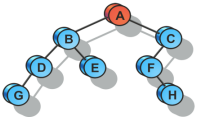
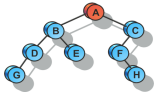
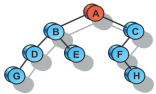


Алгоритм сортировки вставками



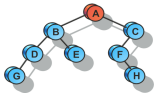
Задача сортировки

- Задача сортировки чисел в порядке возрастания:



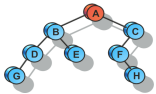
Задача сортировки

- Задача сортировки чисел в порядке возрастания:
  - Вход: последовательность из *n* чисел $< a_1, a_2, ..., a_n >$



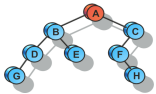
Задача сортировки

- Задача сортировки чисел в порядке возрастания:
  - Вход: последовательность из n чисел $< a_1, a_2, ..., a_n >$
  - Выход: перестановка входной последовательности $< a'_1, a'_2, ..., a'_n >$, такая что $a'_1 \leqslant a'_2 \leqslant ... \leqslant a'_n$



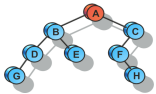
Задача сортировки

- Задача сортировки чисел в порядке возрастания:
  - Вход: последовательность из n чисел $< a_1, a_2, ..., a_n >$
  - Выход: перестановка входной последовательности $< a'_1, a'_2, ..., a'_n >$, такая что $a'_1 \leqslant a'_2 \leqslant ... \leqslant a'_n$
- Пример экземпляра задачи сортировки:



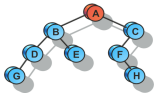
Задача сортировки

- Задача сортировки чисел в порядке возрастания:
  - Вход: последовательность из n чисел $< a_1, a_2, ..., a_n >$
  - Выход: перестановка входной последовательности $< a'_1, a'_2, ..., a'_n >$, такая что $a'_1 \leqslant a'_2 \leqslant ... \leqslant a'_n$
- Пример экземпляра задачи сортировки:
  - Вход: $< 10, 8, 2, 12, 5, 100 >$



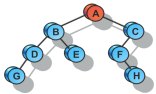
Задача сортировки

- Задача сортировки чисел в порядке возрастания:
  - Вход: последовательность из n чисел $< a_1, a_2, ..., a_n >$
  - Выход: перестановка входной последовательности $< a'_1, a'_2, ..., a'_n >$, такая что $a'_1 \leqslant a'_2 \leqslant ... \leqslant a'_n$
- Пример экземпляра задачи сортировки:
  - Вход: $< 10, 8, 2, 12, 5, 100 >$
  - Выход: $< 2, 5, 8, 10, 12, 100 >$



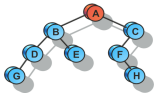
Задача сортировки

- Задача сортировки чисел в порядке возрастания:
  - Вход: последовательность из n чисел $< a_1, a_2, ..., a_n >$
  - Выход: перестановка входной последовательности $< a'_1, a'_2, ..., a'_n >$, такая что $a'_1 \leqslant a'_2 \leqslant ... \leqslant a'_n$
- Пример экземпляра задачи сортировки:
  - Вход: $< 10, 8, 2, 12, 5, 100 >$
  - Выход: $< 2, 5, 8, 10, 12, 100 >$
- В этой лекции будет изучен алгоритм сортировки вставками



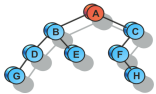
Задача сортировки

- Задача сортировки чисел в порядке возрастания:
  - Вход: последовательность из n чисел $< a_1, a_2, ..., a_n >$
  - Выход: перестановка входной последовательности $< a'_1, a'_2, ..., a'_n >$, такая что $a'_1 \leqslant a'_2 \leqslant ... \leqslant a'_n$
- Пример экземпляра задачи сортировки:
  - Вход: $< 10, 8, 2, 12, 5, 100 >$
  - Выход: $< 2, 5, 8, 10, 12, 100 >$
- В этой лекции будет изучен алгоритм сортировки вставками
- На следующей неделе — другие алгоритмы сортировки, основанные на сравнении



Задача сортировки

- Задача сортировки чисел в порядке возрастания:
  - Вход: последовательность из n чисел $< a_1, a_2, ..., a_n >$
  - Выход: перестановка входной последовательности $< a'_1, a'_2, ..., a'_n >$, такая что $a'_1 \leqslant a'_2 \leqslant ... \leqslant a'_n$
- Пример экземпляра задачи сортировки:
  - Вход: $< 10, 8, 2, 12, 5, 100 >$
  - Выход: $< 2, 5, 8, 10, 12, 100 >$
- В этой лекции будет изучен алгоритм сортировки вставками
- На следующей неделе — другие алгоритмы сортировки, основанные на сравнении
  - Могут быть применены к сортировке любых сравнимых между собой элементов



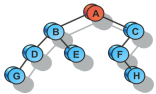
Алгоритм сортировки вставками

Пример:

5 2 4 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



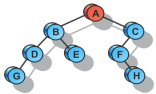
Алгоритм сортировки вставками

Пример:

5 2 4 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



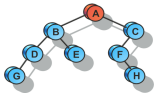
Алгоритм сортировки вставками

Пример:

2 **5** 4 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:             temp ← A[i]
6:             A[i] ← A[i + 1]
7:             A[i + 1] ← temp
8:         i ← i − 1
```



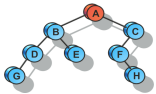
Алгоритм сортировки вставками

Пример:

2 5 4 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



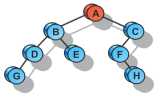
Алгоритм сортировки вставками

Пример:

2 **5** 4 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



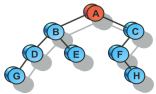
Алгоритм сортировки вставками

Пример:

2 4 **5** 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



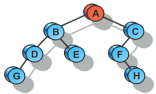
Алгоритм сортировки вставками

Пример:

2 4 5 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



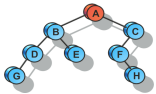
Алгоритм сортировки вставками

Пример:

2 4 ***5*** 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



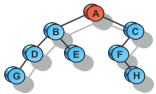
Алгоритм сортировки вставками

Пример:

2 4 ***5*** 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



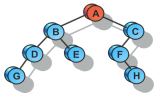
Алгоритм сортировки вставками

Пример:

2 4 5 6 1 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



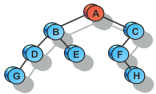
Алгоритм сортировки вставками

Пример:

1 2 4 5 6 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



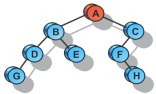
Алгоритм сортировки вставками

Пример:

1 2 4 5 6 3

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



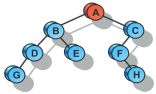
Алгоритм сортировки вставками

Пример:

1 **2** 3 4 5 6

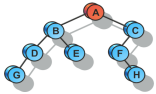
InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```



Оценка времени работы сортировки вставками

Оценим время однократного выполнения каждой строки псевдокода алгоритма сортировки вставками

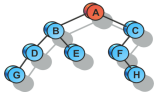


Оценка времени работы сортировки вставками

Оценим время однократного выполнения каждой строки псевдокода алгоритма сортировки вставками

InsertionSort(A)

		Время
1:	**for** $j \leftarrow 2$ ***to*** n **do**	c_1
2:	$i \leftarrow j - 1$	
3:	**while** $i > 0$ and $A[i] > A[i+1]$ **do**	
4:	swap$(A[i], A[i+1])$	
5:	$i \leftarrow i - 1$	

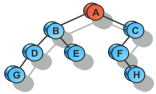


Оценка времени работы сортировки вставками

Оценим время однократного выполнения каждой строки псевдокода алгоритма сортировки вставками

InsertionSort(A)

		Время
1:	**for** $j \leftarrow 2$ **to** n **do**	c_1
2:	$i \leftarrow j - 1$	c_2
3:	**while** $i > 0$ and $A[i] > A[i+1]$ **do**	
4:	swap$(A[i], A[i+1])$	
5:	$i \leftarrow i - 1$	

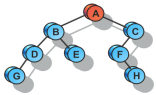


Оценка времени работы сортировки вставками

Оценим время однократного выполнения каждой строки псевдокода алгоритма сортировки вставками

InsertionSort(A)

		Время
1:	**for** $j \leftarrow 2$ **to** n **do**	c_1
2:	$i \leftarrow j - 1$	c_2
3:	**while** $i > 0$ ***and*** $A[i] > A[i+1]$ **do**	c_3
4:	swap$(A[i], A[i+1])$	
5:	$i \leftarrow i - 1$	

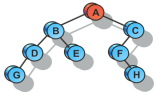


Оценка времени работы сортировки вставками

Оценим время однократного выполнения каждой строки псевдокода алгоритма сортировки вставками

InsertionSort(A)

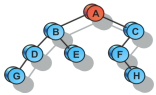
		Время
1:	**for** $j \leftarrow 2$ **to** n **do**	c_1
2:	$i \leftarrow j - 1$	c_2
3:	**while** $i > 0$ and $A[i] > A[i+1]$ **do**	c_3
4:	***swap***$(A[i], A[i+1])$	c_4
5:	$i \leftarrow i - 1$	



Оценка времени работы сортировки вставками

Оценим время однократного выполнения каждой строки псевдокода алгоритма сортировки вставками

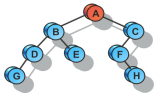
InsertionSort(A)	Время
1: **for** $j \leftarrow 2$ **to** n **do**	c_1
2: $i \leftarrow j - 1$	c_2
3: **while** $i > 0$ and $A[i] > A[i+1]$ **do**	c_3
4: swap$(A[i], A[i+1])$	c_4
5: $i \leftarrow i - 1$	c_5



Оценка времени работы сортировки вставками

Оценим общее число выполнений каждой строки

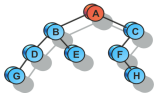
InsertionSort(A)	Время	Число раз
1: **for** $j \leftarrow 2$ ***to*** n **do**	c_1	n
2: $i \leftarrow j - 1$	c_2	
3: **while** $i > 0$ and $A[i] > A[i+1]$ **do**	c_3	
4: swap$(A[i], A[i+1])$	c_4	
5: $i \leftarrow i - 1$	c_5	



Оценка времени работы сортировки вставками

Оценим общее число выполнений каждой строки

InsertionSort(A)	Время	Число раз
1: **for** $j \leftarrow 2$ **to** n **do**	c_1	n
2: $i \leftarrow j - 1$	c_2	$n-1$
3: **while** $i > 0$ and $A[i] > A[i+1]$ **do**	c_3	
4: swap$(A[i], A[i+1])$	c_4	
5: $i \leftarrow i - 1$	c_5	



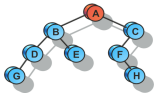
Оценка времени работы сортировки вставками

Оценим общее число выполнений каждой строки

t_j — число проверок условия while на j-ой итерации цикла for

InsertionSort(A)

		Время	Число раз
1:	**for** $j \leftarrow 2$ **to** n **do**	c_1	n
2:	$i \leftarrow j - 1$	c_2	$n-1$
3:	**while** $i > 0$ ***and*** $A[i] > A[i+1]$ **do**	c_3	
4:	swap$(A[i], A[i+1])$	c_4	
5:	$i \leftarrow i - 1$	c_5	



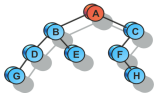
Оценка времени работы сортировки вставками

Оценим общее число выполнений каждой строки

t_j — число проверок условия while на j-ой итерации цикла for

InsertionSort(A)

		Время	Число раз
1:	**for** $j \leftarrow 2$ **to** n **do**	c_1	n
2:	$i \leftarrow j - 1$	c_2	$n-1$
3:	**while** $i > 0$ ***and*** $A[i] > A[i+1]$ **do**	c_3	$\sum_{j=2}^{n} t_j$
4:	swap$(A[i], A[i+1])$	c_4	
5:	$i \leftarrow i - 1$	c_5	



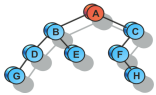
Оценка времени работы сортировки вставками

Оценим общее число выполнений каждой строки

t_j — число проверок условия while на j-ой итерации цикла for

InsertionSort(A)

		Время	Число раз
1:	**for** $j \leftarrow 2$ **to** n **do**	c_1	n
2:	$i \leftarrow j - 1$	c_2	$n-1$
3:	**while** $i > 0$ and $A[i] > A[i+1]$ **do**	c_3	$\sum_{j=2}^{n} t_j$
4:	***swap***$(A[i], A[i+1])$	c_4	$\sum_{j=2}^{n} (t_j - 1)$
5:	$i \leftarrow i - 1$	c_5	



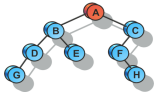
Оценка времени работы сортировки вставками

Оценим общее число выполнений каждой строки

t_j — число проверок условия while на j-ой итерации цикла for

InsertionSort(A)

		Время	Число раз
1:	**for** $j \leftarrow 2$ **to** n **do**	c_1	n
2:	$i \leftarrow j - 1$	c_2	$n-1$
3:	**while** $i > 0$ and $A[i] > A[i+1]$ **do**	c_3	$\sum_{j=2}^{n} t_j$
4:	swap$(A[i], A[i+1])$	c_4	$\sum_{j=2}^{n} (t_j - 1)$
5:	$i \leftarrow i - 1$	c_5	$\sum_{j=2}^{n} (t_j - 1)$



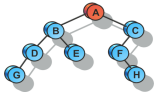
Оценка времени работы сортировки вставками

Оценим общее число выполнений каждой строки

t_j — число проверок условия while на j-ой итерации цикла for

InsertionSort(A)

		Время	Число раз
1:	**for** $j \leftarrow 2$ **to** n **do**	c_1	n
2:	$i \leftarrow j - 1$	c_2	$n-1$
3:	**while** $i > 0$ and $A[i] > A[i+1]$ **do**	c_3	$\sum_{j=2}^{n} t_j$
4:	swap$(A[i], A[i+1])$	c_4	$\sum_{j=2}^{n} (t_j - 1)$
5:	$i \leftarrow i - 1$	c_5	$\sum_{j=2}^{n} (t_j - 1)$



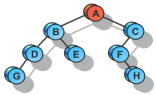
Оценка времени работы сортировки вставками

Оценим суммарное время работы алгоритма
$T(n) =$

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```

Время	Число раз
c_1	n
c_2	$n-1$
c_3	$\sum_{j=2}^{n} t_j$
c_4	$\sum_{j=2}^{n} (t_j - 1)$
c_5	$\sum_{j=2}^{n} (t_j - 1)$



Оценка времени работы сортировки вставками

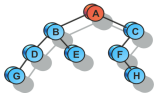
Оценим суммарное время работы алгоритма

$T(n) = c_1 n +$

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```

Время	Число раз
c_1	n
c_2	$n-1$
c_3	$\sum_{j=2}^{n} t_j$
c_4	$\sum_{j=2}^{n} (t_j - 1)$
c_5	$\sum_{j=2}^{n} (t_j - 1)$



Оценка времени работы сортировки вставками

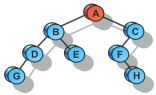
Оценим суммарное время работы алгоритма

$T(n) = c_1 n + c_2(n-1) +$

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```

Время	Число раз
c_1	n
c_2	$n-1$
c_3	$\sum_{j=2}^{n} t_j$
c_4	$\sum_{j=2}^{n} (t_j - 1)$
c_5	$\sum_{j=2}^{n} (t_j - 1)$



Оценка времени работы сортировки вставками

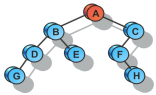
Оценим суммарное время работы алгоритма

$T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j +$

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```

Время	Число раз
c_1	n
c_2	$n-1$
c_3	$\sum_{j=2}^{n} t_j$
c_4	$\sum_{j=2}^{n} (t_j - 1)$
c_5	$\sum_{j=2}^{n} (t_j - 1)$



Оценка времени работы сортировки вставками

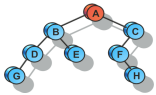
Оценим суммарное время работы алгоритма

$T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) +$

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```

Время	Число раз
c_1	n
c_2	$n-1$
c_3	$\sum_{j=2}^{n} t_j$
c_4	$\sum_{j=2}^{n} (t_j - 1)$
c_5	$\sum_{j=2}^{n} (t_j - 1)$



Оценка времени работы сортировки вставками

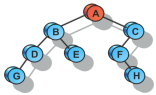
Оценим суммарное время работы алгоритма

$$T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$$

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```

Время	Число раз
c_1	n
c_2	$n-1$
c_3	$\sum_{j=2}^{n} t_j$
c_4	$\sum_{j=2}^{n} (t_j - 1)$
c_5	$\sum_{j=2}^{n} (t_j - 1)$



Оценка времени работы сортировки вставками

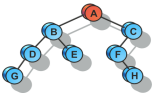
Оценим суммарное время работы алгоритма

$$T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$$

InsertionSort(A)

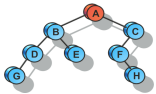
```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
```

Время	Число раз
c_1	n
c_2	$n-1$
c_3	$\sum_{j=2}^{n} t_j$
c_4	$\sum_{j=2}^{n} (t_j - 1)$
c_5	$\sum_{j=2}^{n} (t_j - 1)$



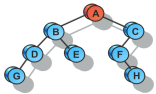
Время работы сортировки вставками в различных случаях

- Время работы сортировки вставками в общем случае :
  $T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$



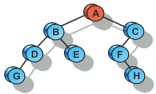
Время работы сортировки вставками в различных случаях

- Время работы сортировки вставками в общем случае :
  $T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$
- Лучший случай: элементы входной последовательности отсортированы в требуемом порядке



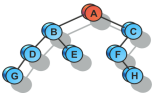
Время работы сортировки вставками в различных случаях

- Время работы сортировки вставками в общем случае :
  $T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$
- Лучший случай: элементы входной последовательности отсортированы в требуемом порядке
  - $t_j = 1$



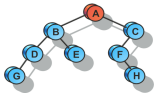
Время работы сортировки вставками в различных случаях

- Время работы сортировки вставками в общем случае :
  $T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$
- Лучший случай: элементы входной последовательности отсортированы в требуемом порядке
  - $t_j = 1$
  - $T(n) = c_1 n + c_2(n-1) + c_3(n-1) = (c_1 + c_2 + c_3)n - (c_2 + c_3) = \Theta(n)$



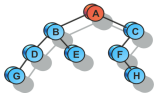
Время работы сортировки вставками в различных случаях

- Время работы сортировки вставками в общем случае :
  $T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$
- Лучший случай: элементы входной последовательности отсортированы в требуемом порядке
  - $t_j = 1$
  - $T(n) = c_1 n + c_2(n-1) + c_3(n-1) = (c_1 + c_2 + c_3)n - (c_2 + c_3) = \Theta(n)$
- Худший случай: элементы входной последовательности отсортированы в порядке, обратном требуемому



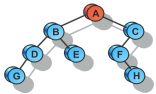
Время работы сортировки вставками в различных случаях

- Время работы сортировки вставками в общем случае :
  $T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$
- Лучший случай: элементы входной последовательности отсортированы в требуемом порядке
  - $t_j = 1$
  - $T(n) = c_1 n + c_2(n-1) + c_3(n-1) = (c_1 + c_2 + c_3)n - (c_2 + c_3) = \Theta(n)$
- Худший случай: элементы входной последовательности отсортированы в порядке, обратном требуемому
  - $t_j = j$



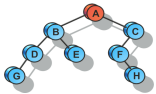
Время работы сортировки вставками в различных случаях

- Время работы сортировки вставками в общем случае :
  $T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$
- Лучший случай: элементы входной последовательности отсортированы в требуемом порядке
  - $t_j = 1$
  - $T(n) = c_1 n + c_2(n-1) + c_3(n-1) = (c_1 + c_2 + c_3)n - (c_2 + c_3) = \Theta(n)$
- Худший случай: элементы входной последовательности отсортированы в порядке, обратном требуемому
  - $t_j = j$
  - По формуле суммы арифметической прогрессии
    $\sum_{j=2}^{n} (j) = \frac{n(n+1)}{2} - 1, \qquad \sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$



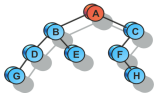
Время работы сортировки вставками в различных случаях

- Время работы сортировки вставками в общем случае :
  $T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$
- Лучший случай: элементы входной последовательности отсортированы в требуемом порядке
  - $t_j = 1$
  - $T(n) = c_1 n + c_2(n-1) + c_3(n-1) = (c_1 + c_2 + c_3)n - (c_2 + c_3) = \Theta(n)$
- Худший случай: элементы входной последовательности отсортированы в порядке, обратном требуемому
  - $t_j = j$
  - По формуле суммы арифметической прогрессии
    $\sum_{j=2}^{n} (j) = \frac{n(n+1)}{2} - 1, \qquad \sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$
  - $T(n) = (\frac{c_3}{2} + \frac{c_4}{2} + \frac{c_5}{2})n^2 + (c_1 + c_2 + \frac{c_3}{2} - \frac{c_4}{2} - \frac{c_5}{2})n - (c_2 + c_3)$



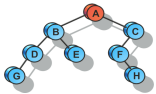
Время работы сортировки вставками в различных случаях

- Время работы сортировки вставками в общем случае :
  $T(n) = c_1 n + c_2(n-1) + c_3 \sum_{j=2}^{n} t_j + c_4 \sum_{j=2}^{n} (t_j - 1) + c_5 \sum_{j=2}^{n} (t_j - 1)$
- Лучший случай: элементы входной последовательности отсортированы в требуемом порядке
  - $t_j = 1$
  - $T(n) = c_1 n + c_2(n-1) + c_3(n-1) = (c_1 + c_2 + c_3)n - (c_2 + c_3) = \Theta(n)$
- Худший случай: элементы входной последовательности отсортированы в порядке, обратном требуемому
  - $t_j = j$
  - По формуле суммы арифметической прогрессии
    $\sum_{j=2}^{n} (j) = \frac{n(n+1)}{2} - 1, \qquad \sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$
  - $T(n) = (\frac{c_3}{2} + \frac{c_4}{2} + \frac{c_5}{2})n^2 + (c_1 + c_2 + \frac{c_3}{2} - \frac{c_4}{2} - \frac{c_5}{2})n - (c_2 + c_3)$
- В общем случае, время работы сортировки вставками оценивается сверху как $O(n^2)$



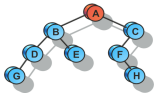
Верхняя оценка времени работы сортировки вставками

- При оценке времени работы алгоритмов часто используется оценка сверху



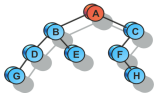
Верхняя оценка времени работы сортировки вставками

- При оценке времени работы алгоритмов часто используется оценка сверху
  - Верхняя оценка дает гарантии на время выполнения алгоритма, не требуя уточнения условий выполнения оценки



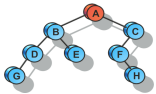
Верхняя оценка времени работы сортировки вставками

- При оценке времени работы алгоритмов часто используется оценка сверху
  - Верхняя оценка дает гарантии на время выполнения алгоритма, не требуя уточнения условий выполнения оценки
  - В некоторых алгоритмах худший случай достигается часто



Верхняя оценка времени работы сортировки вставками

- При оценке времени работы алгоритмов часто используется оценка сверху
  - Верхняя оценка дает гарантии на время выполнения алгоритма, не требуя уточнения условий выполнения оценки
  - В некоторых алгоритмах худший случай достигается часто
  - Среднее время работы алгоритмов во многих случаях не лучше, чем худшее

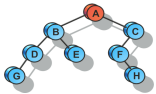


Верхняя оценка времени работы сортировки вставками

- При оценке времени работы алгоритмов часто используется оценка сверху
  - Верхняя оценка дает гарантии на время выполнения алгоритма, не требуя уточнения условий выполнения оценки
  - В некоторых алгоритмах худший случай достигается часто
  - Среднее время работы алгоритмов во многих случаях не лучше, чем худшее

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
6:     A[i + 1] ← key
```



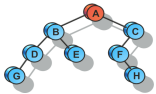
Верхняя оценка времени работы сортировки вставками

- При оценке времени работы алгоритмов часто используется оценка сверху
  - Верхняя оценка дает гарантии на время выполнения алгоритма, не требуя уточнения условий выполнения оценки
  - В некоторых алгоритмах худший случай достигается часто
  - Среднее время работы алгоритмов во многих случаях не лучше, чем худшее

InsertionSort(A)

$T_{for}(n) = O(n)$

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
6:     A[i + 1] ← key
```



Верхняя оценка времени работы сортировки вставками

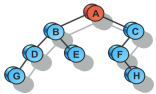
- При оценке времени работы алгоритмов часто используется оценка сверху
  - Верхняя оценка дает гарантии на время выполнения алгоритма, не требуя уточнения условий выполнения оценки
  - В некоторых алгоритмах худший случай достигается часто
  - Среднее время работы алгоритмов во многих случаях не лучше, чем худшее

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
6:     A[i + 1] ← key
```

$T_{\text{for}}(n) = O(n)$

$T_{\textit{while}}(n) = O(n)$



Верхняя оценка времени работы сортировки вставками

- При оценке времени работы алгоритмов часто используется оценка сверху
  - Верхняя оценка дает гарантии на время выполнения алгоритма, не требуя уточнения условий выполнения оценки
  - В некоторых алгоритмах худший случай достигается часто
  - Среднее время работы алгоритмов во многих случаях не лучше, чем худшее

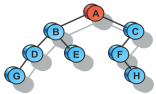
InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
6:     A[i + 1] ← key
```

$T_{\text{for}}(n) = O(n)$

$T_{\text{while}}(n) = O(n)$

$T(n) = T_{\textit{for}}(n) \times T_{\textit{while}}(n) = O(n^2)$



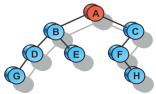
Верхняя оценка времени работы сортировки вставками

- При оценке времени работы алгоритмов часто используется оценка сверху
  - Верхняя оценка дает гарантии на время выполнения алгоритма, не требуя уточнения условий выполнения оценки
  - В некоторых алгоритмах худший случай достигается часто
  - Среднее время работы алгоритмов во многих случаях не лучше, чем худшее

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
6:     A[i + 1] ← key
```

$T_{\text{for}}(n) = O(n)$
$T_{\text{while}}(n) = O(n)$
$T(n) = T_{\text{for}}(n) \times T_{\text{while}}(n) = O(n^2)$



Верхняя оценка времени работы сортировки вставками

- При оценке времени работы алгоритмов часто используется оценка сверху
  - Верхняя оценка дает гарантии на время выполнения алгоритма, не требуя уточнения условий выполнения оценки
  - В некоторых алгоритмах худший случай достигается часто
  - Среднее время работы алгоритмов во многих случаях не лучше, чем худшее

InsertionSort(A)

```
1: for j ← 2 to n do
2:     i ← j − 1
3:     while i > 0 and A[i] > A[i + 1] do
4:         swap(A[i], A[i + 1])
5:         i ← i − 1
6:     A[i + 1] ← key
```

$$T_{\text{for}}(n) = O(n)$$
$$T_{\text{while}}(n) = O(n)$$
$$T(n) = T_{\text{for}}(n) \times T_{\text{while}}(n) = O(n^2)$$

Спасибо за внимание!