Алгоритмы на строках
Сортировка строк

Василий Николенко

АО «НИЦ СПб ЭТУ»

2017

Основные определения

Основные определения

Строка S — это произвольная конечная последовательность символов (конечного алфавита).

Основные определения

Строка S — это произвольная конечная последовательность символов (конечного алфавита).
Длина строки S — количество символов в последовательности ($|S|$).

Основные определения

Строка S — это произвольная конечная последовательность символов (конечного алфавита).
Длина строки S — количество символов в последовательности (|S|).
$S[i..j]$ — *сплошная подстрока строки S*, которая начинается в позиции i и заканчивается в позиции j строки S.

Основные определения

Строка S — это произвольная конечная последовательность символов (конечного алфавита).
Длина строки S — количество символов в последовательности (|S|).
$S[i..j]$ — *сплошная подстрока строки S*, которая начинается в позиции i и заканчивается в позиции j строки S.
S[0..i] — *префикс строки S*.

Основные определения

Строка S — это произвольная конечная последовательность символов (конечного алфавита).
Длина строки S — количество символов в последовательности (|S|).
$S[i..j]$ — *сплошная подстрока строки S*, которая начинается в позиции i и заканчивается в позиции j строки S.
S[0..i] — *префикс строки S*.
S[i..|S|-1] — *суффикс строки S*.

Основные определения

Строка S — это произвольная конечная последовательность символов (конечного алфавита).
Длина строки S — количество символов в последовательности (|S|).
$S[i..j]$ — *сплошная подстрока строки S*, которая начинается в позиции i и заканчивается в позиции j строки *S*.
S[0..i] — *префикс строки S*.
S[i..|S|-1] — *суффикс строки S*.
Собственные префикс, суффикс, подстрока S — это, соответственно префикс, суффикс и подстрока, которые не сопадают с S и не пустые.

Основные определения

Строка S — это произвольная конечная последовательность символов (конечного алфавита).
Длина строки S — количество символов в последовательности (|S|).
$S[i..j]$ — *сплошная подстрока строки S*, которая начинается в позиции i и заканчивается в позиции j строки *S*.
S[0..i] — *префикс строки S*.
S[i..|S|-1] — *суффикс строки S*.
Собственные префикс, суффикс, подстрока S — это, соответственно префикс, суффикс и подстрока, которые не сопадают с S и не пустые.

Пример

абракадабра —это строка длины 11
брак —это подстрока
абрака —это префикс
кадабра —это суффикс

Задача сортировки

Задача сортировки

Пусть есть n-записей

$$R_1, R_1, \ldots, R_n$$

Каждая запись R_i имеет ключ k_i.

Задача сортировки

Пусть есть n-записей

$$R_1, R_1, \ldots, R_n$$

Каждая запись R_i имеет ключ k_i.
На множестве ключей введено отношение линейного порядка($\leq$), т. е. отношение :

- Рефлексивно: $\forall a \in K : a \leq a$.
- Антисимметрично: $\forall a, b \in K$:если $a \leq b$ и $b \leq a$, то $a = b$.
- Транзитивно: $\forall a, b, c \in K$:если $a \leq b$ и $b \leq c$, то $a \leq c$.
- Обладает свойством: $\forall a \in K \ \forall b \in K$ либо $a \leq b$, либо $b \leq a$.

Задача сортировки

Пусть есть n-записей

$$R_1, R_1, \ldots, R_n$$

Каждая запись R_i имеет ключ k_i.
На множестве ключей введено отношение линейного порядка($\leq$), т. е. отношение :

- Рефлексивно: $\forall a \in K : a \leq a$.
- Антисимметрично: $\forall a, b \in K$:если $a \leq b$ и $b \leq a$, то $a = b$.
- Транзитивно: $\forall a, b, c \in K$:если $a \leq b$ и $b \leq c$, то $a \leq c$.
- Обладает свойством: $\forall a \in K \ \forall b \in K$ либо $a \leq b$, либо $b \leq a$.

Задача сортировки — найти такую перестановку π индексов $\{1, 2, \ldots, n\}$, что:

$$k_{\pi(0)} \leq k_{\pi(1)} \leq \ldots \leq k_{\pi(n)}$$

Задача сортировки

Пусть есть n-записей

$$R_1, R_1, \ldots, R_n$$

Каждая запись R_i имеет ключ k_i.
На множестве ключей введено отношение линейного порядка($\leq$), т. е. отношение :

- Рефлексивно: $\forall a \in K : a \leq a$.
- Антисимметрично: $\forall a, b \in K$:если $a \leq b$ и $b \leq a$, то $a = b$.
- Транзитивно: $\forall a, b, c \in K$:если $a \leq b$ и $b \leq c$, то $a \leq c$.
- Обладает свойством: $\forall a \in K \ \forall b \in K$ либо $a \leq b$, либо $b \leq a$.

Задача сортировки — найти такую перестановку π индексов $\{1, 2, \ldots, n\}$, что:

$$k_{\pi(0)} \leq k_{\pi(1)} \leq \ldots \leq k_{\pi(n)}$$

Сортировка называется *устойчивой*, если записи с равными ключами, остаются в прежнем порядке.

Сложность решения задачи сортировки

Любой алгоритм сортировки сравнением в наихудшем случае требует выполнения $\Omega(n \log n)$ сравнений.

Сложность решения задачи сортировки

Любой алгоритм сортировки сравнением в наихудшем случае требует выполнения $\Omega(n \log n)$ сравнений.

Способы сортировки строк

- Сортировка по младшим разрядам (least-significant-digit - **LSD**).
  Просматривает символы в ключах справа налево.
  Рекомендуется, когда все ключи имеют одинаковую длину.
- Сортировка по старшим разрядам (most-significant-digit - **MSD**).
  Просматривает символы в ключах слево направо, начиная с наиболее значащего символа.
- Трехчастная быстрая сортировка строк.
  Модификация трехчастной быстрой сортировки общего назначения.

Распределяющий подсчет (сортировка подсчетом, counting sort)

Пусть множество ключей *конечно* ($|K| = k$).

Распределяющий подсчет (сортировка подсчетом, counting sort)

Пусть множество ключей *конечно* ($|K| = k$).

Идея

Для каждого входного элемента R_i с ключом k_i определить количество элементов, у которых ключи меньше k_i. На основе этой информации каждый R_i можно поместить в нужную позицию выходного массива.

Распределяющий подсчет. Пример

Входные данные

(фамилия, полет)	
Беляев П. И.	8
Быковский В. Ф.	5
Гагарин Ю. А.	1
Егоров Б. Б.	7
Комаров В. М.	7
Леонов А. А.	8
Николаев А. Г.	3
Титов Г. С.	2
Феоктистов К. П.	7

Распределяющий подсчет. Пример

Входные данные (фамилия, полет)		**Упорядоченный результат** (по полету)	
Беляев П. И.	8	Гагарин Ю. А.	1
Быковский В. Ф.	5	Титов Г. С.	2
Гагарин Ю. А.	1	Николаев А. Г.	3
Егоров Б. Б.	7	Быковский В. Ф.	5
Комаров В. М.	7	Егоров Б. Б.	7
Леонов А. А.	8	Комаров В. М.	7
Николаев А. Г.	3	Феоктистов К. П.	7
Титов Г. С.	2	Беляев П. И.	8
Феоктистов К. П.	7	Леонов А. А.	8

Распределяющий подсчет. Шаги

1. Вычисление счетчиков повторений
2. Преобразование счетчиков в индексы
3. Распределение данных
4. Копирование данных в исходный массив

Вычисление счетчиков повторений

0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0

Вычисление счетчиков повторений

Беляев П. И. 8

0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	1

Вычисление счетчиков повторений

Быковский В. Ф. 5

0	1	2	3	4	5	6	7
0	0	0	0	1	0	0	1

Вычисление счетчиков повторений

Гагарин Ю. А. 1

0	1	2	3	4	5	6	7
1	0	0	0	1	0	0	1

Вычисление счетчиков повторений

Егоров Б. Б. 7

0	1	2	3	4	5	6	7
1	0	0	0	1	0	1	1

Вычисление счетчиков повторений

Комаров В. М. 7

0	1	2	3	4	5	6	7
1	0	0	0	1	0	2	1

Вычисление счетчиков повторений

Леонов А. А. 8

0	1	2	3	4	5	6	7
1	0	0	0	1	0	2	2

Вычисление счетчиков повторений

Николаев А. Г. 3

0	1	2	3	4	5	6	7
1	0	1	0	1	0	2	2

Вычисление счетчиков повторений

Титов Г. С. 2

0	1	2	3	4	5	6	7
1	1	1	0	1	0	2	2

Вычисление счетчиков повторений

Феоктистов К. П. 7

0	1	2	3	4	5	6	7
1	1	1	0	1	0	3	2

Преобразование счетчиков в индексы

0	1	2	3	4	5	6	7
1	1	1	0	1	0	3	2

Преобразование счетчиков в индексы

0	1	2	3	4	5	6	7
1	2	1	0	1	0	3	2

Преобразование счетчиков в индексы

0	1	2	3	4	5	6	7
1	2	3	0	1	0	3	2

Преобразование счетчиков в индексы

0	1	2	3	4	5	6	7
1	2	3	3	1	0	3	2

Преобразование счетчиков в индексы

0	1	2	3	4	5	6	7
1	2	3	3	4	0	3	2

Преобразование счетчиков в индексы

0	1	2	3	4	5	6	7
1	2	3	3	4	4	3	2

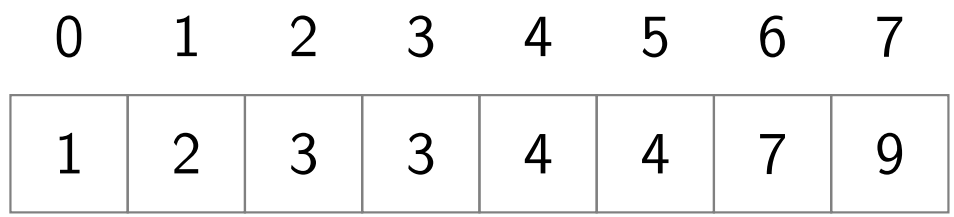
Преобразование счетчиков в индексы

0	1	2	3	4	5	6	7
1	2	3	3	4	4	7	2

Преобразование счетчиков в индексы

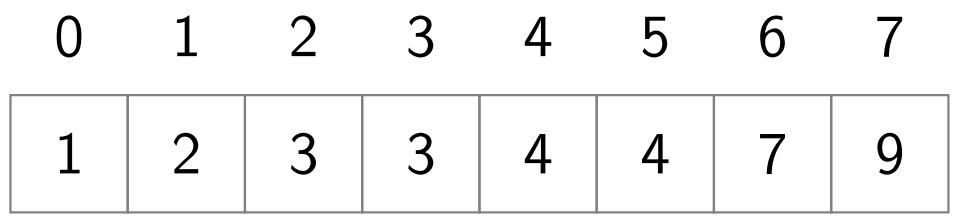
0	1	2	3	4	5	6	7
1	2	3	3	4	4	7	9

Распределение данных



Распределение данных

Феоктистов К. П. 7



Распределение данных

6. Феоктистов К. П. 7

Распределение данных

Титов Г. С. 2

0	1	2	3	4	5	6	7
1	2	3	3	4	4	6	9

6. Феоктистов К. П. 7

Распределение данных

1. Титов Г. С. 2

6. Феоктистов К. П. 7

Распределение данных

Николаев А. Г. 3

1. Титов Г. С. 2

6. Феоктистов К. П. 7

0	1	2	3	4	5	6	7
1	1	3	3	4	4	6	9

Распределение данных

1. Титов Г. С. 2
2. Николаев А. Г. 3

0	1	2	3	4	5	6	7
1	1	2	3	4	4	6	9

6. Феоктистов К. П. 7

Распределение данных

Леонов А. А. 8

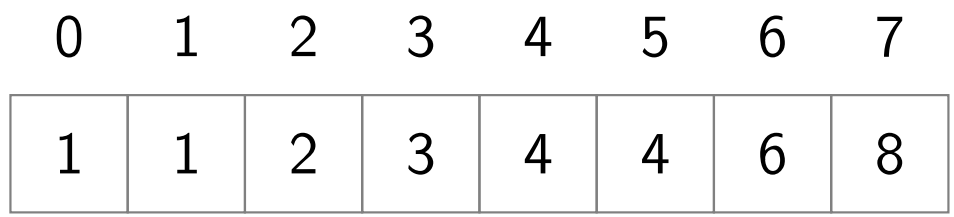
1. Титов Г. С. 2
2. Николаев А. Г. 3

0	1	2	3	4	5	6	7
1	1	2	3	4	4	6	9

6. Феоктистов К. П. 7

Распределение данных

1. Титов Г. С. 2
2. Николаев А. Г. 3



6. Феоктистов К. П. 7

8. Леонов А. А. 8

Распределение данных

Комаров В. М. 7

1. Титов Г. С. 2
2. Николаев А. Г. 3

0	1	2	3	4	5	6	7
1	1	2	3	4	4	6	8

6. Феоктистов К. П. 7

8. Леонов А. А. 8

Распределение данных

1. Титов Г. С. 2
2. Николаев А. Г. 3

0	1	2	3	4	5	6	7
1	1	2	3	4	3	6	8

5. Комаров В. М. 7
6. Феоктистов К. П. 7

8. Леонов А. А. 8

Распределение данных

Егоров Б. Б. 7

1. Титов Г. С. 2
2. Николаев А. Г. 3

0	1	2	3	4	5	6	7
1	1	2	3	4	3	6	8

5. Комаров В. М. 7
6. Феоктистов К. П. 7

8. Леонов А. А. 8

Распределение данных

1. Титов Г. С. 2
2. Николаев А. Г. 3

4. Егоров Б. Б. 7
5. Комаров В. М. 7
6. Феоктистов К. П. 7

8. Леонов А. А. 8

0	1	2	3	4	5	6	7
1	1	2	3	4	3	5	8

Распределение данных

Гагарин Ю. А. 1

1. Титов Г. С. 2
2. Николаев А. Г. 3

4. Егоров Б. Б. 7
5. Комаров В. М. 7
6. Феоктистов К. П. 7

8. Леонов А. А. 8

0	1	2	3	4	5	6	7
1	1	2	3	4	3	5	8

Распределение данных

0. Гагарин Ю. А. 1
1. Титов Г. С. 2
2. Николаев А. Г. 3

4. Егоров Б. Б. 7
5. Комаров В. М. 7
6. Феоктистов К. П. 7

8. Леонов А. А. 8

0	1	2	3	4	5	6	7
0	1	2	3	4	3	5	8

Распределение данных

Быковский В. Ф. 5

0. Гагарин Ю. А. 1
1. Титов Г. С. 2
2. Николаев А. Г. 3

4. Егоров Б. Б. 7
5. Комаров В. М. 7
6. Феоктистов К. П. 7

8. Леонов А. А. 8

0	1	2	3	4	5	6	7
0	1	2	3	4	3	5	8

Распределение данных

0. Гагарин Ю. А. 1
1. Титов Г. С. 2
2. Николаев А. Г. 3
3. Быковский В. Ф. 5
4. Егоров Б. Б. 7
5. Комаров В. М. 7
6. Феоктистов К. П. 7

8. Леонов А. А. 8

0	1	2	3	4	5	6	7
0	1	2	3	3	3	5	8

Распределение данных

Беляев П. И. 8

0. Гагарин Ю. А. 1
1. Титов Г. С. 2
2. Николаев А. Г. 3
3. Быковский В. Ф. 5
4. Егоров Б. Б. 7
5. Комаров В. М. 7
6. Феоктистов К. П. 7

8. Леонов А. А. 8

0	1	2	3	4	5	6	7
0	1	2	3	3	3	5	8

Распределение данных

0. Гагарин Ю. А. 1
1. Титов Г. С. 2
2. Николаев А. Г. 3
3. Быковский В. Ф. 5
4. Егоров Б. Б. 7
5. Комаров В. М. 7
6. Феоктистов К. П. 7
7. Беляев П. И. 8
8. Леонов А. А. 8

0	1	2	3	4	5	6	7
0	1	2	3	2	3	5	8

```
1: procedure COUNTING-SORT(records, keysCard)
2:     n ← records.length
3:     c[0..keysCard − 1] - массив, инициализированный 0
4:     for j ← 0, n − 1 do
5:         c[records[j].key()] ← c[records[j].key()] + 1
6:     end for
7:     //c[i]- содержит число ключей, равных i
8:     for i ← 1, keysCard − 1 do
9:         c[i] ← c[i] + c[i − 1]
10:    end for
11:    // c[i]- содержит число ключей, НЕ превышающих i
12:    tmp[0..n − 1]
13:    for j ← n − 1, 0 do
14:        tmp[c[records[j].key()]] ← records[j]
15:        c[records[j].key()] ← c[records[j].key()] − 1
16:    end for
17:    for i ← 0, n − 1 do
18:        records[i] ← tmp[i]
19:    end for
20: end procedure
```

Распределяющий подсчет. Анализ

Для устойчивой сортировки N записей, ключи которых представляют собой целые числа от 0 до $K-1$, распределяющему подсчету требуется $\Theta(N+K)$ обращений к массиву.

Распределяющий подсчет. Анализ

Для устойчивой сортировки N записей, ключи которых представляют собой целые числа от 0 до $K-1$, распределяющему подсчету требуется $\Theta(N+K)$ обращений к массиву.

Доказательство.

Инициализация массивов — $\Theta(N), \Theta(K)$
Первый цикл увеличивает счетчик для каждого из N элементов — $\Theta(N)$
Второй выполняет K сложений — $\Theta(K)$
Третий выполняет N уменьшений счетчиков и N перемещений данных — $\Theta(N)$
Четвертый цикл выполняет N перемещений данных — $\Theta(N)$
Т. о.: $\Theta(N+K)$ □

Распределяющий подсчет. Анализ

Для устойчивой сортировки N записей, ключи которых представляют собой целые числа от 0 до $K-1$, распределяющему подсчету требуется $\Theta(N+K)$ обращений к массиву.

Доказательство.

Инициализация массивов — $\Theta(N), \Theta(K)$
Первый цикл увеличивает счетчик для каждого из N элементов — $\Theta(N)$
Второй выполняет K сложений — $\Theta(K)$
Третий выполняет N уменьшений счетчиков и N перемещений данных — $\Theta(N)$
Четвертый цикл выполняет N перемещений данных — $\Theta(N)$
Т. о.: $\Theta(N+K)$ □

На практике $K = O(N)$, в этом случае $\boldsymbol{\Theta(N)}$.

LSD-сортировка строк

Необходимо отсортировать N строк длины L каждая. Символы строк принадлежат алфавиту $\mathscr{A}$ размера A ($A = |\mathscr{A}|$).

LSD-сортировка строк

Необходимо отсортировать N строк длины L каждая. Символы строк принадлежат алфавиту $\mathscr{A}$ размера A ($A = |\mathscr{A}|$).

Идея

L раз выполнить распределяющий подсчет, используя в качестве ключей каждый символ справа налево.

LSD-сортировка строк. Пример

```
C065MK
X015MC
A346OE
B349PT
A146PM
E065XX
```

LSD-сортировка строк. Пример

	$d = 5$
C065MK	A346O**E**
X015MC	C065M**K**
A346OE	A146P**M**
B349PT	X015M**C**
A146PM	B349P**T**
E065XX	E065X**X**

LSD-сортировка строк. Пример

	$d = 5$	$d = 4$
C065MK	A346O**E**	C065**M**K
X015MC	C065M**K**	X015**M**C
A346OE	A146P**M**	A346**O**E
B349PT	X015M**C**	A146**P**M
A146PM	B349P**T**	B349**P**T
E065XX	E065X**X**	E065**X**X

LSD-сортировка строк. Пример

	$d = 5$	$d = 4$	$d = 3$
C065MK	A346O**E**	C065**M**K	C06**5**MK
X015MC	C065M**K**	X015**M**C	X01**5**MC
A346OE	A146P**M**	A346**O**E	E06**5**XX
B349PT	X015M**C**	A146**P**M	A34**6**OE
A146PM	B349P**T**	B349**P**T	A14**6**PM
E065XX	E065X**X**	E065**X**X	B34**9**PT

LSD-сортировка строк. Пример

	$d = 5$	$d = 4$	$d = 3$	$d = 2$
C065MK	A346O**E**	C065**M**K	C06**5**MK	X0**1**5MC
X015MC	C065M**K**	X015**M**C	X01**5**MC	A3**4**6OE
A346OE	A146P**M**	A346**O**E	E06**5**XX	A1**4**6PM
B349PT	X015M**C**	A146**P**M	A34**6**OE	B3**4**9PT
A146PM	B349P**T**	B349**P**T	A14**6**PM	C0**6**5MK
E065XX	E065X**X**	E065**X**X	B34**9**PT	E0**6**5XX

LSD-сортировка строк. Пример

	$d = 5$	$d = 4$	$d = 3$	$d = 2$	$d = 1$
C065MK	A346O**E**	C065**M**K	C06**5**MK	X0**1**5MC	X**0**15MC
X015MC	C065M**K**	X015**M**C	X01**5**MC	A3**4**6OE	C**0**65MK
A346OE	A146P**M**	A346**O**E	E06**5**XX	A1**4**6PM	E**0**65XX
B349PT	X015M**C**	A146**P**M	A34**6**OE	B3**4**9PT	A**1**46PM
A146PM	B349P**T**	B349**P**T	A14**6**PM	C0**6**5MK	A**3**46OE
E065XX	E065X**X**	E065**X**X	B34**9**PT	E0**6**5XX	B**3**49PT

LSD-сортировка строк. Пример

	$d = 5$	$d = 4$	$d = 3$	$d = 2$	$d = 1$	$d = 0$
C065MK	A346O**E**	C065**M**K	C06**5**MK	X0**1**5MC	X**0**15MC	**A**146PM
X015MC	C065M**K**	X015**M**C	X01**5**MC	A3**4**6OE	C**0**65MK	**A**346OE
A346OE	A146P**M**	A346**O**E	E06**5**XX	A1**4**6PM	E**0**65XX	**B**349PT
B349PT	X015M**C**	A146**P**M	A34**6**OE	B3**4**9PT	A**1**46PM	**C**065MK
A146PM	B349P**T**	B349**P**T	A14**6**PM	C0**6**5MK	A**3**46OE	**E**065XX
E065XX	E065X**X**	E065**X**X	B34**9**PT	E0**6**5XX	B**3**49PT	**X**015MC

```
1: procedure LSD-Sort(strings, L, A)
2:     for d ← L − 1, 0 do
3:         Counting − Sort (strings, A, d) // сортировка по
   d-символу
4:     end for
5: end procedure
```

LSD-сортировка. Анализ

LSD-сортировка устойчиво упорядочивает строки фиксированной длины.

LSD-сортировка. Анализ

LSD-сортировка устойчиво упорядочивает строки фиксированной длины.
Общее время выполнения: $\Theta(L * (N + A))$

LSD-сортировка. Анализ

LSD-сортировка устойчиво упорядочивает строки фиксированной длины.
Общее время выполнения: $\Theta(L * (N + A))$

На практике $A = O(N)$, в этом случае $\boldsymbol{\Theta(L * N)}$.

MSD-сортировка строк

Необходимо отсортировать N строк *разной* длины. Символы строк принадлежат алфавиту $\mathscr{A}$ размера A ($A = |\mathscr{A}|$).

MSD-сортировка строк

Необходимо отсортировать N строк *разной* длины. Символы строк принадлежат алфавиту $\mathscr{A}$ размера A ($A = |\mathscr{A}|$).

Идея

Строки упорядочиваются с помощью распределяющего подсчета по первому символу, затем подмассивы, соответствующие каждому символу, рекурсивно упорядочиваются по остальным символам (кроме первого, который один и тот же у всех строк подмассива).

Соглашение об окончании строк

Для правильного выполнения сортировки необходим подмассив для строк(все символы которых уже просмотрены), чтобы оказаться в первом подмассиве, и его рекурсивно сортировать уже не нужно.
Можно ввести метод ($charAt(s, d)$)для получения d-символа строки s, который в случае $d \geq |s|$ возвращал бы определенное значение-маркер (-1).

1: $SIZE\_FOR\_INSERTION\_SORT \leftarrow 15$ // $\sim \sqrt[2]{|A|}$
2: $A \leftarrow 256$ // для ext ASCII
3: **procedure** MSD-SORT(strings)
4: $n \leftarrow strings.length$
5: $tmp[0..n-1]$
6: $msd\_sort\,(strings, tmp, 0, n-1, 0)$
7: **end procedure**

```
 1: procedure MSD_SORT(strings, tmp, lo, hi, d)
 2:     if hi ≤ lo + SIZE_FOR_INSERTION_SORT then
 3:         insertion_sort_d (strings, lo, hi, d)
 4:         return
 5:     end if
 6:     c[0..A] - массив, инициализированный 0
 7:     for i ← 0, A do
 8:         charNum ← charAt(strings[i], d)
 9:         c[charNum] ← c[charNum] + 1
10:     end for
11:     for j ← 1, A do
12:         c[j] ← c[j] + c[j − 1]
13:     end for
14:     for i ← n − 1, 0 do
15:         symbolIndex ← charAt(strings[i], d) + 1
16:         tmp[c[symbolIndex]] ← strings[i]
17:         c[symbolIndex] ← c[symbolIndex] − 1
18:     end for
19:     ...
20: end procedure
```

```
1:  procedure MSD_SORT(strings, tmp, lo, hi, d)
2:      ...
3:      for i ← n − 1, 0 do
4:          symbolIndex ← charAt(strings[i], d) + 1
5:          tmp[c[symbolIndex]] ← strings[i]
6:          c[symbolIndex] ← c[symbolIndex] − 1
7:      end for
8:      for i ← lo, hi do
9:          strings[i] ← tmp[i − lo]
10:     end for
11:     for j ← 1, A − 1 do
12:         msd_sort (strings, tmp, lo + c[j − 1], lo + c[j] − 1, d)
13:     end for
14: end procedure
```

```
1: //сортировка вставками для строк с одинаковыми первыми
   d символами
2: procedure INSERTION_SORT_D(strings, lo, hi, d)
3:     for i ← lo, hi do
4:         for
   j ← i; j > lo&&less(strings[j], strings[j − 1], d); j ← j − 1 do
5:             exch (strings, j, j − 1)
6:         end for
7:     end for
8: end procedure
```

MSD-сортировка. Особенности

- работает со строками разной длины
- может не проверять все символы ключей
- заданный алфавит
- отсечка для небольших подмассивов
- равные ключи (или длинные общие префиксы)

MSD-сортировка. Анализ

Для методов сортировки, основанных на сравнениях был важен *порядок* ключей, в случае MSD-метода - *значение* ключей.

- Для *случайных* данных MSD-сортировка проверяет лишь столько символов, сколько нужно для различения ключей, поэтому время *сублинейно* относительно количества символов в данных.

MSD-сортировка. Анализ

Для методов сортировки, основанных на сравнениях был важен *порядок* ключей, в случае MSD-метода - *значение* ключей.

- Для *случайных* данных MSD-сортировка проверяет лишь столько символов, сколько нужно для различения ключей, поэтому время *сублинейно* относительно количества символов в данных.
- Для *неслучайных* данных MSD-сортировка может выполняться за *сублинейно* время, но ей может потребоваться проверка большего числа символов, чем для случайных данных, поэтому время выполнения приближается к линейному.

MSD-сортировка. Анализ

Для методов сортировки, основанных на сравнениях был важен *порядок* ключей, в случае MSD-метода - *значение* ключей.

- Для *случайных* данных MSD-сортировка проверяет лишь столько символов, сколько нужно для различения ключей, поэтому время *сублинейно* относительно количества символов в данных.
- Для *неслучайных* данных MSD-сортировка может выполняться за *сублинейно* время, но ей может потребоваться проверка большего числа символов, чем для случайных данных, поэтому время выполнения приближается к линейному.
- В *худшем* случае — время выполнения *линейно* относительно числа символов в данных (как в LSD-сортировке).

MSD-сортировка. Анализ

Для сортировки N случайных строк из A-символьного алфавита MSD-сортировке требуется просмотреть в среднем порядка $N \log_A N$ символов.

Длина ключей не играет роли. Для любого заданного числа символов существует отличная от нуля вероятность, что два ключа совпадут, но она настолько мала, что ею можно пренебречь в оценках производительности.

MSD-сортировка. Анализ

Для сортировки N строк из A-символьного алфавита MSD-сортировке требуется от $8N + 3A$ до $7lN + 3lR$ обращений к массиву, где l-средняя длина строк.

MSD-сортировка. Анализ

Для сортировки N строк из A-символьного алфавита MSD-сортировке требуется от $8N + 3A$ до $7lN + 3lR$ обращений к массиву, где l-средняя длина строк.

Для сортировки N строк из A-символьного алфавита MSD-сортировка требует в худшем случае объем памяти $\sim A * L_{max} + N$, где L_{max} — максимальная длина строки.

Массив *c* должен быть создан в методе msd_sort, поэтому общий объем необходимой памяти пропорционален A, умноженному на глубину рекурсии (плюс N для массива tmp).

Трехчастная быстрая сортировка строк

Идея

Чтобы избежать пустых подмассивов (в случае большого числа равных ключей), предлагается разбивать массив строк на три части (как в быстрой сортировке) по старшему символу ключей, а затем (рекурсивно) сортировать эти массивы. Переход к следующему символу осуществляется только в среднем массиве.

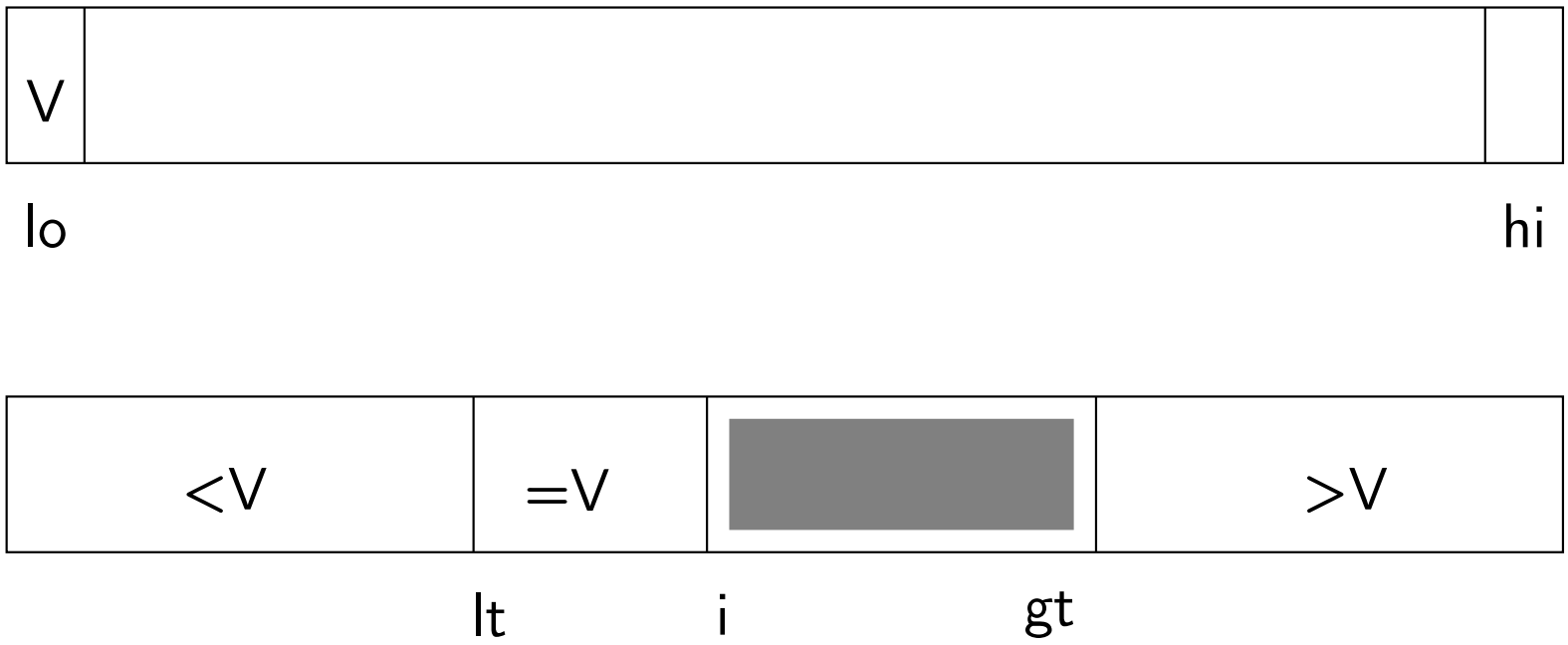
Быстрая сортировка. Напоминание

Основа метода—процесс разбиения, который упорядочивает массив a так, чтобы выполнялись три условия:

- центральный элемент $a[j]$ находится на своем окончательном месте (центральный элемент определяется в процессе разбиения)
- ни один элемент от $a[lo]$ до $a[j-1]$ не больше, чем $a[j]$
- ни один элемент от $a[j+1]$ до $a[hi]$ не меньше, чем $a[j]$

```
1: procedure QUICK_SORT(a, lo, hi)
2:     if hi ≤ lo then
3:         return
4:     end if
5:     j ← partition (a, lo, hi)
6:     quick_sort (a, lo, j − 1)
7:     quick_sort (a, j + 1, hi)
8: end procedure
```

Трехчастное разбиение



```
 1: procedure QUICK_SORT_3(a, lo, hi)
 2:     if hi ≤ lo then
 3:         return
 4:     end if
 5:     lt ← lo; i ← lo + 1; gt ← hi
 6:     v ← a[lo]
 7:     while i ≤ gt do
 8:         cmp ← a[i].compareTo (v)
 9:         if cmp < 0 then
10:             exch(a, lt++, i++)
11:         else if cmp > 0 then
12:             exch(a, i, gt–)
13:         else
14:             i++
15:         end if
16:     end while
17:     quick_sort_3 (a, lo, lt − 1)
18:     quick_sort_3 (a, gt + 1, hi)
19: end procedure
```

```
1: procedure QUICK_SORT_3_STR(a)
2:     _quick_sort_3_str(a, 0, a.length − 1, 0)
3: end procedure
```

```
 1: procedure _QUICK_SORT_3_STR(a, lo, hi, d)
 2:     if hi ≤ lo then
 3:         return
 4:     end if
 5:     lt ← lo; gt ← hi
 6:     v ← charAt(a[lo], d)
 7:     i ← lo + 1
 8:     while i ≤ gt do
 9:         t ← charAt(a[i], d)
10:         if t < v then
11:             exch(a, lt++, i++)
12:         else if t > 0 then
13:             exch(a, i, gt–)
14:         else
15:             i++
16:         end if
17:     end while
18:     ...
19: end procedure
```

```
 1: procedure _QUICK_SORT_3_STR(a, lo, hi, d)
 2:     ...
 3:     while i ≤ gt do
 4:         t ← charAt(a[i], d)
 5:         if t < v then
 6:             exch(a, lt++, i++)
 7:         else if t > 0 then
 8:             exch(a, i, gt–)
 9:         else
10:             i++
11:         end if
12:     end while
13:     //a[lo, .., lt − 1] < v = a[lt, .., gt] < a[gt + 1, .., hi]
14:     _quick_sort_3_str(a, lo, lt − 1, d)
15:     if v ≥ 0 then
16:         _quick_sort_3_str(a, lt, gt, d + 1)
17:     end if
18:     _quick_sort_3_str(a, gt + 1, hi, d)
19: end procedure
```

Сравнение алгоритмов сортировки строк

Алгоритм	**Устойчив?**	**На месте?**	**Время выполнения**	**Доп. память**	**Сильные стороны**
LSD-сортировка	Да	Нет	NL	N	Короткие строки фиксированной длины
MSD-сортировка	Да	Нет	от N до NI	$N + LR$	Случайные строки
Трехчастная сортировка строк	Нет	Да	от N до NI	$L + \log N$	Общего назначения, строки с одинаковыми длинными префиксами

Литература

- Седжвик Р., Уэйн К. Алгоритмы на Java. 4-е издание
- Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 3-е издание