Алгоритмы и структуры данных

Стохастические алгоритмы. **SELECT** & **SIMULATED ANNEALING**



Практическое занятие **07**

2026–2027 учебный год

План

1. Понятие порядковых статистик — задача SELECT
2. Случайный поиск порядковой статистики
   за ожидаемое линейное время — RANDOMIZED SELECT
3. Детерминированный поиск порядковой статистики
   за гарантированное линейное время — LINEAR SELECT
4. Управляемый перебор и метод имитации отжига —
   алгоритм SIMULATED ANNEALING

Warm-Up

FAQ Случайные алгоритмы

1. Что отличает случайные алгоритмы от детерминированных?
2. Какие преимущества дает применение случайных алгоритмов?
3. В чем разница между случайными алгоритмами типа Монте Карло и Лас Вегас?
4. Может ли случайный алгоритм всегда возвращать неверный ответ?

Порядковые статистики

Задача 1




SELECT

Вход: массив A, содержащий n различных элементов, сравнимых друг с другом и целое число k.

Выход: элемент $a \in A$, который в точности больше других $k - 1$ элементов A.

- $k = 1$ — минимум массива
- $k = n$ — максимум массива
- $k = n/2$ — медиана массива

...

Задача 1




NAÏVE SELECT для min ИЛИ max

```
minimum(A)
1 min = A[1]
2 for i = 2 to A.size
3     if min > A[i]
4         min = A[i]
5 return min
```

Стандартный подход потребует $n - 1$ сравнений

[Q] Можно ли найти минимум более оптимально?

Задача 1




NAÏVE SELECT для min ИЛИ max

```
minimum(A)
1 min = A[1]
2 for i = 2 to A.size
3     if min > A[i]
4         min = A[i]
5 return min
```

Стандартный подход потребует $n - 1$ сравнений

[Q] Можно ли найти минимум более оптимально? НЕТ

Задача 1




NAÏVE SELECT для min И max

```
minMax(A)
1 min = minimum(A)
2 max = maximum(A)
3 return min, max
```

Такой подход одновременного поиска потребует $2n - 2$ сравнений

[Q] Можно ли найти уменьшить число сравнений?

Задача 1




NAÏVE SELECT для min И max

```
minMax(A)
1 min = minimum(A)
2 max = maximum(A)
3 return min, max
```

Такой подход одновременного поиска потребует $2n - 2$ сравнений

[Q] Можно ли найти уменьшить число сравнений? ДА

с помощью «разделяй-и-властвуй»

Задача 1

NAÏVE SELECT для min И max



Задача 1

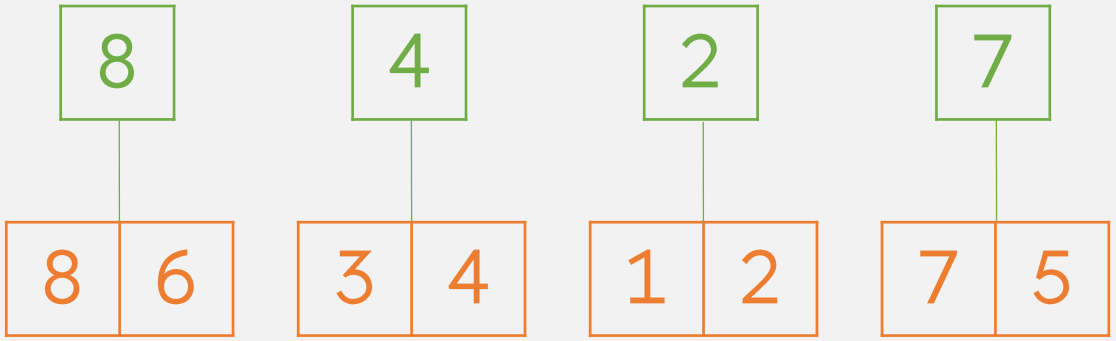
NAÏVE SELECT для min И max



Задача 1

NAÏVE SELECT для min И max

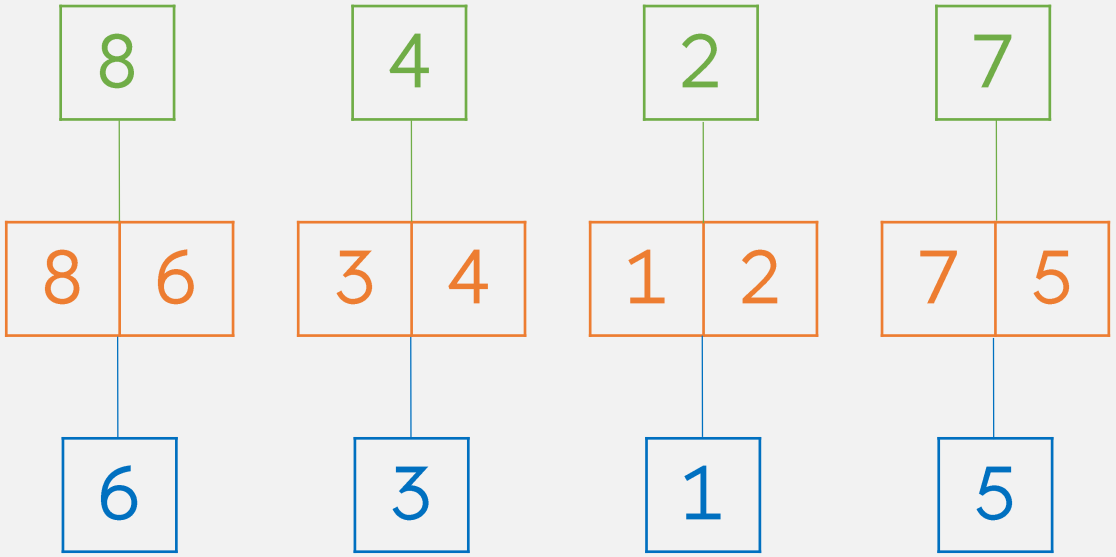
максимумы среди пар



Задача 1

NAÏVE SELECT для min И max

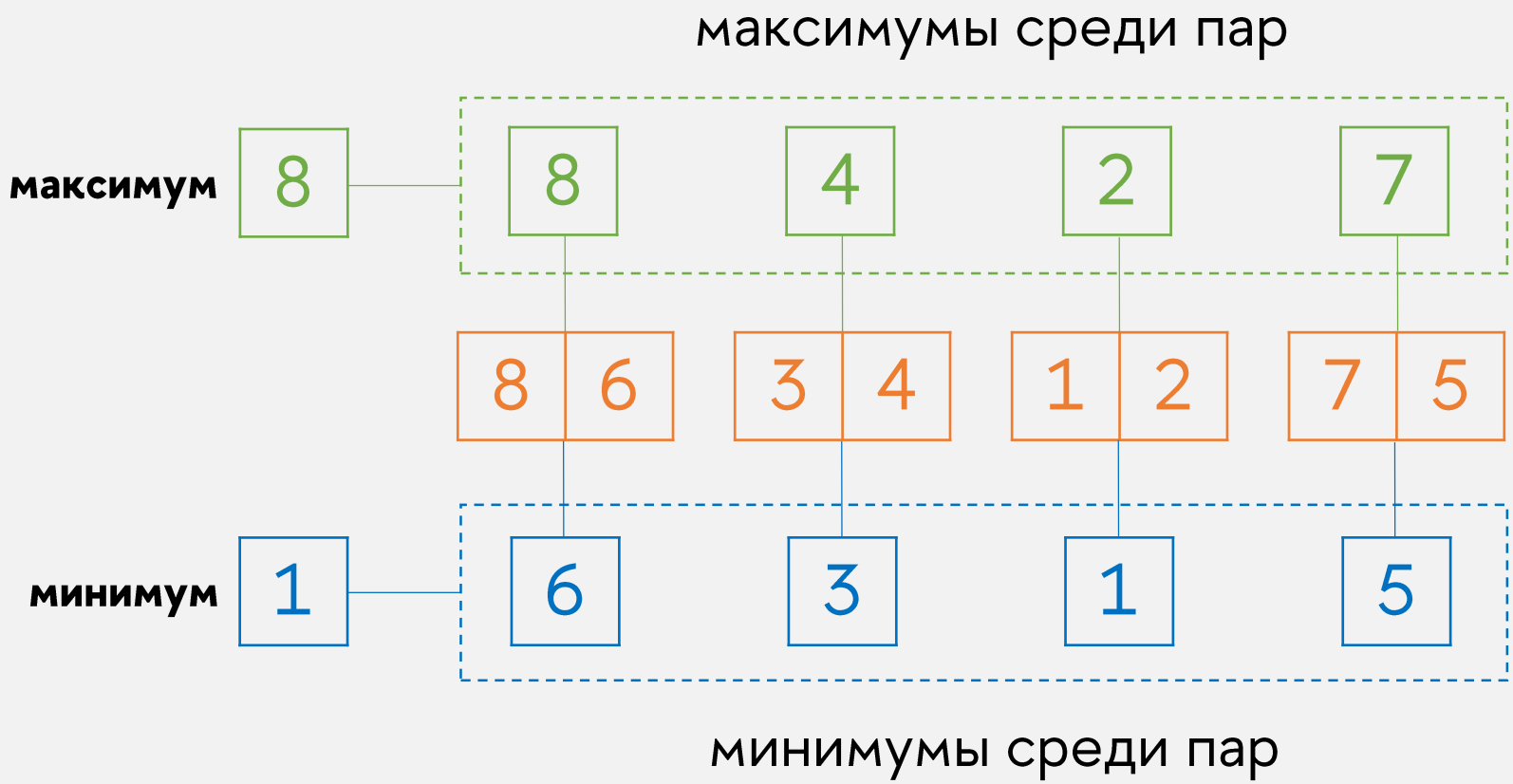
максимумы среди пар



минимумы среди пар

Задача 1

NAÏVE SELECT для min И max



DaC-подход потребует $^{n}/_{2} + {}^{n}/_{2} + {}^{n}/_{2} = {}^{3n}/_{2}$ сравнений $\leq 2n - 2$

Задача 1





NAÏVE SELECT для общих целей

Чтобы найти ***k*-ую порядковую статистику** необходимо:

1. **Отсортировать** массив любым доступным способом.
2. **Выбрать *k*-ый элемент** в отсортированном массиве.

сложность такого подхода определяется сложностью сортировки

Порядковые статистики
RANDOMIZED SELECT

Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```

Алгоритм типа Лас Вегас повторяет логику QUICK SORT

`partition(mas, l, r)`

- случайный выбор опорного элемента
- перестановка его на свое место pos в массиве

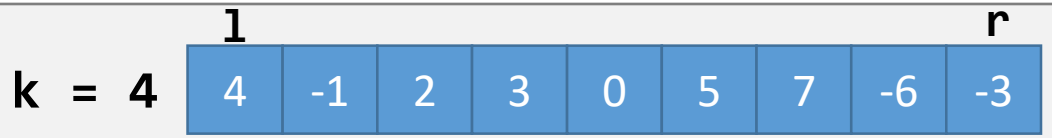
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



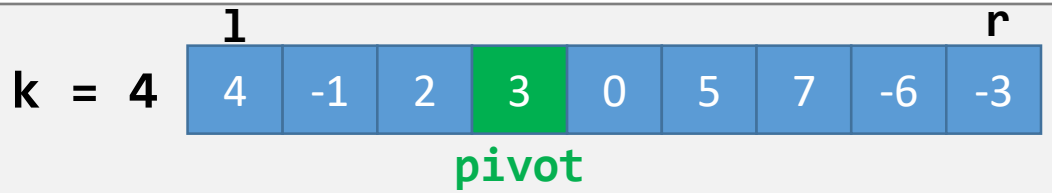
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



- Для удобства можем переместить `pivot` в конец
- Остальные элементы массива переставляем в зависимости от сравнения с `pivot`
- Еще один способ...

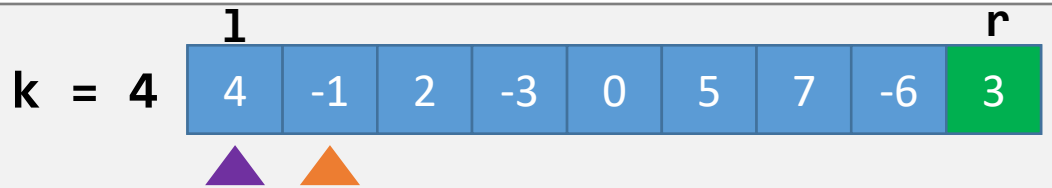
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



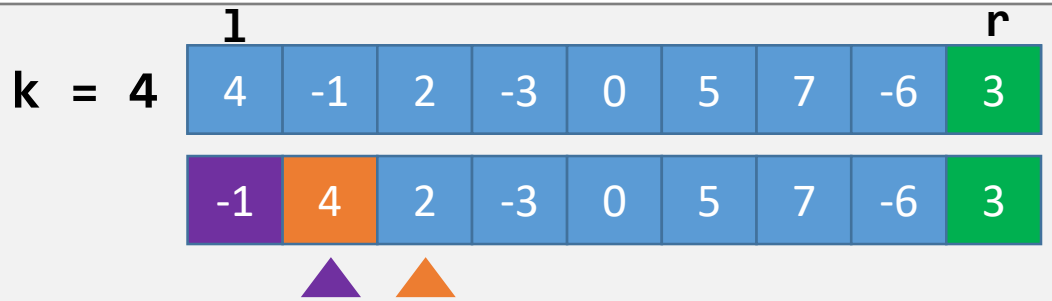
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



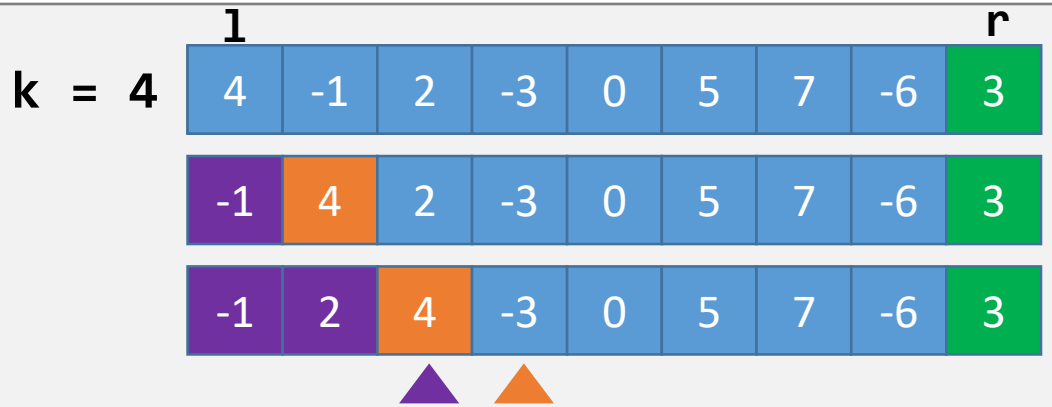
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



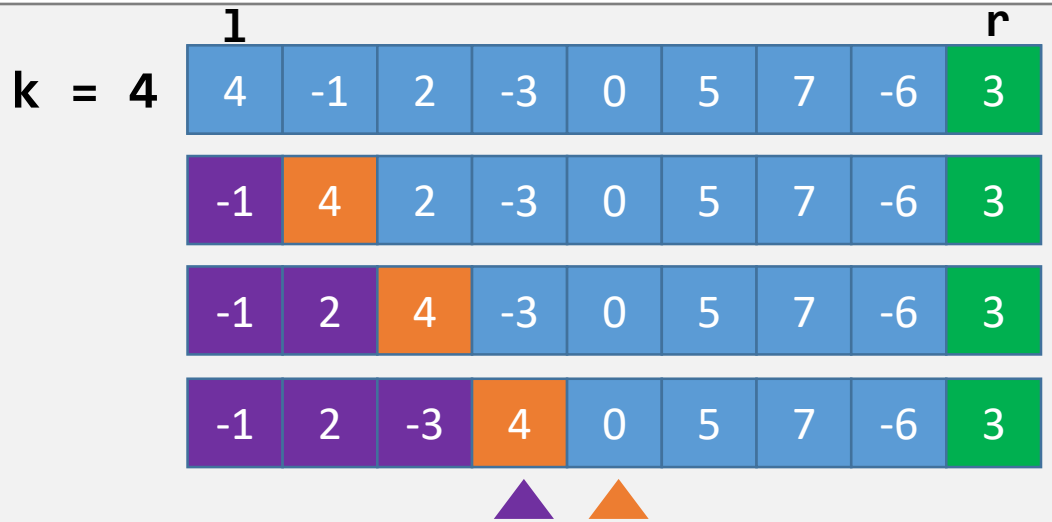
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



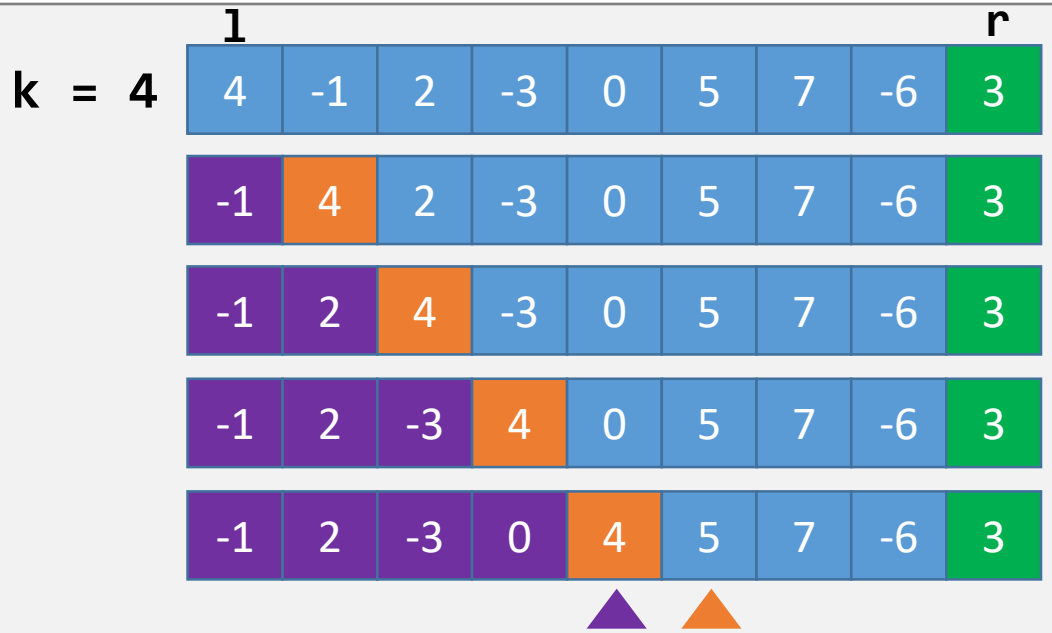
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



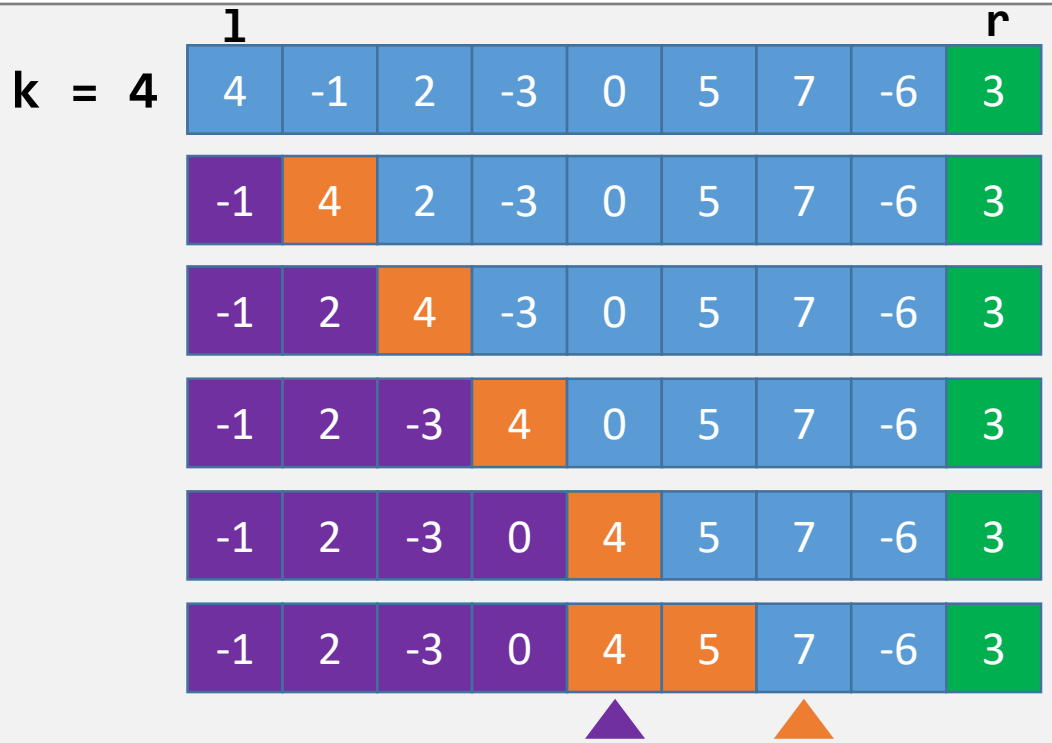
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



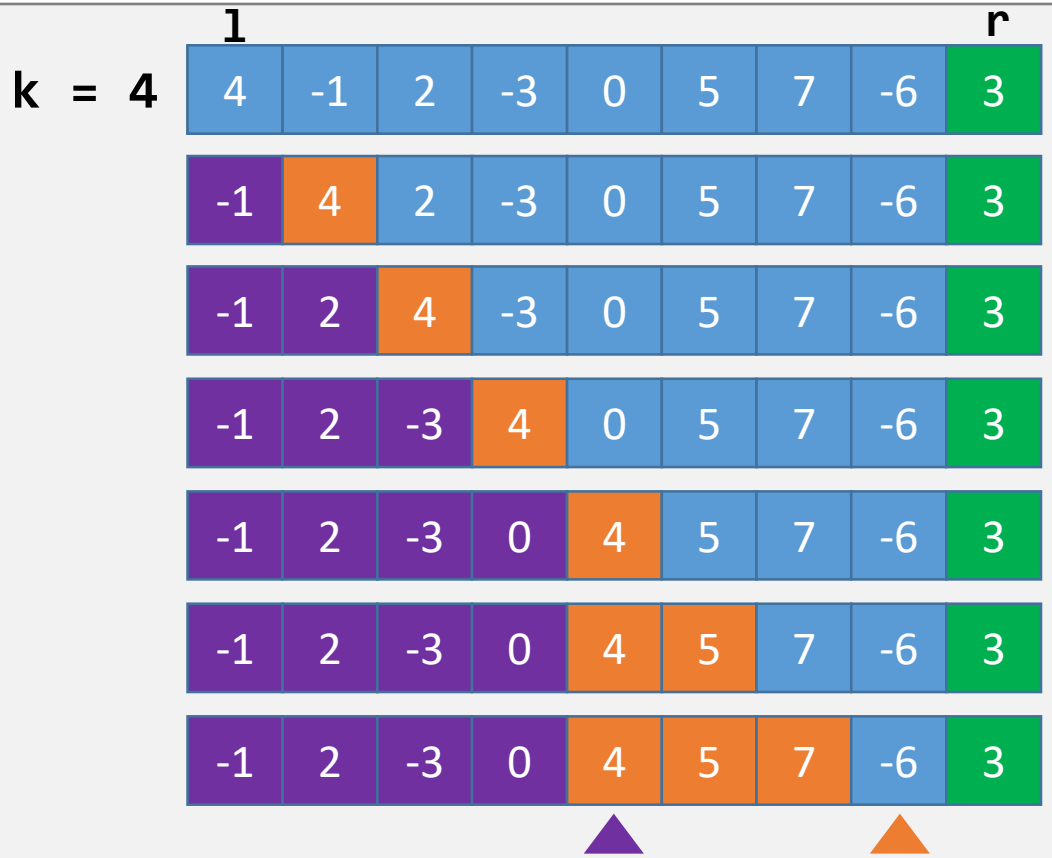
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



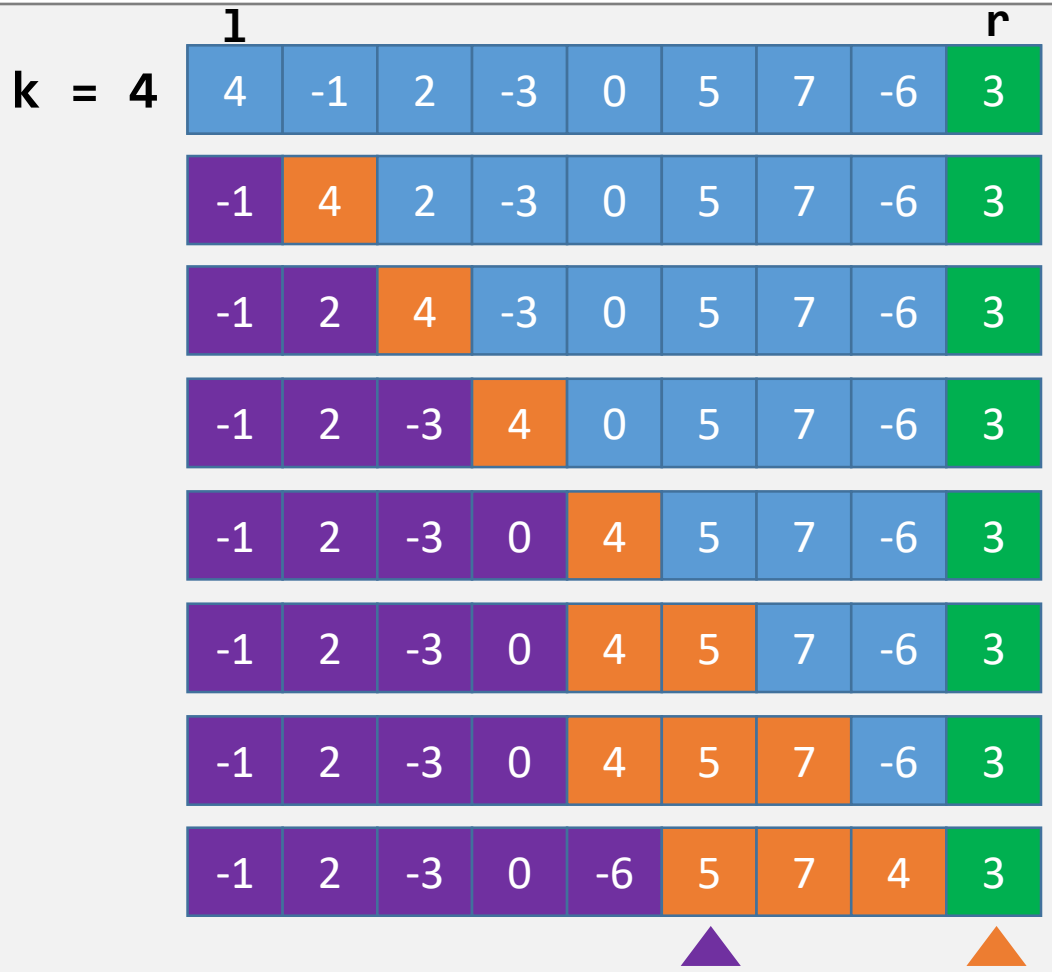
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



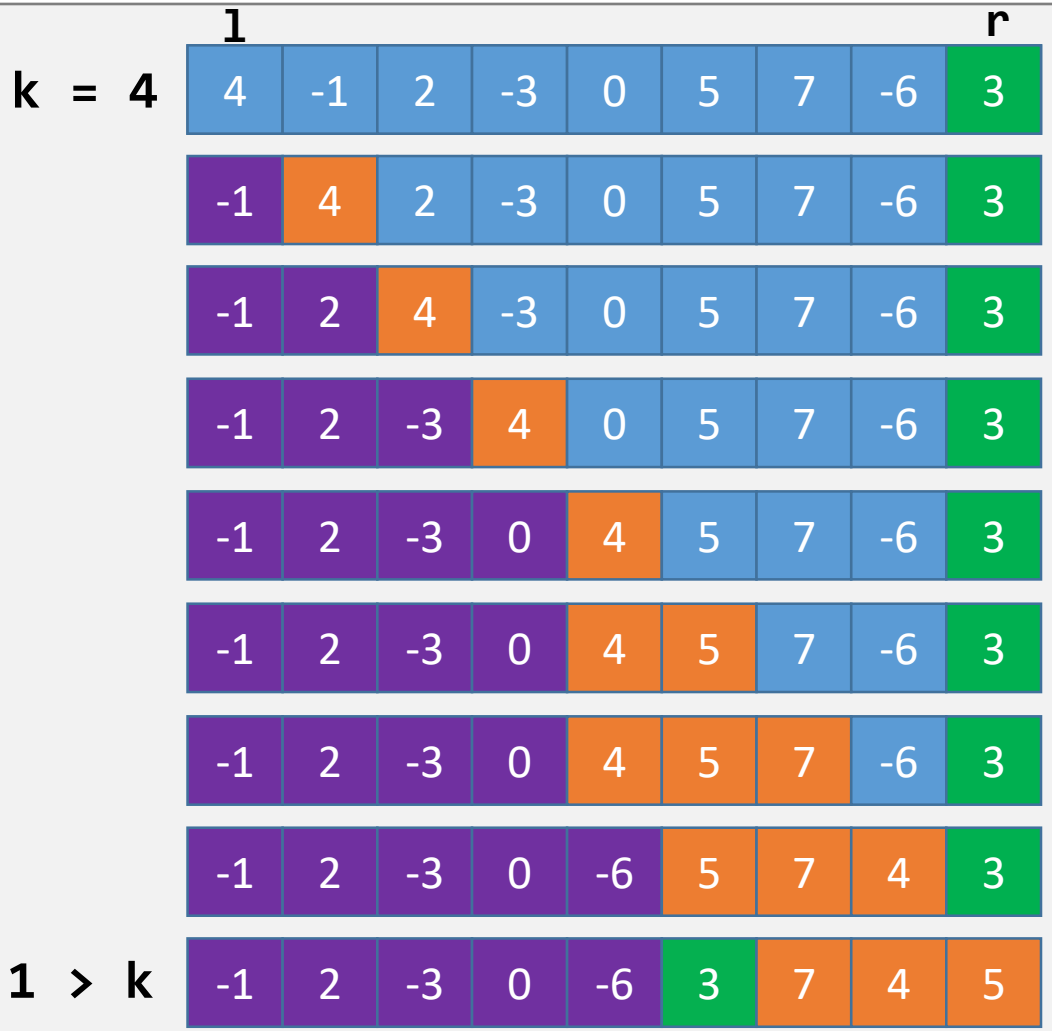
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



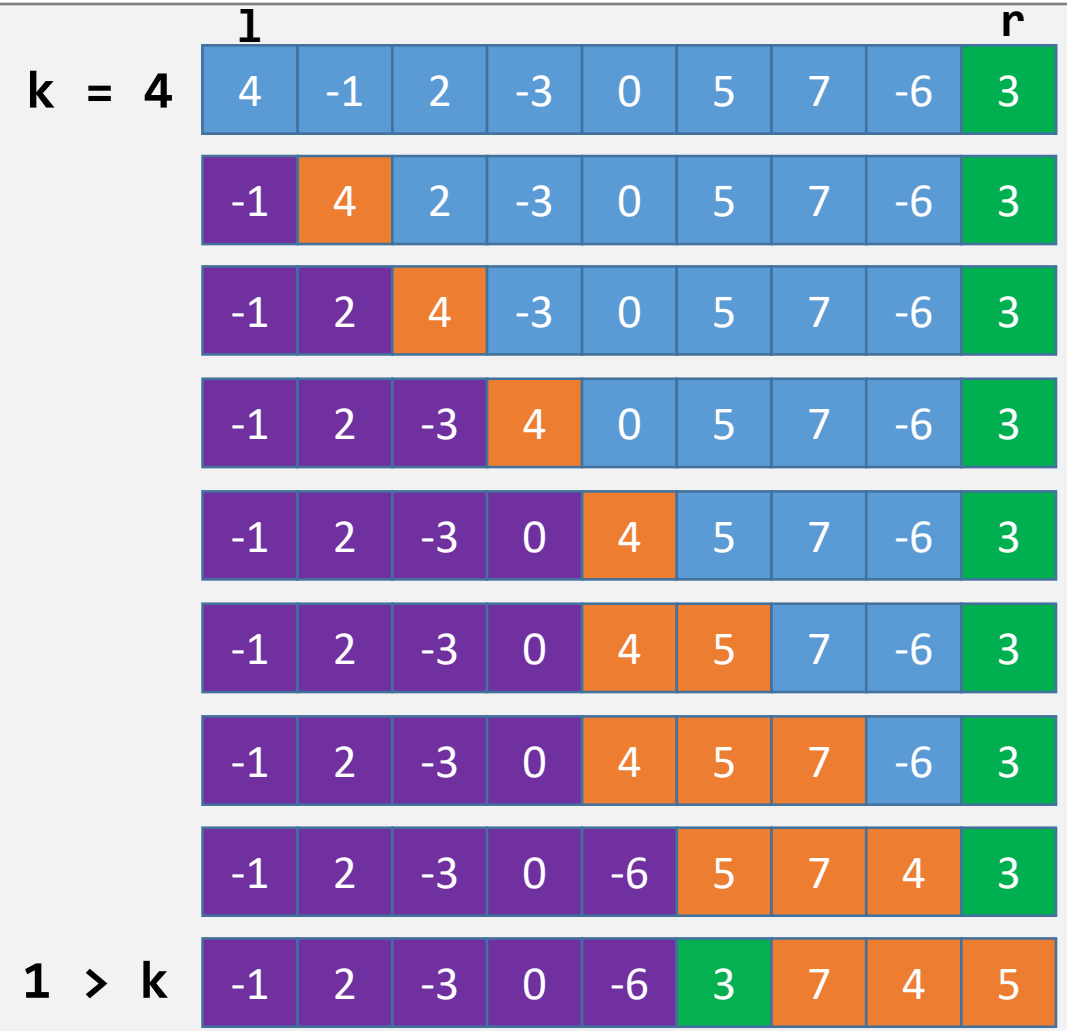
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



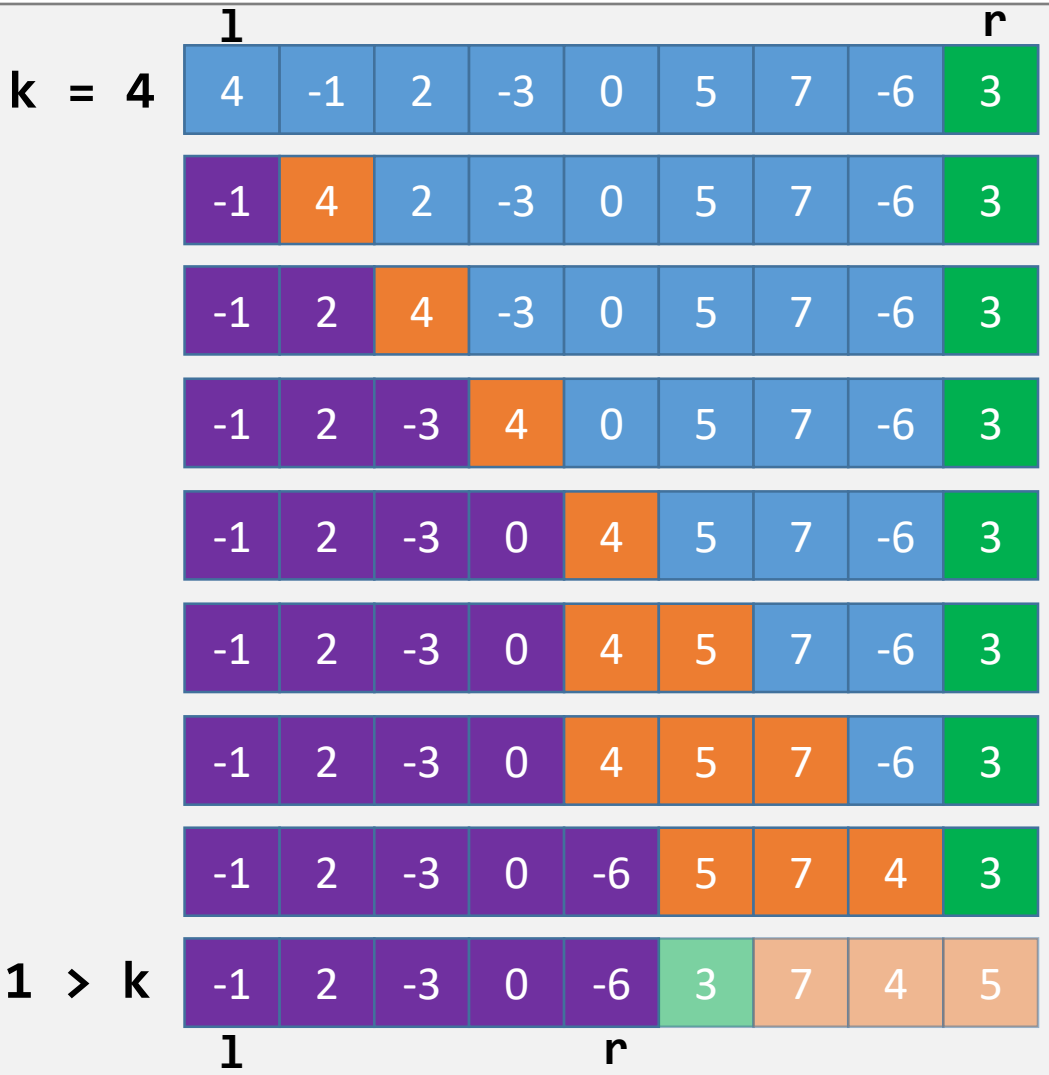
Задача 1

RANDOMIZED SELECT

RANDOMIZED_SELECT.cpp

```cpp
void randomizeSelect(std::vector<int> mas, int k) {
    int left = 0;
    int right = mas.size() - 1;

    for (;;) {
        int pos = partition(mas, left, right);
        int m = pos - left + 1;
        if (m = k) {
            return mas[pos];
        } else if (k < m) {
            right = pos - 1;
        } else {
            left = pos + 1;
            k = k - m;
        }
    }
}
```



Задача 1

Оценка сложности RANDOMIZED SELECT

Соотношение: $T(n) = T(|\mathbf{Left}|) + kn$ **ИЛИ** $T(n) = T(|\mathbf{Right}|) + kn$

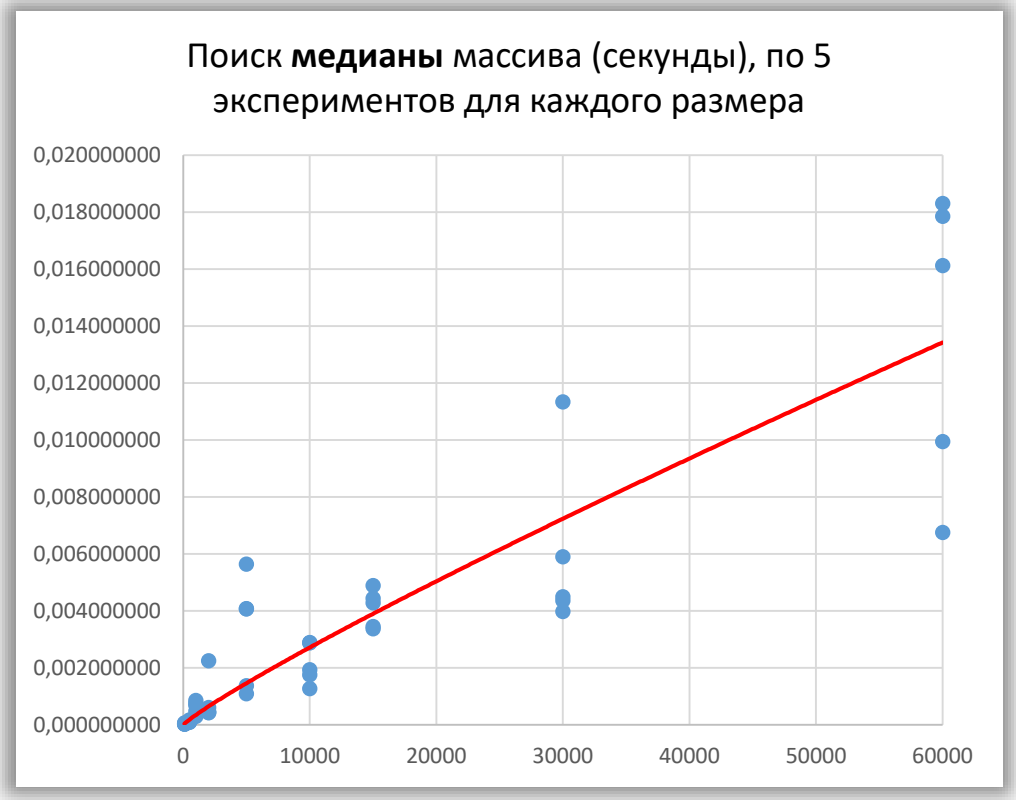
- **Лучший** случай [pivot — медиана] $T(n) = T(n/2) + kn = O(n)$
- **Худший** случай [pivot — min/max] $T(n) = T(n-1) + kn = O(n^2)$
- **Удачный** случай [pivot разбивает в некотором соотношении]

$$T(n) = T\left(\frac{9n}{10}\right) + kn = O(n)$$

- Можно показать, что математическое ожидание сложности $E(T(n))$ алгоритма RANDOMIZED SELECT составит $O(n)$ [в случае равновероятного выбора опорного элемента]

Задача 1

Оценка сложности RANDOMIZED SELECT





Задача 1




RANDOMIZED SELECT ↔ QUICK SORT

Алгоритмы предложены и разработаны **Тони Хоаром**

В основе обоих алгоритмов лежит парадигма «разделяй-и-властвуй»

- Худший случай временной сложности — $O(n^2)$
- Алгоритмы показывают хорошие практические результаты
- Чтобы избежать худшего случая, можно «регуляризовать» отношение, в котором `pivot` делит исходный массив…

Порядковые статистики

LINEAR SELECT

Блюм, Флойд, Пратт, Ривест, Тарьян – 1973

Нужное





Вспомогательное утверждение

1. Пусть $a, b \geq 0$, для которых верно $a + b < 1$
2. Задано соотношение $T(n) = T(an) + T(bn) + kn$, $k > 0$

Тогда верно, что $T(n) = O(n)$.

Нужное




Вспомогательное утверждение

1. Пусть $a, b \geq 0$, для которых верно $a + b < 1$
2. Задано соотношение $T(n) = T(an) + T(bn) + kn, k > 0$

Тогда верно, что $T(n) = O(n)$.

- Основное предположение — $\exists c, n_0 > 0 \; \forall n \geq n_0 : T(n) \leq cn$
- Индуктивная гипотеза — $\forall m < n : T(m) \leq T(n)$
- Выполним подстановку:

$$T(n) \leq can + cbn + kn$$
$$\leq n\big(k + c(a + b)\big)$$

Нужное




Вспомогательное утверждение

1. Пусть $a, b \geq 0$, для которых верно $a + b < 1$
2. Задано соотношение $T(n) = T(an) + T(bn) + kn, k > 0$

Тогда верно, что $T(n) = O(n)$.

- Основное предположение — $\exists c, n_0 > 0\ \forall n \geq n_0 : T(n) \leq cn$
- Индуктивная гипотеза — $\forall m < n : T(m) \leq T(n)$
- Выполним подстановку:

$$T(n) \leq can + cbn + kn$$
$$\leq n\big(k + c(a + b)\big) \leq cn$$

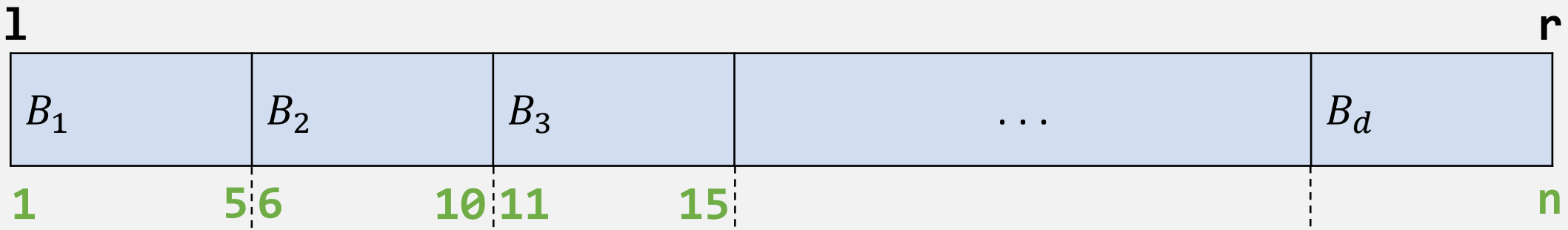
- При условии, что $k + c(a + b) \leq c \Rightarrow c \geq \frac{k}{1-(a+b)} \Rightarrow a + b < 1$

Задача 1

LINEAR SELECT

linearSelect(A, l, r, k), n = r – l + 1 $T(n) = ???$

ШАГ 1 — Разбить исходный массив A на группы по 5 элементов



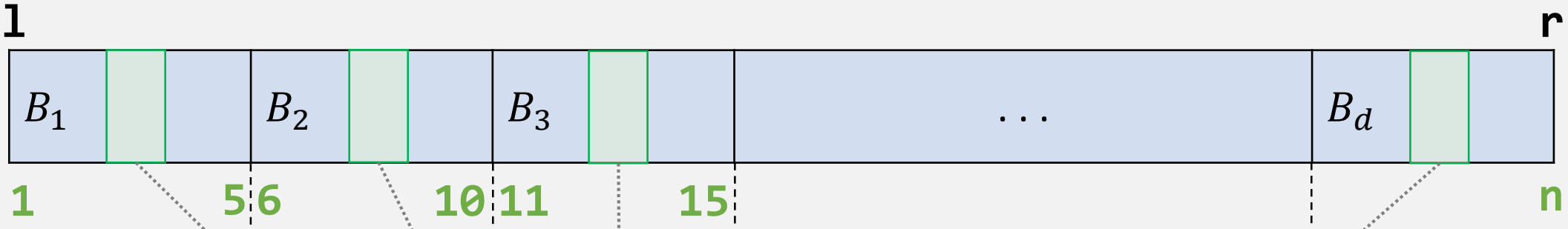
- Число блоков по 5 элементов — $d = {}^{n}/_{5}$
- Последний блок может содержать менее 5 элементов

Задача 1

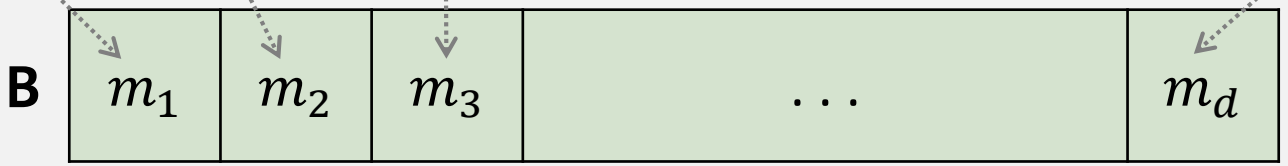
LINEAR SELECT

linearSelect(A, l, r, k), n = r – l + 1 $T(n) = ???$

ШАГ 1 — Разбить исходный массив A на группы по 5 элементов



В каждом блоке B_j найти медиану m_j и записать их в массив B:



$d = {}^{n}/_{5}$

Поиск медианы в блоке — **постоянное** время $0(1)$. ПОЧЕМУ?

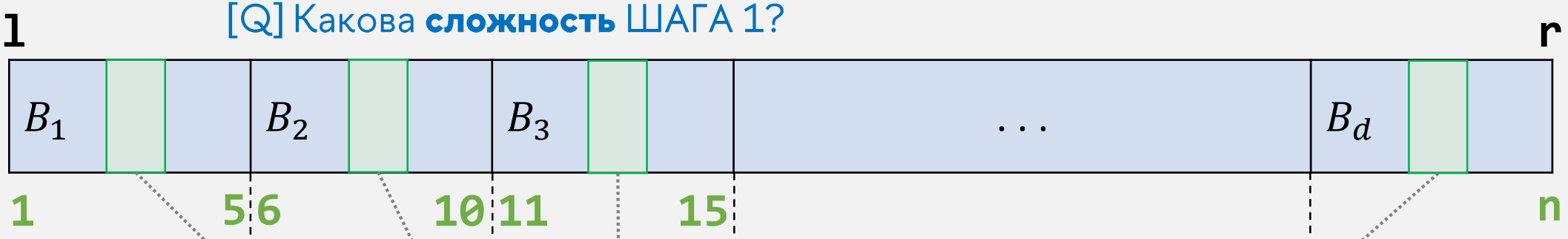
Задача 1

LINEAR SELECT

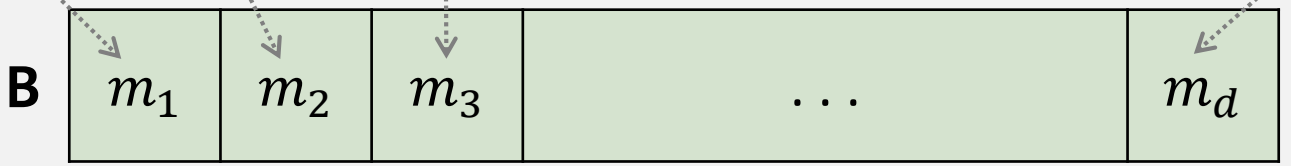
linearSelect(A, l, r, k), n = r – l + 1 $T(n) = ???$

ШАГ 1 — Разбить исходный массив A на группы по 5 элементов

[Q] Какова **сложность** ШАГА 1?



В каждом блоке B_j найти медиану m_j и записать их в массив B:



$d = {}^{n}/_{5}$

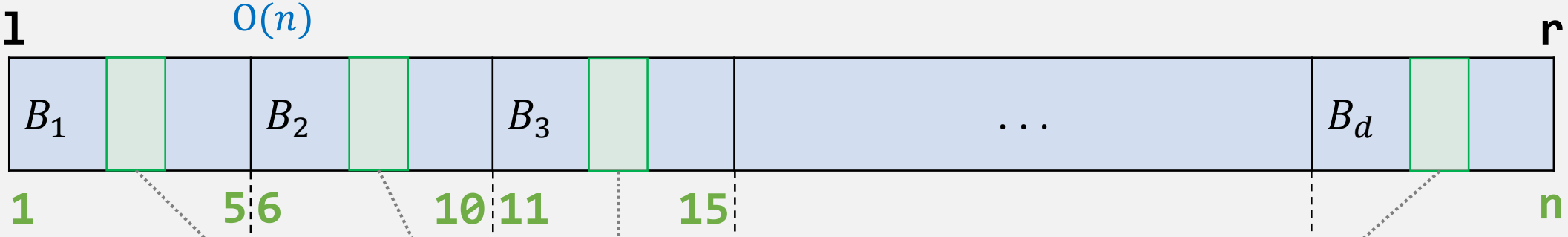
Поиск медианы в блоке — **постоянное** время $O(1)$.

Задача 1

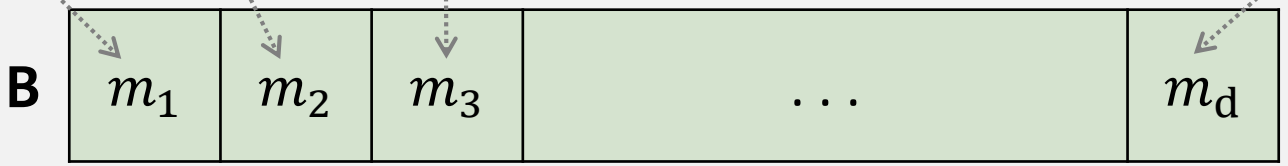
LINEAR SELECT

linearSelect(A, l, r, k), n = r – l + 1 $T(n) = ???$

ШАГ 1 — Разбить исходный массив A на группы по 5 элементов



В каждом блоке B_j найти медиану m_j и записать их в массив B:



$d = {}^{n}/_{5}$

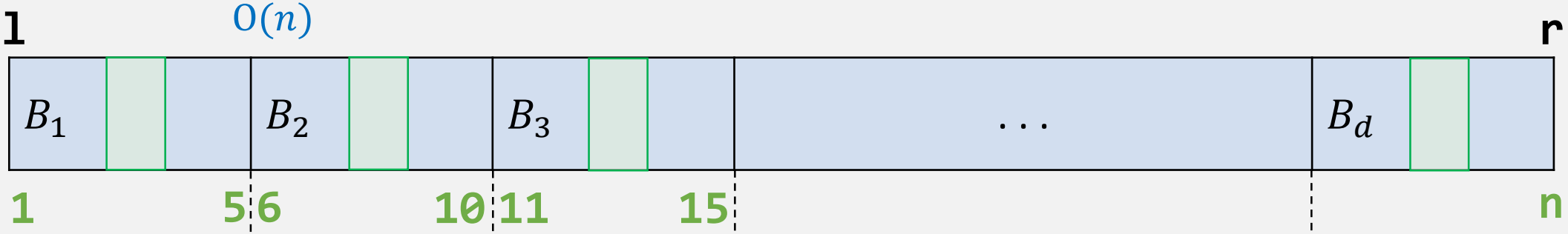
Поиск медианы в блоке — **постоянное** время $O(1)$.

Задача 1

LINEAR SELECT

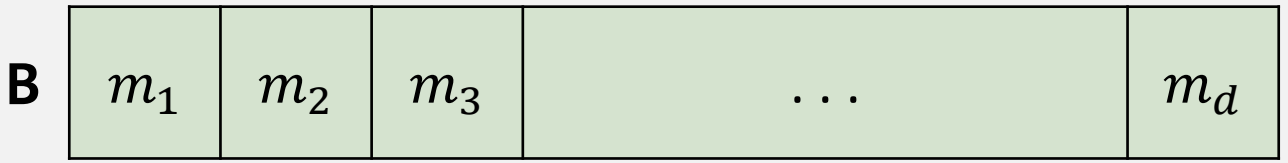
linearSelect(A, l, r, k), **n** = r − l + 1 $T(n) = ???$

ШАГ 1 — Разбить исходный массив A на группы по 5 элементов



ШАГ 2 — Рекурсивный вызов **linearSelect**(B, 1, d, d/2)

$T(n/5)$



$d = n/5$

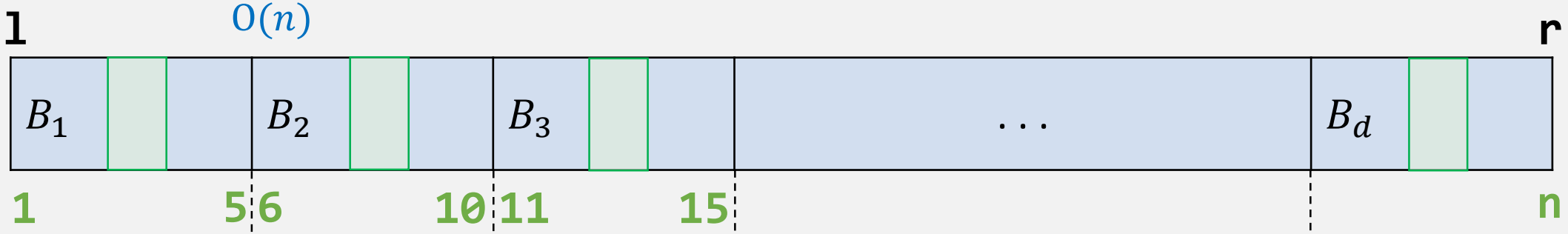
Рекурсивный вызов выполняется для поиска m — **медианы** B.

Задача 1

LINEAR SELECT

linearSelect(A, l, r, k), n = r – l + 1 $T(n) = ???$

ШАГ 3 — Выполнить **partition** для A по медиане m массива B



В результате, m займет корректное место pos в массиве A:



Элемент m поделит массив A в некотором соотношении.

Задача 1

LINEAR SELECT

linearSelect(A, l, r, k), n = r – l + 1 $T(n) = ???$

ШАГ 4 — Рекурсивный **linearSelect** в правильной части A

$T(s)$, где s — размер **наибольшей** части массива после **partition**



небоходимо правильно поменять номер искомой статистики!

- если k = pos – l + 1, то **return** A[pos]
- если k < pos – l + 1, то **linearSelect**(A, l, pos - 1, k)
- если k > pos – l + 1, то **linearSelect**(A, pos + 1, r, k – pos + l – 1)

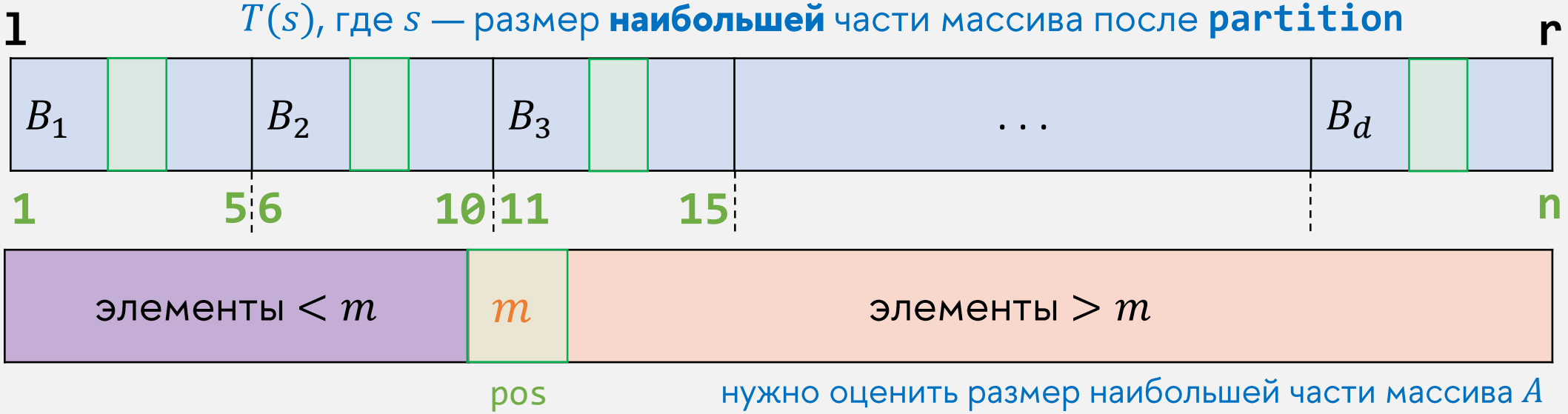
Задача 1

LINEAR SELECT

linearSelect(A, l, r, k), n = r – l + 1

ШАГ 4 — Рекурсивный **linearSelect** в правильной части A

$T(s)$, где s — размер **наибольшей** части массива после **partition**

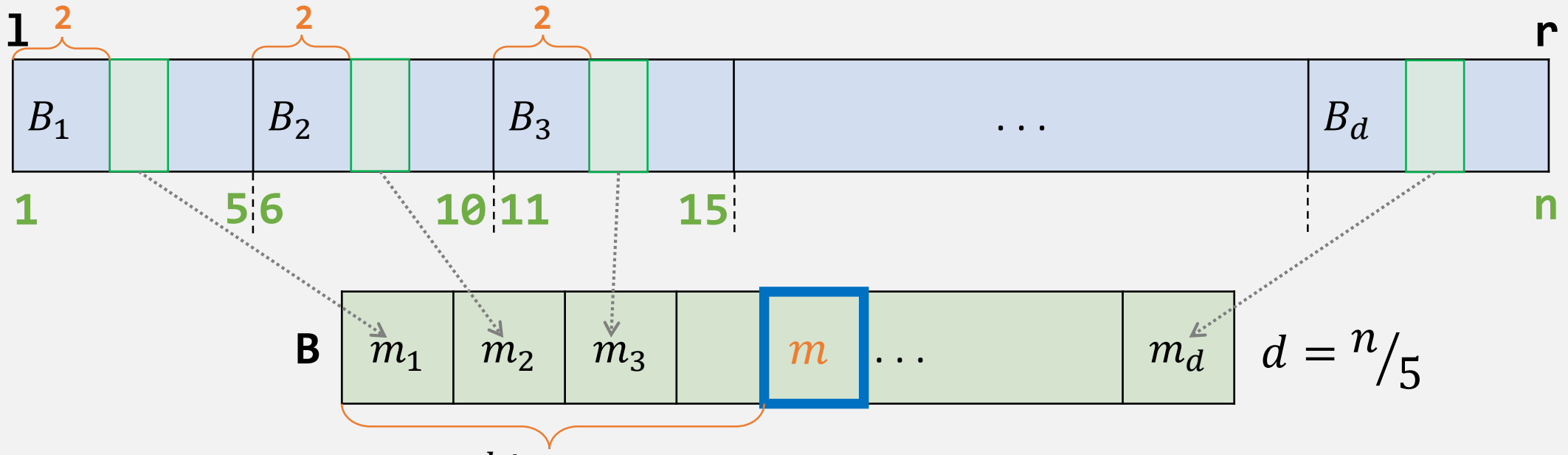


нужно оценить размер наибольшей части массива A

- если k = pos – l + 1, то **return** A[pos]
- если k < pos – l + 1, то **linearSelect**(A, l, pos - 1, k)
- если k > pos – l + 1, то **linearSelect**(A, pos + 1, r, k – pos + l – 1)

Задача 1

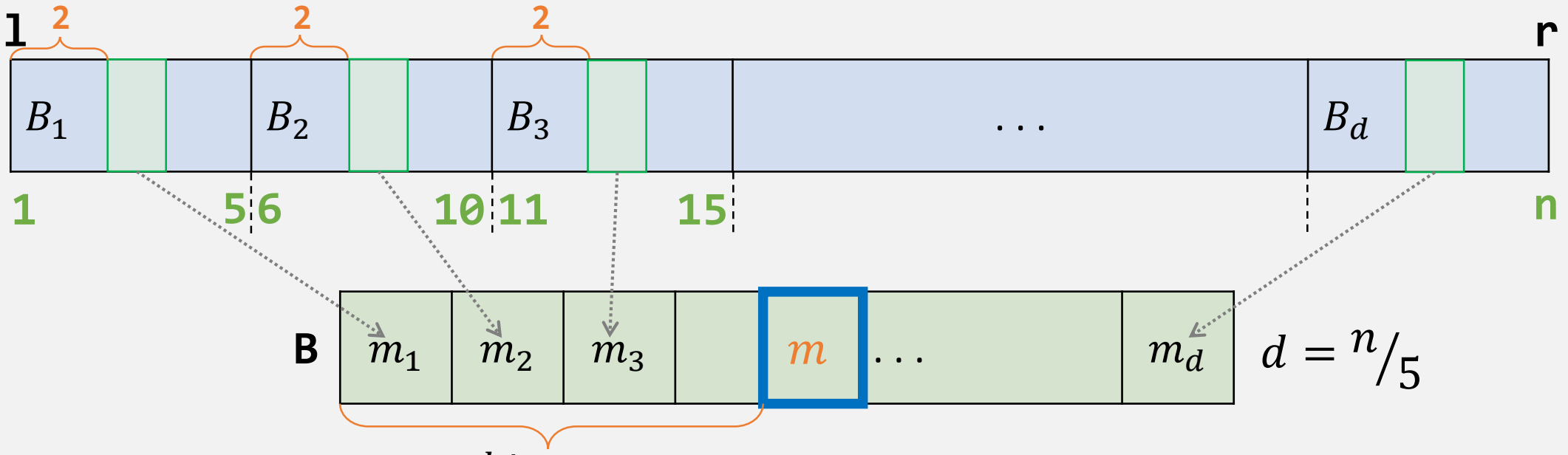
LINEAR SELECT



1. В массиве B всего ${}^{d}/_{2}$ элементов, меньших m
2. Для медианы m_j из каждой группы B_j в массиве A найдется еще по два элемента, меньших m_j

Задача 1

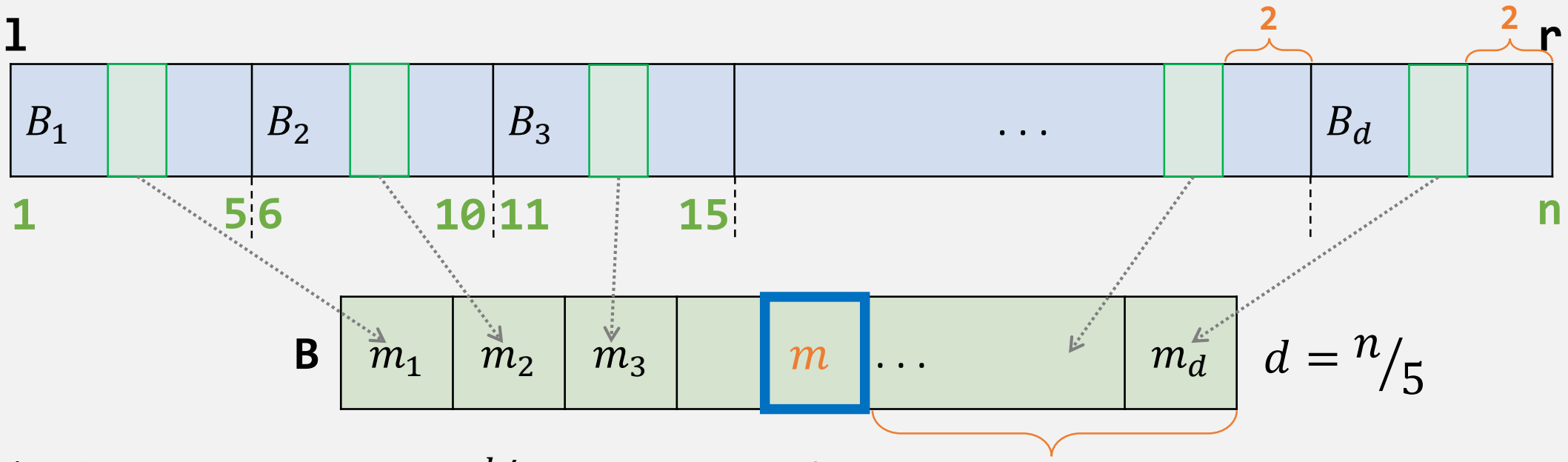
LINEAR SELECT



1. В массиве B всего ${}^{d}/_{2}$ элементов, меньших m
2. Для медианы m_j из каждой группы B_j в массиве A найдется еще по два элемента, меньших m_j
3. Следовательно, число элементов в A, меньших m, $\geq {}^{3d}/_{2} = {}^{3n}/_{10}$
4. Тогда, число элементов в A, больших m, $\leq n - {}^{3n}/_{10} = {}^{7n}/_{10}$

Задача 1

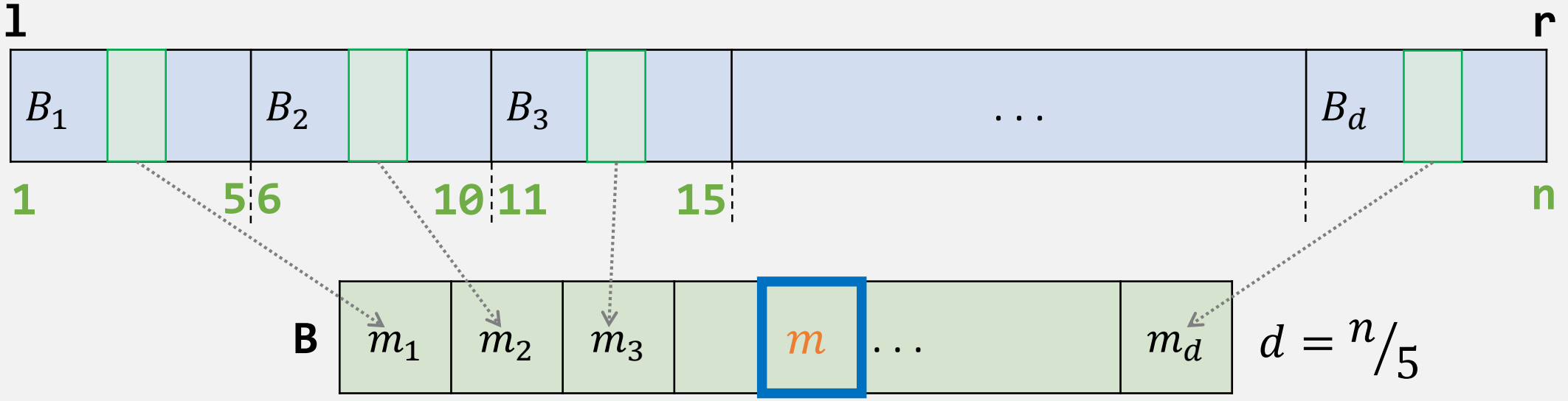
LINEAR SELECT



1. В массиве B всего ${}^{d}/_{2}$ элементов, больших m
2. Для медианы m_j из каждой группы B_j в массиве A найдется еще по два элемента, больших m_j
3. Следовательно, число элементов в A, больших m, $\geq {}^{3d}/_{2} = {}^{3n}/_{10}$
4. Тогда, число элементов в A, меньших m, $\leq n - {}^{3n}/_{10} = {}^{7n}/_{10}$

Задача 1

LINEAR SELECT





Оценка размера **большей** части массива A после **partition** — ${}^{7n}/_{10}$

Задача 1




LINEAR SELECT

linearSelect(A, l, r, k)

1. $O(n)$ Разбиение массива A на блоки по 5 элементов и формирование массива B из медиан блоков
2. $T(n/5)$ Рекурсивный поиск медианы в массиве B
3. $O(n)$ Выполнение **partition** массива A по медиане B
4. $T(7n/10)$ Рекурсивный поиск в правильной части массива A

Задача 1

LINEAR SELECT

linearSelect(A, l, r, k)

1. $O(n)$ Разбиение массива A на блоки по 5 элементов и формирование массива B из медиан блоков
2. $T(n/5)$ Рекурсивный поиск медианы в массиве B
3. $O(n)$ Выполнение **partition** массива A по медиане B
4. $T(7n/10)$ Рекурсивный поиск в правильной части массива A

$$T(n) = T(n/5) + T(7n/10) + k_1 n + k_2 n$$

- Установим, что $1/5 + 7/10 = 9/10 \leq 1$
- По вспомогательному утверждению, $T(n) = O(n)$

Задача 1




LINEAR SELECT

Теоретическая оценка сложности LINEAR SELECT действительно **дает гарантию** линейной сложности, но необходимо учесть, что здесь скрывается большая константа...

$$T(n) = T(n/5) + T(7n/10) + Kn \leq cn, \text{ где } K = k_1 + k_2$$

- В рамках индуктивной гипотезы $\forall m < n: T(m) \leq T(n)$:

$$T(n) \leq cn/5 + 7cn/10 + Kn = (9c/10 + K)n \leq cn$$

- При условии, что $9c/10 + K \leq c \Rightarrow c \geq 10K$
- Базу рекуррентного соотношения проверьте самостоятельно

Задача 1



LINEAR SELECT — пример k = 7

6	2	12	4	5	17	7	14	3	18	1	15	9	10	19	11	8	13	16

ШАГ 1 — Разбиваем массив на блоки, сортируем их и находим медианы

Задача 1

LINEAR SELECT — пример k = 7

6	2	12	4	5	17	7	14	3	18	1	15	9	10	19	11	8	13	16

ШАГ 1 — Разбиваем массив на блоки, сортируем их и находим медианы

2	4	5	6	12	3	7	14	17	18	1	9	10	15	19	8	11	13	16

5	14	10	11

Задача 1

LINEAR SELECT — пример k = 7

6	2	12	4	5	17	7	14	3	18	1	15	9	10	19	11	8	13	16

ШАГ 1 — Разбиваем массив на блоки, сортируем их и находим медианы

2	4	5	6	12	3	7	14	17	18	1	9	10	15	19	8	11	13	16

5	14	10	11

ШАГ 2 — Рекурсивно ищем медиану m в массиве медиан блоков

5	10	11	14

Задача 1

LINEAR SELECT — пример k = 7

6	2	12	4	5	17	7	14	3	18	1	15	9	10	19	11	8	13	16

ШАГ 1 — Разбиваем массив на блоки, сортируем их и находим медианы

2	4	5	6	12	3	7	14	17	18	1	9	10	15	19	8	11	13	16

5	14	10	11

ШАГ 2 — Рекурсивно ищем медиану m в массиве медиан блоков

5	10	11	14

ШАГ 3 — Выполняем **partition** исходного массива относительно m

6	2	8	4	5	9	7	1	3	10	14	15	17	16	19	11	12	13	18

Задача 1

LINEAR SELECT — пример k = 7

6	2	12	4	5	17	7	14	3	18	1	15	9	10	19	11	8	13	16

ШАГ 1 — Разбиваем массив на блоки, сортируем их и находим медианы

2	4	5	6	12	3	7	14	17	18	1	9	10	15	19	8	11	13	16

5	14	10	11

ШАГ 2 — Рекурсивно ищем медиану m в массиве медиан блоков

5	10	11	14

ШАГ 3 — Выполняем **partition** исходного массива относительно m

6	2	8	4	5	9	7	1	3	10	~~14~~	~~15~~	~~17~~	~~16~~	~~19~~	~~11~~	~~12~~	~~13~~	~~18~~

ШАГ 4 — Рекурсивно переходим в нужную часть массива относительно позиции m...

Задача 1




QUICK SORT + LINEAR SELECT

Как можно обеспечить сложность $O(n \log n)$ для QUICK SORT в худшем случае?

- Выполнить поиск медианы исходного массива с помощью LINEAR SELECT
- Использовать найденную медиану в качестве опорного элемента **pivot**

Метод имитации отжига

улучшение перебора

Задача 2





Физические аспекты процесса отжига

Процесс термической обработки металлов и сплавов

1. Быстрый нагрев металла до определенной температуры T
2. Медленное охлаждение металла до нормальных условий T_0

- В результате происходит рекристаллизация — атомы формируют кристаллическую решетку и возвращаются в оптимальное состояние с **минимальной** энергией
- При этом, некоторые атомы в процессе отжига могут с некоторой вероятностью переходить в состояние с **более высокой** энергией

Задача 2




Глобальная оптимизация

ЦЕЛЬ — Найти минимальное значение функции энергии $f(x_1, x_2, \ldots, x_n)$

- Текущая температура T и коэффициент охлаждения α, близкий к 1
- Текущий случайный кандидат $x_1^0, x_2^0, \ldots, x_n^0$ и $f_{\text{cur}} = f(x_1^0, x_2^0, \ldots, x_n^0)$

Задача 2

Глобальная оптимизация

ЦЕЛЬ — Найти минимальное значение функции энергии $f(x_1, x_2, \dots, x_n)$

- Текущая температура T и коэффициент охлаждения α, близкий к 1
- Текущий случайный кандидат $x_1^0, x_2^0, \dots, x_n^0$ и $f_{\text{cur}} = f(x_1^0, x_2^0, \dots, x_n^0)$

Запускаем имитацию процесса отжига `SIMULATED ANNEALING`:

1. Охлаждение — $T_{\text{cur}} = \alpha \cdot T$
2. Случайное изменение — $x_1' = x_1^0 \pm a, x_2' = x_2^0 \pm a, \dots, x_n' = x_n^0 \pm a$
3. Разность энергий — $\Delta f = f(x_1', x_2', \dots, x_n') - f_{\text{cur}}$
   - если $\Delta f < 0$, то $x_1', x_2', \dots, x_n'$ — точно новый кандидат
   - если $\Delta f > 0$, то $x_1', x_2', \dots, x_n'$ — новый кандидат с вероятностью $P = e^{-\Delta f / T_{\text{cur}}}$
4. Повторяем 1–3, пока не достигнем нормальных условий T_0

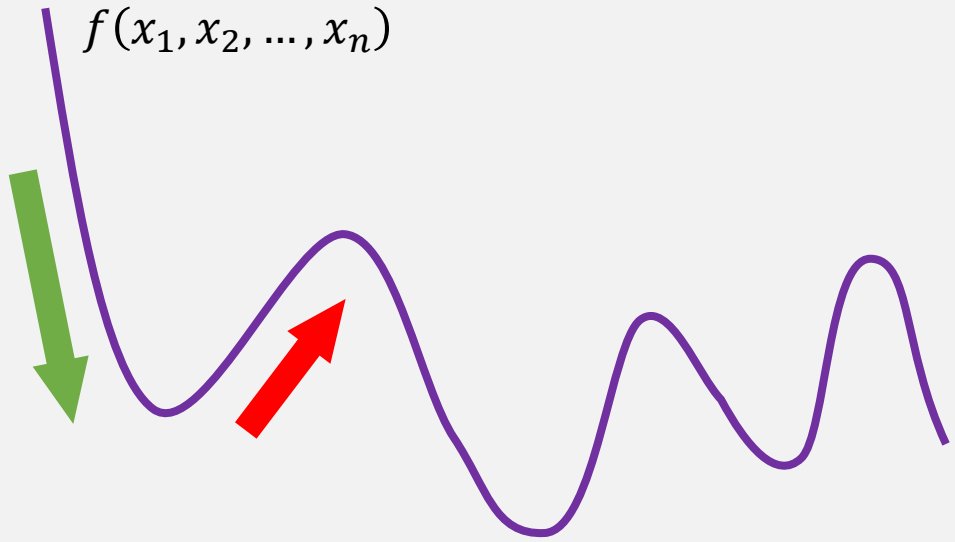
Задача 2

Глобальная оптимизация

ЦЕЛЬ — Найти минимальное значение функции энергии $f(x_1, x_2, \dots, x_n)$

- Текущая температура T и коэффициент охлаждения α, близкий к 1
- Текущий случайный кандидат $x_1^0, x_2^0, \dots, x_n^0$ и $f_{\text{cur}} = f(x_1^0, x_2^0, \dots, x_n^0)$

[Q] Почему иногда необходимо выполнять **НЕОПТИМАЛЬНЫЕ шаги**?

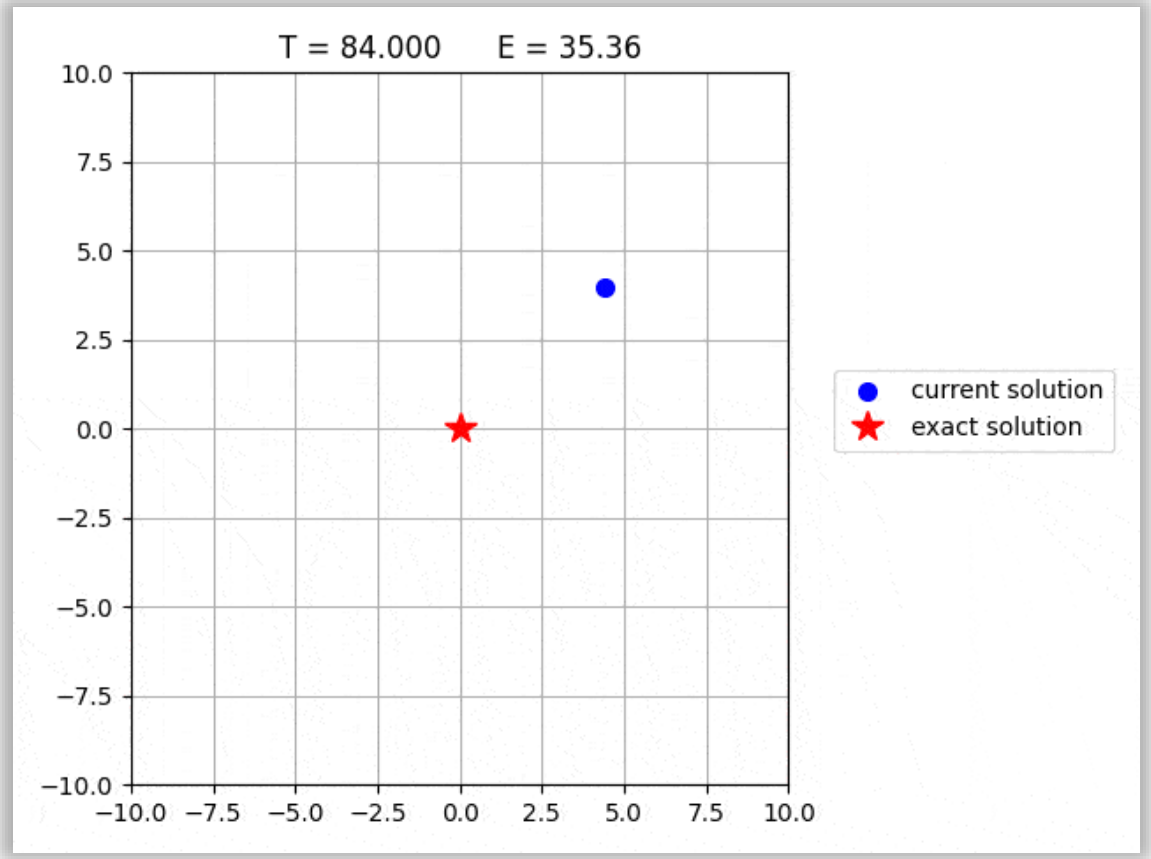


- Вероятностный переход в состояние с более высокой энергией поможет нам избежать локально оптимальных решений
- Вероятность $P = e^{-\Delta f / T_{\text{cur}}}$ выполнения неоптимальных шагов падает с падением температуры в `SIMULATED ANNEALING`

Задача 2

Глобальная оптимизация — пример

ЦЕЛЬ — Найти минимальное значение функции энергии $f(x, y) = x^2 + y^2$



T — температура

E — энергия, значение функции

- Пространство поиска ограничено $(x, y) \in [-10, 10] \times [-10, 10]$
- Коэффициент охлаждения $\alpha = 0.84$
- Начальный кандидат $(4,4)$
- Случайное изменение ± 0.2

Истинно оптимальное решение может быть не найдено!

Задача 2





SIMULATED ANNEALING

[Q1] Как реализовать вероятностный выбор неоптимального перехода в состояние с более высокой энергией?

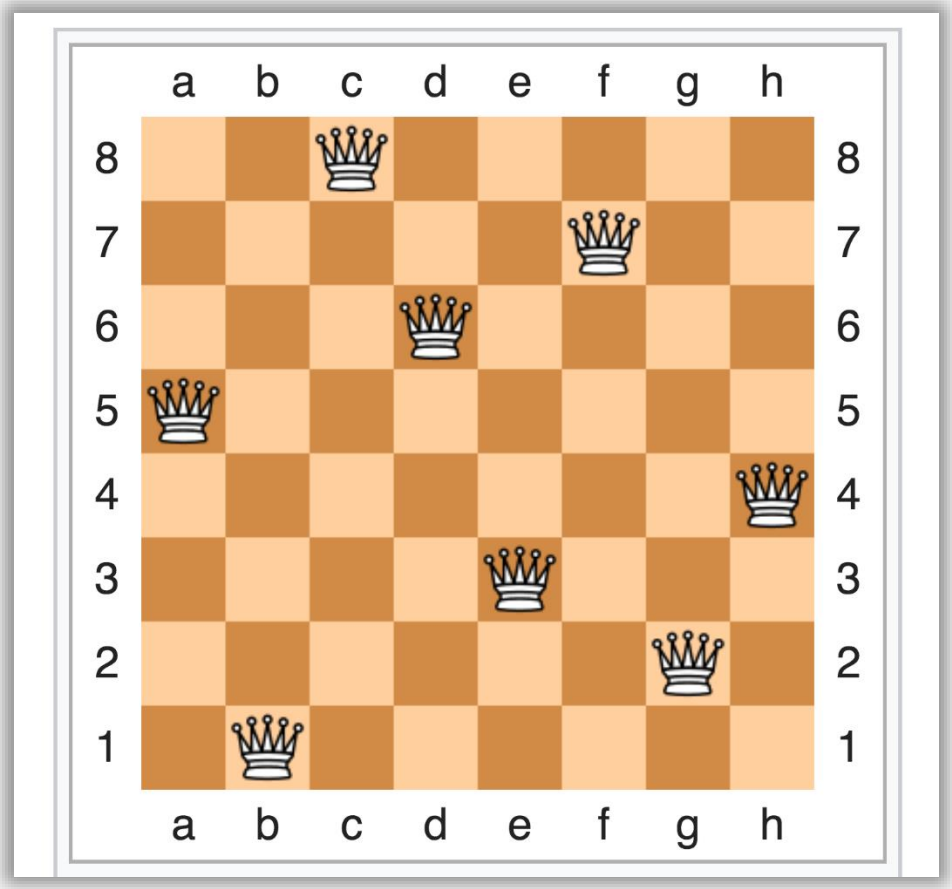
случайное число в интервале $[0,1) < P = e^{-\Delta f / T_{\text{cur}}}$

[Q2] Как ограничить число итераций алгоритма?

1. Определить минимальную температуру T_0 для окончания алгоритма
2. Определить минимальную энергию E_0 для окончания алгоритма
3. Явным образом задать максимальное число итераций

Задача 2

N-QUEENS PROBLEM



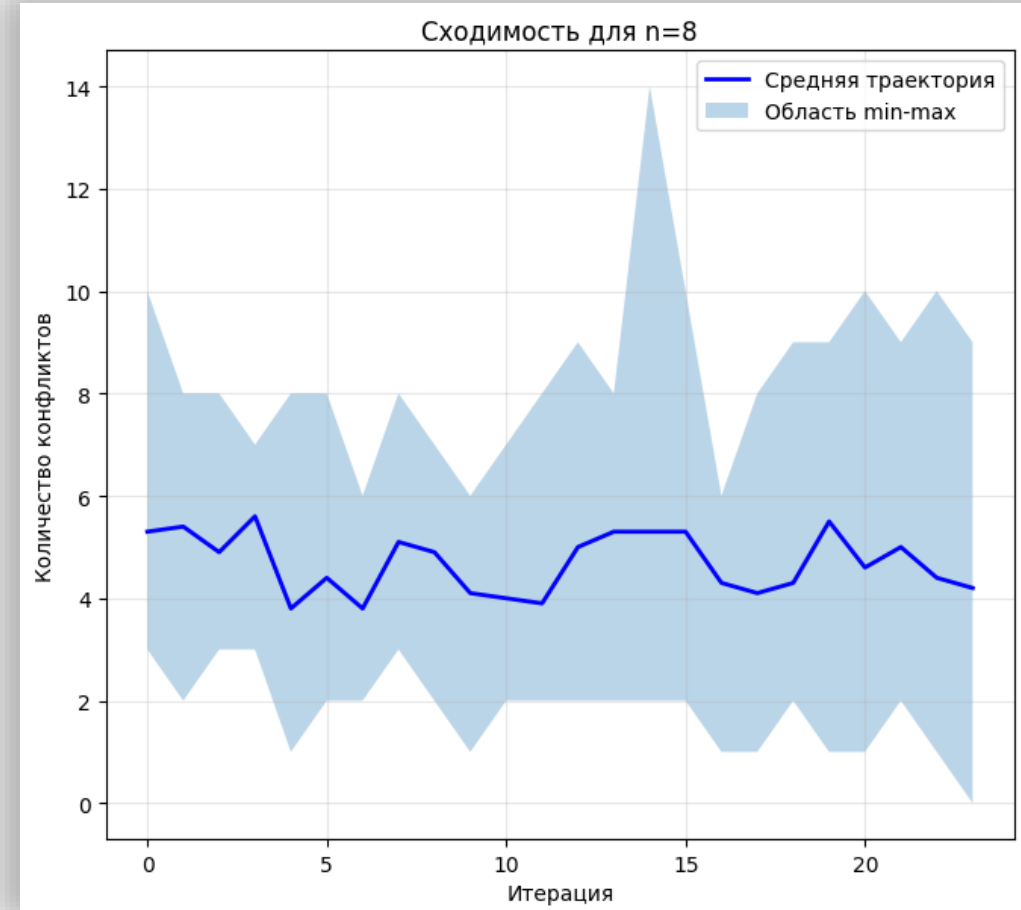
ЦЕЛЬ — расставить N ферзей на шахматной доске $N \times N$

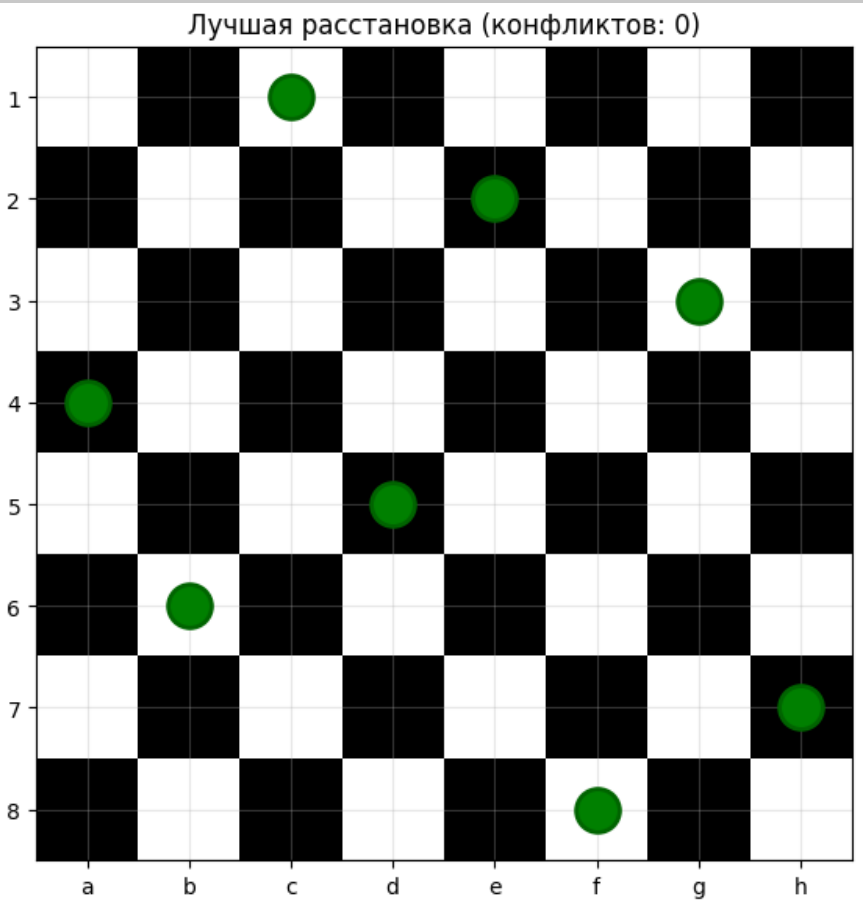
- функция энергии f
  количество пар ферзей, которые могут атаковать друг друга
- случайное изменение
  обмен горизонтальных позиций двух ферзей, выбранных случайно

Требуется найти минимальное значение функции энергии f

Задача 2

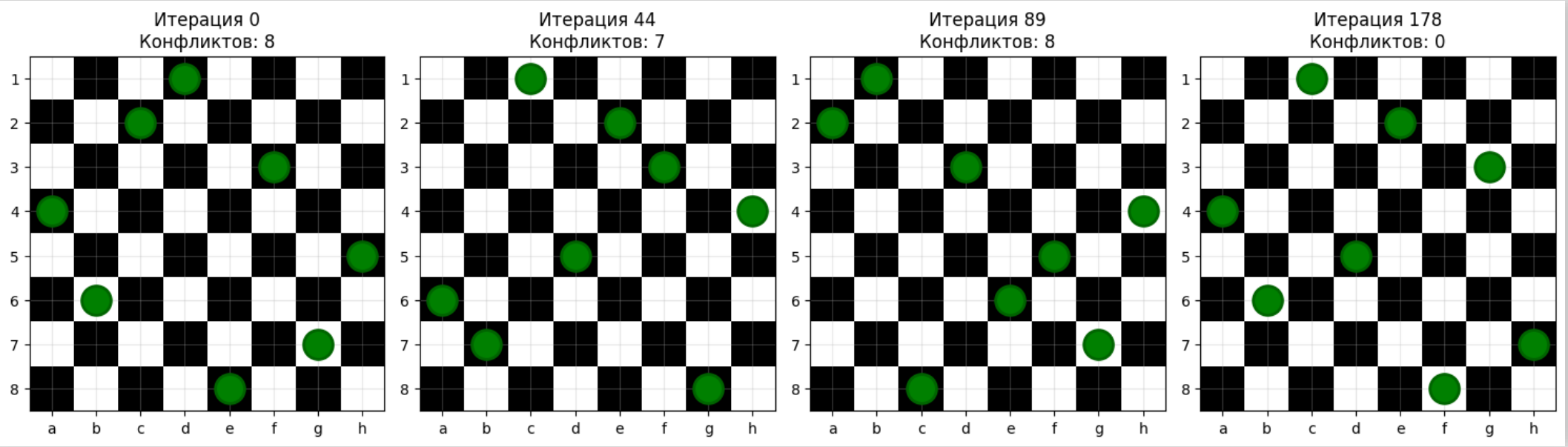
8-QUEENS PROBLEM





Задача 2

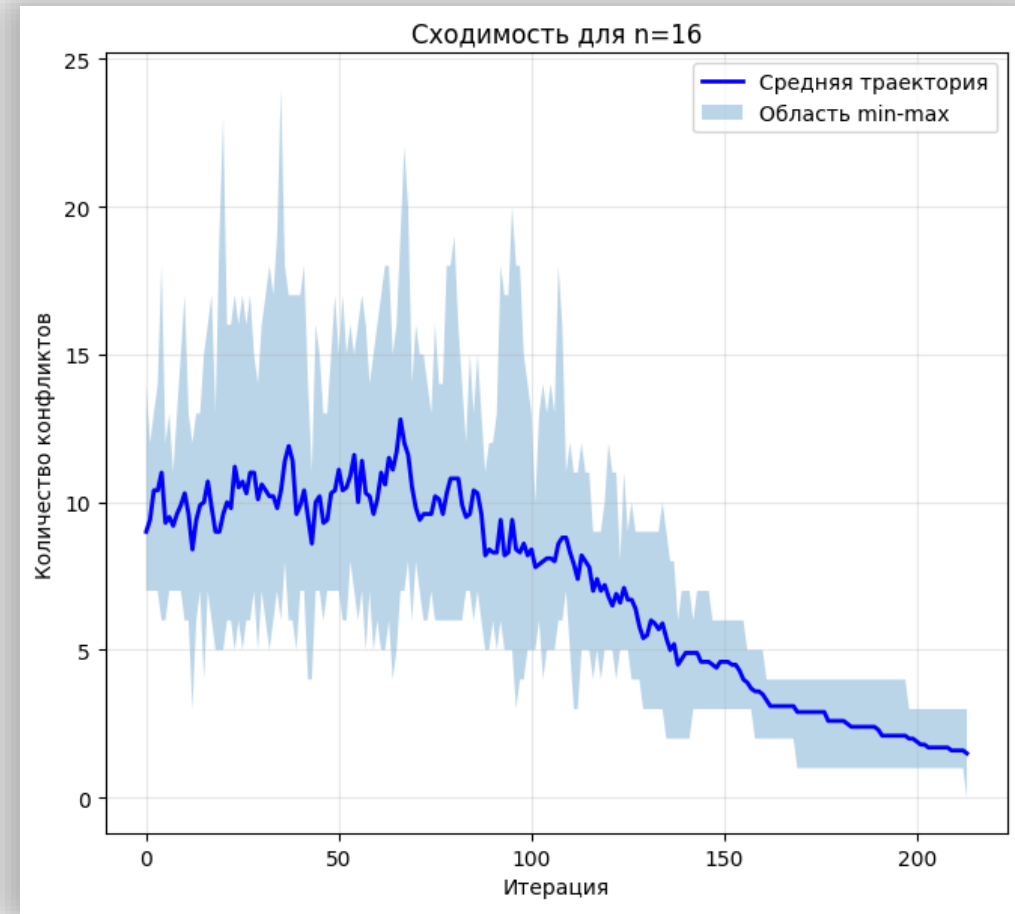
8-QUEENS PROBLEM

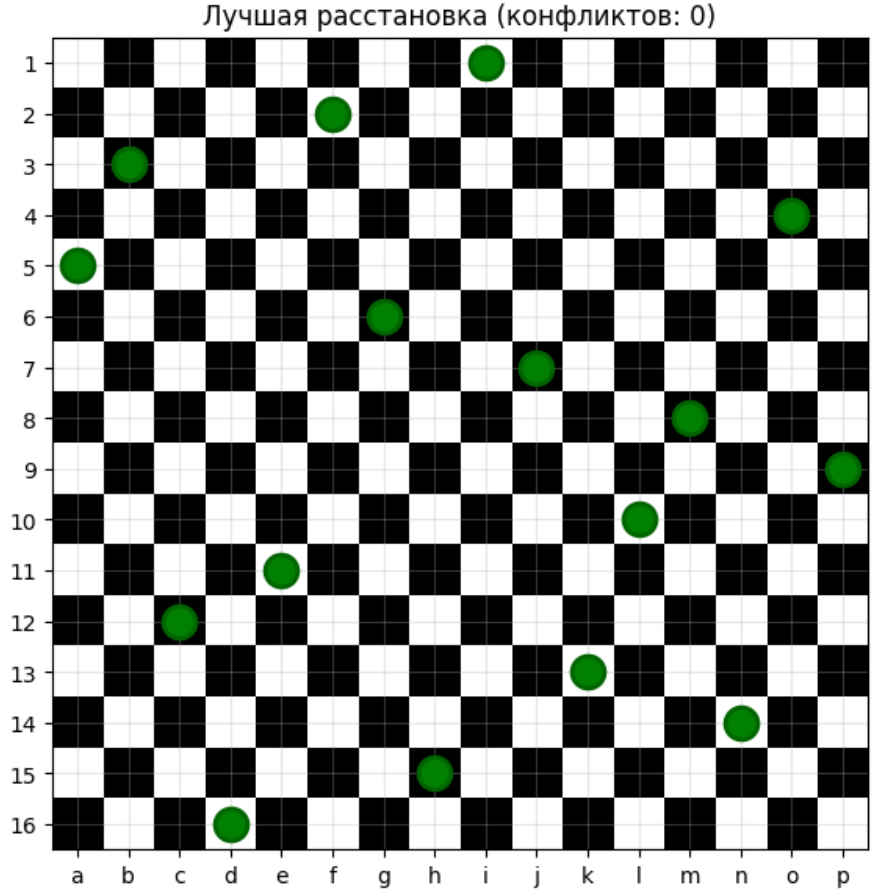


Анализ промежуточных состояний показывает, что число ферзей, которые могут атаковать друг друга, может вырасти из-за того, что мы иногда принимаем решения с большей энергией.

Задача 2

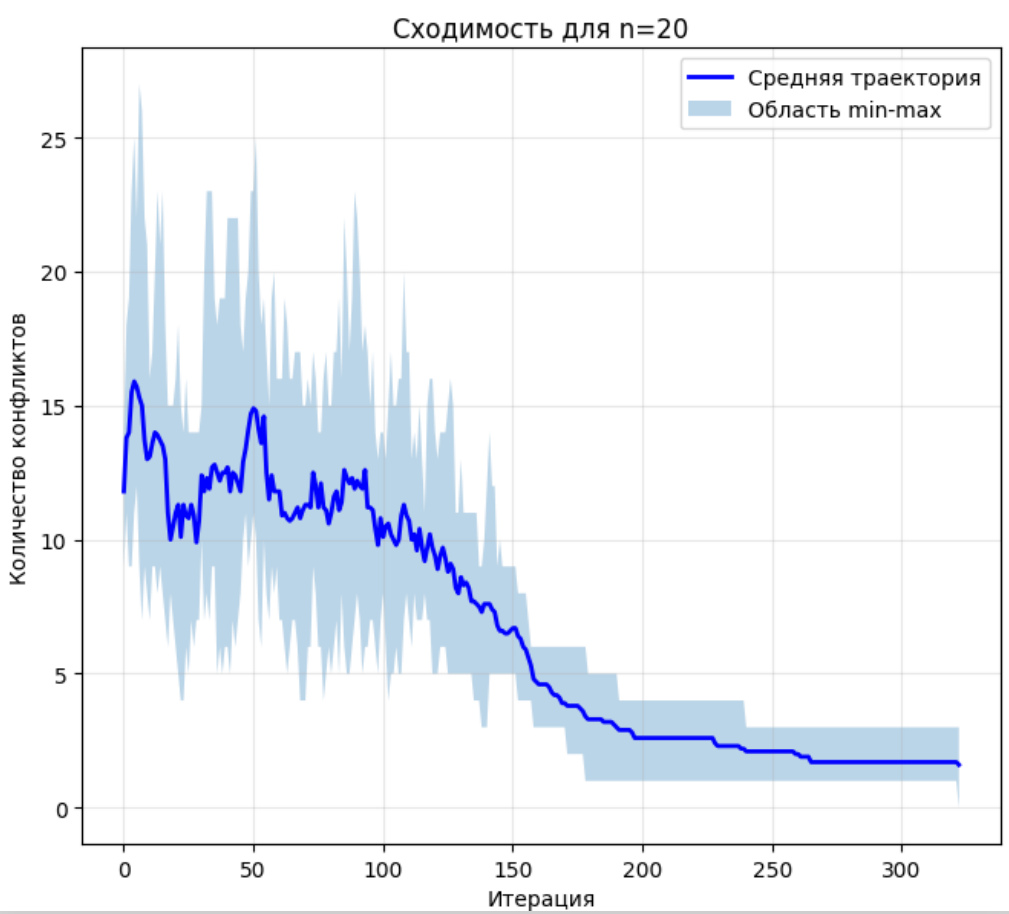
16-QUEENS PROBLEM

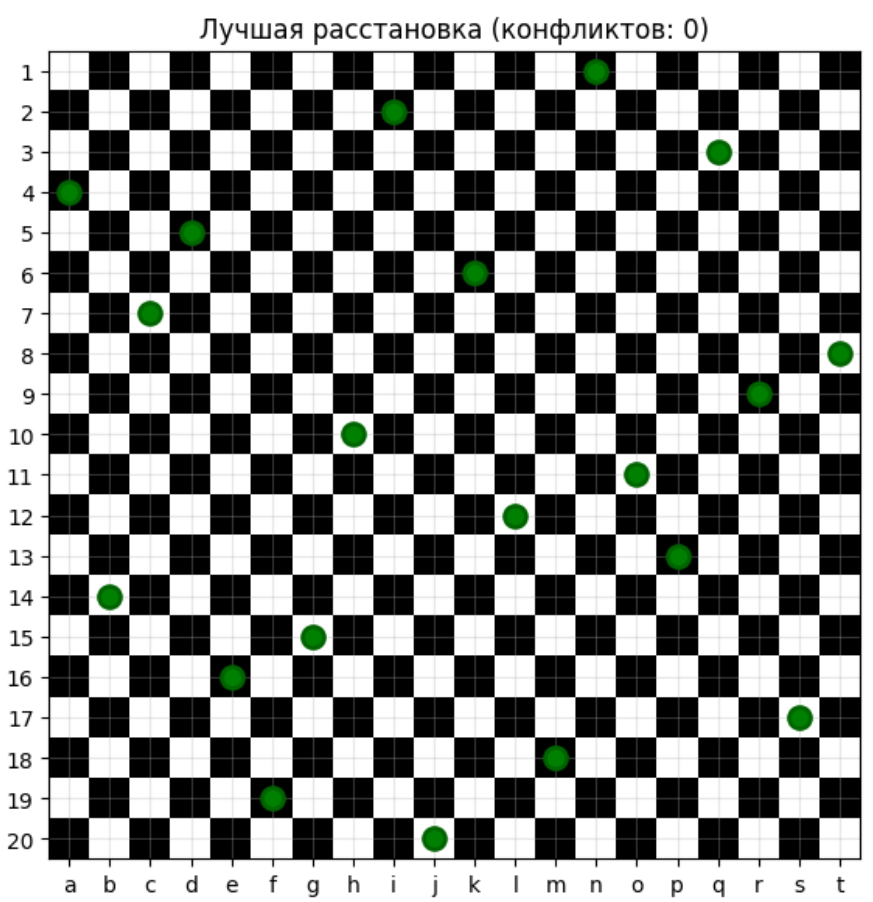




Задача 2

20-QUEENS PROBLEM





Задача 2




SIMULATED ANNEALING

Случайный метаэвристический алгоритм типа Монте-Карло для решения оптимизационных проблем с самой разной конфигурацией

- 👍 простота реализации и способность выхода из локальных решений
- 👍 гибкость настройки и масштабируемость [параллелизация]
- 👎 чувствительность к параметрам и отсутствие гарантий
- 👎 медленная или преждевременная сходимость

Резюме

1. Поиск k-ой порядковой статистики
   - `RANDOMIZED SELECT` — ожидаемая $O(n)$
   - `LINEAR SELECT` — гарантированная* $O(n)$
2. Метод имитации отжига SIMULATED ANNEALING
   - Поиск *как можно более близкого* к оптимальному решению задачи
   - `N-QUEENS PROBLEM` — безопасная расстановка ферзей на доске