Структуры и алгоритмы обработки данных, 2

Лекция 5

Сортировка (продолжение)

1. Нижняя граница задачи сортировки
2. Оптимальная сортировка
3. Поразрядная сортировка

Нижняя граница задачи сортировки

Вопрос: существует ли алгоритм сортировки, основанный на сравнениях и решающий задачу за меньшее число операций,

чем $O(n \log n)$ при любых входных данных?

Сортировка с минимальным числом сравнений

(избыточные сравнения не выполняются)

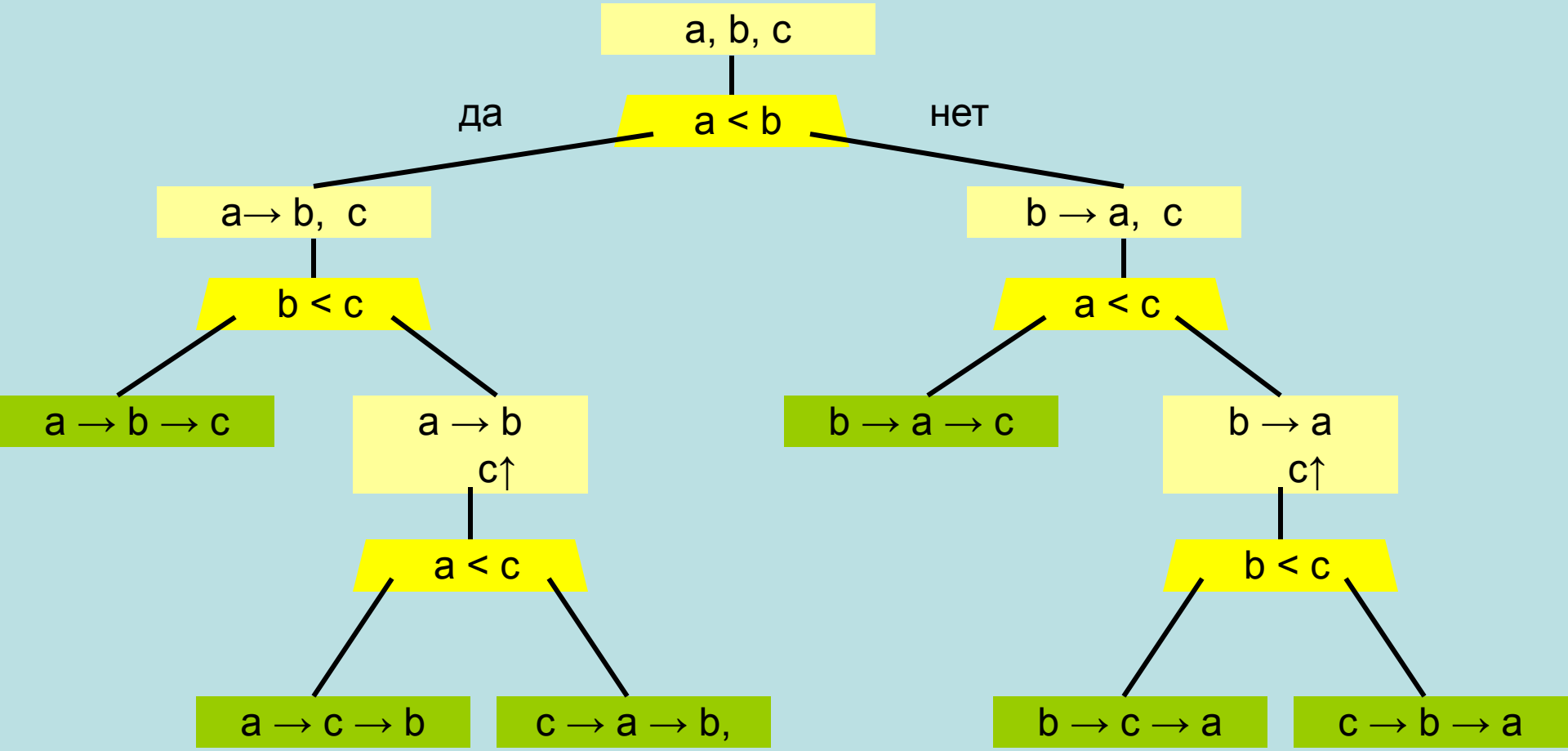
Деревья решений (сравнений)

Для простоты – все ключи различны.

Основная операция сравнения: $k_i < k_j$

Сортировка с минимальным числом сравнений

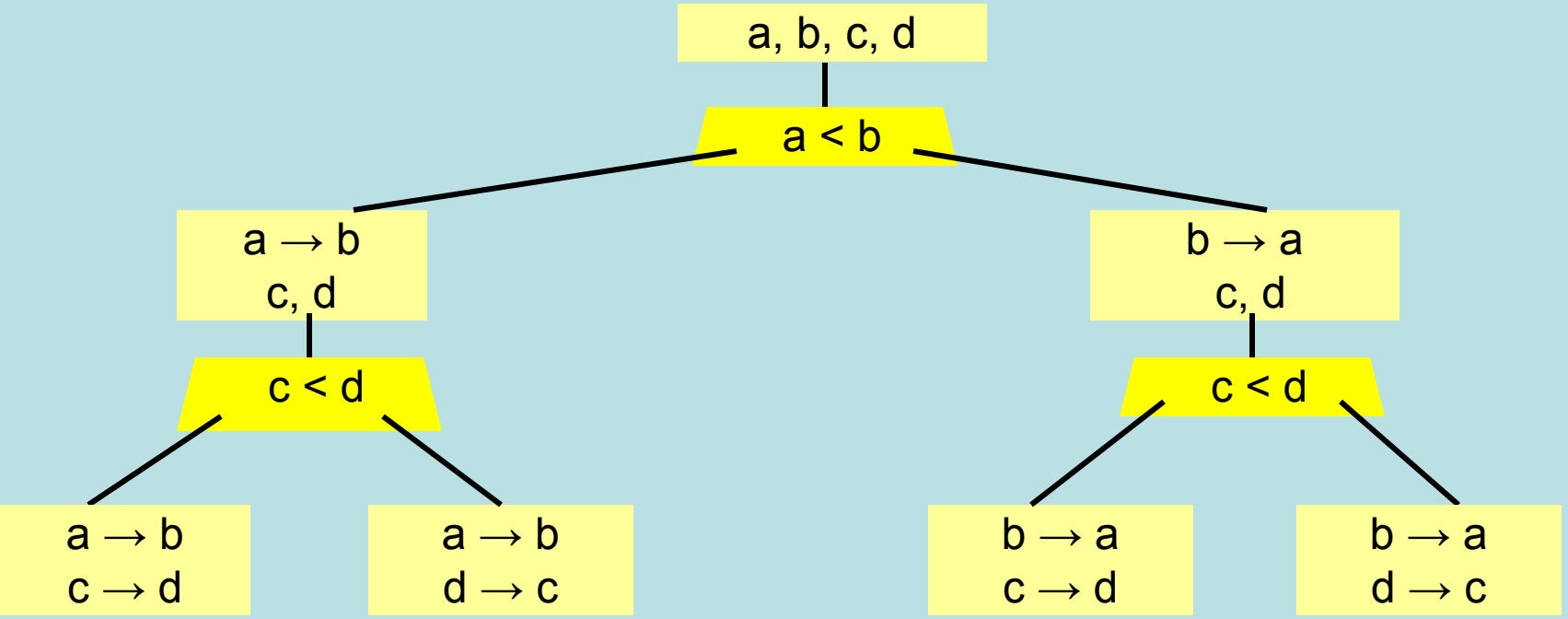
Деревья решений (сравнений)



n = 3, число перестановок и листьев = 3! = 6

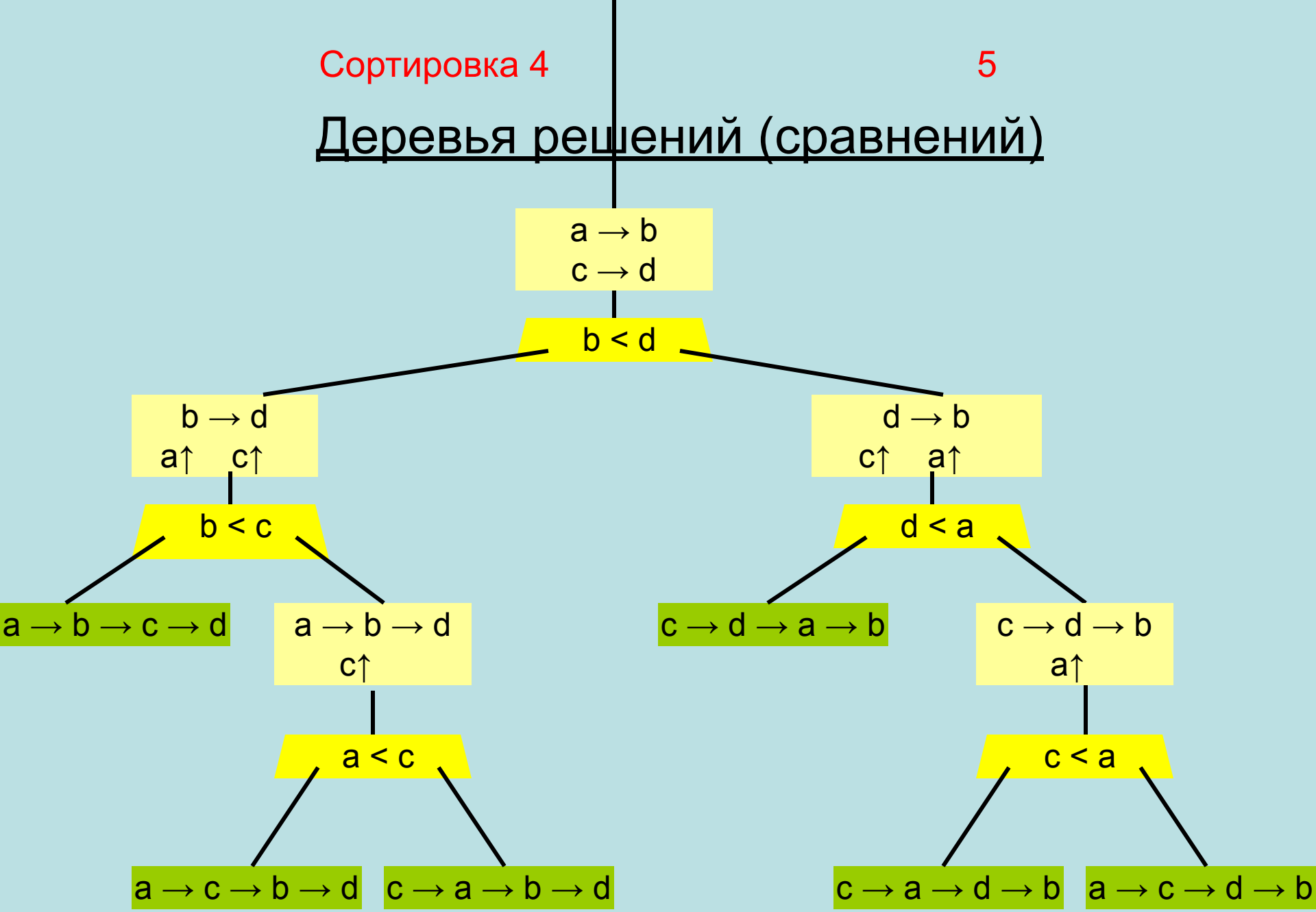
Сортировка с минимальным числом сравнений

Деревья решений (сравнений)



n = 4, число перестановок и листьев = 4! = 24

Деревья решений (сравнений)



n = 4, число перестановок и листьев = 4! = 24

Деревья решений (сравнений)

Пусть m высота дерева (максимальное число сравнений).

$2^m \geq$ число листьев $\geq n!$

(число листьев должно быть не менее числа исходов)

$m \geq \lceil \log_2 n! \rceil$

Итак, утверждение:

Для любого алгоритма сортировки, основанного на сравнениях, всегда можно подобрать такие исходные данные, что применение алгоритма потребует не менее $\lceil \log_2 n! \rceil$ шагов.

Формула Стирлинга

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$\lceil \log_2 n! \rceil \le \log_2 n! + 1$$

$$\lceil \log_2 n! \rceil = n \log_2 n - n \log_2 e + (1/2) \log_2 n + O(1)$$

Итак, **сложность задачи** сортировки есть $\Omega(n \log_2 n)$,

а алгоритмы быстрой сортировки и пирамидальной сортировки решают задачу за время $\Theta(n \log_2 n)$

Оптимальная сортировка

Бинарные вставки: $B(n) = \sum_{k=1}^{n} \lceil \log_2 k \rceil = n\lceil \log_2 n \rceil - 2^{\lceil \log_2 n \rceil} + 1$

n	2	3	4	5	6	7	8	9	12	16
$\lceil \log_2 n! \rceil$	1	3	5	7	10	13	16	19	29	45
$B(n)$	1	3	5	8	11	14	17	21	33	49

Пусть $S(n)$ – минимальное число сравнений, достаточное для сортировки. $S(n) \geq \lceil \log_2 n! \rceil$. ? $S(n) = \lceil \log_2 n! \rceil$?

Например, $S(5) = 7$ (см.далее), а $S(12) = 30$ (!)

Оптимальная сортировка

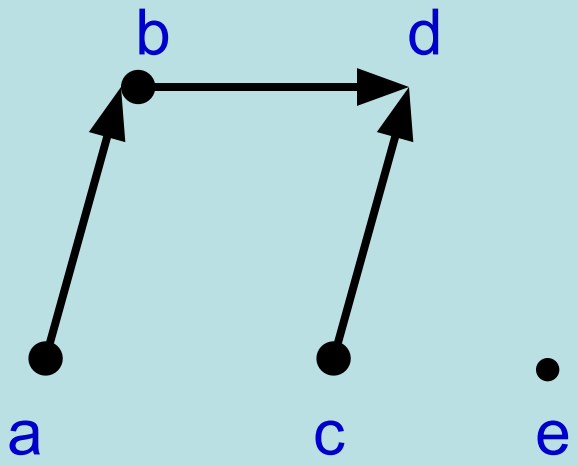
$S(5) = 7$?

k_1, k_2, k_3, k_4, k_5

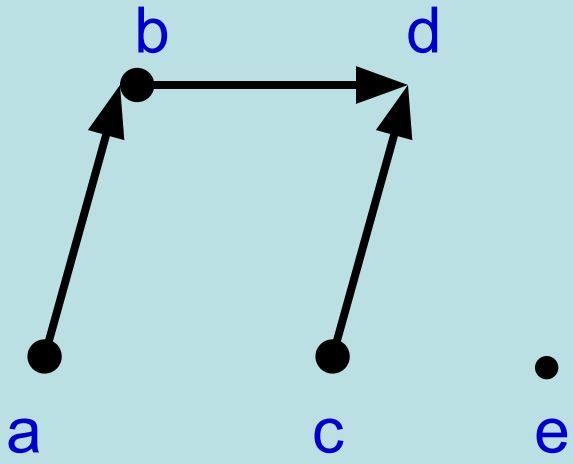
1-е сравнение: $k_1 : k_2 \rightarrow \max(k_1, k_2)$

2-е сравнение: $k_3 : k_4 \rightarrow \max(k_3, k_4)$

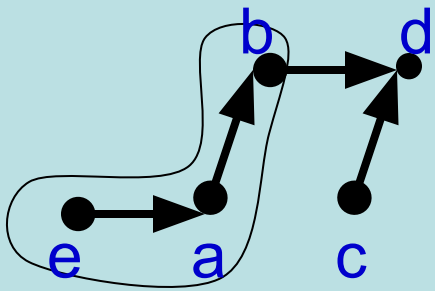
3-е сравнение: $\max(k_1, k_2) : \max(k_3, k_4)$



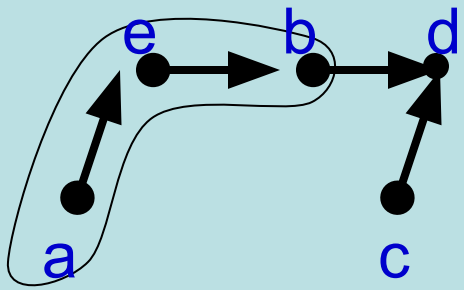
Оптимальная сортировка



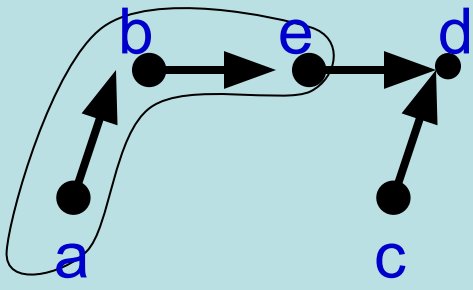
4 и 5-е сравнения: e среди a<b<d



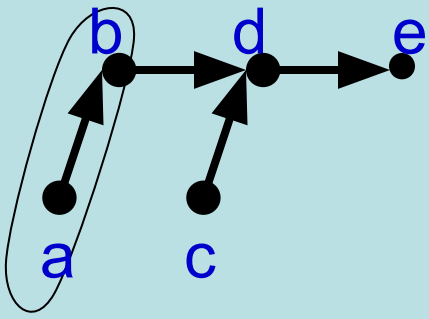
2 сравнения



2 сравнения



2 сравнения



1 или 2 сравнения

Поразрядная сортировка

(Распределяющая сортировка)

Пример 1 (с картами).

Карта = (масть, достоинство)

Масть: ♣ - трефа, ♦ - бубна, ♥ - черва, ♠ - пика

Достоинство: 2, 3, 4, 5, 6, 7, 8, 9, 10, В, Д, К, Т

Порядок по масти: ♣ < ♦ < ♥ < ♠.

Порядок по достоинству: 2<3<4<5<6<7<8<9<10<В<Д<К<Т

Порядок карт *лексикографический*:

♣2<♣3<♣4<♣5<♣6<♣7<♣8<♣9<♣10<♣В<♣Д<♣К<♣Т<

♦2<♦3<♦4<♦5<♦6<♦7<♦8<♦9<♦10<♦В<♦Д<♦К<♦Т<

♥2<♥3<♥4<♥5<♥6<♥7<♥8<♥9<♥10<♥В<♥Д<♥К<♥Т<

♠2<♠3<♠4<♠5<♠6<♠7<♠8<♠9<♠10<♠В<♠Д<♠К<♠Т

Поразрядная сортировка

Сортировка колоды карт в лексикографическом порядке:

1 *шаг* – распределяем по достоинству (раскладываем) на кучки («лицом» вверх)

2	3	4	5	6	7	8	9	10	В	Д	К	Т

Затем складываем кучки друг на друга (последовательно кучки большего достоинства сверху).

2 шаг – держим колоду рубашкой вниз и распределяем на кучки по масти (карты кладем лицом вверх).

При этом в каждой «кучке по масти» карты лягут в порядке возрастания достоинства вниз кучки.

Поразрядная сортировка

В итоге получим 4 кучки с верхними картами: ♣ < ♦ < ♥ < ♠

Заключительный шаг – последовательно складываем, начиная справа, левую кучку на правую.

Задание: найти колоду карт и поупражняться

Поразрядная сортировка

Пример 2 (с числами).

Рассмотрим целые (положительные) *трехразрядные* числа

в позиционной **шестиричной** системе счисления,

т.е. каждый разряд содержит цифры 0, 1, 2, 3, 4, 5

Например, пусть дана последовательность из 9 чисел

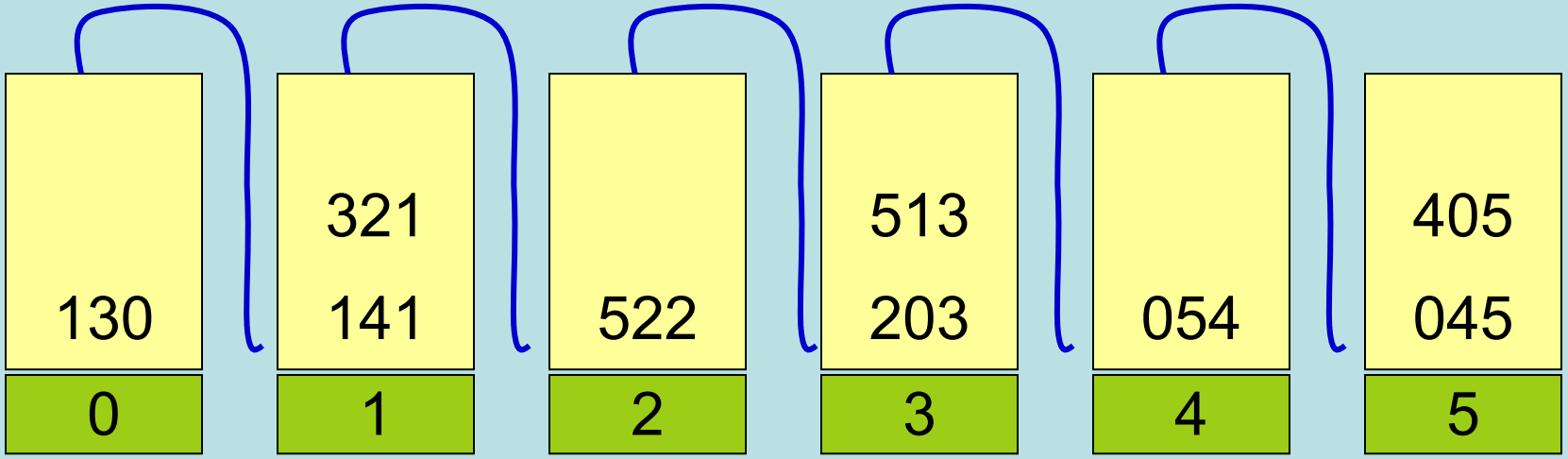
203, 045, 141, 405, 321, 522, 130, 054, 513

1 *шаг* – распределяем числа последовательности сначала по младшей цифре по ящикам с «именами»: «0», «1», «2», «3», «4», «5».

Ящики заполняются снизу вверх.

Поразрядная сортировка

203, 045, 141, 405, 321, 522, 130, 054, 513



Сцепляем последовательно слева направо содержимое ящиков в одну последовательность, так, что нижнее число правого ящика следует за верхним числом левого соседнего ящика (внутри ящиков последовательность расположена снизу вверх). Таким образом, получим последовательность

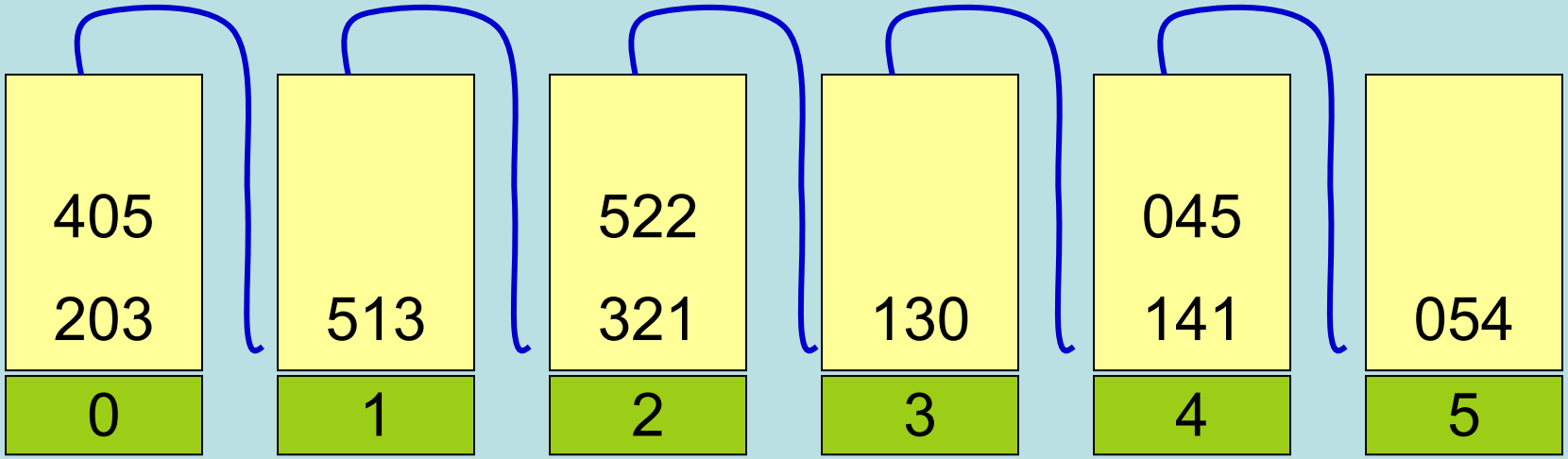
130, 141, 321, 522, 203, 513, 054, 045, 405.

Заметим, что младшие цифры этой последовательности упорядочены по неубыванию.

Поразрядная сортировка

130, 141, 321, 522, 203, 513, 054, 045, 405

2 шаг – аналогичным образом распределяем по ящикам числа полученной последовательности *по средней цифре.*



Повторяя сцепление содержимого ящиков, получим последовательность

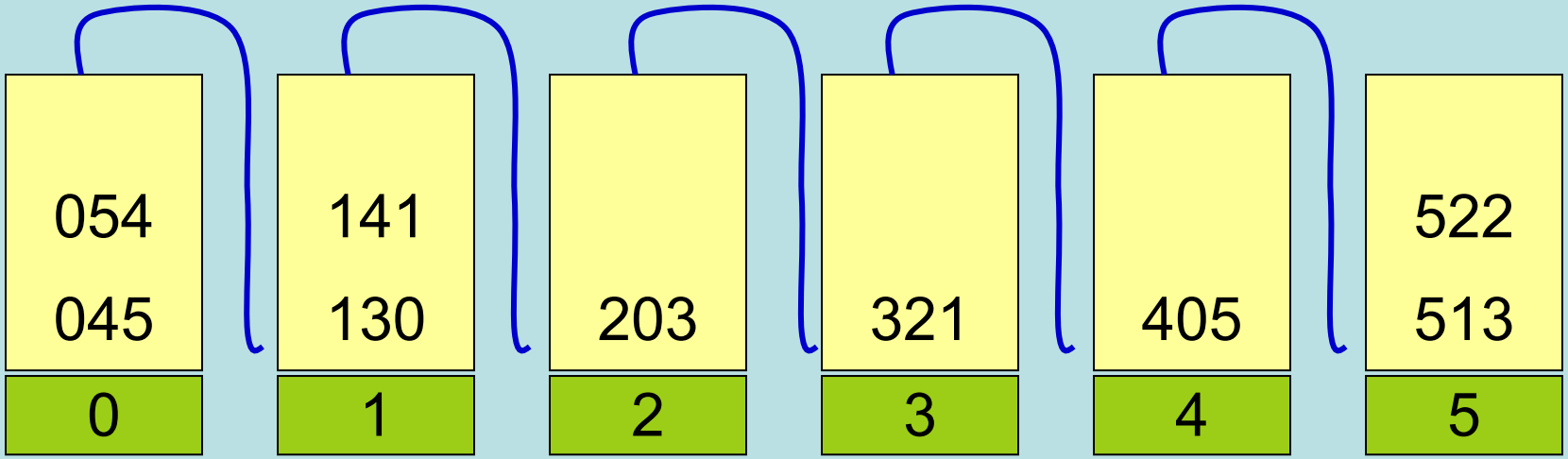
203, 405, 513, 321, 522, 130, 141, 045, 054,

упорядоченную по двум младшим цифрам.

Поразрядная сортировка

203, 405, 513, 321, 522, 130, 141, 045, 054

3 шаг – аналогичным образом распределяем по ящикам числа полученной последовательности *по старшей цифре.*



Повторяя сцепление содержимого ящиков, получим упорядоченную последовательность

045, 054, 130, 141, 203, 321, 405, 513, 522.

Алгоритм поразрядной сортировки

В общем случае даны числа (ключи):

$$x_1, x_2, x_3, \ldots, x_n.$$

Каждый ключ состоит из p разрядов:

$$(\forall\ i \in 1..n: x_i = (x_i(p), x_i(p-1), \ldots, x_i(1))),$$

а каждый q-й разряд представлен цифрой $x_i(q) \in 0..r-1$.

Тогда

$(x_i < x_j) \leftrightarrow$

$(\exists\ t \in 1..p: (\forall\ q \in t+1..p: x_i(q) = x_j(q)) \ \&\ (x_i(t) < x_j(t))$)).

Алгоритм поразрядной сортировки

Для представления «ящиков» и их сцепления в новую последовательность используем ***очереди***:

- Q – общая очередь (исходная, промежуточные и результирующая последовательности);
- $q[0], q[1], \ldots, q[r-1]$ – очереди по каждой r-ичной цифре («ящики»).

Алгоритм поразрядной сортировки

Алгоритм:

Create(*Q*); *Create*(*q*[*]);
for *j* := 1 **to** *p* **do**
  for *k* := 0 **to** *r* − 1 **do** *Empty*(*q*[*k*]) **od**;
  while not *Null*(*Q*) **do**
    x ← *Q*;
    Пусть *x* = (*x*(*p*), *x*(*p*−1),…, *x*(1)), тогда
    q[*x*(*j*)] ← *x*
  od {**while**};
  Q ← сцепление(*q*[0], *q*[1],…, *q*[*r* − 1])
od {**for** *j*}

Алгоритм поразрядной сортировки

Подсчитаем количество операций.

Внешний цикл по разрядам выполняется p раз.

Каждая *итерация* этого цикла для каждого из n чисел выполняет извлечение и запись в очередь, т.е. всего $2n$ таких операций.

Кроме того, на каждой итерации происходит $r - 1$ сцепление очередей.

Таким образом на одной итерации требуется $2n + r - 1$ операций,

а всего по всем разрядам $O(p\,n + p\,r)$ операций.

Сравнение с $O(n \log n)$!...

КОНЕЦ ЛЕКЦИИ