Алгоритмы и структуры данных-1

Асимптотический анализ и рекуррентные соотношения. Часть 2

Практическое занятие 4 23-28.09.2024

2024-2025 учебный год

Разделяй-и-властвуй. Recap

Стратегия разделяй-и-властвуй (DaC)

Поиск ближайшей пары точек

CLOSEST PAIR

Вход: P – конечное множество n точек, заданных координатами в d-мерном пространстве.

Выход: пара точек p_1 и p_2 из множества P, расстояние между которыми минимально.

Наивный подход для d измерений

???

Наивный подход для d измерений

1. Рассчитать попарные расстояния между заданными точками (по формуле Евклида).
2. Найти минимум среди всех вычисленных расстояний.

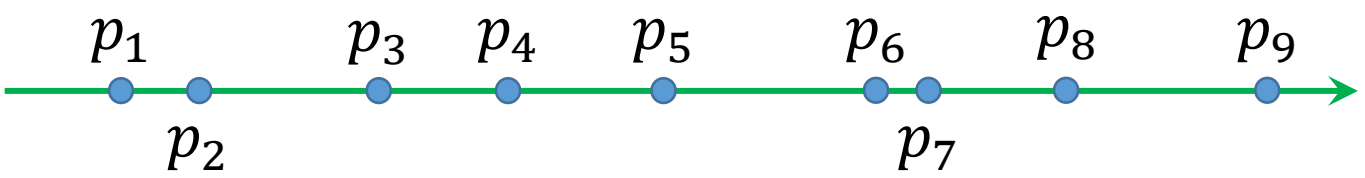
Сложность - ???

Наивный подход для d измерений

1. Рассчитать попарные расстояния между заданными точками (по формуле Евклида).
2. Найти минимум среди всех вычисленных расстояний.

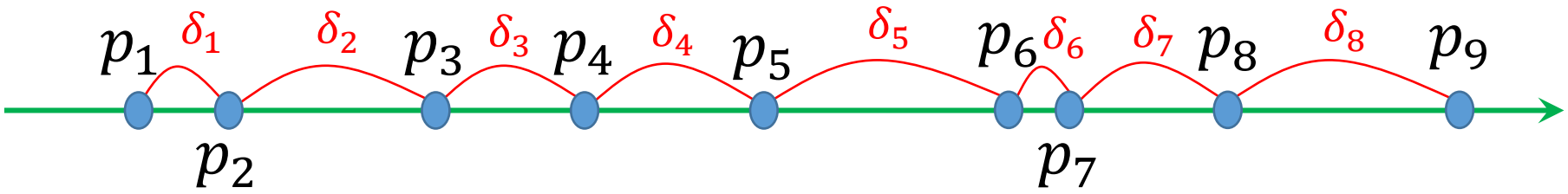
Сложность - $O(dn^2)$

Случай одного измерения – Прямая



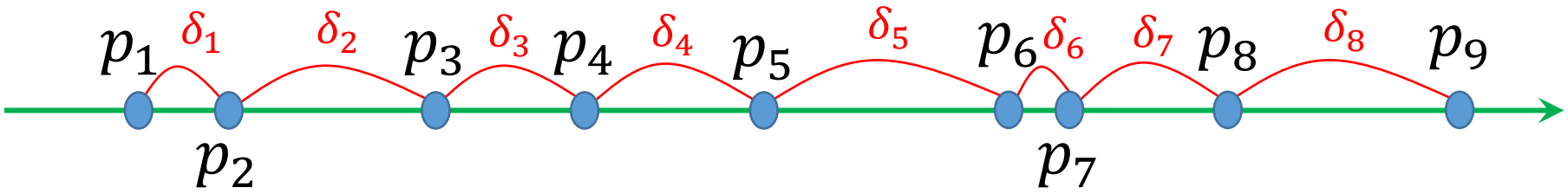
???

Случай одного измерения – Прямая



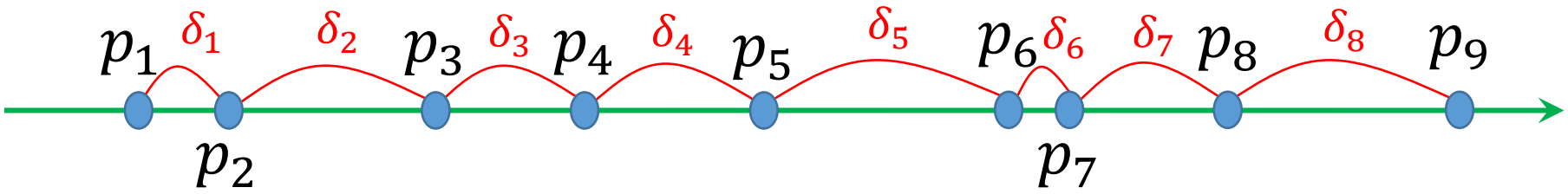
1. Отсортировать точки по координате
2. Вычислить расстояния между соседними точками
3. Найти минимальное расстояние

Случай одного измерения – Прямая



1. $O(?)$ Отсортировать множество точек по координате
2. $O(?)$ Вычислить расстояния между соседними точками
3. $O(?)$ Найти минимальное расстояние среди вычисленных

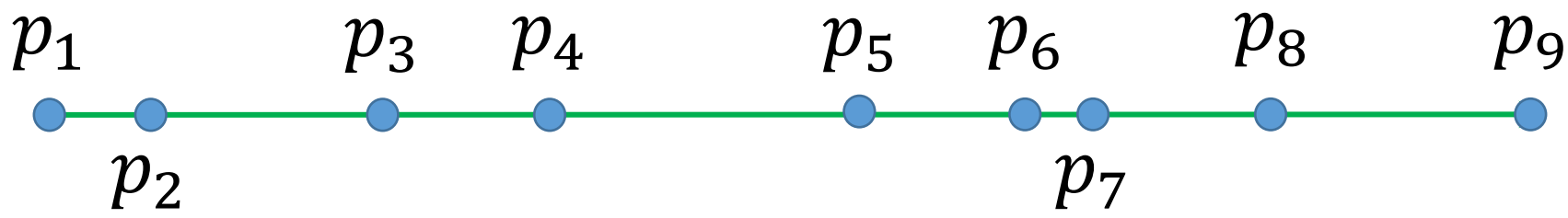
Случай одного измерения – Прямая



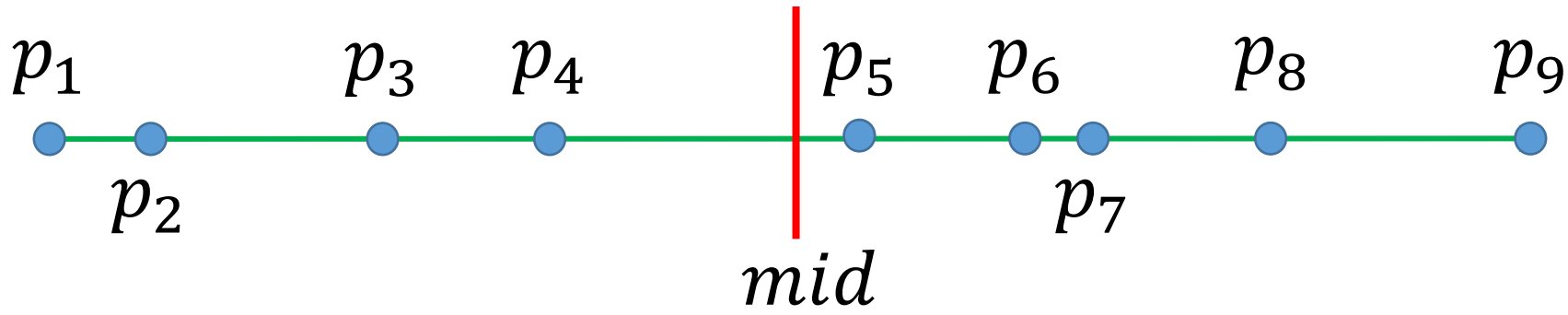
1. $\boldsymbol{O(n \log n)}$ Отсортировать множество точек по координате
2. $O(n)$ Вычислить расстояния между соседними точками
3. – Найти минимальное расстояние среди вычисленных

Поиск ближайшей пары точек на прямой при помощи DaC

Случай одного измерения – DaC

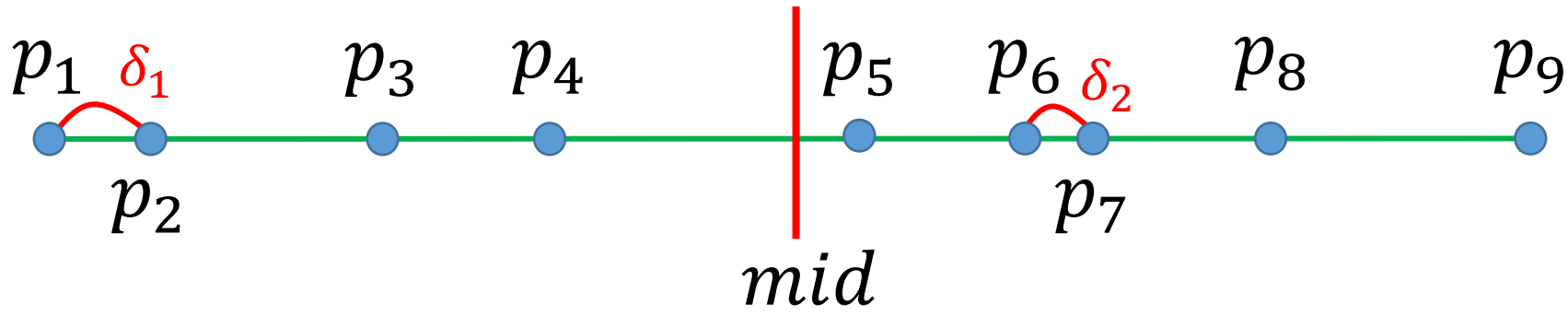


Случай одного измерения – DaC



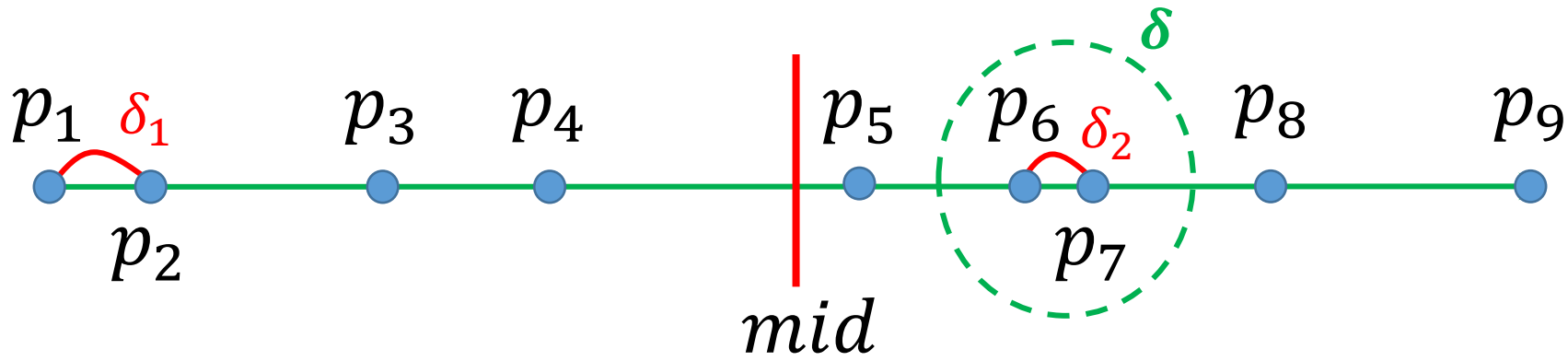
1. Разбить множество заданных точек по средней координате $mid = (p_{max} - p_{min})/2$

Случай одного измерения – DaC



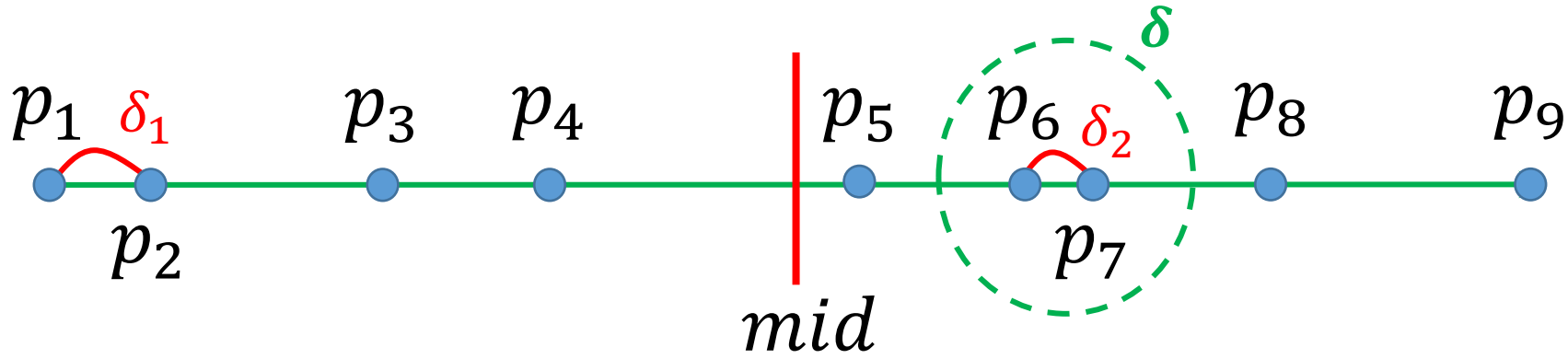
1. Разбить множество заданных точек по средней координате $mid = (p_{max} - p_{min})/2$
2. Выполнить рекурсивный поиск ближайших точек в каждой из двух образовавшихся половин

Случай одного измерения – DaC



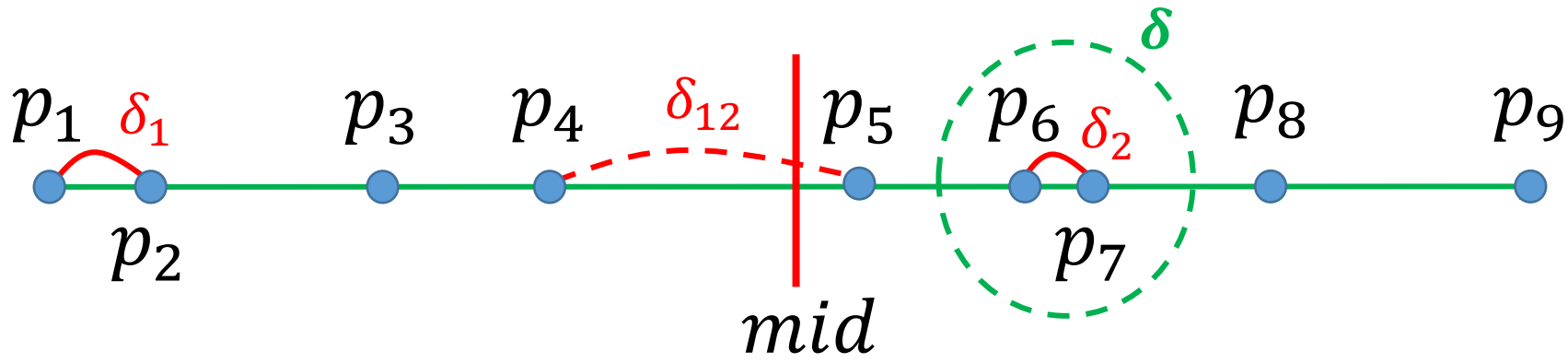
1. Разбить множество заданных точек по средней координате $mid = (p_{max} - p_{min})/2$
2. Выполнить рекурсивный поиск пар ближайших точек в каждой из двух образовавшихся половин
3. $\delta = \min(\delta_1, \delta_2)$

Случай одного измерения – DaC



1. Разбить множество заданных точек по средней координате $mid = (p_{max} - p_{min})/2$
2. Выполнить рекурсивный поиск пар ближайших точек в каждой из двух образовавшихся половин
3. $\delta = \min(\delta_1, \delta_2)$
4. **Что дальше?**

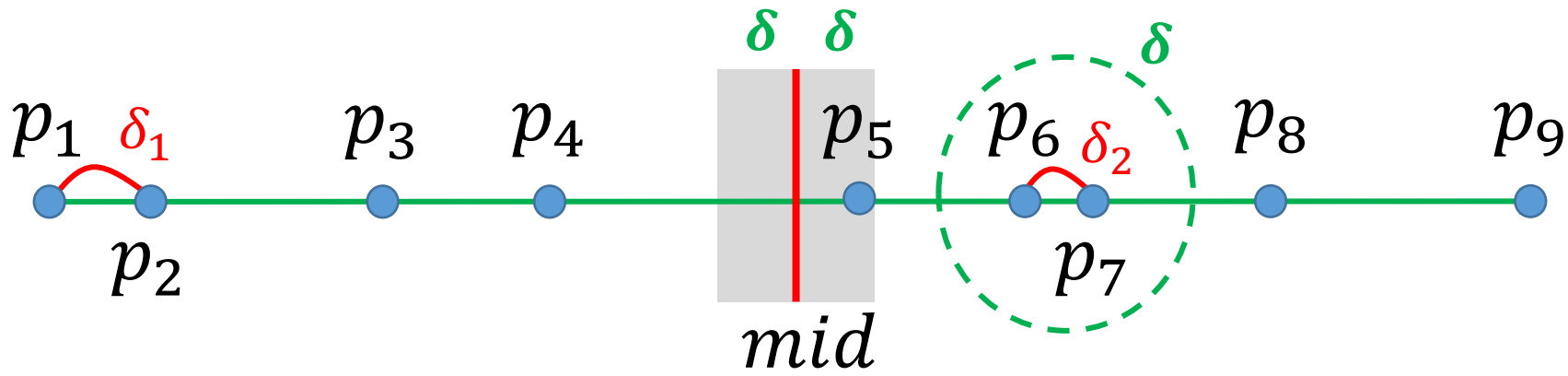
Случай одного измерения – DaC



Осталось проверить точки по разные стороны от mid (δ_{12})

$$\mathbf{answer} = \mathbf{min}(\boldsymbol{\delta}, \boldsymbol{\delta_{12}})$$

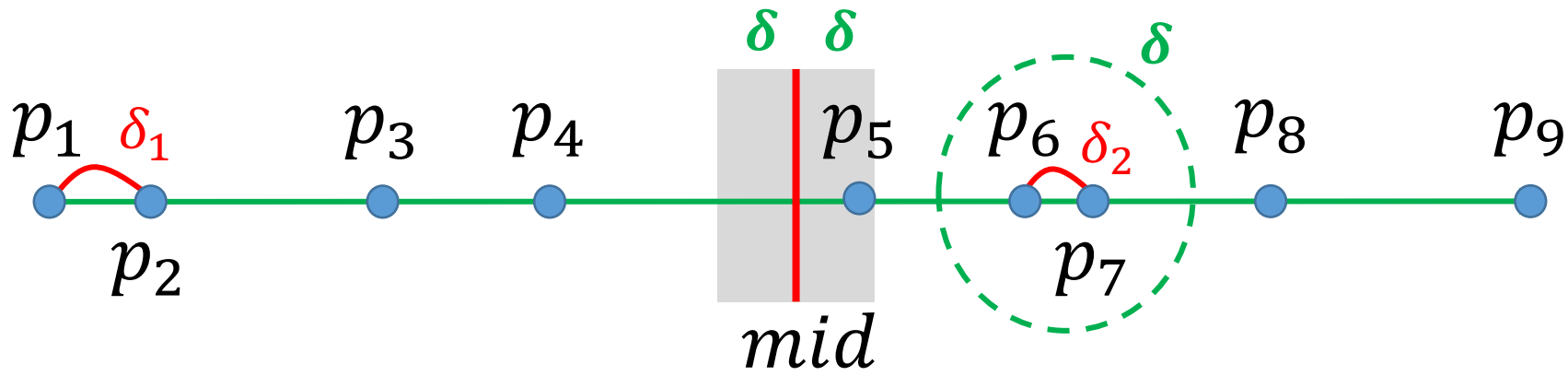
Случай одного измерения – DaC



Нас интересуют только точки на расстоянии δ по обе стороны от mid

Сколько пар точек попадает в эту область?

Случай одного измерения – DaC



Нас интересуют только точки на расстоянии δ по обе стороны от mid

Сколько пар точек попадает в эту область?

только одна пара точек

CLOSEST PAIR 1D – DaC

	closestPair(*P*)	$T(n)$
1	**if** $\|P\| = 1$ **return** ∞	…
2	**if** $\|P\| = 2$ **return** $\|p_1 - p_2\|$	
3	m – медианная координата в P	?
4	разбить P на P_1 и P_2 относительно m	
5	δ_1 = **closestPair**(P_1)	?
6	δ_2 = **closestPair**(P_2)	
7	δ_{12} - расстояние между точками по разные стороны от m	…
8	**return** $\mathbf{min}(\delta_1, \delta_2, \delta_{12})$	

CLOSEST PAIR 1D – DaC

	closestPair(*P*)	$T(n)$
1	**if** $\|P\| = 1$ **return** ∞	…
2	**if** $\|P\| = 2$ **return** $\|p_1 - p_2\|$	
3	m – медианная координата в P	$O(n)$
4	разбить P на P_1 и P_2 относительно m	
5	δ_1 = **closestPair**(P_1)	$2T(n/2)$
6	δ_2 = **closestPair**(P_2)	
7	δ_{12} - расстояние между точками по разные стороны от m	…
8	**return min**$(\delta_1, \delta_2, \delta_{12})$	

CLOSEST PAIR 1D – DaC

Рекуррентное соотношение DaC:

$$T(n) = 2 \cdot T\left({}^{n}/_{2}\right) + O(n)$$

Что здесь не учтено?

CLOSEST PAIR 1D – DaC

Рекуррентное соотношение DaC:

$$T(n) = 2 \cdot T\left(n/2\right) + O(n)$$

Нужно **отсортировать** точки

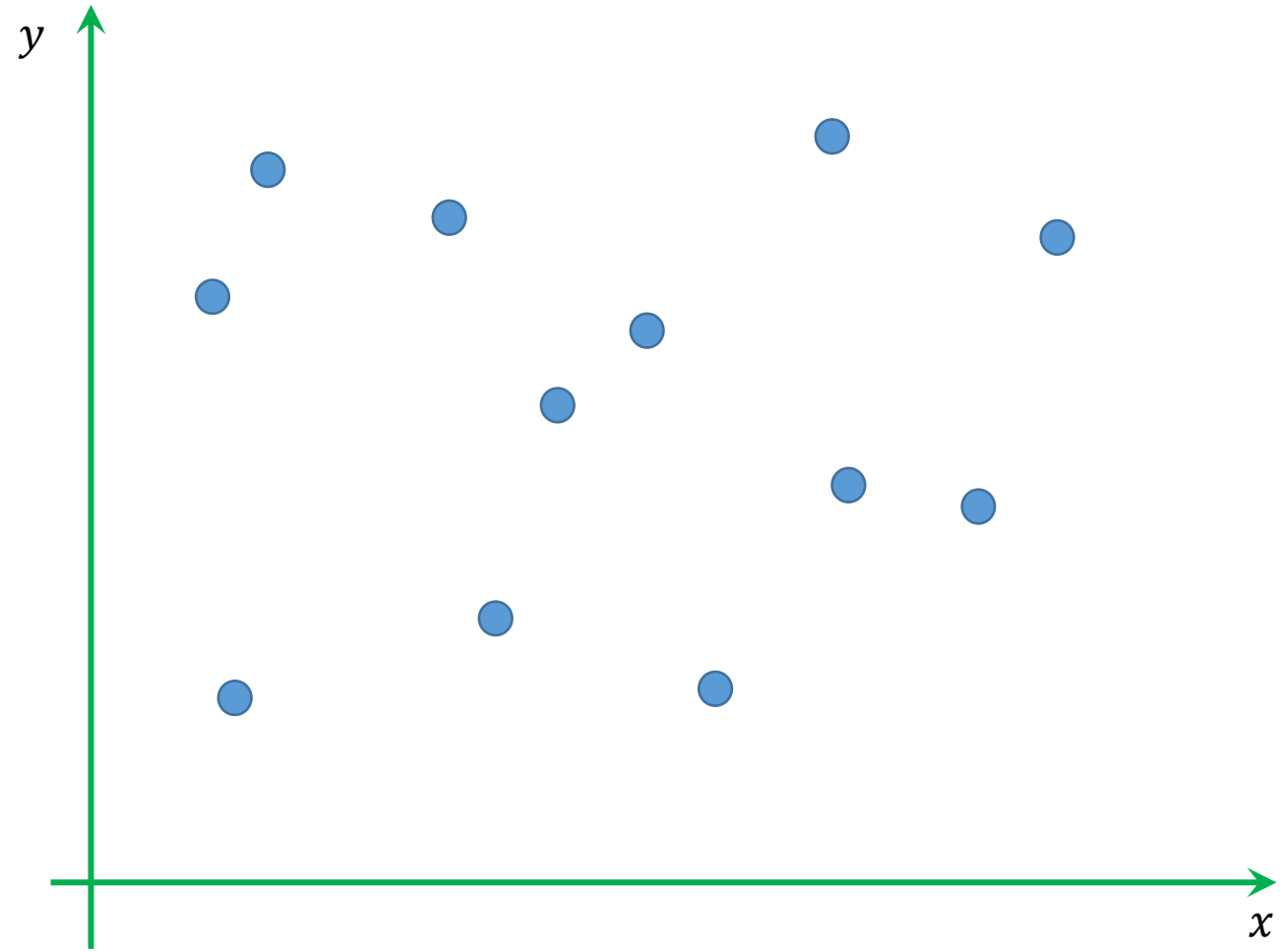
Общая сложность CLOSEST PAIR 1D:

$$\boldsymbol{T_{total} = T(n) + O(n \log n)}$$

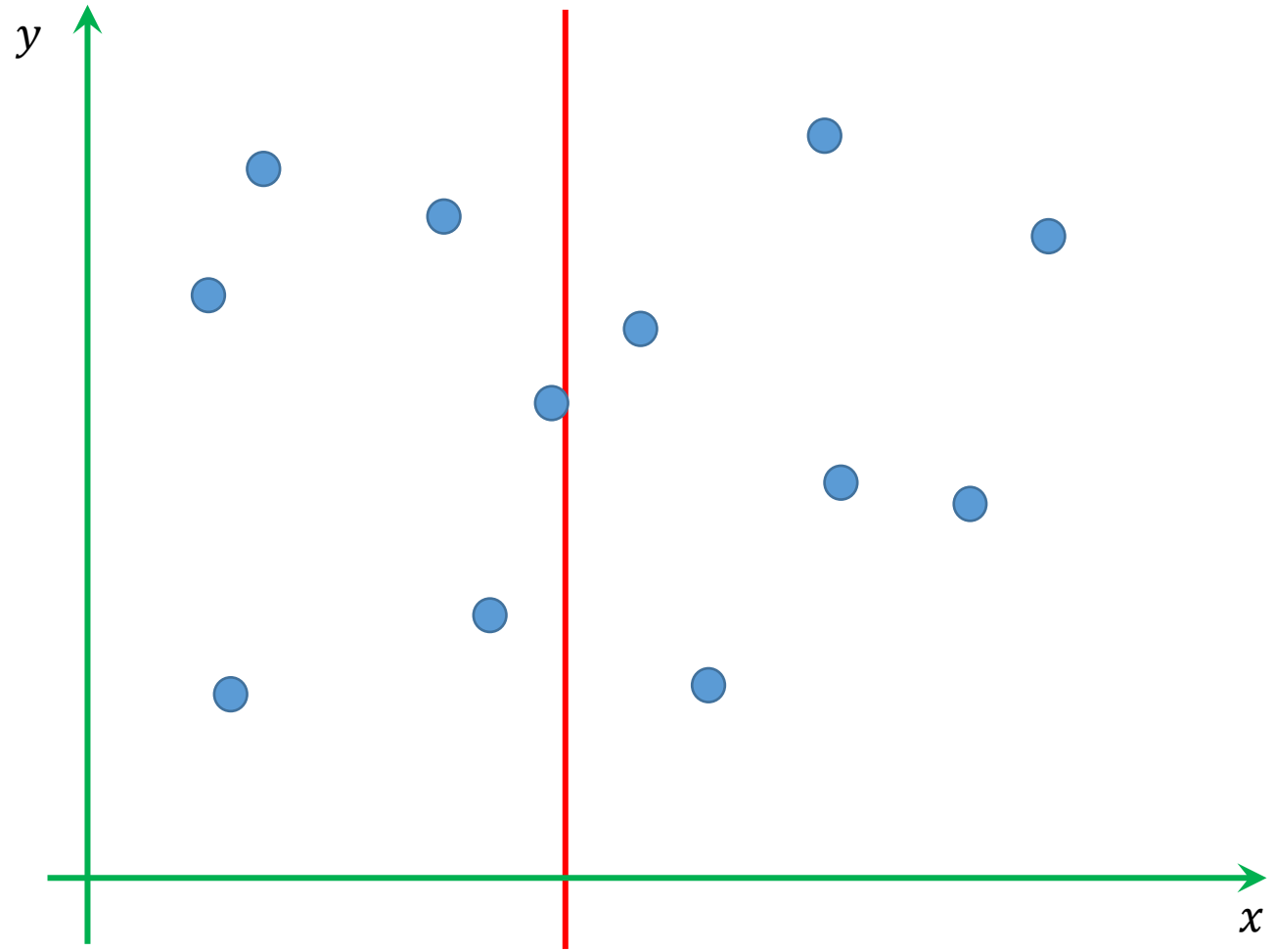
Выведем верхнюю границу...

Поиск ближайшей пары точек на плоскости при помощи DaC

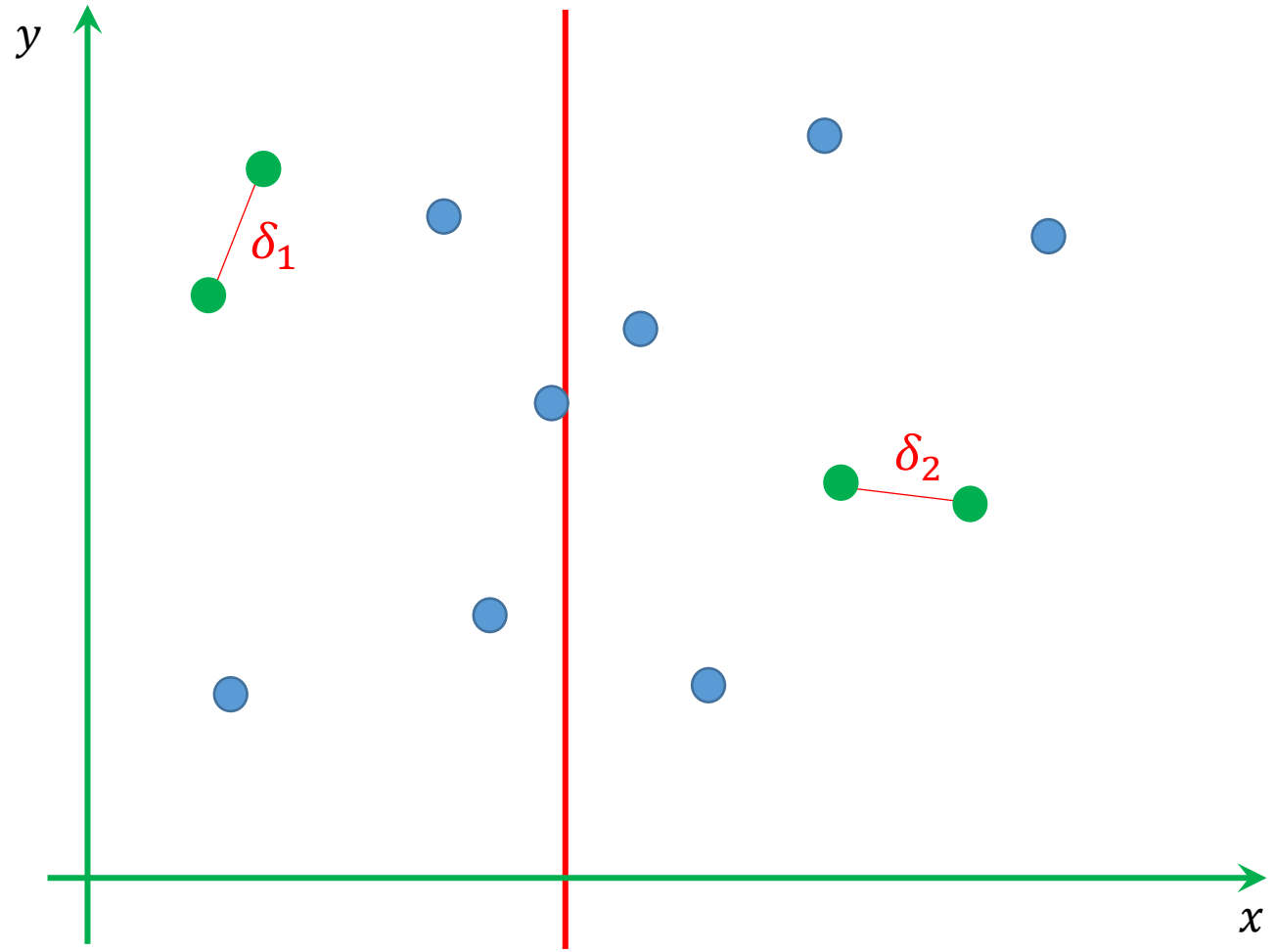
CLOSEST PAIR 2D – DaC



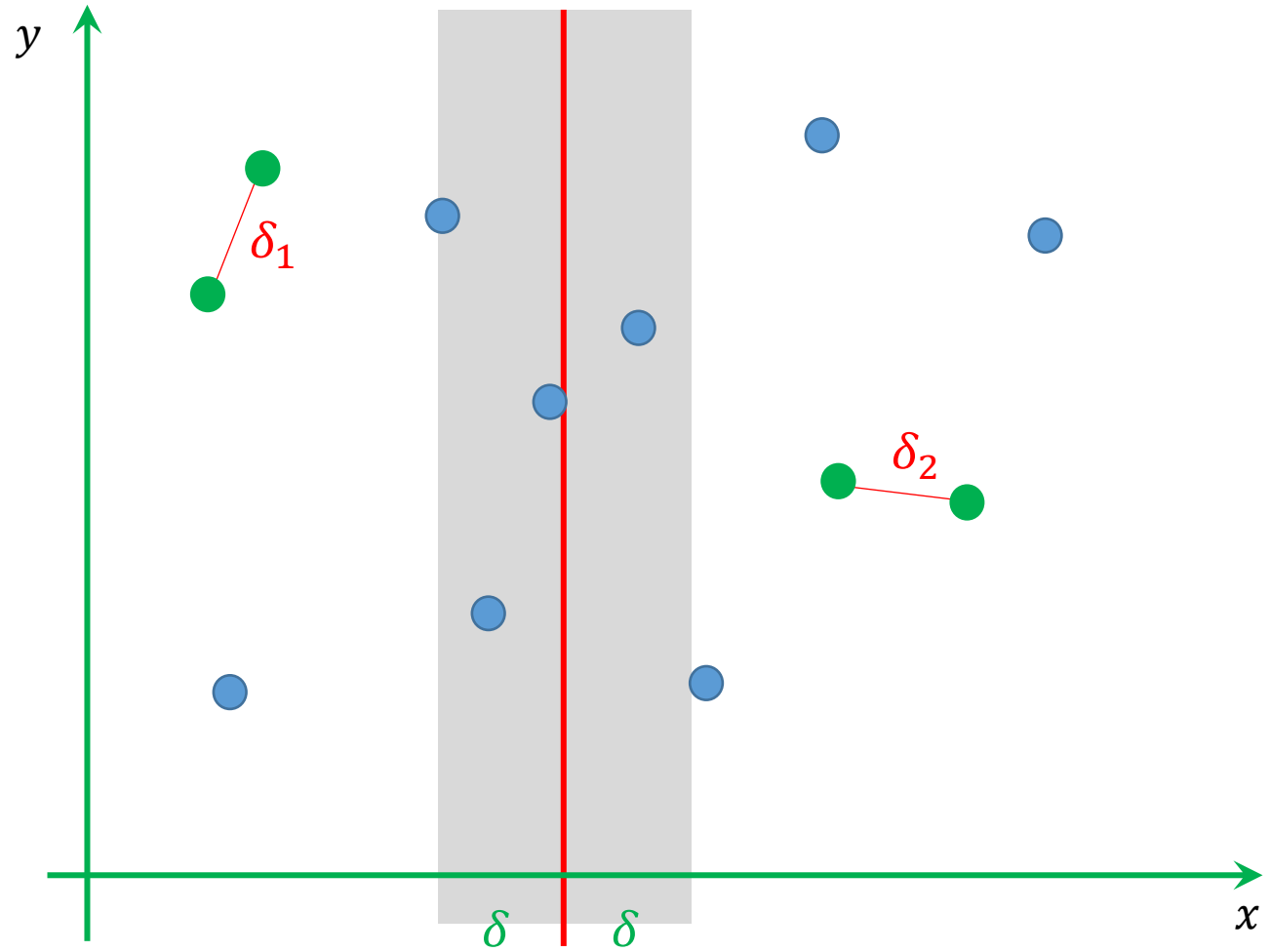
CLOSEST PAIR 2D – DaC



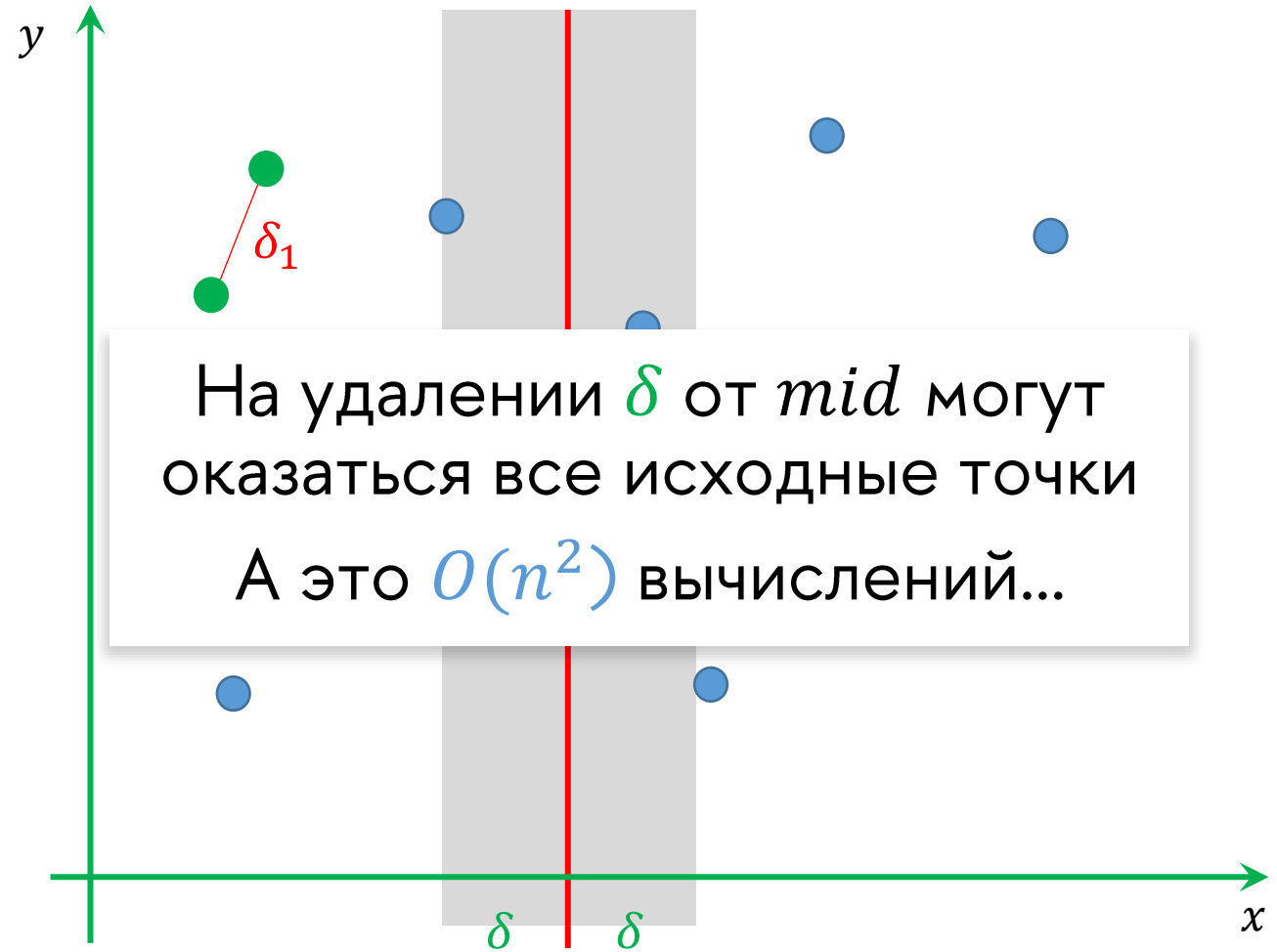
CLOSEST PAIR 2D – DaC



CLOSEST PAIR 2D – DaC



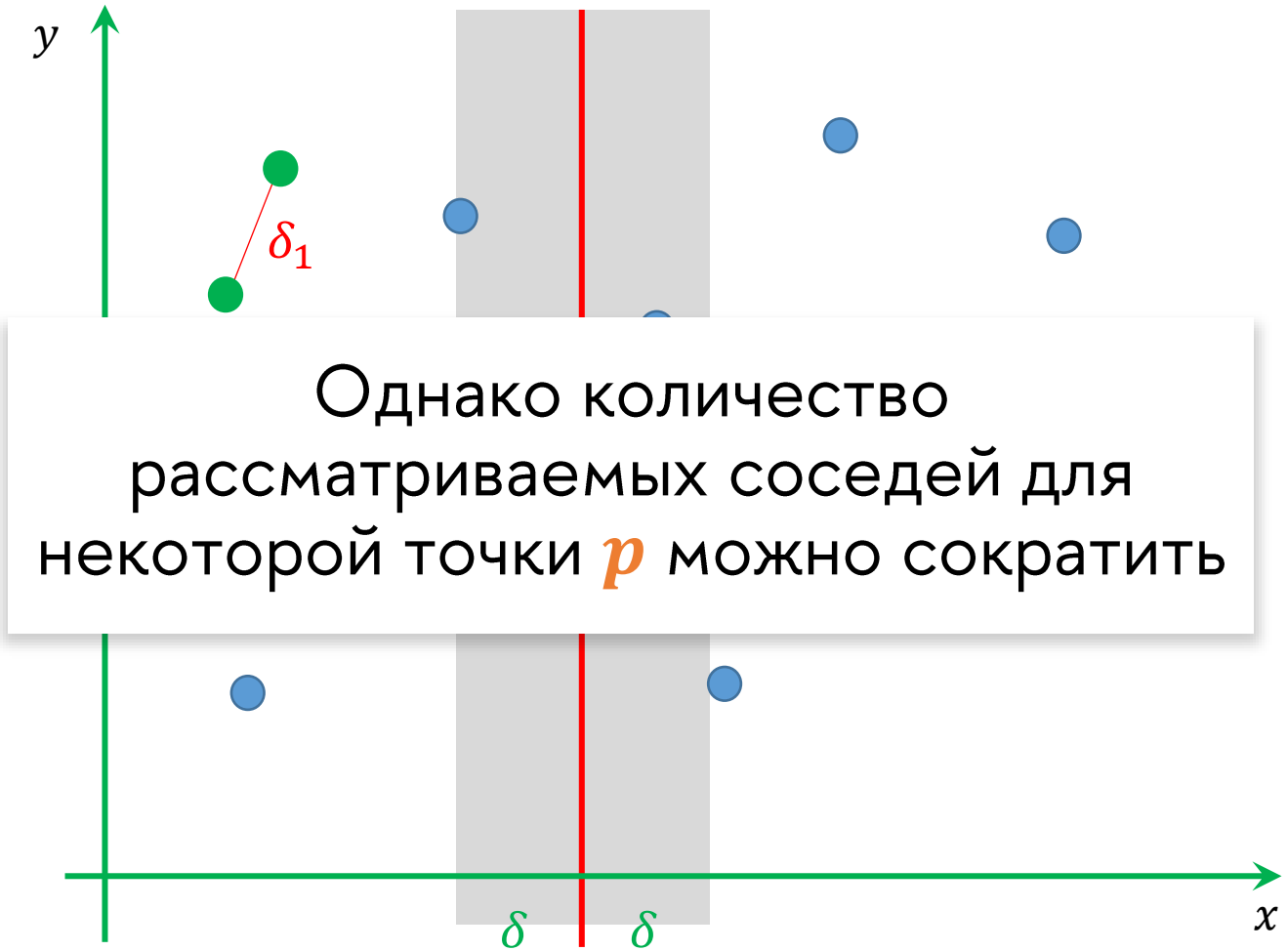
CLOSEST PAIR 2D – DaC



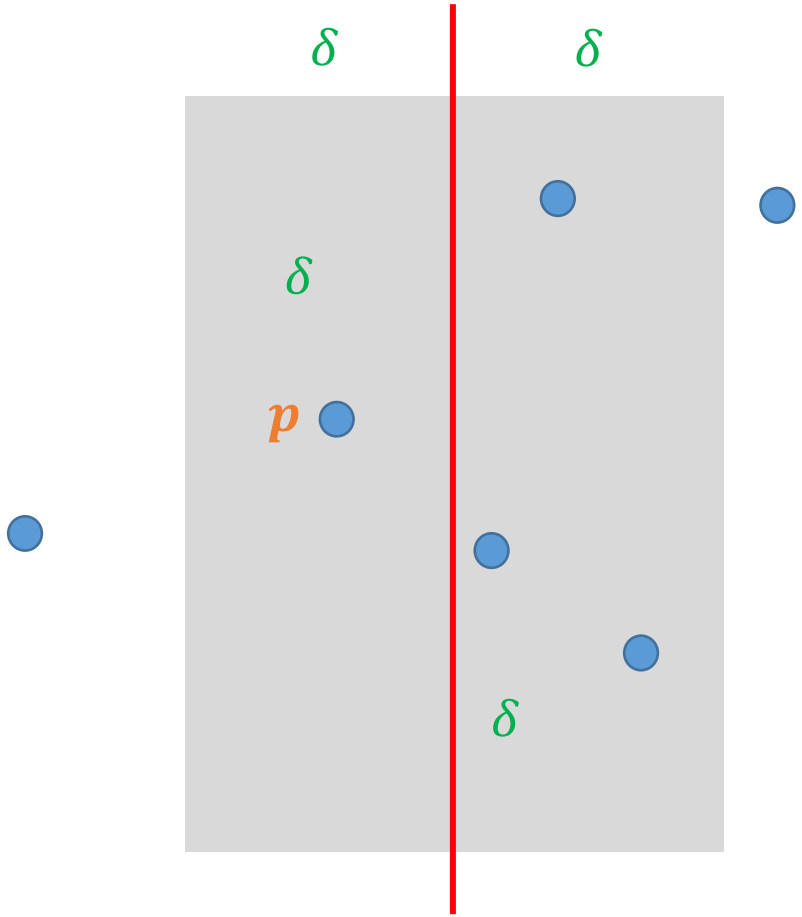
На удалении δ от mid могут оказаться все исходные точки

А это $O(n^2)$ вычислений...

CLOSEST PAIR 2D – DaC

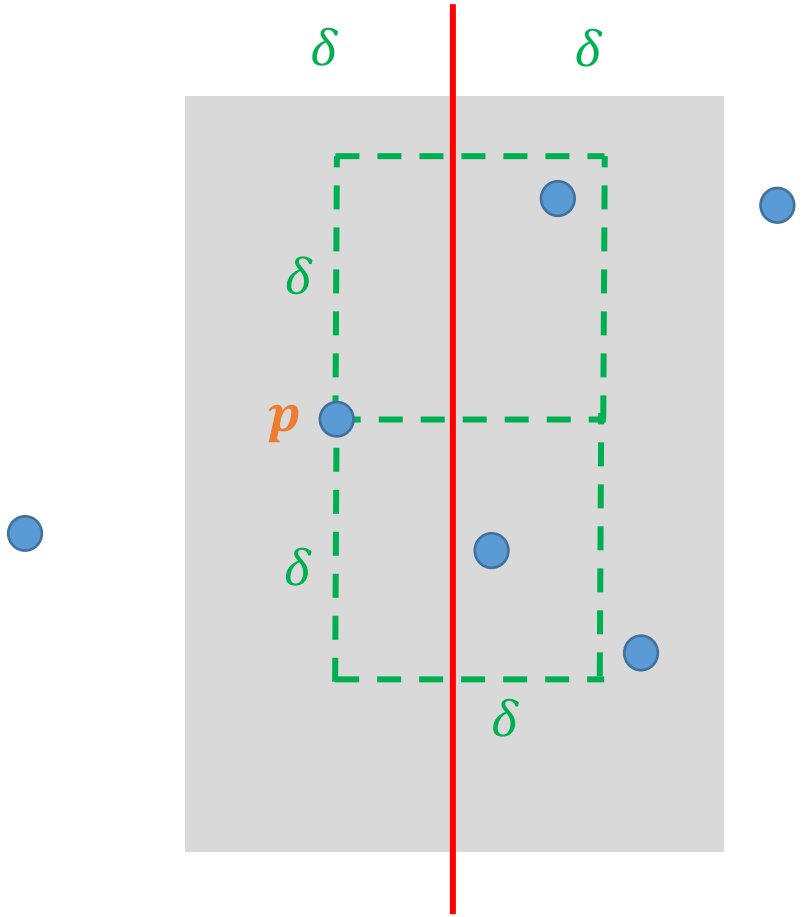


CLOSEST PAIR 2D – DaC



Вокруг p достаточно рассмотреть только соседей на расстоянии $\leq \delta$

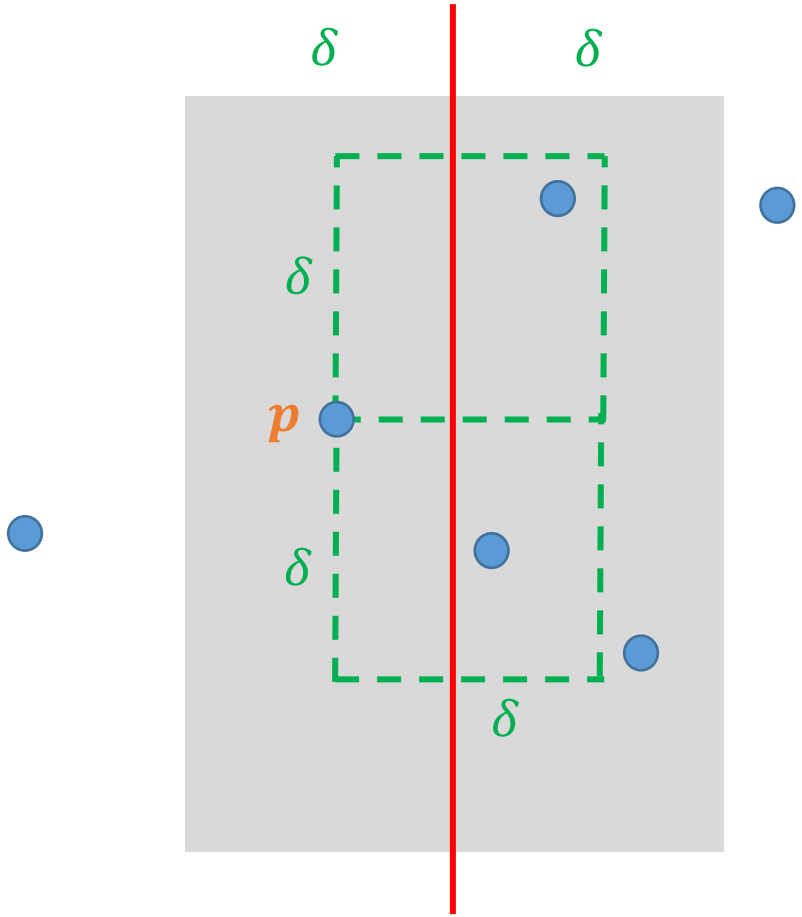
CLOSEST PAIR 2D – DaC



Вокруг p достаточно рассмотреть только соседей на расстоянии $\leq \delta$

Эти соседи точно в пределах прямоугольника $\delta \times 2\delta$

CLOSEST PAIR 2D – DaC

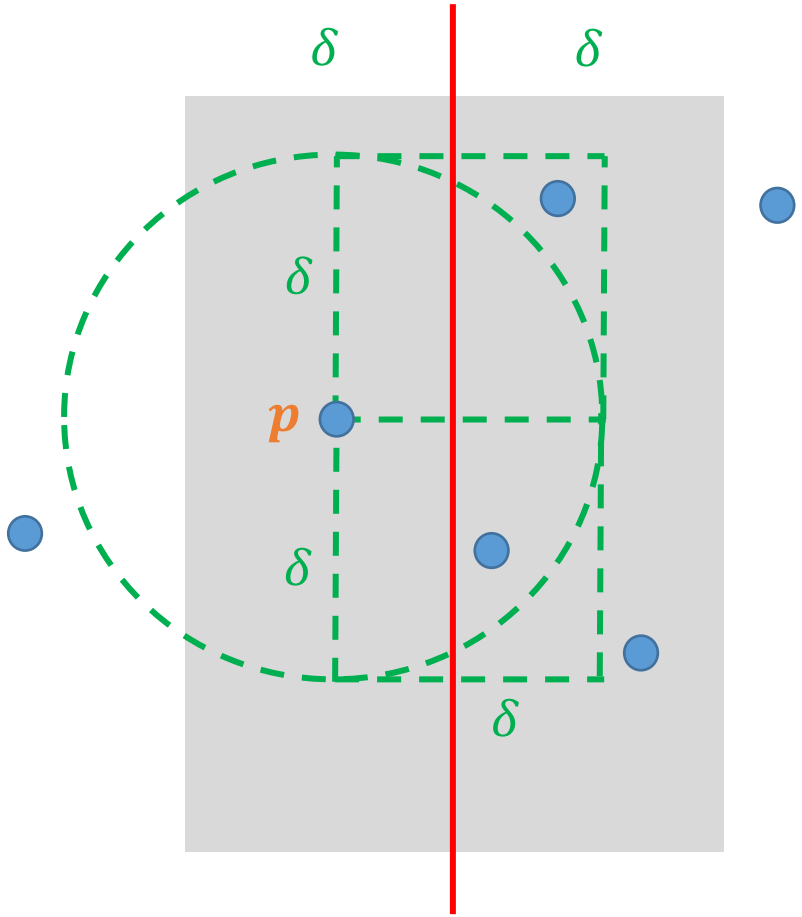


Вокруг p достаточно рассмотреть только соседей на расстоянии $\leq \delta$

Эти соседи точно в пределах прямоугольника $\delta \times 2\delta$

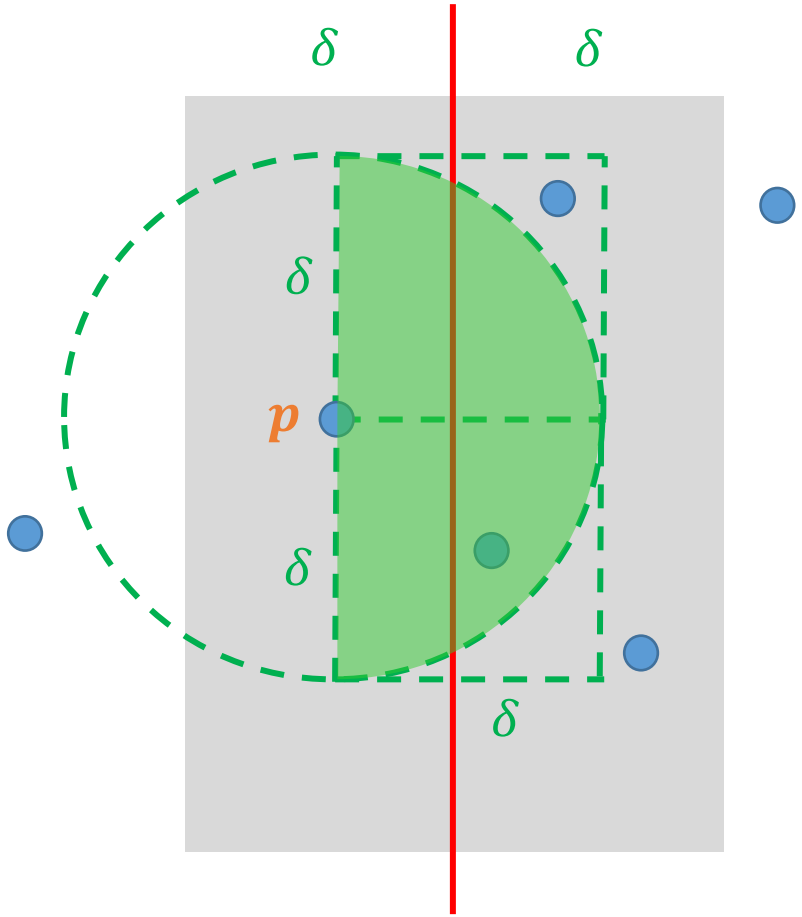
Сколько точек попадет внутрь одного квадрата $\delta \times \delta$?

CLOSEST PAIR 2D – DaC



Соседи точно в пределах окружности с радиусом δ

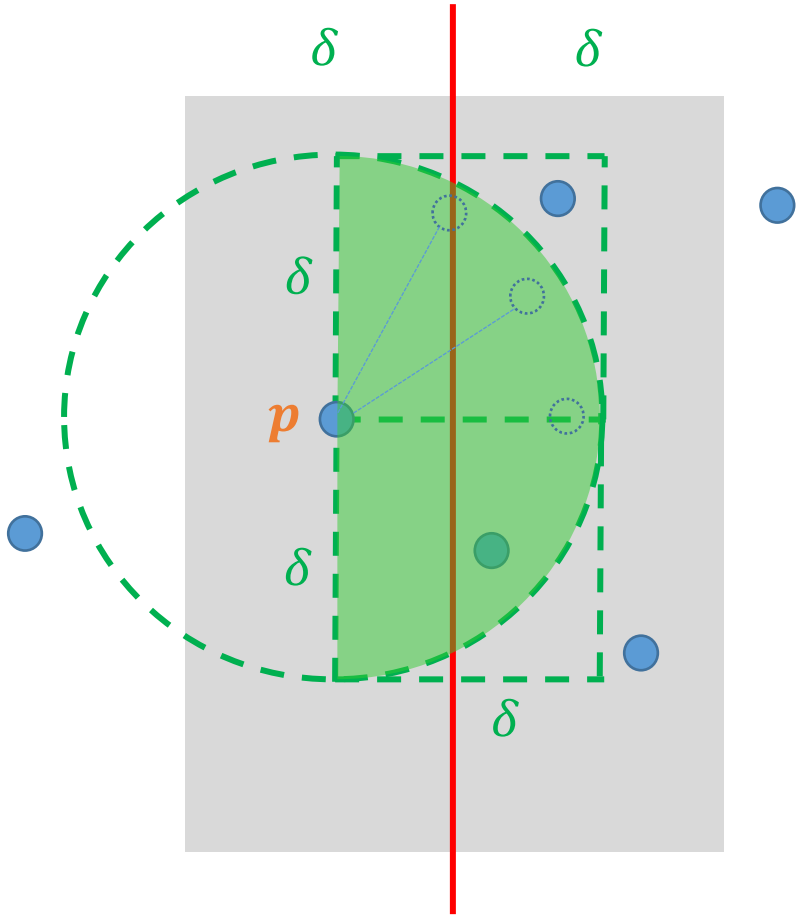
CLOSEST PAIR 2D – DaC



Соседи точно в пределах половины окружности с радиусом δ

Максимальное число соседей = ?

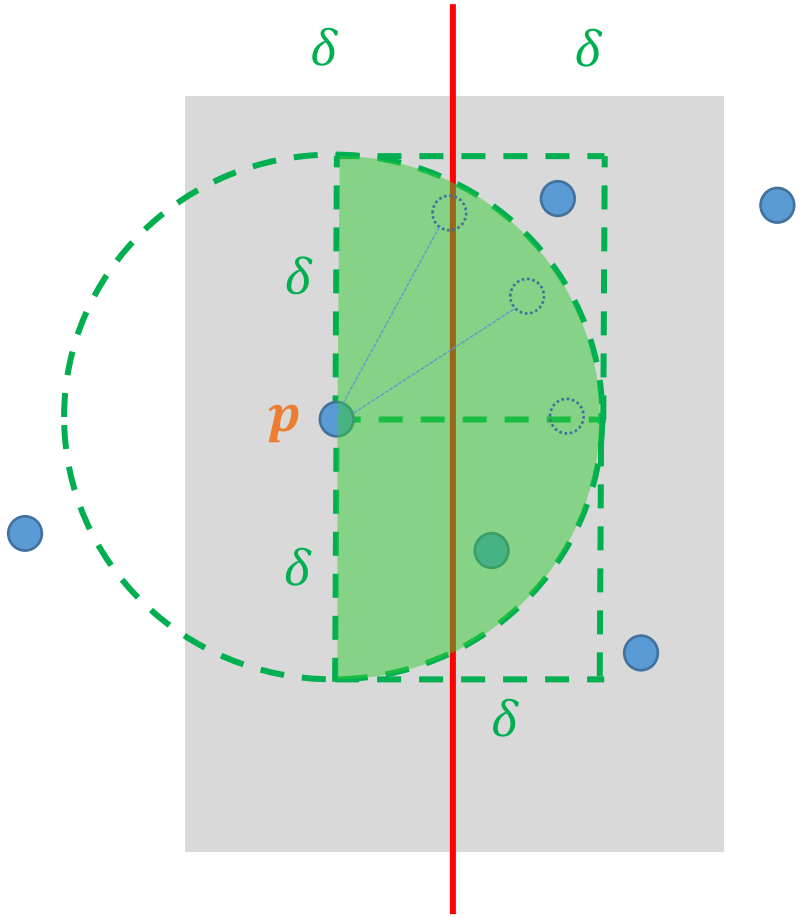
CLOSEST PAIR 2D – DaC



Соседи точно в пределах половины окружности с радиусом δ

Максимальное число соседей = 6

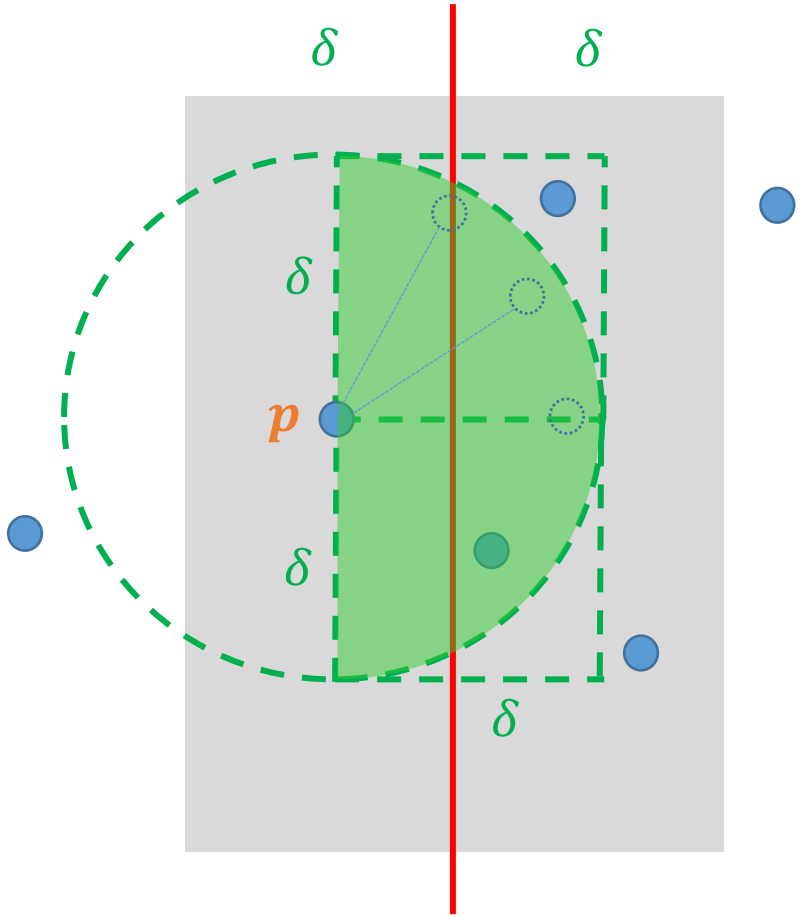
CLOSEST PAIR 2D – DaC



Максимальное число соседей = 6

А значит для точки p потребуется $\leq 6n$ вычислений

CLOSEST PAIR 2D – DaC



Максимальное число соседей = 6

А значит для точки p потребуется $\leq 6n$ вычислений

CLOSEST PAIR 2D – DaC

1. Разбить входное множество P точек на P_1 и P_2 по медианной x-координате ℓ.
2. Применить **`closestPair2D`** для P_1 и P_2.
   Найти $\delta = \min(\delta_1, \delta_2)$.
3. Исключить из P_1 и P_2 точки, удаленные от ℓ более, чем на δ.
   Получить два множества точек P_1' и P_2'.
4. Отсортировать P_1' и P_2' по y-координате.
5. Для каждой точки в P_1' вычислить расстояния до ее шести соседей, попадающих в прямоугольник $\delta \times 2\delta$. Если среди них есть меньшие δ, то обновить текущий минимум.

CLOSEST PAIR 2D – DaC

1. Разбить входное множество P точек на P_1 и P_2 по медианной x-координате ℓ.
2. При
   Най
3. Иск
   Пол
4. Отсортировать P_1 и P_2 по y-координате.
5. Для каждой точки в P_1' вычислить расстояния до ее шести соседей, попадающих в прямоугольник $\delta \times 2\delta$. Если среди них есть меньшие δ, то обновить текущий минимум.

Запишем рекуррентное выражение для `closestPair2D` и оценим его сложность

CLOSEST PAIR 2D – DaC

1. Разбить входное множество P точек на P_1 и P_2 по медианной x-координате ℓ.
2. Применить **`closestPair2D`** для P_1 и P_2.
   Найти $\delta = \min(\delta_1, \delta_2)$.
3. Исключить из P_1 и P_2 точки, удаленные от ℓ более, чем на δ.
   Получить два множества точек P_1' и P_2'.
4. **Отсортировать P_1' и P_2' по y-координате.**
5. Для каждой точки в P_1' вычислить расстояния до ее шести соседей, попадающих в прямоугольник $\delta \times 2\delta$. Если среди них есть меньшие δ, то обновить текущий минимум.

Применение CLOSEST PAIR

Построение остовных деревьев графовых структур

Проектирование крыш (!)

Иерархическая кластеризация объектов

Эвристики для задачи коммивояжёра

...

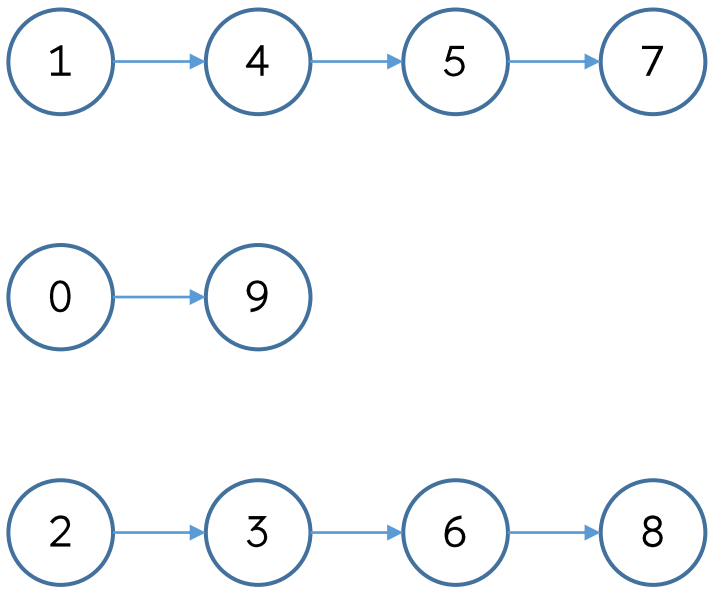
Слияние k отсортированных списков K-WAY MERGE

K-WAY MERGE

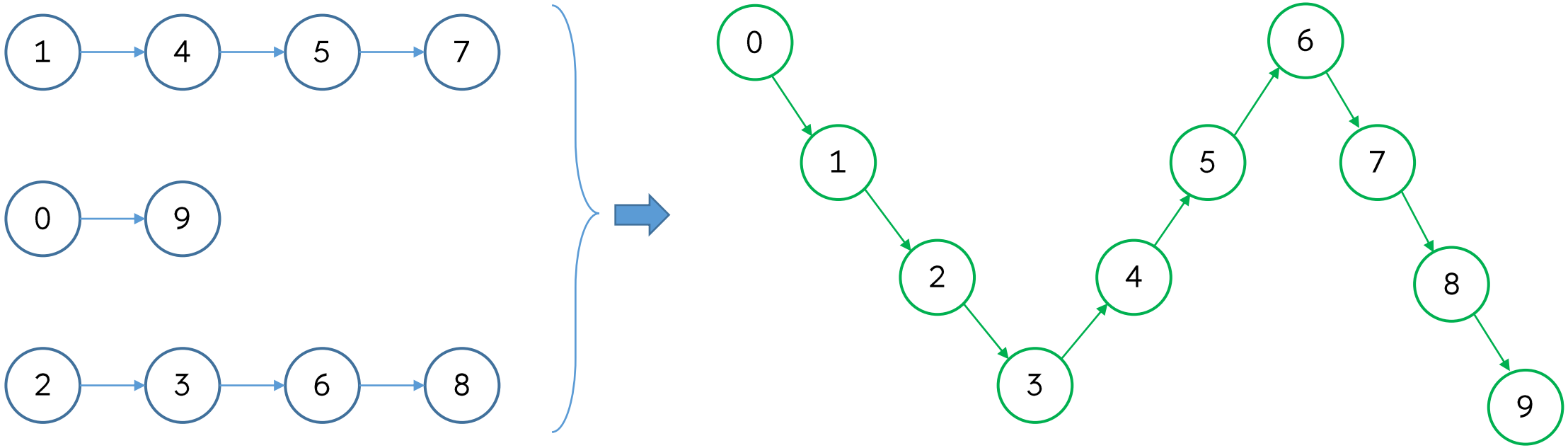
Вход: k связных списков, значения в узлах которых отсортированы по некоторому критерию.

Выход: связный список L, полученный в результате слияния входных списков, значения в узлах которого отсортированы по тому же критерию.

K-WAY MERGE



K-WAY MERGE



K-WAY MERGE грубой силой

1. Обойти исходные списки и записать значения, содержащиеся в узлах списка, в массив A

K-WAY MERGE грубой силой

1. Обойти исходные списки и записать значения, содержащиеся в узлах списка, в массив A
2. Отсортировать массив A

K-WAY MERGE грубой силой

1. Обойти исходные списки и записать значения, содержащиеся в узлах списка, в массив A
2. Отсортировать массив A
3. Создать выходной список L и переписать в него значения из массива A

K-WAY MERGE грубой силой

1. Обойти все исходные списки и записать значения, содержащиеся в узлах списка, в массив A
2. **Отсортировать массив A**
3. Создать выходной список L и переписать в него значения из массива A

Сложность сортировки доминирует над любыми другими шагами данного варианта алгоритма

K-WAY MERGE грубой силой

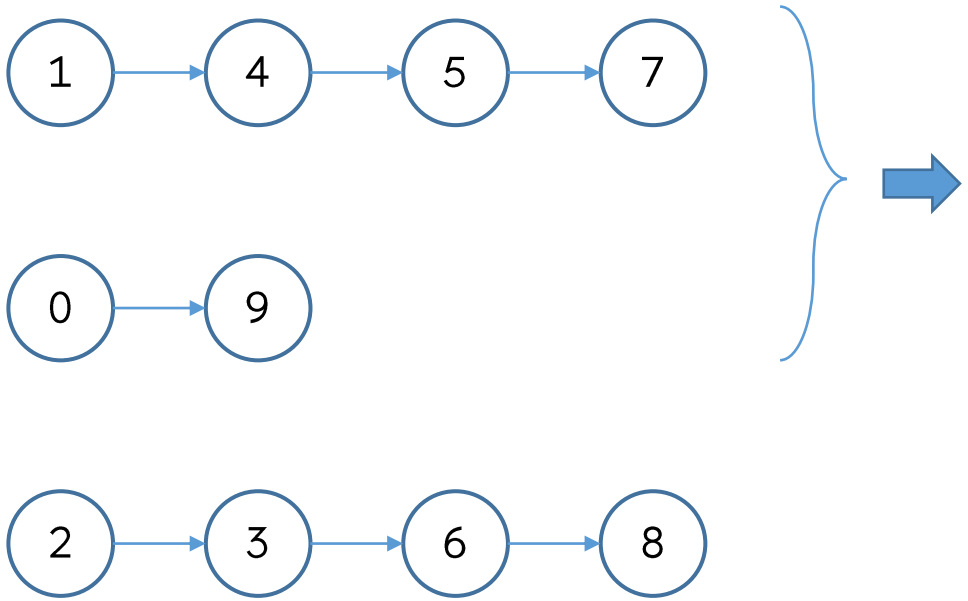
1. Обойти все исходные списки и записать значения,

С `mergeSort` получим $O(N \log N)$,

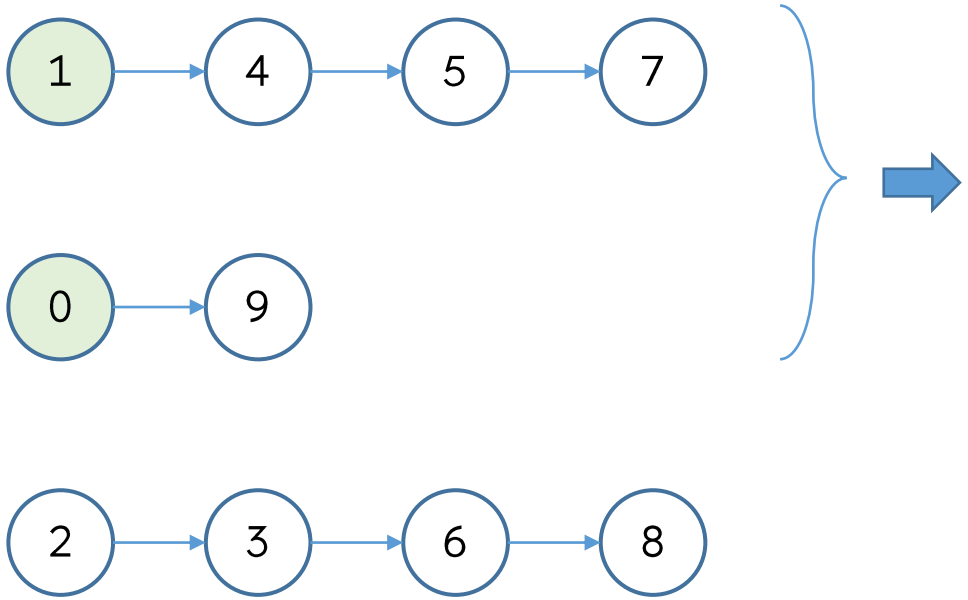
где N – общее количество значений в k списках

значения из массива A

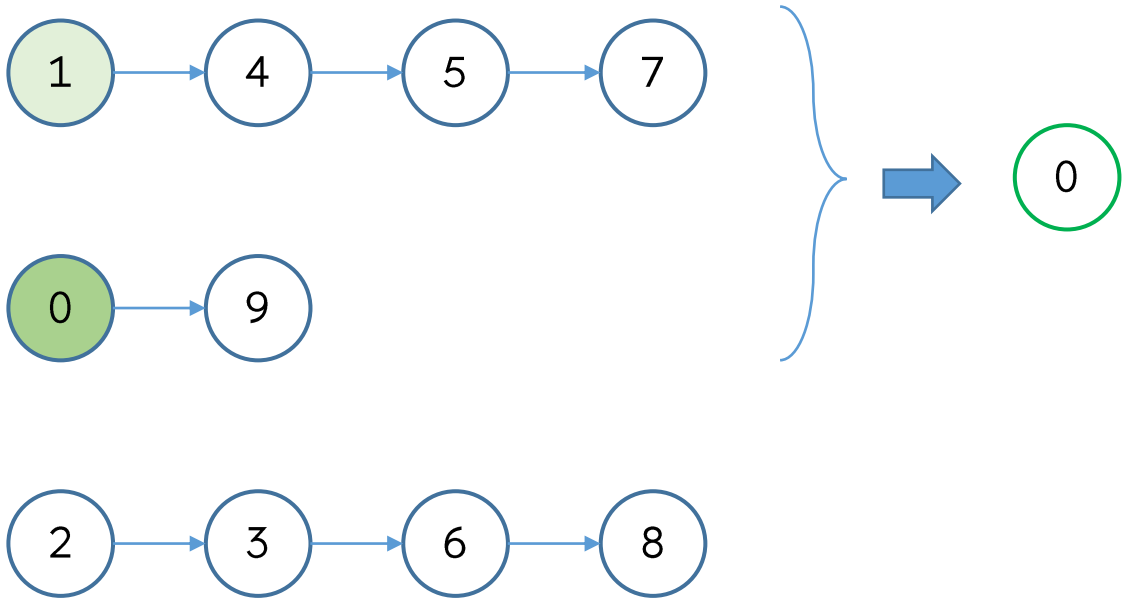
K-WAY MERGE друг за другом



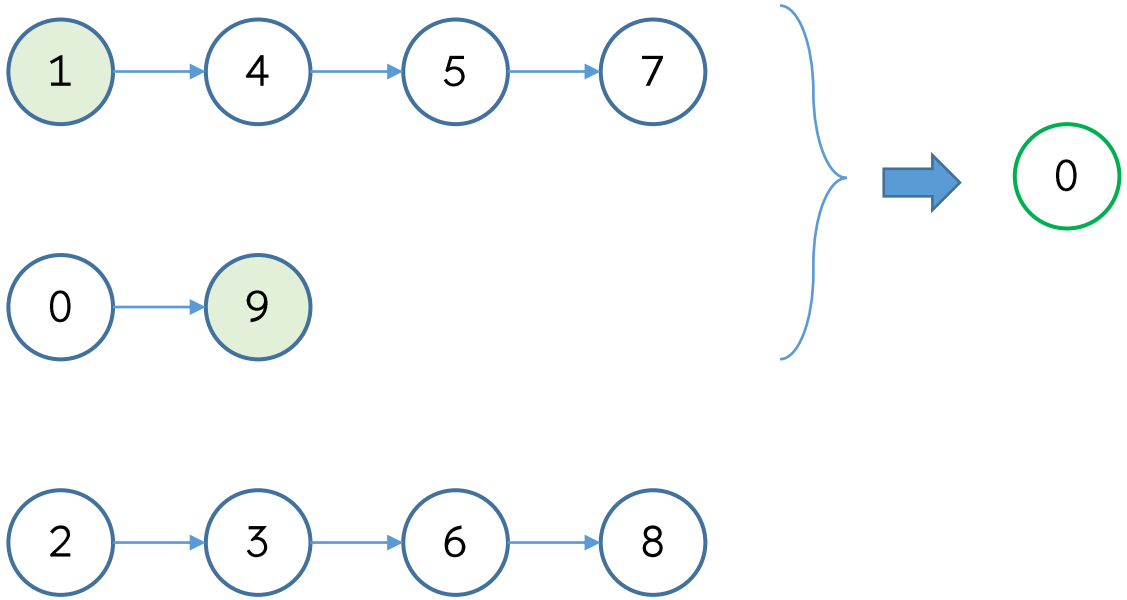
K-WAY MERGE друг за другом



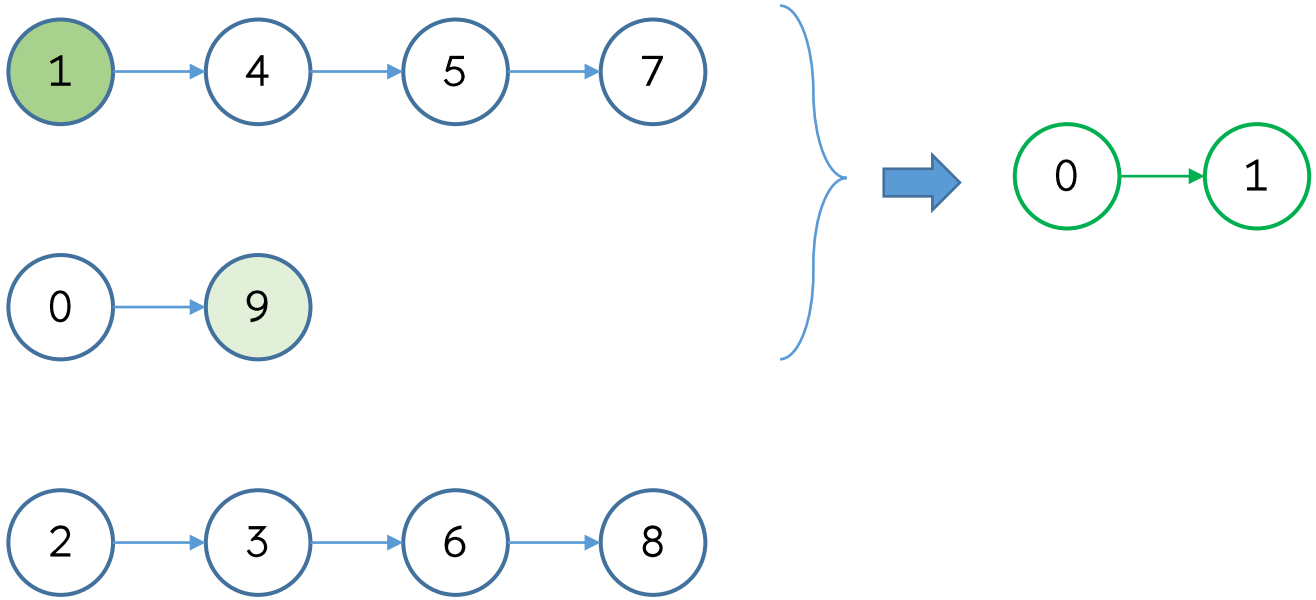
K-WAY MERGE друг за другом



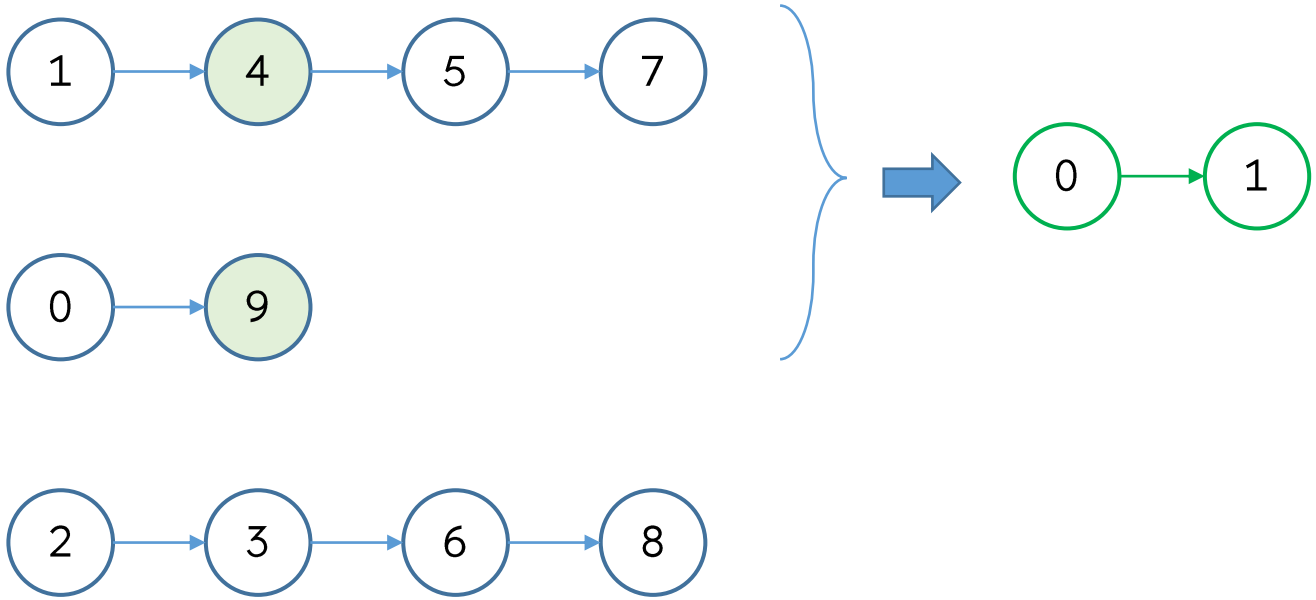
K-WAY MERGE друг за другом



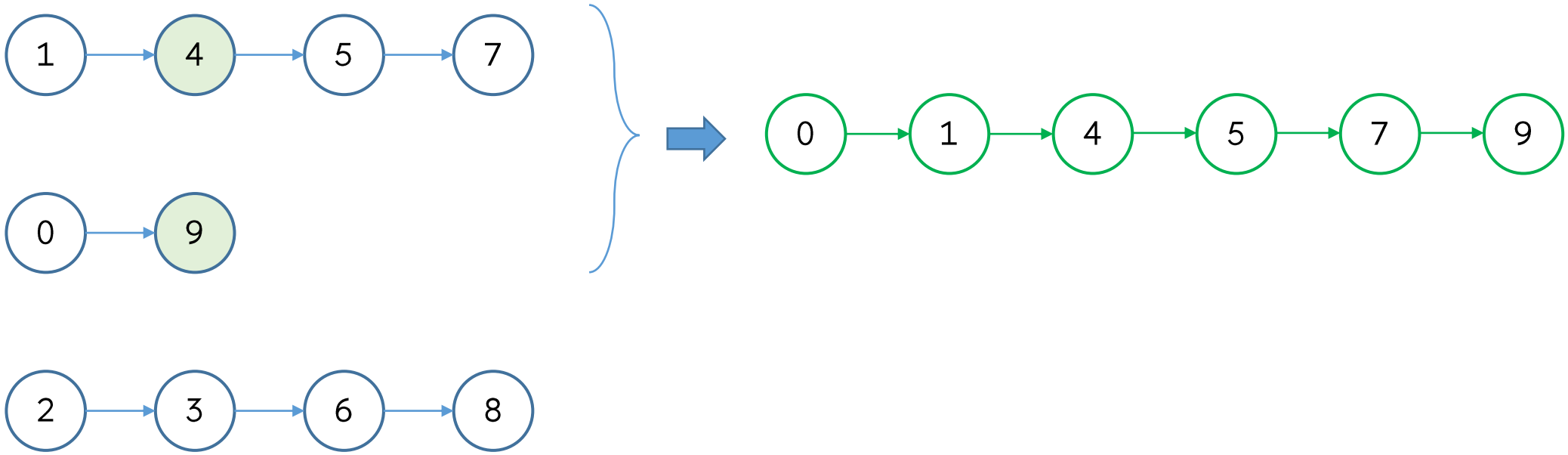
K-WAY MERGE друг за другом



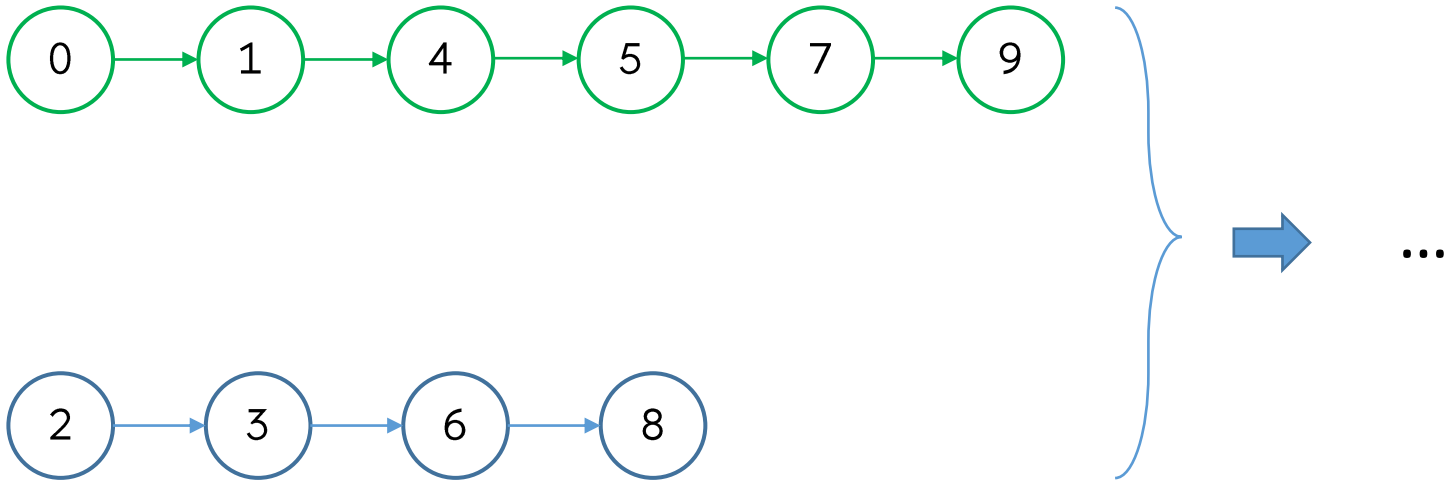
K-WAY MERGE друг за другом



K-WAY MERGE друг за другом



K-WAY MERGE друг за другом



K-WAY MERGE друг за другом

kWayMergeOneByOne(*lists, k*)

```
1  if k = 0 return null
2  if k = 1 return lists[0]
3  head = mergeTwoSortedLists(lists[0], lists[1])
4  for i = 2 to k
5      head = mergeTwoSortedLists(head, lists[i])
6  return head
```

K-WAY MERGE друг за другом

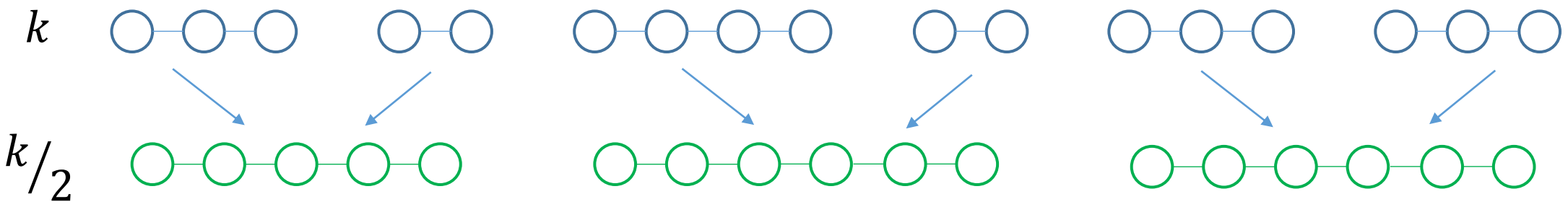
kWayMergeOneByOne(*lists*, *k*)

```
1  if k = 0 return null
2  if k = 1 return lists[0]
3  head = mergeTwoSortedLists(lists[0], lists[1])
4  for i = 2 to k
5      head = mergeTwoSortedLists(head, lists[i])
6  return head
```

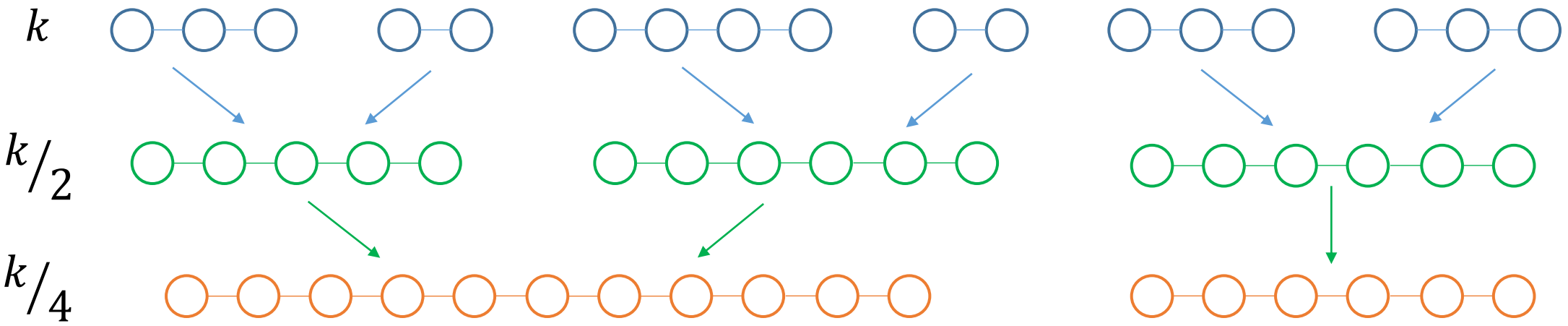
Выведем оценку верхней границы

K-WAY MERGE DaC

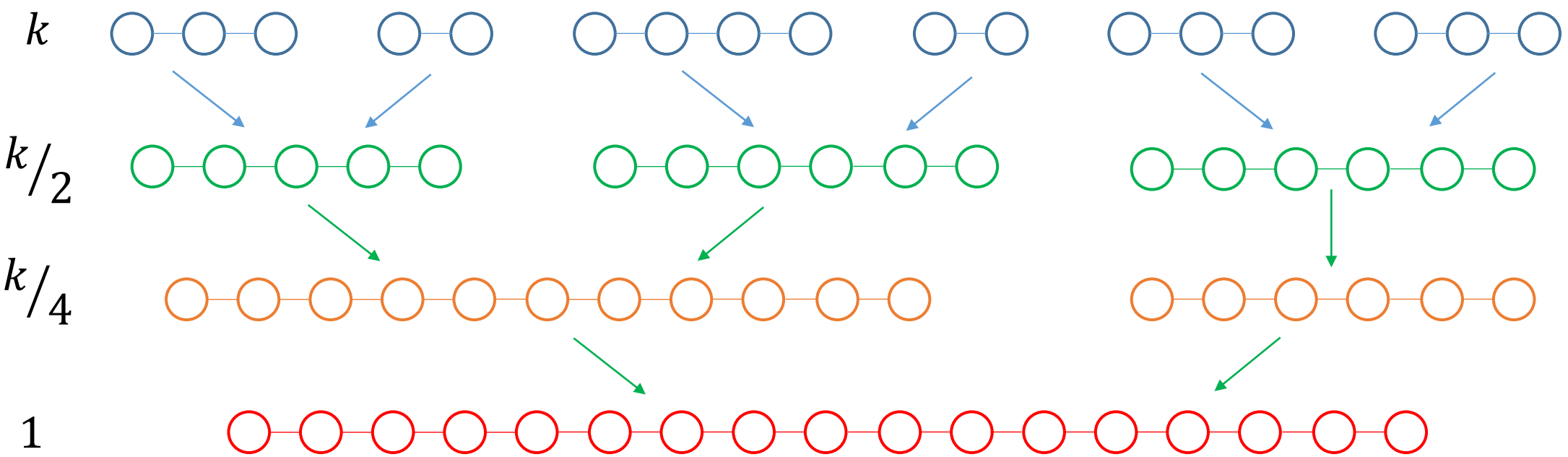
K-WAY MERGE DaC



K-WAY MERGE DaC



K-WAY MERGE DaC



K-WAY MERGE DaC

kWayMergeDaC(*lists*, *k*)

```
1  if k = 0 return null
2  intrv = 1
3  while intrv < k
4      i = 0
5      while i + intrv < k
6          lists[i] = mergeTwoSortedLists(lists[i], lists[i + intrv])
7          i = i + intrv * 2
8      intrv *= 2
9  return lists[0]
```

Выведем оценку верхней границы

Recap

Проектирование и анализ сложности алгоритмов по стратегии «разделяй-и-властвуй», сравнение с другими подходами

Задача поиска ближайшей пары точек

Задача k-проходного слияния отсортированных списков