http://rain.ifmo.ru/cat

# Сортировка вектора $_{<3>}$ InsertionSort

© С. Е. Столяр, 2010
ses@mail.ifmo.ru










# Вспомогательный алгоритм ...

## Постановка задачи

- Вход: массив $M_{[1..k]}, k > 1$.
- Подмассив $M_{[1..(k-1)]}$ упорядочен *по неубыванию*.
- Выход: тот же массив, полностью упорядоченный.




Back
Close



# Вспомогательный алгоритм ...

## Постановка задачи

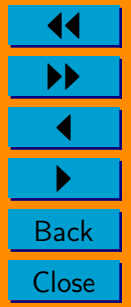
- Вход: массив $M_{[1..k]}, k > 1$.
- Подмассив $M_{[1..(k-1)]}$ упорядочен *по неубыванию*.
- Выход: тот же массив, полностью упорядоченный.

## Пример

- Вход: $M_{[1..5]} = \boxed{\boxed{2\ 7\ 8\ 17}\ 3}$.
- Выход: $M_{[1..5]} = \boxed{2\ 3\ 7\ 8\ 17}$.

## Реализация

- Алгоритм $\mathrm{StraightIns}(M_{[1..(k-1)]}, M_k)$ — *простая вставка*.





# . . . Вспомогательный алгоритм

**Algorithm 3.1:** STRAIGHTINS$(M_{[1..(k-1)]}, M_k)$

**comment:** расширим массив $M_{[1..k]}$ до $M_{[0..k]}$

$tmp \leftarrow M_k \ ; \ M_0 \leftarrow tmp$

$i \leftarrow (k-1)$

**while** $tmp < M_i$

**do** $\begin{cases} M_{i+1} \leftarrow M_i \\ i \leftarrow (i-1) \end{cases}$

$M_i \leftarrow tmp$

**return** $(M_{[1..k]})$

Back

Close



# . . . Вспомогательный алгоритм

**Algorithm 3.1:** StraightIns$(M_{[1..(k-1)]}, M_k)$

**comment:** расширим массив $M_{[1..k]}$ до $M_{[0..k]}$

$tmp \leftarrow M_k \ ; \ M_0 \leftarrow tmp$

$i \leftarrow (k-1)$

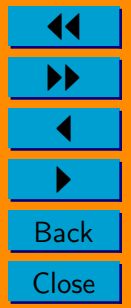
**while** $tmp < M_i$

**do** $\begin{cases} M_{i+1} \leftarrow M_i \\ i \leftarrow (i-1) \end{cases}$

$M_i \leftarrow tmp$

**return** $(M_{[1..k]})$

- Ячейка $M_0$ играет роль *барьера*.
- Без него заголовок цикла надо дополнить вторым условием: проверять $i \in 1..k$.





# . . . Вспомогательный алгоритм

**Algorithm 3.1:** StraightIns$(M_{[1..(k-1)]}, M_k)$

**comment:** расширим массив $M_{[1..k]}$ до $M_{[0..k]}$

$tmp \leftarrow M_k \ ; \ M_0 \leftarrow tmp$

$i \leftarrow (k-1)$

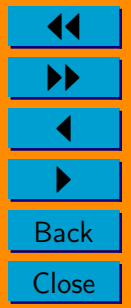
**while** $tmp < M_i$

**do** $\begin{cases} M_{i+1} \leftarrow M_i \\ i \leftarrow (i-1) \end{cases}$

$M_i \leftarrow tmp$

**return** $(M_{[1..k]})$

- Ячейка $M_0$ играет роль *барьера*.
- Без него заголовок цикла надо дополнить вторым условием: проверять $i \in 1..k$.





# Сортировка простыми вставками

- $n = 1$ : вырожденный массив $M_{[1..n]}$ не требует упорядочения, задача $\mathcal{M}_1$ решена.
- $n > 1$ : возникает цепочка подзадач $\mathcal{M}_2, \mathcal{M}_3, \ldots, \mathcal{M}_n$, которые последовательно решаются с помощью $\mathrm{StraightIns}$.

Back
Close



# Сортировка простыми вставками

- $n = 1$ : вырожденный массив $M_{[1..n]}$ не требует упорядочения, задача $\mathcal{M}_1$ решена.
- $n > 1$ : возникает цепочка подзадач $\mathcal{M}_2, \mathcal{M}_3, \ldots, \mathcal{M}_n$, которые последовательно решаются с помощью StraightIns.

**Algorithm 4.1:** InsertionSort$(M_{[1..n]})$

**for** $i \leftarrow 2$ **to** $n$
**do** $\{$StraightIns$(M_{[1..(i-1)]}, M_i)$




Back
Close



# Сортировка простыми вставками

- $n = 1$ : вырожденный массив $M_{[1..n]}$ не требует упорядочения, задача $\mathcal{M}_1$ решена.
- $n > 1$ : возникает цепочка подзадач $\mathcal{M}_2, \mathcal{M}_3, \ldots, \mathcal{M}_n$, которые последовательно решаются с помощью StraightIns.

**Algorithm 4.1:** InsertionSort$(M_{[1..n]})$

**for** $i \leftarrow 2$ **to** $n$
**do** $\{$StraightIns$(M_{[1..(i-1)]}, M_i)$




Back
Close

# Пример

- Вход: $M_{[1..5]} = \boxed{8 \;\; 2 \;\; 17 \;\; 7 \;\; 3}$






Back
Close

# Пример

- Вход: $M_{[1..5]} = \boxed{8 \;\; 2 \;\; 17 \;\; 7 \;\; 3}$

- $\mathcal{M}_2$ : $\boxed{\boxed{8} \;\; 2 \;\; 17 \;\; 7 \;\; 3}$







Back
Close



# Пример

- Вход: $M_{[1..5]} = \boxed{8\ \ 2\ \ 17\ \ 7\ \ 3}$

- $\mathcal{M}_2$ : $\boxed{\boxed{8}\ \ 2\ \ 17\ \ 7\ \ 3}$
- $\mathcal{M}_3$ : $\boxed{2\ \ \boxed{8}\ \ 17\ \ 7\ \ 3}$

Back
Close



# Пример

- Вход: $M_{[1..5]} =$ | 8 | 2 | 17 | 7 | 3 |

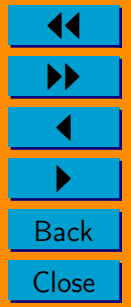
- $\mathcal{M}_2$ : | 8 | 2 | 17 | 7 | 3 |
- $\mathcal{M}_3$ : | 2 | 8 | 17 | 7 | 3 |
- $\mathcal{M}_4$ : | 2 | 8 | 17 | 7 | 3 |



# Пример

- Вход: $M_{[1..5]} = \boxed{8\ \ 2\ \ 17\ \ 7\ \ 3}$

- $\mathcal{M}_2$ : $\boxed{8}\ 2\ \ 17\ \ 7\ \ 3$
- $\mathcal{M}_3$ : $\boxed{2\ \ 8}\ \ 17\ \ 7\ \ 3$
- $\mathcal{M}_4$ : $\boxed{2\ \ 8\ \ 17}\ \ 7\ \ 3$
- $\mathcal{M}_5$ : $\boxed{2\ \ 7\ \ 8\ \ 17}\ \ 3$





# Пример

- Вход: $M_{[1..5]} = \boxed{8\ \ 2\ \ 17\ \ 7\ \ 3}$

- $\mathcal{M}_2$ : $\boxed{\boxed{8}\ \ 2\ \ 17\ \ 7\ \ 3}$
- $\mathcal{M}_3$ : $\boxed{\boxed{2\ \ 8}\ \ 17\ \ 7\ \ 3}$
- $\mathcal{M}_4$ : $\boxed{\boxed{2\ \ 8\ \ 17}\ \ 7\ \ 3}$
- $\mathcal{M}_5$ : $\boxed{\boxed{2\ \ 7\ \ 8\ \ 17}\ \ 3}$

- Выход: $M_{[1..5]} = \boxed{2\ \ 3\ \ 7\ \ 8\ \ 17}$

Back
Close



# Пример

- Вход: $M_{[1..5]} = \boxed{8\ \ 2\ \ 17\ \ 7\ \ 3}$

- $\mathcal{M}_2$ : $\boxed{\boxed{8}\ \ 2\ \ 17\ \ 7\ \ 3}$
- $\mathcal{M}_3$ : $\boxed{\boxed{2\ \ 8}\ \ 17\ \ 7\ \ 3}$
- $\mathcal{M}_4$ : $\boxed{\boxed{2\ \ 8\ \ 17}\ \ 7\ \ 3}$
- $\mathcal{M}_5$ : $\boxed{\boxed{2\ \ 7\ \ 8\ \ 17}\ \ 3}$

- Выход: $M_{[1..5]} = \boxed{2\ \ 3\ \ 7\ \ 8\ \ 17}$

Back
Close




# Оценки трудоемкости

Согласно псевдокоду,

- алгоритм сортирует массив *на месте*,
  объем дополнительной памяти не зависит от длины входа;
- временна́я трудоемкость $O(n^2)$ — квадратичная относительно длины входа.




Back
Close



# Оценки трудоемкости

Согласно псевдокоду,

- алгоритм сортирует массив *на месте*,
  объем дополнительной памяти не зависит от длины входа;
- временна́я трудоемкость $O(n^2)$ — квадратичная относительно длины входа.

? Оцените трудоемкость алгоритма, если входной набор уже упорядочен.




Back
Close



# Оценки трудоемкости

Согласно псевдокоду,

- алгоритм сортирует массив *на месте*,
  объем дополнительной памяти не зависит от длины входа;
- временна́я трудоемкость $O(n^2)$ — квадратичная относительно длины входа.

? Оцените трудоемкость алгоритма, если входной набор уже упорядочен.

? Оцените трудоемкость алгоритма, если входной набор упорядочен в противоположном направлении.




Back
Close



# Оценки трудоемкости

Согласно псевдокоду,

- алгоритм сортирует массив *на месте*,
  объем дополнительной памяти не зависит от длины входа;
- временна́я трудоемкость $O(n^2)$ — квадратичная относительно длины входа.

? Оцените трудоемкость алгоритма, если входной набор уже упорядочен.

? Оцените трудоемкость алгоритма, если входной набор упорядочен в противоположном направлении.




Back
Close



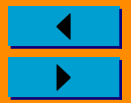
# Модификации

## *BinaryInsertionSort*

- Вспомогательный алгоритм StraightIns последовательно проверяет элементы слева, отыскивая место для вставки.
- Поскольку подмассив слева упорядочен, можно воспользоваться *дихотомическим поиском*.
- Преимущества могут проявиться при *длинных* наборах данных, для *коротких* массивов, напротив, эффект окажется отрицательным.
- Естественное название алгоритма — *сортировка двоичными вставками*.

## *ShellSort*

- Весьма эффективная *сортировка Шелла* (1959) будет рассмотрена позднее.




Back
Close



# Вопросы / упражнения / задания

? Как изменится алгоритм InsertionSort, если требуется упорядочить массив *по убыванию*?

? Является ли алгоритм InsertionSort *устойчивым*?

ДЗ Напишите процедуру, которая реализует алгоритм BinaryInsertionSort.




Back
Close




# Остались вопросы?

Рекомендуем заглянуть в предлагаемые источники:

- *визуализатор* [скачать ZIP (внутри *.EXE)] / И. Зверев. — СПб.: СПбГУ ИТМО, 2000.
- *визуализатор* [Java] / М. Смачных. — СПб.: СПбГУ ИТМО, 2004, 2010.
- С. Е. Столяр, А. А. Владыкин. *Информатика: Представление данных и алгоритмы.* — СПб.: Невский Диалект; М.: БИНОМ. Лаборатория знаний, 2007. — 382 с.
- *Wikipedia / InsertionSort* [ONLINE]

## Доп. литература

- Вирт, Н. *Алгоритмы и структуры данных.* — 2-е изд., испр. — СПб., Невский Диалект, 2001. — 352 с.




Back
Close