ITIVITI

LET IT TRADE

Алгоритмы и структуры данных

Косяков Михаил Сергеевич
к.т.н., доцент кафедры ВТ

УНИВЕРСИТЕТ ИТМО

Принцип divide-and-conquer

ITIVITI

LET IT TRADE

- Решаем задачу рекурсивно, применяя на каждом уровне рекурсии три шага:
- **Divide:** разделяем задачу на несколько таких же задач меньшего размера
- **Conquer:** решаем подзадачи рекурсивно
  - Подзадачи малого размера решаем прямым решением
- **Combine:** комбинируем решения подзадач в решение исходной задачи

УНИВЕРСИТЕТ ИТМО

Оценка сложности

- $T(N) = T(N/2) + T(N/2) + 1$, где $T(N)$ – время выполнения задачи размера N
- Какая сложность?

Оценка сложности

- $T(N) = T(N/2) + T(N/2) + 1$, где $T(N)$ – время выполнения задачи размера N
- Метод №1 – сделаем предположение и докажем его корректность
- По индукции: $T(N) \le cN$ для некоторой константы c
  - Более строгое предположение: $T(N) \le cN - a$, $a \ge 1$
  - База индукции: $T(1) \le c - a$
  - Шаг индукции: пусть $T(N/2) \le cN/2 - a$, надо показать, что $T(\mathrm{N}) \le cN - a$
  - $T(N) = T(N/2) + T(N/2) + 1 \le cN - 2a + 1 \le cN - a$, $a \ge 1$
- Как угадать решение?

Оценка сложности

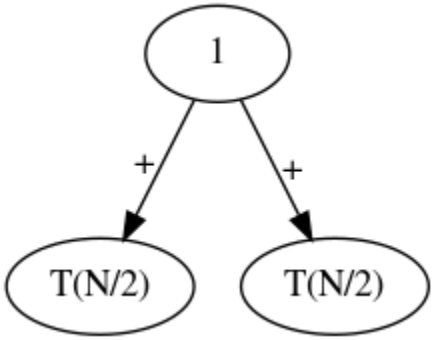
- $T(N) = T(N/2) + T(N/2) + 1$, где $T(N)$ – время выполнения задачи размера N
- Метод №1 – сделаем предположение и докажем его корректность
- По индукции: $T(N) \le cN$ для некоторой константы c
  - Более строгое предположение: $T(N) \le cN - a$, $a \ge 1$
  - База индукции: $T(1) \le c - a$
  - Шаг индукции: пусть $T(N/2) \le cN/2 - a$, надо показать, что $T(\mathrm{N}) \le cN - a$
  - $T(N) = T(N/2) + T(N/2) + 1 \le cN - 2a + 1 \le cN - a$, $a \ge 1$
- Метод №2 – построим дерево рекурсии

Дерево рекурсии

- $T(N) = T(N/2) + T(N/2) + 1$

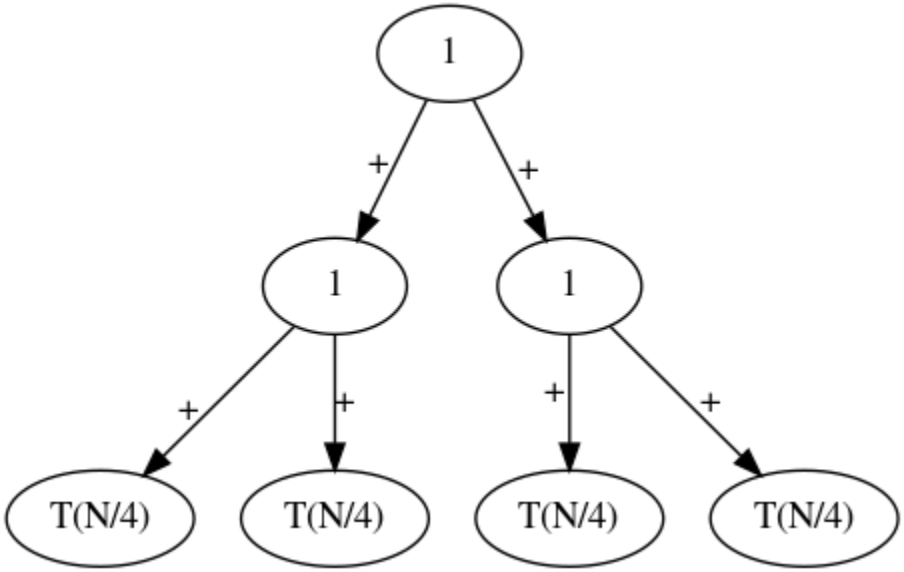
Дерево рекурсии

- $T(N/2) = T(N/4) + T(N/4) + 1$



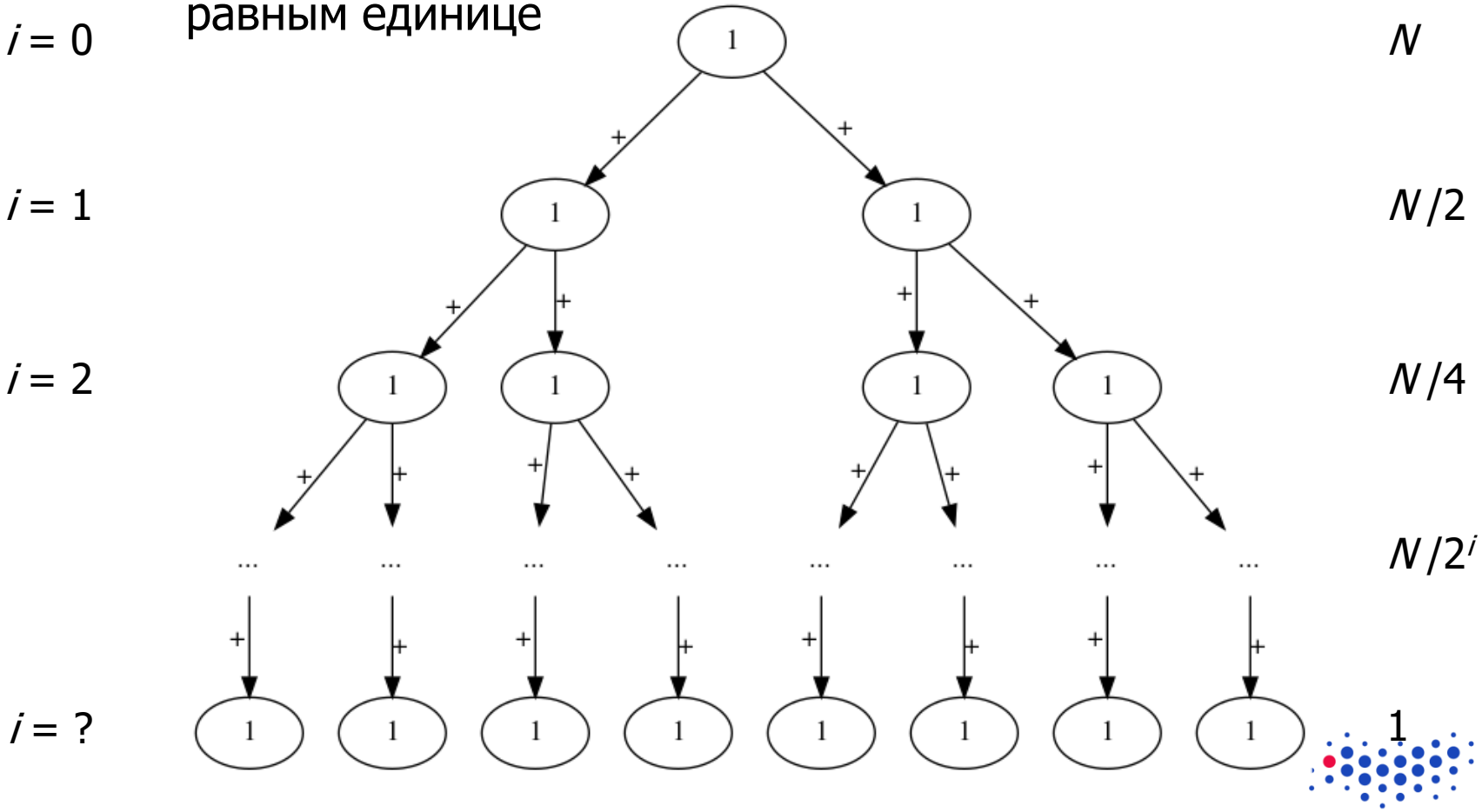
Дерево рекурсии

- $T(N/4) = T(N/8) + T(N/8) + 1$



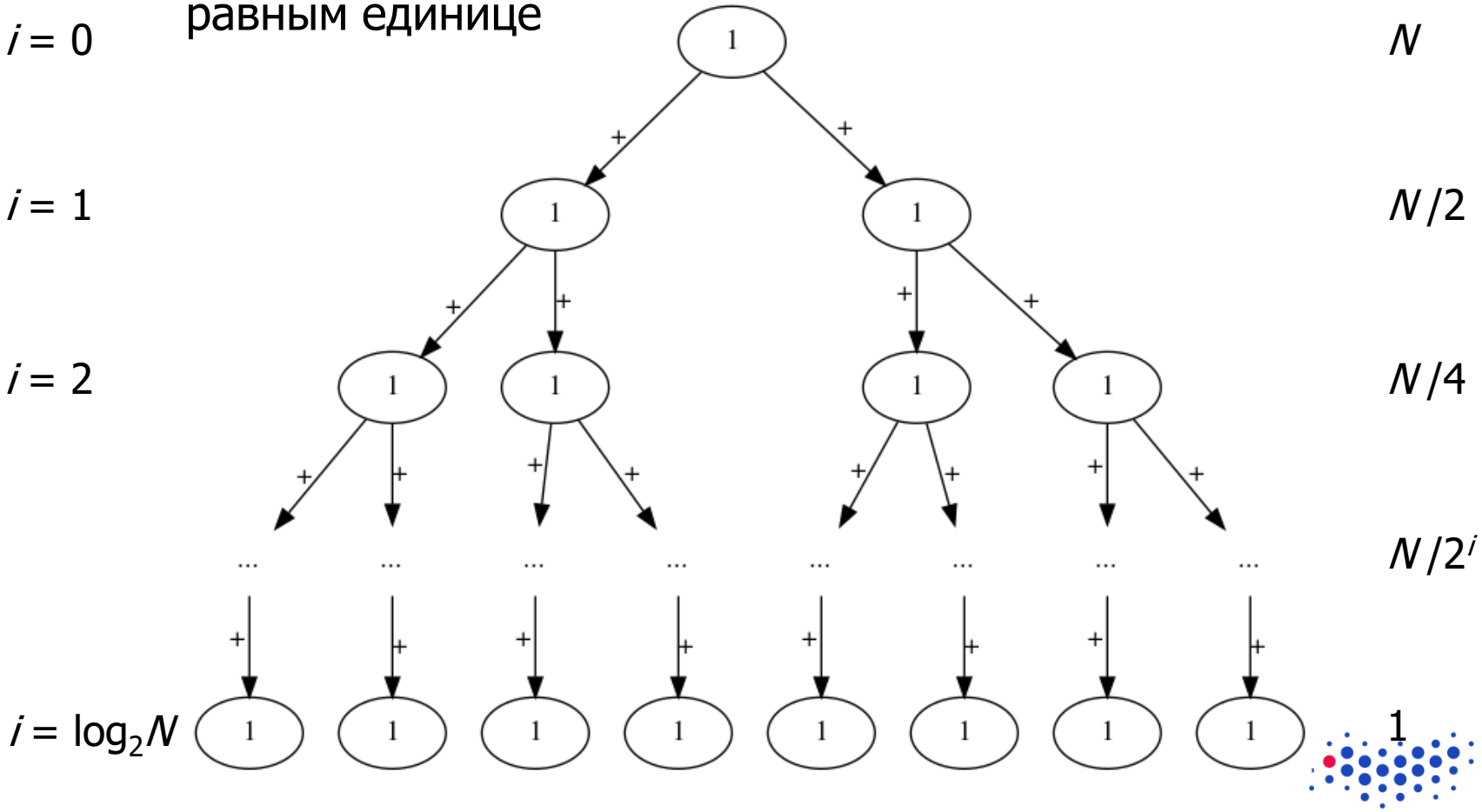
Дерево рекурсии

- Рекурсия останавливается, когда размер подзадачи станет равным единице



Дерево рекурсии

- Рекурсия останавливается, когда размер подзадачи станет равным единице



Оценка сложности

- На нижнем уровне N узлов (подзадач) каждая со временем выполнения $T(1)$, поэтому время работы этого уровня $\Theta(N)$
- Общее время работы:

$$1 + 2^1 + 2^2 + \ldots + 2^{\log N - 1} + \Theta(N) =$$
$$= (2^{\log N} - 1) / (2 - 1) + \Theta(N) = N - 1 + \Theta(N)$$

- $T(N) = \Theta(N)$
- Предположение: количество элементов в исходной задаче равно степени двойки
  - Не влияет на порядок роста: почему?

Умножение чисел

- Требуется перемножить два n -значных числа
- В столбик? Оценка сложности: $\Theta(N^2)$
- Divide: разобьем каждое число на две $n/2$ –значные половины
  - $num_1 = x_1 * 10^{n/2} + x_0$
  - $num_2 = y_1 * 10^{n/2} + y_0$
- Conquer: найдем произведения $x_1 * y_1$; $x_1 * y_0$; $x_0 * y_1$; $x_0 * y_0$
  - Произведение однозначных чисел выполняется за $O(1)$
- Combine: $num_1 * num_2 = (x_1 * 10^{n/2} + x_0)(y_1 * 10^{n/2} + y_0) =$

$$x_1 * y_1 * 10^n + (x_1 * y_0 + x_0 * y_1) * 10^{n/2} + x_0 * y_0$$

Умножение чисел

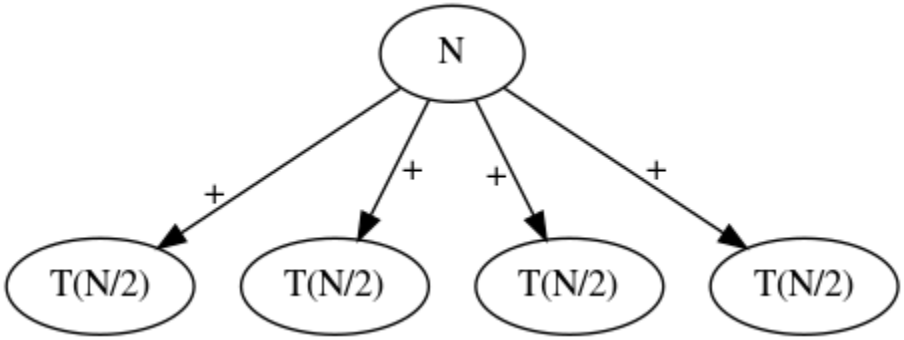
- $x_1 * y_1 * 10^n + (x_1 * y_0 + x_0 * y_1) * 10^{n/2} + x_0 * y_0$
- Предположим, что:
  - Умножение на 10^N делаем сдвигом цифр в массиве => $O(N)$
  - Сложение делаем поразрядно с переносом. Разрядность цифр в формуле не превосходит $2N$ => $O(N)$
- $T(N) = 4T(N/2) + O(N)$
- $T(N) = \Theta(?)$

Рекурсивное умножение

- $T(N) = 4T(N/2) + N$

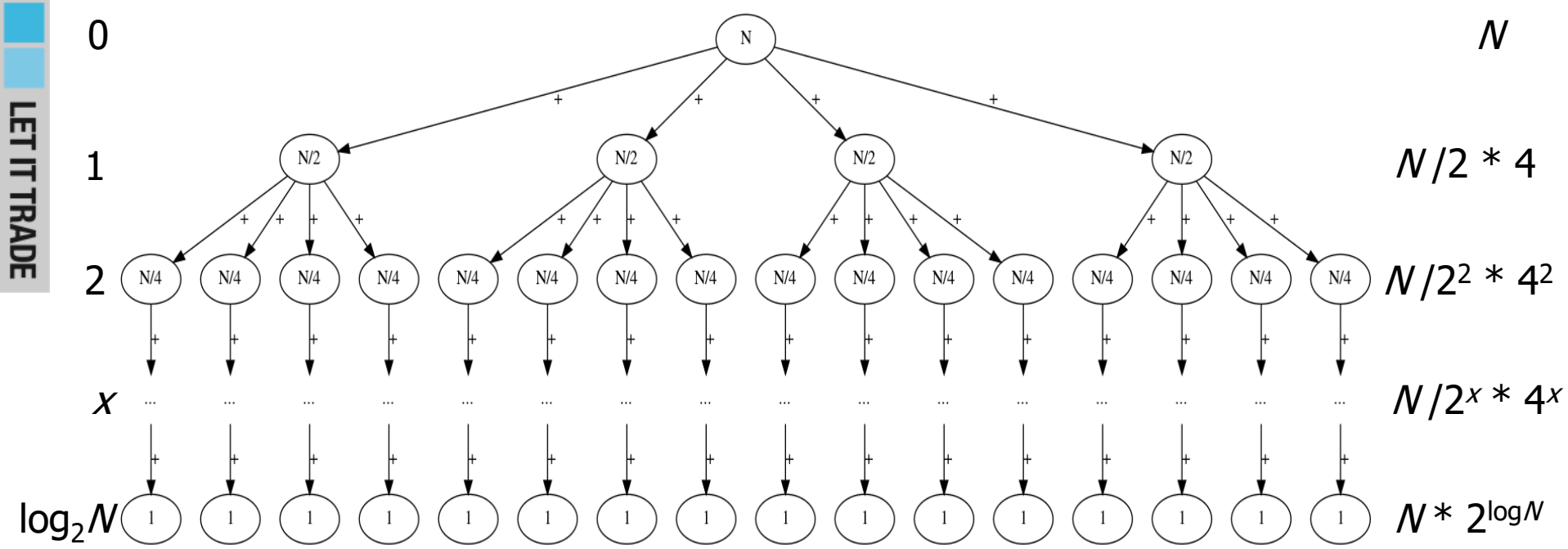
Рекурсивное умножение

- $T(N/2) = 4T(N/4) + N/2$



Рекурсивное умножение

- Каждая цифра из num_1 перемножается с каждой цифрой из num_2 => N^2 операций на нижнем уровне



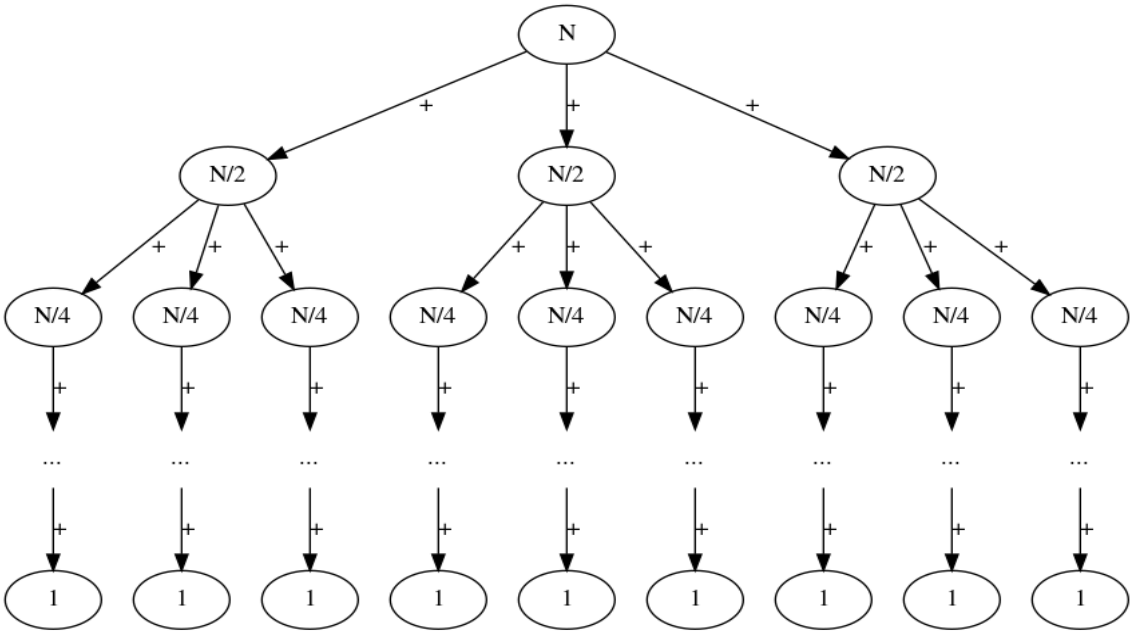
- $T(N) = N(2^{\log N + 1} - 1) / (2 - 1) = N(2N - 1) = \Theta(N^2)$

Алгоритм Карацубы

- Divide: как и прежде
- Conquer: вместо четырех подзадач требуется решить три
  - $A = x_1 * y_1$
  - $B = x_0 * y_0$
  - $C = (x_1 + x_0)(y_0 + y_1)$
- Combine: $num_1 * num_2 = A * 10^n + (C - A - B) * 10^{n/2} + B$
  - $C - A - B = (x_1 + x_0)(y_0 + y_1) - x_1 * y_1 - x_0 * y_0 =$
    $= x_1 * y_0 + x_0 * y_1$
- $T(N) = 3T(N/2) + O(N)$
- $T(N) = \Theta(\ ?\ )$

Алгоритм Карацубы

- Такое же число уровней дерева рекурсии, что и в предыдущем алгоритме



- $(3/2)^{\log N} * N = (3^{\log N} / 2^{\log N}) * N = (N^{\log 3} / N) * N = N^{\log 3}$
- $T(N) = \Theta (N^{\log 3})$ как сумма геом. прогрессии

Умножение полиномов

- $A(x) = \sum_{k=0}^{n} a_k x^k$ $\qquad$ $B(x) = \sum_{k=0}^{n} b_k x^k$
- Divide: $A(x) = A_1(x)\, x^{\frac{n}{2}} + A_0(x)$ $\qquad$ $B(x) = B_1(x)\, x^{\frac{n}{2}} + B_0(x)$
  - Пример: $1 + 3x + x^2 + 7x^3 = (1 + 3x) + x^2(1 + 7x)$
- Conquer (алгоритм Карацубы):
  - $C_2(x) = A_1(x)\, B_1(x)$
  - $C_1(x) = (A_0(x) + A_1(x))(B_0(x) + B_1(x)) - A_0(x)\, B_0(x) - A_1(x)\, B_1(x)$
  - $C_0(x) = A_0(x)\, B_0(x)$
- Combine: $A(x)\, B(x) = C_2(x)\, x^n + C_1(x)\, x^{\frac{n}{2}} + C_0(x)$

Содержание курса

- Введение в теорию алгоритмов
- **Алгоритмы сортировок**
- Структуры данных
  - Линейные структуры
  - Бинарные деревья поиска
  - Хеши и хеш-функции
- Алгоритмы на графах
  - Обходы графов в ширину и глубину
  - Минимальные остовные деревья
  - Поиск кратчайших путей в графе

Мотивация

ITIVITI

LET IT TRADE

- Одна из самых естественных и распространенных задач:
  - Словарь – слова расположены в алфавитном порядке
  - Рейтинг студентов – у кого больше баллов, тот выше в списке
  - Расписание поездов, самолетов – по времени прибытия / отправления
  - Многие алгоритмы требуют отсортированные входные данные: различные геометрические задачи, ORDER BY, GROUP BY, DISTINCT в СУБД, быстрое объединение структур данных (в т.ч. JOIN в СУБД) и т.д.

Мотивация

- Таблица "Customers"

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
...	...	...	...	...	...	...

https://www.w3schools.com/sql/

Мотивация
Сортировка

ITIVITI

- Пример ORDER BY

```
1.SELECT CustomerID, Country
2.FROM Customers
3.ORDER BY Country;
```

CustomerID	Country
12	Argentina
54	Argentina
64	Argentina
20	Austria
59	Austria
50	Belgium
76	Belgium
15	Brazil
…	…

LET IT TRADE

https://www.w3schools.com/sql/

УНИВЕРСИТЕТ ИТМО

Мотивация
Агрегация

ITIVITI

- Пример GROUP BY

```
1.SELECT COUNT(CustomerID), Country
2.FROM Customers
3.GROUP BY Country;
```

COUNT(CustomerID)	Country
3	Argentina
2	Austria
2	Belgium
9	Brazil
3	Canada
2	Denmark
2	Finland
11	France
…	…

LET IT TRADE

https://www.w3schools.com/sql/

УНИВЕРСИТЕТ ИТМО

Мотивация
Дедупликация

ITIVITI

- Пример DISTINCT

```
1.SELECT DISTINCT Country FROM Customers;
```

Number of Records: 21

Country
Argentina
Austria
Belgium
Brazil
Canada
Denmark
Finland
France
Germany
Ireland
…

LET IT TRADE

https://www.w3schools.com/sql/

УНИВЕРСИТЕТ ИТМО

Бинарный поиск

ITIVITI

- Найдем идентификаторы всех таймзон:

```
TimeZoneIdentifier using_linear_str(const char * timezone_name)
{
    if (::strcmp(timezone_name, "America/Porto_Acre") == 0)
        { return TZ_STD_AMERICA_PORTO_ACRE; }
    if (::strcmp(timezone_name, "Australia/Perth") == 0)
        { return TZ_STD_AUSTRALIA_PERTH; }
    if (::strcmp(timezone_name, "Europe/Madrid") == 0)
        { return TZ_DST_EUROPE_MADRID; }
...
```

- Повторим эксперимент тысячу раз:

```
                          #       50%         95%         99.9%
=============================================================
linear_str_begin -
linear_str_end            1000    3831 µs     4062 µs     4530 µs
```

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Бинарный поиск

ITIVITI

- Найдем идентификаторы всех таймзон:

```
TimeZoneIdentifier using_linear(const char * timezone_name)
{
    const uint32_t hash_key = hash(timezone_name);

    if (0x013129B9 == hash_key) { return  TZ_STD_AMERICA_PORTO_ACRE; }
    if (0x014A1BF1 == hash_key) { return  TZ_STD_AUSTRALIA_PERTH; }
    if (0x03C9E0D7 == hash_key) { return  TZ_DST_EUROPE_MADRID; }
...
```

- Повторим эксперимент тысячу раз:

```
                          #       50%         95%        99.9%
=============================================================
linear_begin -
linear_end                1000    122 µs      140 µs     439 µs
```

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Бинарный поиск

ITIVITI

- Найдем идентификаторы всех таймзон:

```
TimeZoneIdentifier using_switch(const char * timezone_name)
{
    const uint32_t hash_key = hash(timezone_name);
    switch(hash_key)
    {
        case 0x013129B9: return  TZ_STD_AMERICA_PORTO_ACRE;
        case 0x014A1BF1: return  TZ_STD_AUSTRALIA_PERTH;
        case 0x03C9E0D7: return  TZ_DST_EUROPE_MADRID;
...
```

- Повторим эксперимент тысячу раз:

```
                          #      50%        95%       99.9%
=============================================================
switch_begin -
switch_end                1000   25 µs      28 µs     89 µs
```

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Бинарный поиск и сортировка

ITIVITI

- Бинарный поиск:
  - $T(N) = O(1) + T(N/2)$
  - Число атомов во Вселенной ~10^{80}, найдем за ~260 шагов
  - ? $x_0 : f(x_0) = c$, $f(x)$ монотонно возрастает
- Сортировка:
  - Отношение порядка < на множестве ключей:
  - Иррефлексивно и транзитивно
  - Трихотомия: $\forall\ a, b : (a < b) \vee (b < a) \vee (a = b)$
  - Игра «камень, ножницы, бумага»
- Задача сортировки - получить перестановку, в которой ключи расположены в порядке неубывания

LET IT TRADE

Характеристики сортировок

ITIVITI

- Оценка эффективности алгоритма сортировки:
  - Время работы алгоритма в лучшем, среднем и худшем случае
  - Объем используемой дополнительной памяти
- Свойства алгоритма сортировки:
  - Рекурсивность
  - Стабильность
  - Адаптивность
  - Тип (на основе сравнения, подсчета, …)

LET IT TRADE

УНИВЕРСИТЕТ ИТМО

Стабильность

По времени	->По фамилии (нестабильная)	->По фамилии (стабильная)
Петров 09.00	Беляев 10.00	Беляев 10.00
Петров 10.00	Иванов 12.00	Иванов 11.00
Беляев 10.00	Иванов 11.00	Иванов 12.00
Торопов 11.00	Косяков 13.00	Косяков 13.00
Иванов 11.00	Петров 10.00	Петров 09.00
Иванов 12.00	Петров 09.00	Петров 10.00
Косяков 13.00	Торопов 11.00	Торопов 11.00

Сортировка выбором

- Самый простой алгоритм сортировки: на каждой итерации находим минимум и кладем его в начало подмассива

```
1.  void selection_sort(int * a, int n)
2.  {
3.      for(int i = 0; i < n - 1; ++i) {
4.          for(int j = i + 1; j < n; ++j) {
5.              if (a[j] < a[i]) {
6.                  int t = a[i];
7.                  a[i] = a[j];
8.                  a[j] = t;
9.              }
10.         }
11.     }
12. }
```

- Инвариант: после i-ой итерации `a[0..i]` отсортирован, каждый элемент из `a[i+1..n]` не меньше `a[i]`

Сортировка выбором

- Самый простой алгоритм сортировки: на каждой итерации находим минимум и кладем его в начало подмассива

```
1.  void selection_sort(int * a, int n)
2.  {
3.      for(int i = 0; i < n - 1; ++i) {
4.          for(int j = i + 1; j < n; ++j) {
5.              if (a[j] < a[i]) {
6.                  int t = a[i];
7.                  a[i] = a[j];
8.                  a[j] = t;
9.              }
10.         }
11.     }
12. }
```

- Можно через рекурсию: $T(N) = O(N) + T(N-1)$

Сортировка выбором

```
1.  void selection_sort(int * a, int n) {
2.      for(int i = 0; i < n - 1; ++i) {
3.          int min = i;
4.          for(int j = i + 1; j < n; ++j) {
5.              if (a[j] < a[min]) {
6.                  min = j;
7.              }
8.          }
9.          int t = a[i];
10.         a[i] = a[min];
11.         a[min] = t;
12.     }
13. }
```

- Оценка сложности: $\Theta(N^2)$
- Дополнительная память: $O(1)$

Сортировка пузырьком

```
1. void bubble_sort(int * a, int n) {
2.     bool swap = true;
3.     while(swap) {
4.         swap = false; --n;
5.         for(int i = 0; i < n; ++i) {
6.             if (a[i + 1] < a[i]) {
7.                 int t = a[i];
8.                 a[i] = a[i + 1];
9.                 a[i + 1] = t;
10.                swap = true;
11.            }
12.        }
13.    }
14. }
```

- Оценка сложности: $O(N^2)$
- Дополнительная память: $O(1)$

Сортировка слиянием

ITIVITI

LET IT TRADE

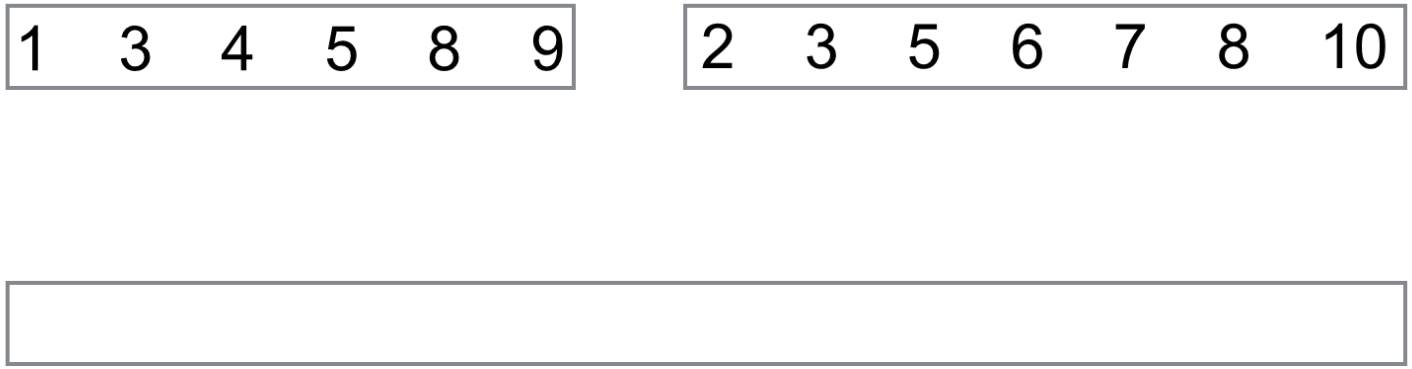
- Метод «разделяй и властвуй» (divide-and-conquer):
  - 1) разделим задачу на подзадачи
  - 2) решаем подзадачи с использованием рекурсии
  - 3) объединяем решения подзадач в решение задачи

Сортировка слиянием

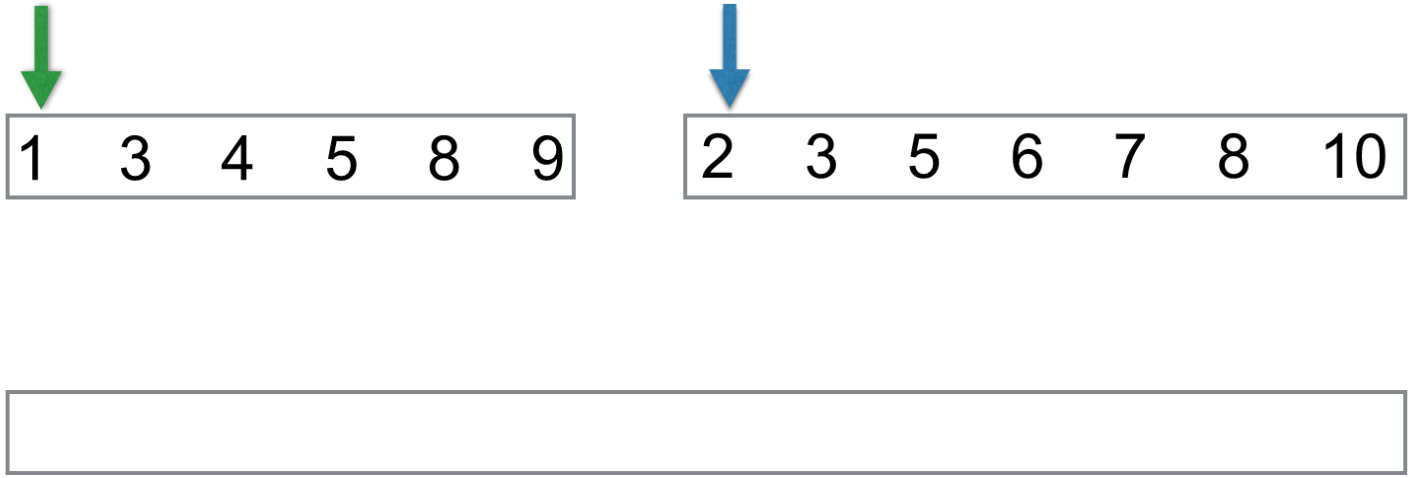
```
1.  void merge_sort(int * a, int * aux, int l, int r) {
2.      if (l < r) {
3.          int m = (l + r) / 2;
4.          merge_sort(a, aux, l, m);
5.          merge_sort(a, aux, m + 1, r);
6.          merge(a, aux, l, m, r);
7.      }
8.  }
```

- Для заданного *N* делим задачу на подзадачи всегда одинаковым образом (в отличии от быстрой сортировки)
- Время работы алгоритма не зависит от входных данных (только от размера задачи)
- Основная работа – после выполнения рекурсивных вызовов при слиянии

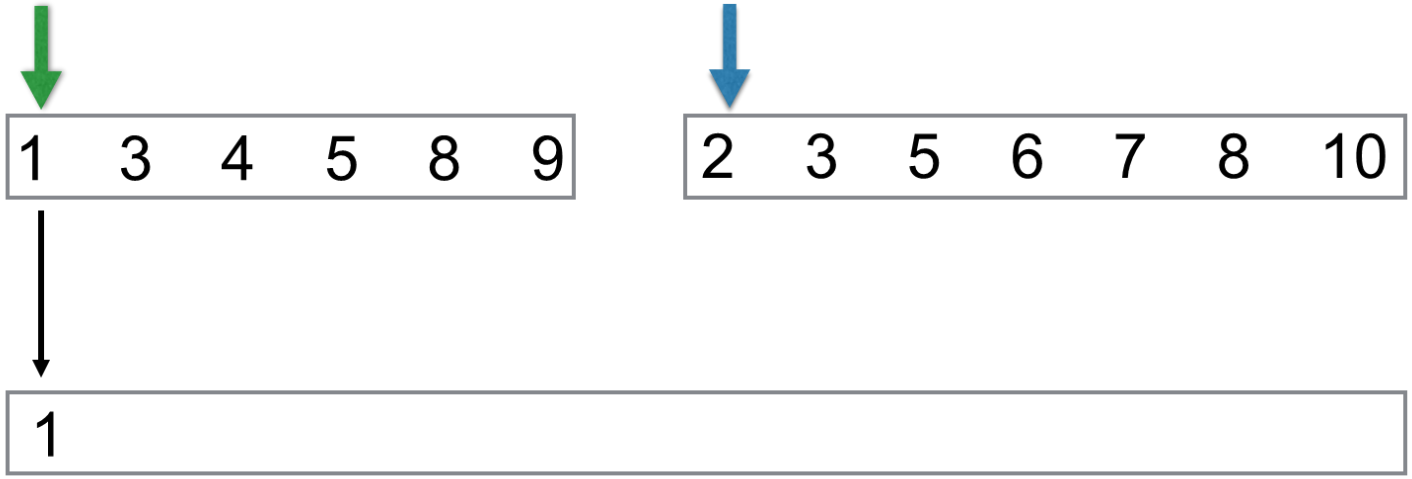
Сортировка слиянием



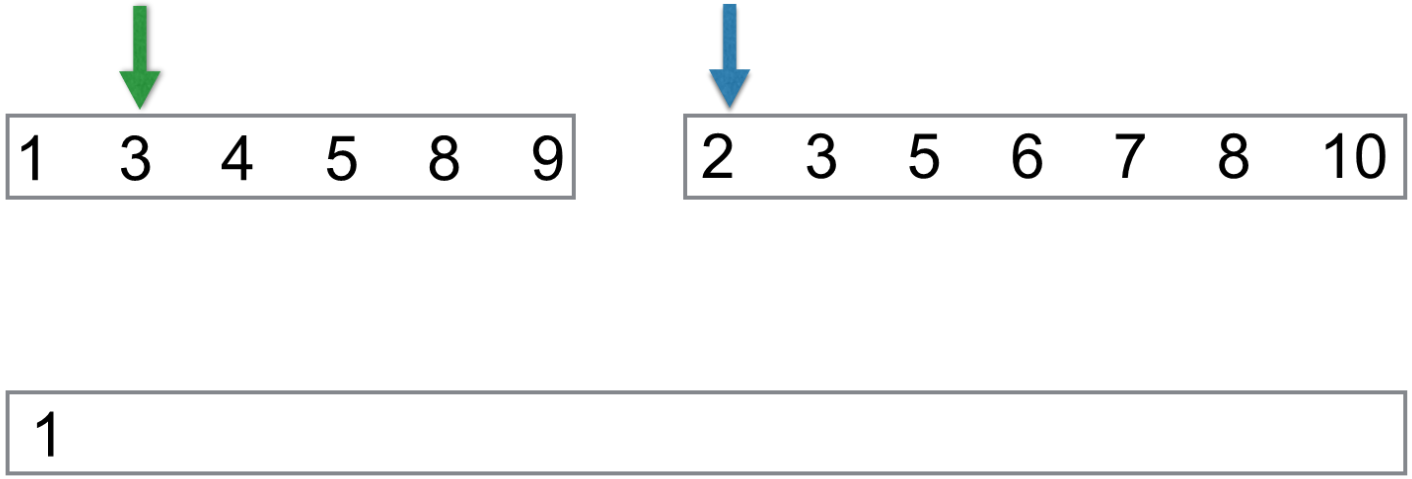
Сортировка слиянием



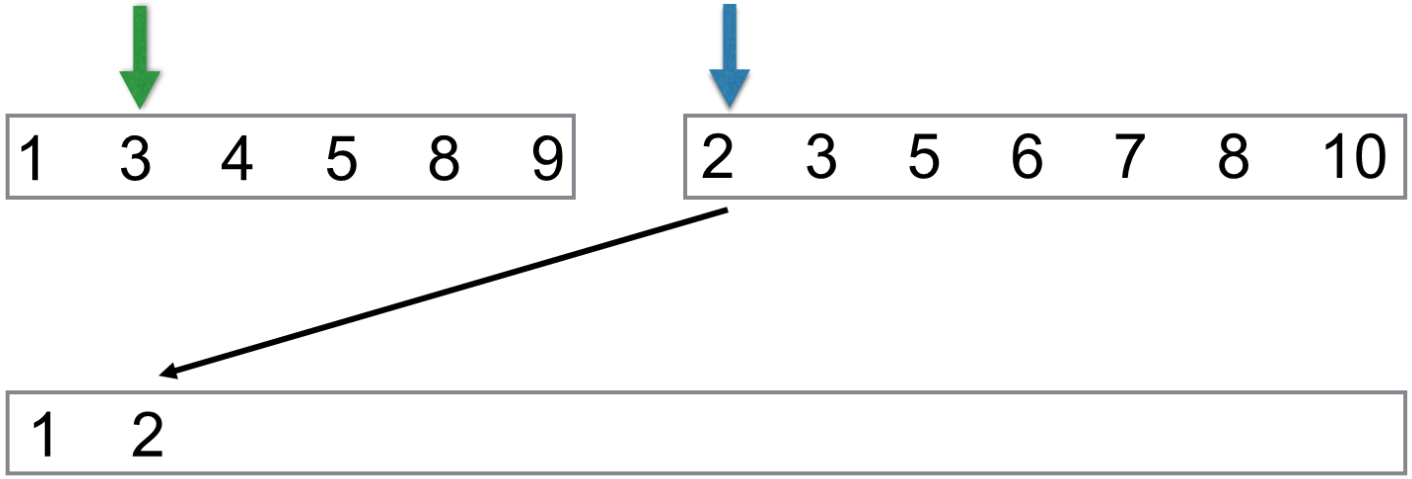
Сортировка слиянием



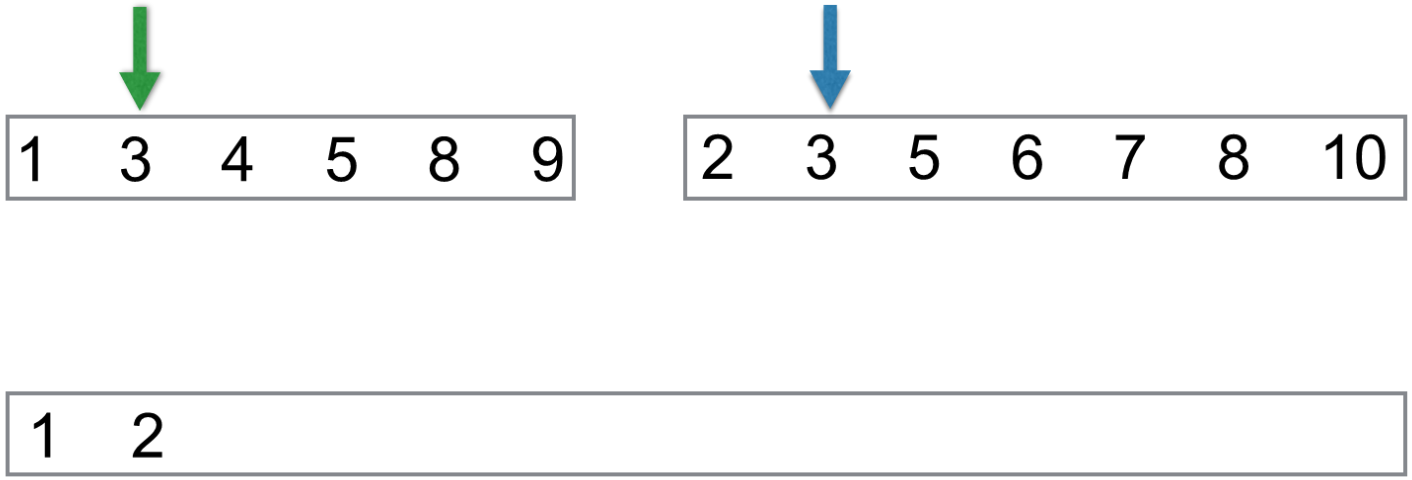
Сортировка слиянием



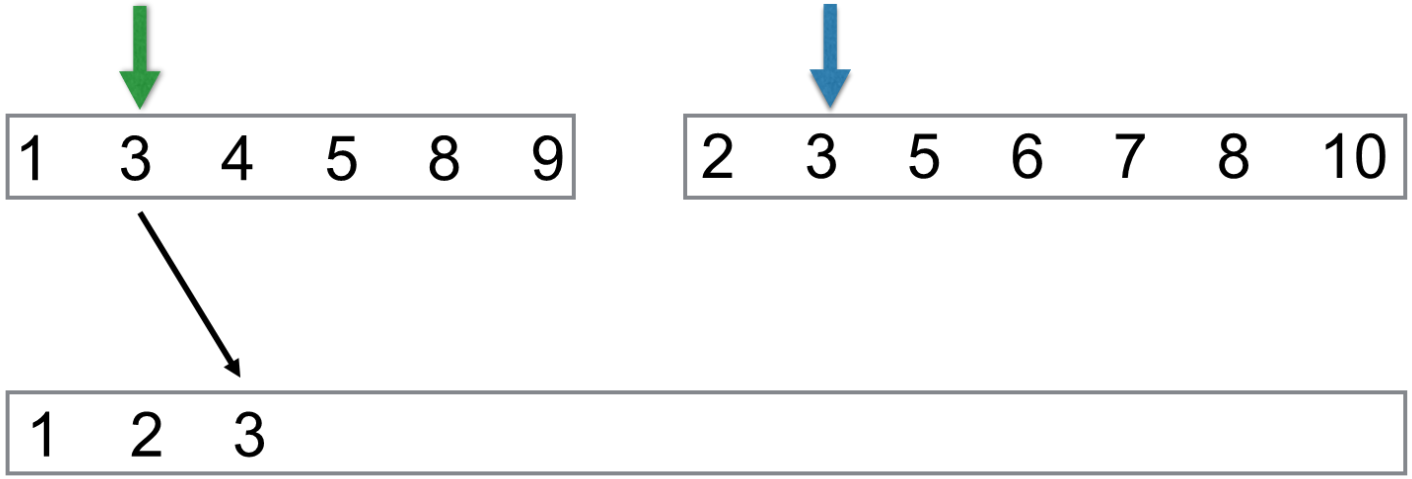
Сортировка слиянием



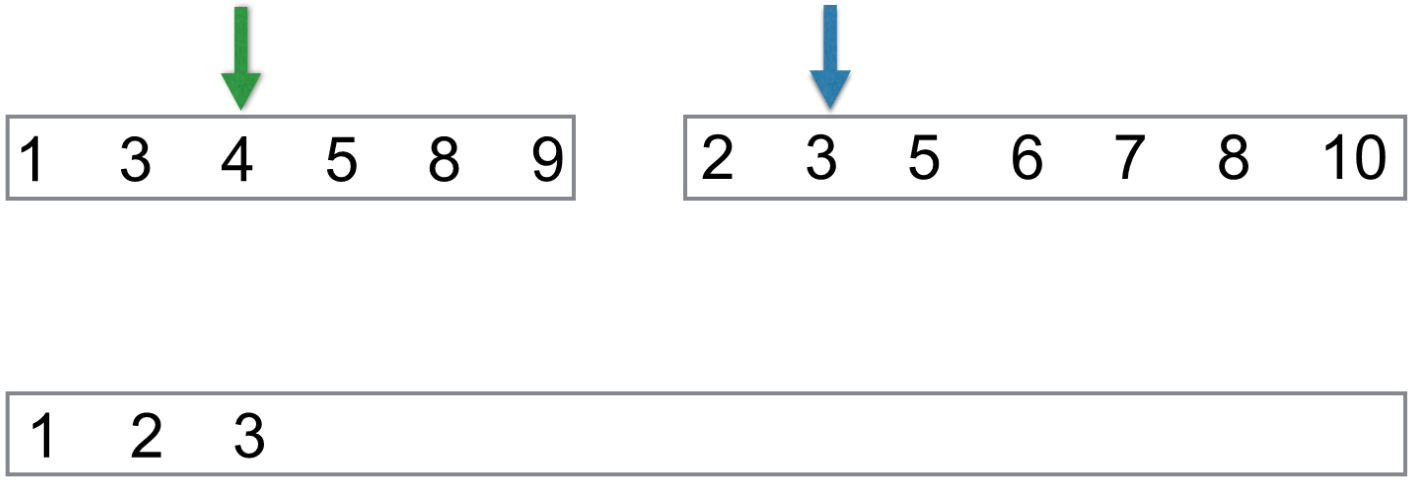
Сортировка слиянием



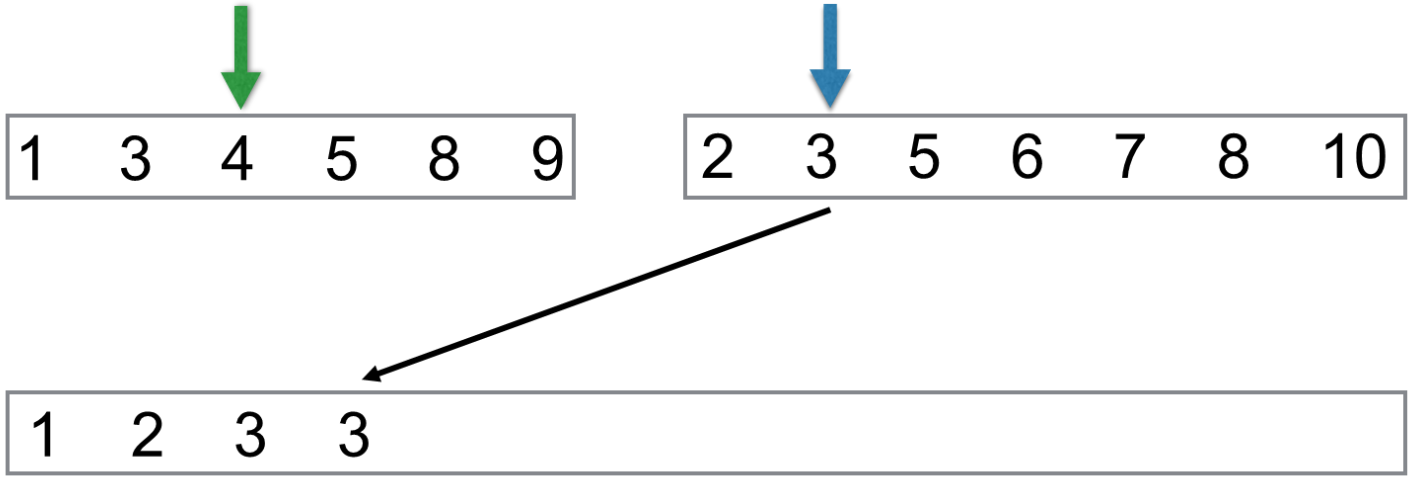
Сортировка слиянием



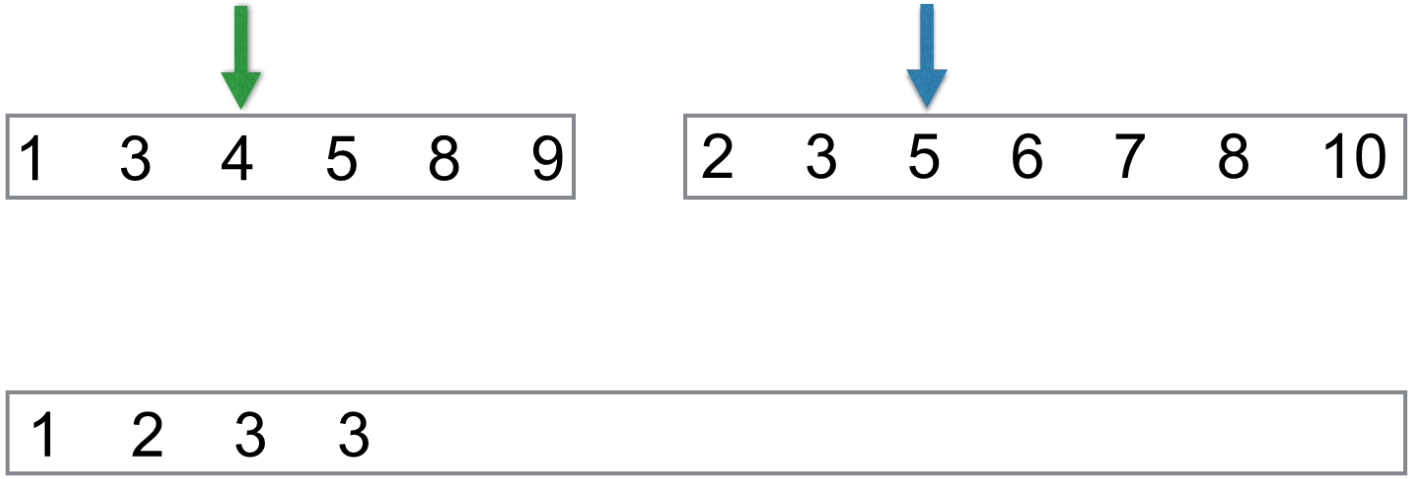
Сортировка слиянием



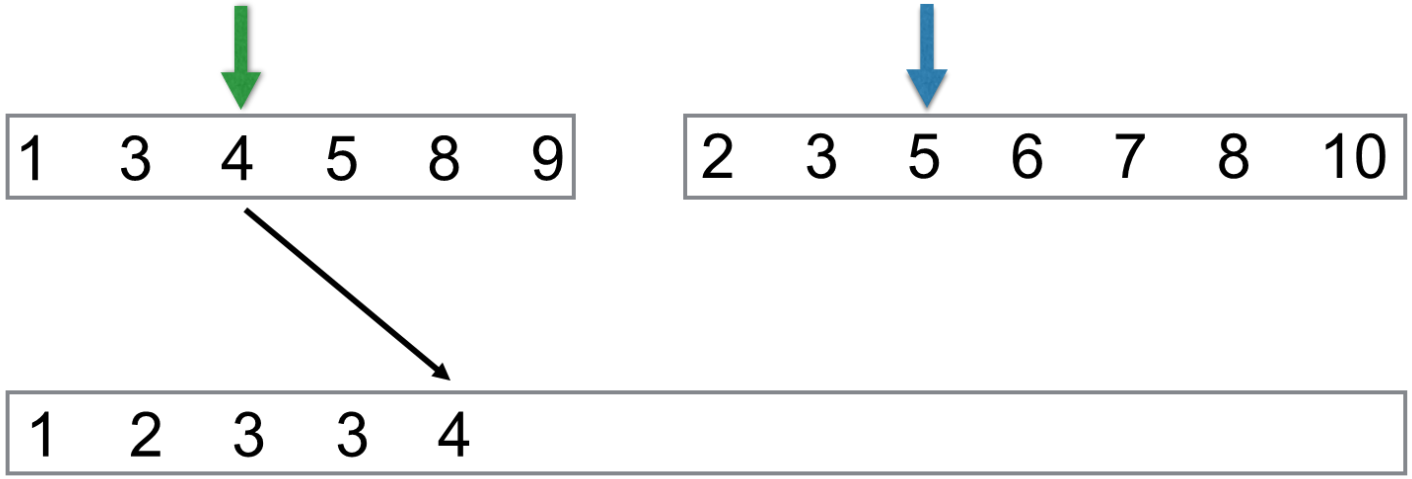
Сортировка слиянием



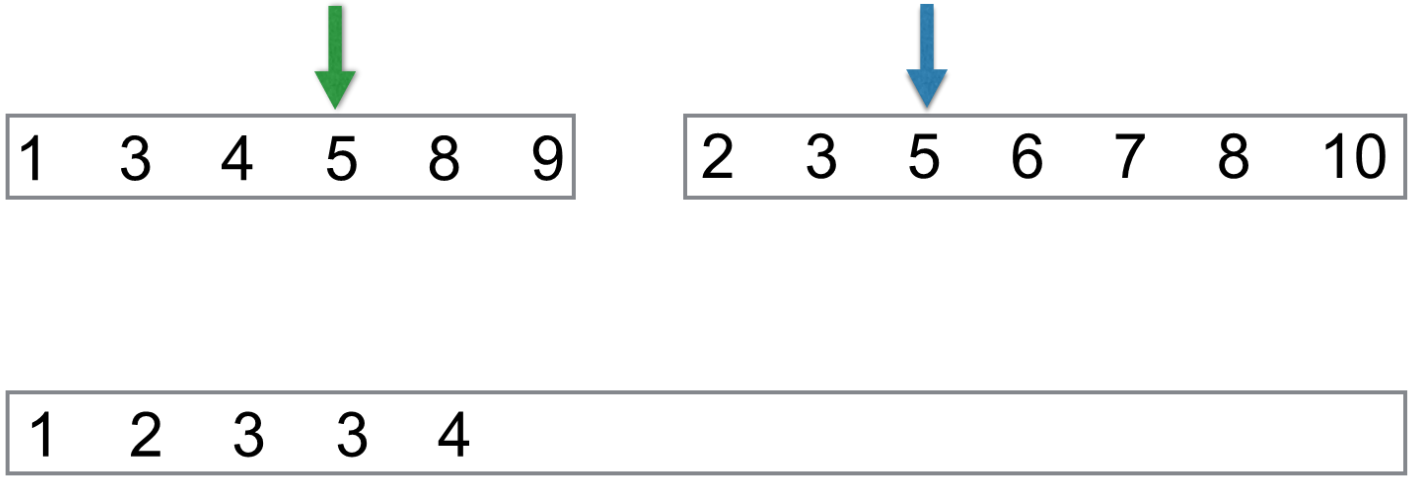
Сортировка слиянием



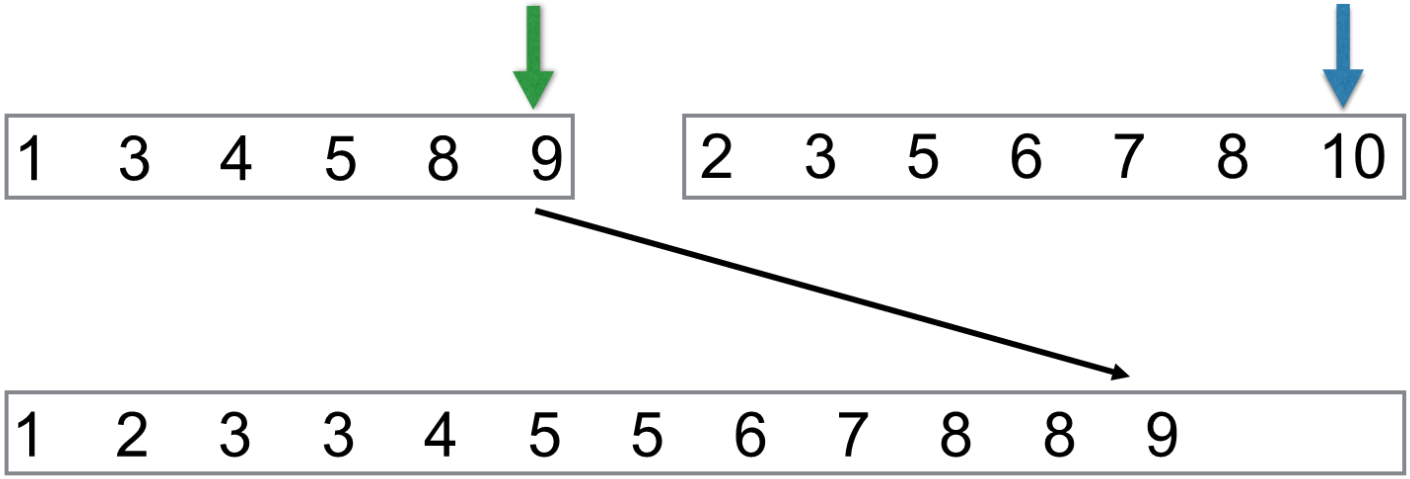
Сортировка слиянием



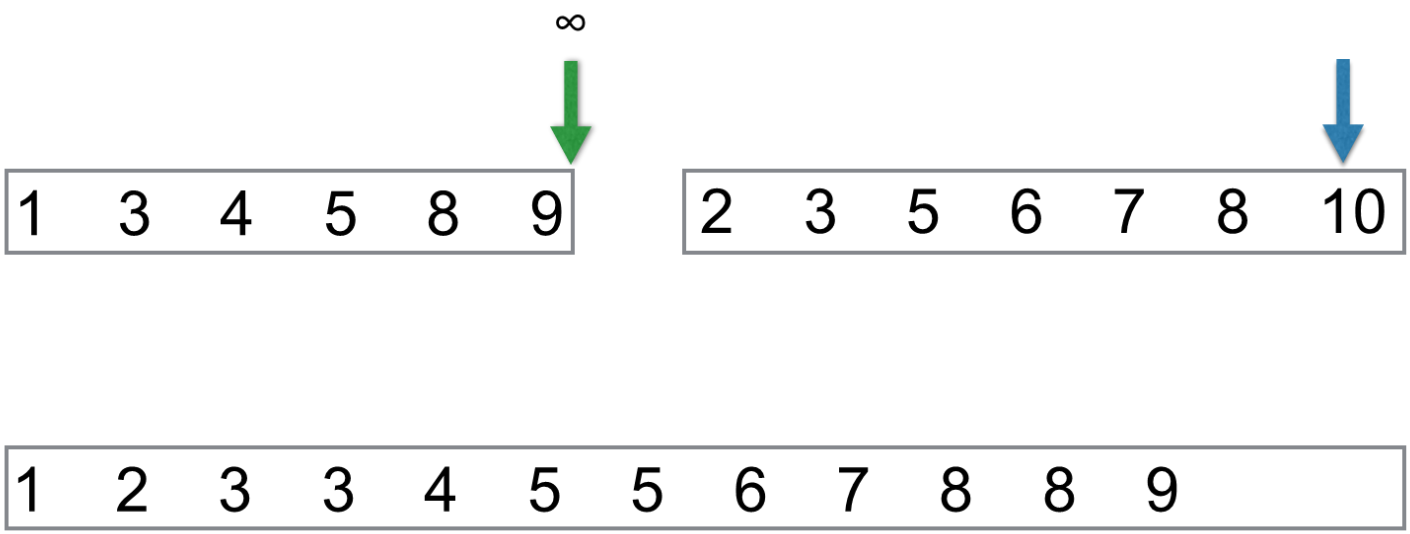
Сортировка слиянием



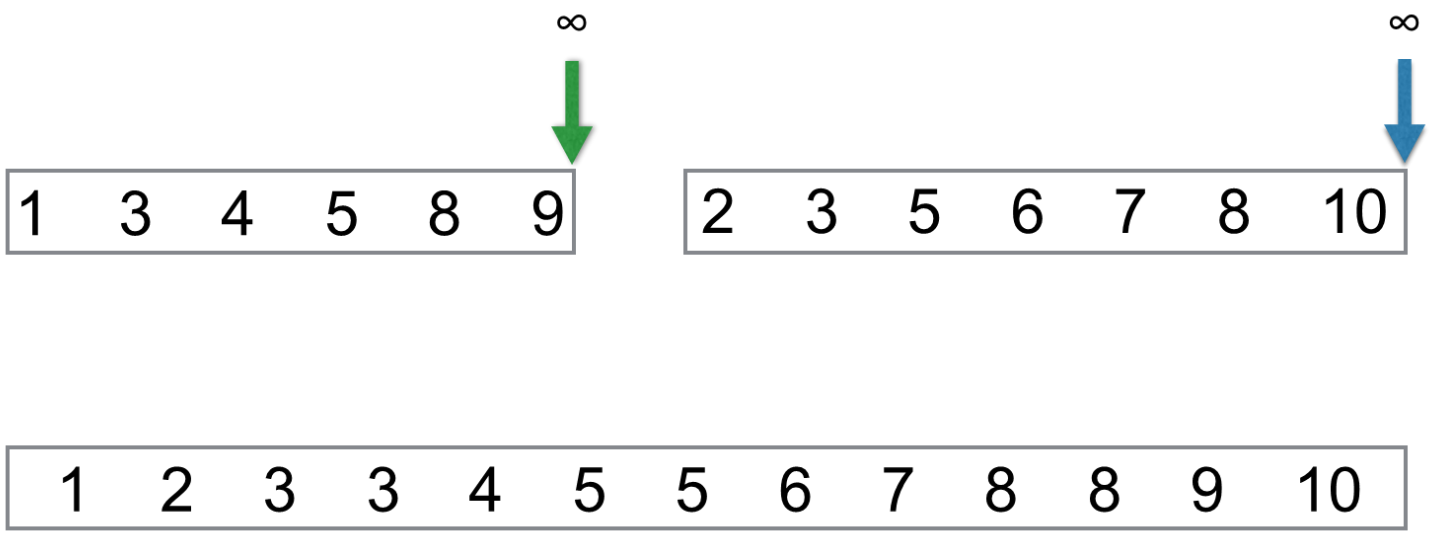
Сортировка слиянием



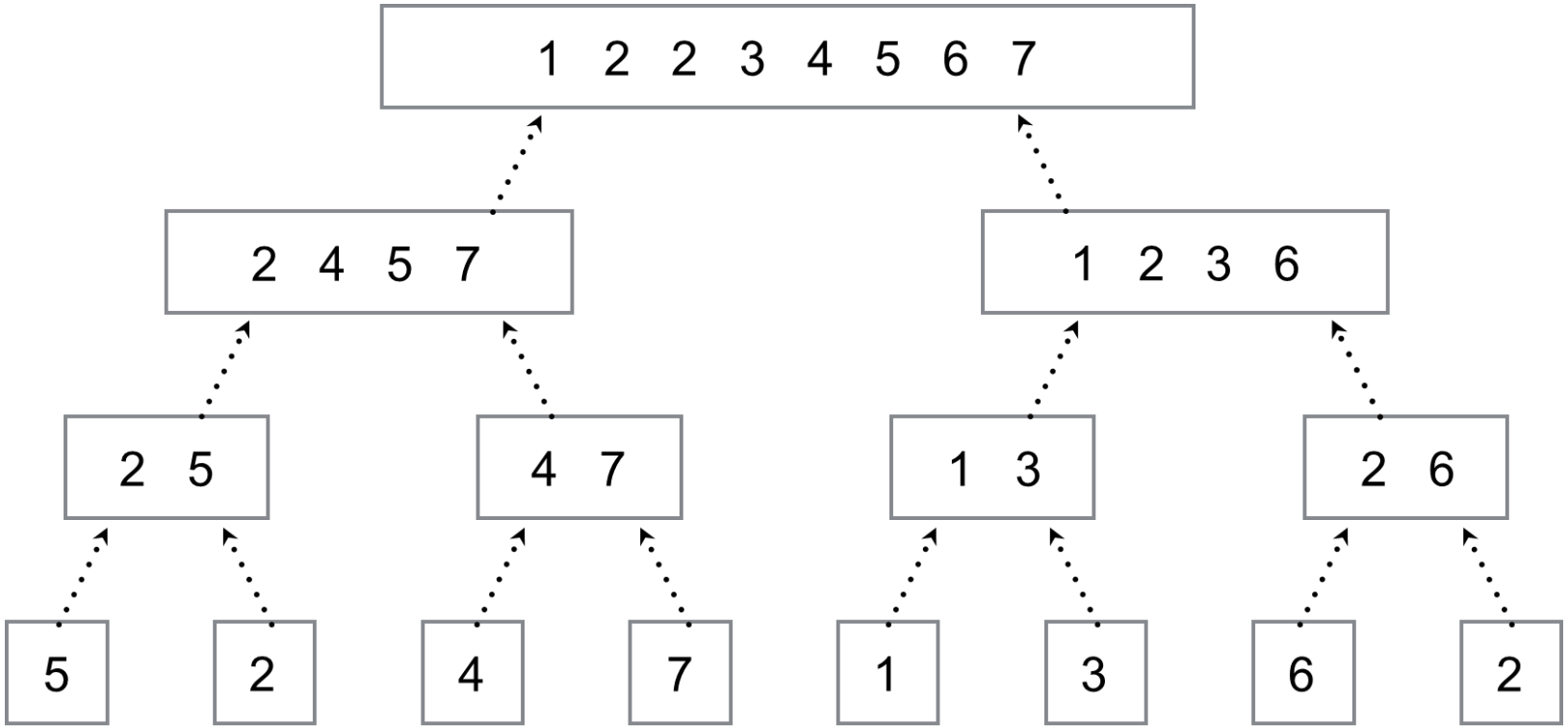
Сортировка слиянием



Сортировка слиянием



Сортировка слиянием



- $T(N) = \Theta(N) + 2T(N/2)$
- Можно считать, что $N = 2^k$, т.е. $k = \log N$; почему?

Сортировка слиянием

- Слияние

```
1. void merge(int * a, int * aux, int l, int m, int r) {
2.     int i = l;
3.     int j = m + 1;
4.     for (int k = l; k <= r; ++k) aux[k] = a[k];
5.
6.     for (int k = l; k <= r; ++k) {
7.         if (i > m) { a[k] = aux[j++]; continue; }
8.         if (j > r) { a[k] = aux[i++]; continue; }
9.         if (aux[j] < aux[i]) { a[k] = aux[j++]; }
10.        else                 { a[k] = aux[i++]; }
11.    }
12.}
```

- Стабильная; в C++ std::stable_sort – обычно сортировка слиянием
- Оценка сложности: $\Theta(N \log N)$; доп. память: $O(N)$

Сортировка слиянием

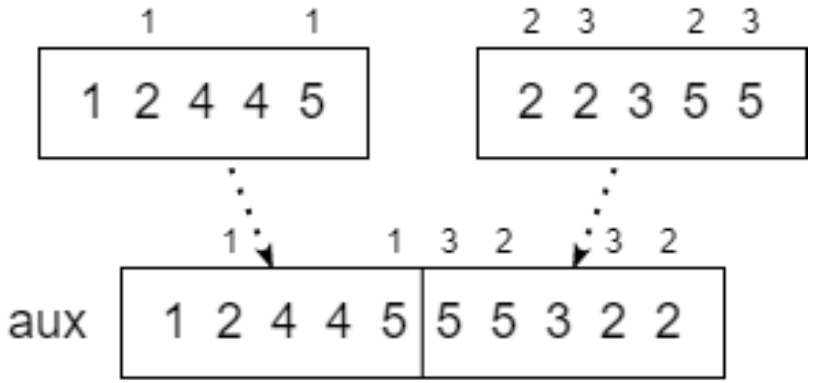
- Слияние через битонический (bitonic) порядок

```
1. void merge_bitonic(int * a, int * aux, int l, int m, int r) {
2.     int i, j;
3.     for (i = m + 1; i > l; --i) aux[i-1] = a[i-1];
4.     for (j = m; j < r; ++j) aux[r+m-j] = a[j+1];
5.     for (int k = l; k <= r; ++k) {
6.         if (aux[j] < aux[i]) {
7.             a[k] = aux[j--];
8.         } else {
9.             a[k] = aux[i++];
10.        }
11.    }
12. }
```

- Стабильность?

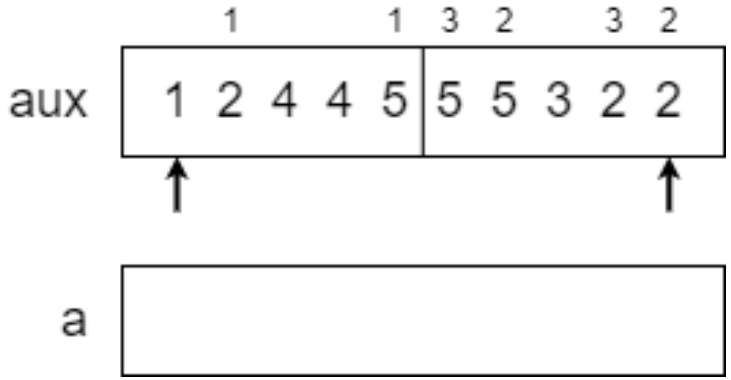
Слияние через битонический порядок


```
3.  for (i = m + 1; i > l; --i) aux[i-1] = a[i-1];
4.  for (j = m; j < r; ++j) aux[r+m-j] = a[j+1];
```



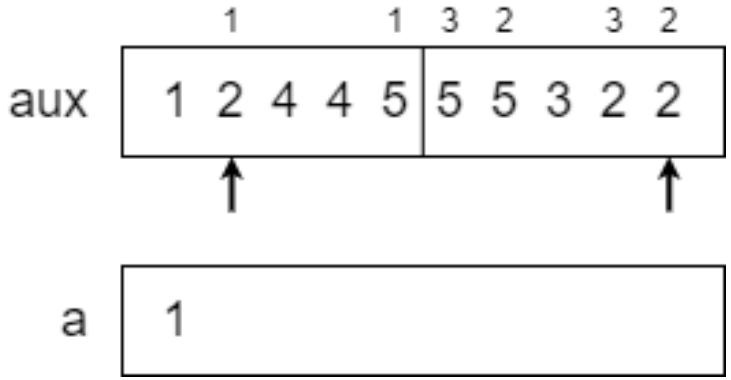
Слияние через битонический порядок

```
5.  for (int k = l; k <= r; ++k) {
6.      if (aux[j] < aux[i]) {
7.          a[k] = aux[j--];
8.      } else {
9.          a[k] = aux[i++];
10.     }
11. }
```



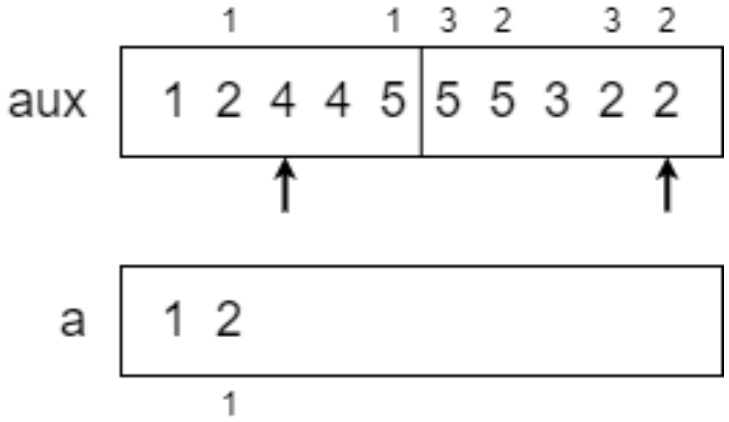
Слияние через битонический порядок

```
5.  for (int k = l; k <= r; ++k) {
6.      if (aux[j] < aux[i]) {
7.          a[k] = aux[j--];
8.      } else {
9.          a[k] = aux[i++];
10.     }
11. }
```



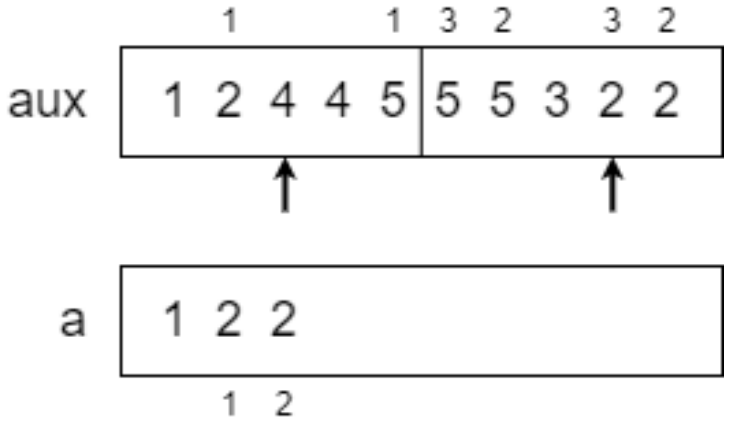
Слияние через битонический порядок

```
5.  for (int k = l; k <= r; ++k) {
6.      if (aux[j] < aux[i]) {
7.          a[k] = aux[j--];
8.      } else {
9.          a[k] = aux[i++];
10.     }
11. }
```



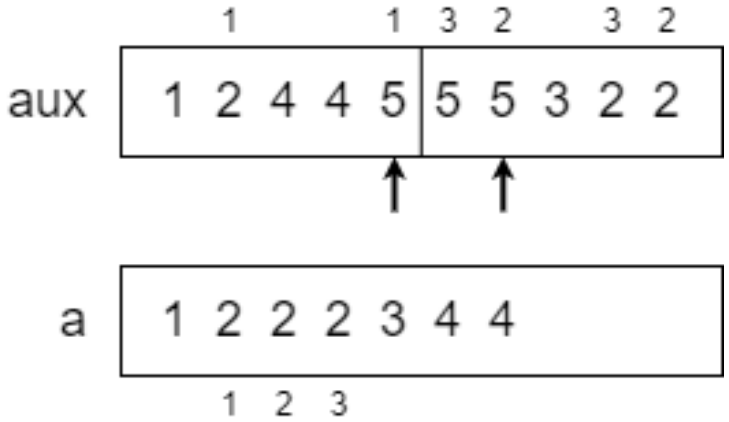
Слияние через битонический порядок

```
5.  for (int k = l; k <= r; ++k) {
6.      if (aux[j] < aux[i]) {
7.          a[k] = aux[j--];
8.      } else {
9.          a[k] = aux[i++];
10.     }
11. }
```



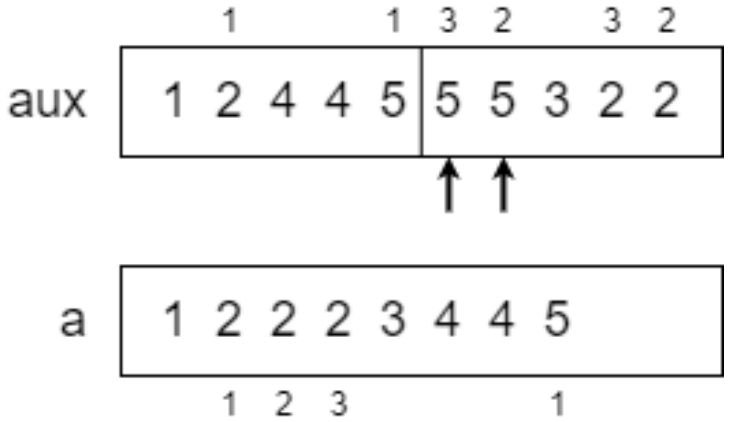
Слияние через битонический порядок

```
5.  for (int k = l; k <= r; ++k) {
6.      if (aux[j] < aux[i]) {
7.          a[k] = aux[j--];
8.      } else {
9.          a[k] = aux[i++];
10.     }
11. }
```



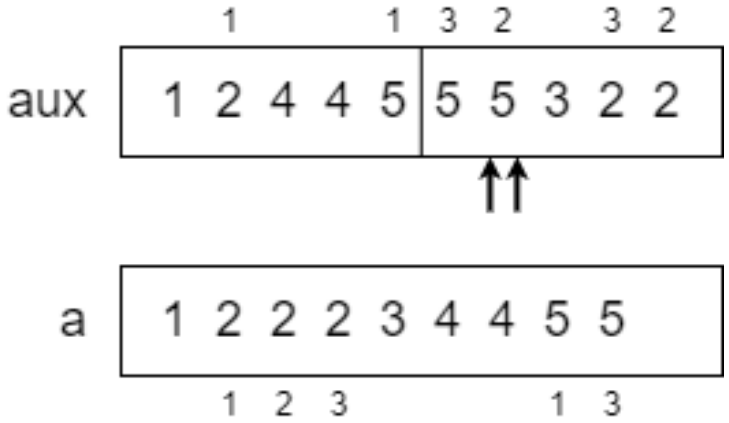
Слияние через битонический порядок

```
5.  for (int k = l; k <= r; ++k) {
6.      if (aux[j] < aux[i]) {
7.          a[k] = aux[j--];
8.      } else {
9.          a[k] = aux[i++];
10.     }
11. }
```



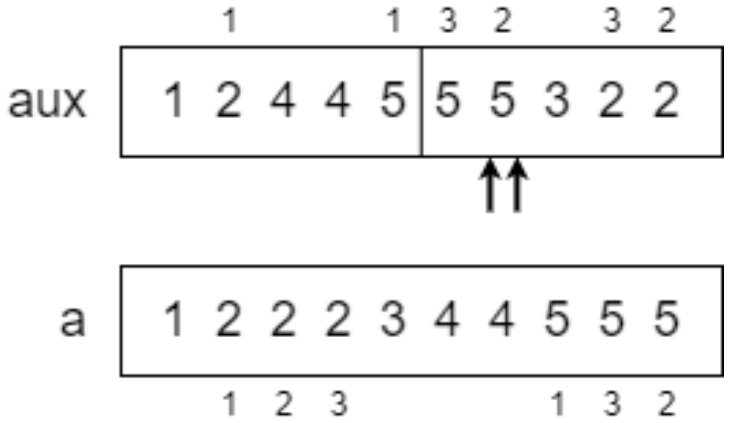
Слияние через битонический порядок

```
5.  for (int k = l; k <= r; ++k) {
6.      if (aux[j] < aux[i]) {
7.          a[k] = aux[j--];
8.      } else {
9.          a[k] = aux[i++];
10.     }
11. }
```



Слияние через битонический порядок

```
5.  for (int k = l; k <= r; ++k) {
6.      if (aux[j] < aux[i]) {
7.          a[k] = aux[j--];
8.      } else {
9.          a[k] = aux[i++];
10.     }
11. }
```



- Реализация нестабильна!

Сортировка слиянием

- Устранение копирования при merge

```
1. void sort(int * a, int N) {
2.     int * aux = new int[N];
3.     for (int k = 0; k < N; ++k) aux[k] = a[k];
4.     merge_sort(a, aux, 0, N-1);
5.     delete[] aux;
6. }
```

```
1. void merge_sort(int * a, int * aux, int l, int r) {
2.     if (l < r) {
3.         int m = (l + r) / 2;
4.         merge_sort(aux, a, l, m);
5.         merge_sort(aux, a, m + 1, r);
6.         merge(a, aux, l, m, r);
7.     }
8.}
```

Bottom-Up сортировка слиянием

ITIVITI

- Альтернатива Top-Down
- Нерекурсивная версия
- Пройдемся по массиву и сольем подмассивы размера 1
- Продолжим далее для подмассивов размера 2, 4, 8, …
- Обход в обратном порядке vs Обход по уровням снизу вверх
- Дерево рекурсии и последовательность слияний для Top-Down при $N = 5$: 1-1; 2-1; 1-1; 3-2 (всего 12 per element слияний)
- Последовательность слияний для Bottom-Up при $N = 5$: 1-1; 1-1; 2-2; 4-1 (всего 13 per element слияний)
- Немного медленнее рекурсивной Top-Down версии

Bottom-Up сортировка слиянием

```
1. void merge_sort_BU(int * a, int * aux, int l, int r) {
2.     for (int sz = 1; sz <= r-l; sz = sz+sz)
3.         for (int i = l; i <= r-sz; i += sz+sz)
4.             merge(a, aux, i, i+sz-1, min(i+sz+sz-1, r));
5.}
```

- Последовательность слияний при $N = 5$

```
sz = 1
merge(a, aux, 0, 0, 1)
merge(a, aux, 2, 2, 3)

sz = 2
merge(a, aux, 0, 1, 3)

sz = 4
merge(a, aux, 0, 3, 4)
```

Sort-Merge JOIN

ITIVITI

- Таблица "Customers"

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico
…	…	…	…

- Таблица "Orders"

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20
…	…	…

LET IT TRADE

https://www.w3schools.com/sql/

УНИВЕРСИТЕТ ИТМО

Sort-Merge JOIN

ITIVITI

- Пример (INNER) JOIN

```
1.SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
2.FROM Orders
3.INNER JOIN Customers
4.ON Orders.CustomerID=Customers.CustomerID;
```

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	1996-09-18
10309	Hungry Owl All-Night Grocers	1996-09-19
10310	The Big Cheese	1996-09-20
10311	Du monde entier	1996-09-20
10312	Die Wandernde Kuh	1996-09-23
…	…	…

- The worst case scenario is if the join attribute for all the tuples in both tables contain the same value

https://www.w3schools.com/sql/

Обновление таблицы

- «На проволоке» таблица лежит по строкам

	11	**22**	**33**
1	X	5	hello
2	Y	10	world
3	Z	15	string
4	X	11	string
5	X	13	itmo

2.11 = Z
1.22 = 0
2.33 = bye
1.22 = 1
1.11 = A

=>

1.11 = A
1.22 = 0
1.22 = 1
2.11 = Z
2.33 = bye

Большие объекты

- Дан массив больших объектов:
  - $B = \langle \text{obj}_1, \text{obj}_2, \ldots, \text{obj}_N \rangle$
- Как лучше отсортировать его?

Большие объекты

- Дан массив больших объектов:
  - $B = \langle \text{obj}_1, \text{obj}_2, \ldots, \text{obj}_N \rangle$
- Можно отсортировать указатели на объекты или индексы:
  - $I = \langle 1, 2, \ldots, N \rangle$
  - Сортируем I так, чтобы выполнялось следующее условие:
  - $B[I[1]] \le B[I[2]] \le \ldots \le B[I[N]]$

Число инверсий

- Число инверсий в массиве A [] равно числу пар (i , j), таких, что $i < j$ и $A[i] > A[j]$
- Используется для оценки численной схожести нескольких списков с рейтингом (оценками) пользователей (numerical similarity measure between ranked lists)
- Сколько инверсий в массиве: 1, 2, 3, 4, 5, 6, 7, 8 ?
- Сколько инверсий в массиве: 1, 4, 5, 3, 2, 6, 7, 8 ?
  - Как вы считали? Какая сложность подсчета?
  - Число инверсий определяется числом пересечений отрезков, соединяющих одинаковые элементы

Число инверсий

- Метод «разделяй и властвуй»
- Хотим $\Theta(N \log N)$
- Используем сортировку слиянием
  - Левая и правая инверсия (можем посчитать рекурсивно)
  - Расщепленная инверсия (как посчитать за линейное время ?)
- Если в массиве $A[]$ отсутствуют расщепленные инверсии, как соотносятся левый и правый подмассивы $A[]$?
- Число расщепленных инверсий, включающих элемент a_j из правого подмассива, совпадает с числом элементов в левом подмассиве на момент копирования a_j

Коэффициент ранговой корреляции Кендалла

ITIVITI

- С помощью сортировки и нехитрой математики Алиса легко сможет выбрать, с кем пойти в кино

ID	Фильм	Алиса	Вася	Саша
1	Звездные войны	6	5	1
2	Терминатор	5	6	2
3	Аватар	2	2	5
4	Властелин колец	3	1	4
5	Гарри Поттер	4	4	3
6	Титаник	1	3	6

$$\tau = \frac{\text{число пар в одном порядке} - \text{число инверсий}}{\text{число всех пар}}$$

LET IT TRADE

ITIVITI

LET IT TRADE

Спасибо за внимание!

УНИВЕРСИТЕТ ИТМО