сортируем элементы кучи

T = O(NlogM) - долго

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Возьмём первые M ключей

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Возьмём первые M ключей
- Добавим к ним очередные M ключей

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Возьмём первые M ключей
- Добавим к ним очередные M ключей
- Из этих 2M ключей оставим M минимальных, T = O(M)

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Возьмём первые M ключей
- Добавим к ним очередные M ключей
- Из этих 2M ключей оставим M минимальных, T = O(M)
- Сортируем оставшиеся в конце M ключей, T = O(MlogM)

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Возьмём первые M ключей
- Добавим к ним очередные M ключей
- Из этих 2M ключей оставим M минимальных, T = O(M)
- Сортируем оставшиеся в конце M ключей, T = O(MlogM)

T = O(MlogM) + O(M) * N/M

Частичная сортровка

Задача: из N ключей найти M наименьших

Что, если не можем запомнить N ключей? Память = O(M)

- Возьмём первые M ключей
- Добавим к ним очередные M ключей
- Из этих 2M ключей оставим M минимальных, T = O(M)
- Сортируем оставшиеся в конце M ключей, T = O(MlogM)

T = O(MlogM) + O(M) * N/M = O(MlogM + N)