Algorithms

Insertion sort

Emanuele Rodolà
rodola@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

2nd semester a.y. 2020/2021 · March 04, 2021

Exercises

In pseudocode or in your favorite language, write an algorithm to solve each of the following problems.

1. Reverse any given sequence of length n:

$$(3, 7, 9, 14) \to (14, 9, 7, 3)$$

2. Given a number x and a sequence $(a_i)_{i=1}^{n}$, find the closest number to x in the sequence.

$$12.1\,,\;\; (3, 31, 7, 11, 52) \to 11$$

3. Given a sequence $(a_i)_{i=1}^{n}$ and a smaller sequence $(b_i)_{i=1}^{m}, m < n$, find the latter inside the former, and return the index of the first occurrence as output.

$$(C, G, A, T, T, G, C, \dots)\,,\;\; (T, T, G \dots) \to 4$$

Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

Intuition: Take one key at a time from the sequence, and insert it into the correct position in the new sequence.

Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

Intuition: Take one key at a time from the sequence, and insert it into the correct position in the new sequence.

	1	2	3	4	5	6
(a)	5	**2**	4	6	1	3

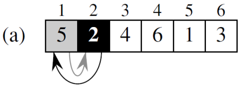
Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

Intuition: Take one key at a time from the sequence, and insert it into the correct position in the new sequence.



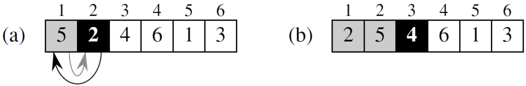
Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \le a'_2 \le \cdots \le a'_n$$

Intuition: Take one key at a time from the sequence, and insert it into the correct position in the new sequence.



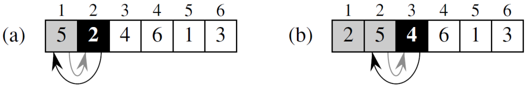
Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

Intuition: Take one key at a time from the sequence, and insert it into the correct position in the new sequence.



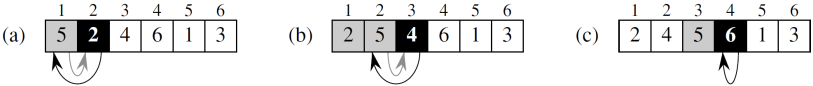
Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \dots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \dots, a'_n)$ such that:

$$a'_1 \le a'_2 \le \cdots \le a'_n$$

Intuition: Take one key at a time from the sequence, and insert it into the correct position in the new sequence.



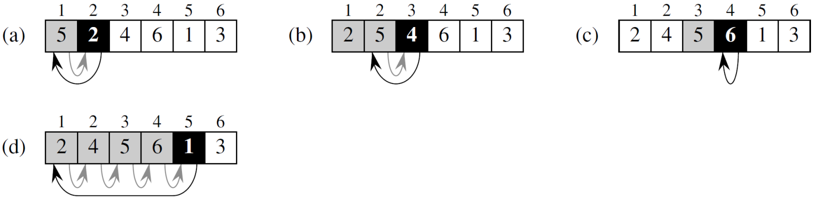
Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

Intuition: Take one key at a time from the sequence, and insert it into the correct position in the new sequence.



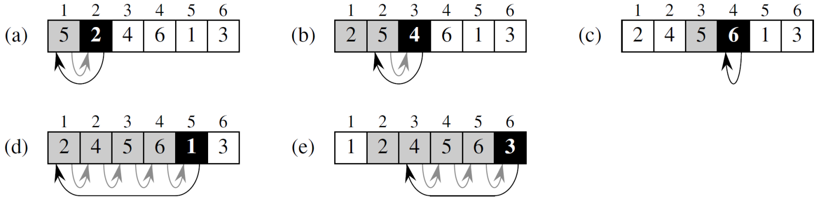
Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

Intuition: Take one key at a time from the sequence, and insert it into the correct position in the new sequence.



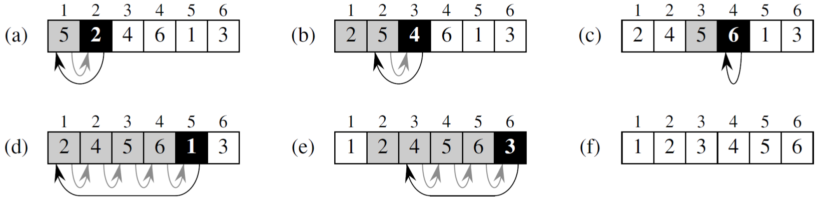
Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$

Output: A reordered sequence $(a'_1, a'_2, \ldots, a'_n)$ such that:

$$a'_1 \leq a'_2 \leq \cdots \leq a'_n$$

Intuition: Take one key at a time from the sequence, and insert it into the correct position in the new sequence.



Pseudocode

INSERTION-SORT(A)

for $j \leftarrow 2$ **to** $length[A]$
    do $key \leftarrow A[j]$
        $\triangleright$ Insert $A[j]$ into the sorted sequence $A[1 \,..\, j-1]$.

Remark: The sequence is sorted in-place.

Pseudocode

```
INSERTION-SORT(A)
for j ← 2 to length[A]
    do key ← A[j]
       ▷ Insert A[j] into the sorted sequence A[1 .. j − 1].
       i ← j − 1
       while i > 0 and A[i] > key
           do A[i + 1] ← A[i]
              i ← i − 1
       A[i + 1] ← key
```

Remark: The sequence is sorted in-place.

Analysis

We want to predict the resources that the algorithm requires.

- Memory
- Bandwidth
- Time
- ...

Analysis

We want to predict the resources that the algorithm requires.

- Memory
- Bandwidth
- Time
- ...

Most of these factors ultimately depend on the size of the input.

As we have seen, we will measure the running time (i.e., the number of steps) as a function of size.

Analysis

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n

Analysis

Insertion-Sort(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$

Analysis

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$

Analysis

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** *length*$[A]$	c_1	n
do *key* $\leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1 \,.\,.\, j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$

Analysis

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	
do $A[i+1] \leftarrow A[i]$	c_6	
$i \leftarrow i-1$	c_7	

Analysis

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1 \,..\, j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	
do $A[i+1] \leftarrow A[i]$	c_6	
$i \leftarrow i-1$	c_7	
$A[i+1] \leftarrow key$	c_8	$n-1$

Analysis

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1 \,..\, j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	
do $A[i+1] \leftarrow A[i]$	c_6	
$i \leftarrow i-1$	c_7	
$A[i+1] \leftarrow key$	c_8	$n-1$

The **while** instruction is not executed a fixed number of times.

It depends on the current number j.

Analysis

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1 \mathinner{..} j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	
do $A[i+1] \leftarrow A[i]$	c_6	
$i \leftarrow i-1$	c_7	
$A[i+1] \leftarrow key$	c_8	$n-1$

The **while** instruction is not executed a fixed number of times.

It depends on the current number j.

We define t_j the number of times **while** is executed for the value j.

Analysis

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
do $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
$i \leftarrow i-1$	c_7	$\sum_{j=2}^{n}(t_j - 1)$
$A[i+1] \leftarrow key$	c_8	$n-1$

The **while** instruction is not executed a fixed number of times.

It depends on the current number j.

We define t_j the number of times **while** is executed for the value j.

Analysis

INSERTION-SORT(A)	*cost*	*times*
for $j \leftarrow 2$ **to** $length[A]$	c_1	n
do $key \leftarrow A[j]$	c_2	$n-1$
$\triangleright$ Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
$i \leftarrow j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
do $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
$i \leftarrow i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
$A[i+1] \leftarrow key$	c_8	$n-1$

Total running time $T(n)$:

$$c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n}(t_j-1) + c_7 \sum_{j=2}^{n}(t_j-1) + c_8(n-1)$$

Best-case analysis

"We define t_j the number of times **while** is executed for the value j"

This means that there is a best-case running time, obtained with $t_j = 1$.

Best-case analysis

"We define t_j the number of times **while** is executed for the value j"

This means that there is a best-case running time, obtained with $t_j = 1$.

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j$$
$$+ c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

Best-case analysis

"We define t_j the number of times **while** is executed for the value j"

This means that there is a best-case running time, obtained with $t_j = 1$.

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1)$$

Best-case analysis

"We define t_j the number of times **while** is executed for the value j"

This means that there is a best-case running time, obtained with $t_j = 1$.

$$T(n) = (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)$$

Best-case analysis

"We define t_j the number of times **while** is executed for the value j"

This means that there is a best-case running time, obtained with $t_j = 1$.

$$T(n) = \underbrace{(c_1 + c_2 + c_4 + c_5 + c_8)}_{a} n - \underbrace{(c_2 + c_4 + c_5 + c_8)}_{b}$$

Best-case analysis

"We define t_j the number of times **while** is executed for the value j"

This means that there is a best-case running time, obtained with $t_j = 1$.

$$T(n) = an + b$$

where a and b are constants.

The best-case cost of insertion sort is linear in n.

Best-case analysis

"We define t_j the number of times **while** is executed for the value j"

This means that there is a best-case running time, obtained with $t_j = 1$.

$$T(n) = an + b$$

where a and b are constants.

The best-case cost of insertion sort is linear in n.

When does the best case happen?

Best-case analysis

"We define t_j the number of times **while** is executed for the value j"

This means that there is a best-case running time, obtained with $t_j = 1$.

$$T(n) = an + b$$

where a and b are constants.

The best-case cost of insertion sort is linear in n.

When does the best case happen?

When the input sequence is already sorted.

Worst-case analysis

What if the input sequence is sorted in decreasing order?

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

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

$$+ c_6 \sum_{j=2}^{n} (j-1) + c_7 \sum_{j=2}^{n} (j-1) + c_8(n-1)$$

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

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

$$+ c_6 \sum_{j=2}^{n} (j-1) + c_7 \sum_{j=2}^{n} (j-1) + c_8(n-1)$$

$\sum_{i=1}^{n} i =$

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} j$$
$$+ c_6 \sum_{j=2}^{n} (j-1) + c_7 \sum_{j=2}^{n} (j-1) + c_8(n-1)$$

$\sum_{i=1}^{n} i = (1+n) +$

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

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

$$+ c_6 \sum_{j=2}^{n} (j-1) + c_7 \sum_{j=2}^{n} (j-1) + c_8(n-1)$$

$\sum_{i=1}^{n} i = (1+n) + \underbrace{(2+(n-1))}_{1+n} +$

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} j$$
$$+ c_6 \sum_{j=2}^{n} (j-1) + c_7 \sum_{j=2}^{n} (j-1) + c_8(n-1)$$

$\sum_{i=1}^{n} i = (1+n) + \underbrace{(2+(n-1))}_{1+n} + \underbrace{(3+(n-2))}_{1+n} + \cdots$

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

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

$$+ c_6 \sum_{j=2}^{n} (j-1) + c_7 \sum_{j=2}^{n} (j-1) + c_8(n-1)$$

$$\sum_{i=1}^{n} i = (1+n) + \underbrace{(2+(n-1))}_{1+n} + \underbrace{(3+(n-2))}_{1+n} + \cdots = (1+n)\frac{n}{2}$$

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

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

$$+ c_6 \sum_{j=2}^{n} (j-1) + c_7 \sum_{j=2}^{n} (j-1) + c_8(n-1)$$

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1)$$

Worst-case analysis

What if the input sequence is sorted in decreasing order?

In this case, $t_j = j$ since we will compare each number with the entire sorted subsequence.

$$T(n) = an^2 + bn + c$$

where a, b, c are constants.

The worst-case cost of insertion sort is quadratic in n.

Average-case analysis

For a given random sequence of numbers, we observe that, on average:

- $A[j] >$ half the elements in $A[1 \ldots j-1]$, and
- $A[j] <$ half the elements in $A[1 \ldots j-1]$

Average-case analysis

For a given random sequence of numbers, we observe that, on average:

- $A[j] >$ half the elements in $A[1 \ldots j-1]$, and
- $A[j] <$ half the elements in $A[1 \ldots j-1]$

So, on average, $t_j = \frac{j}{2}$.

Average-case analysis

For a given random sequence of numbers, we observe that, on average:

- $A[j] >$ half the elements in $A[1 \dots j-1]$, and
- $A[j] <$ half the elements in $A[1 \dots j-1]$

So, on average, $t_j = \frac{j}{2}$.

Following the steps from the worst case, we get again:

$$T(n) = an^2 + bn + c$$

for some constants a, b, c.

Average-case analysis

For a given random sequence of numbers, we observe that, on average:

- $A[j] >$ half the elements in $A[1 \dots j-1]$, and
- $A[j] <$ half the elements in $A[1 \dots j-1]$

So, on average, $t_j = \frac{j}{2}$.

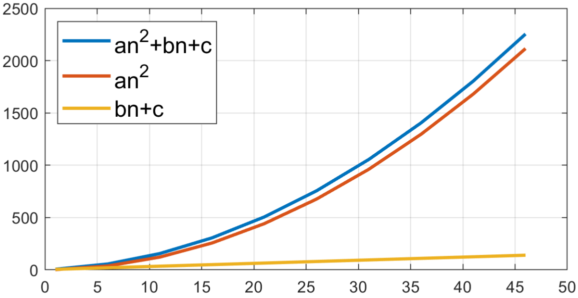
Following the steps from the worst case, we get again:

$$T(n) = an^2 + bn + c$$

for some constants a, b, c.

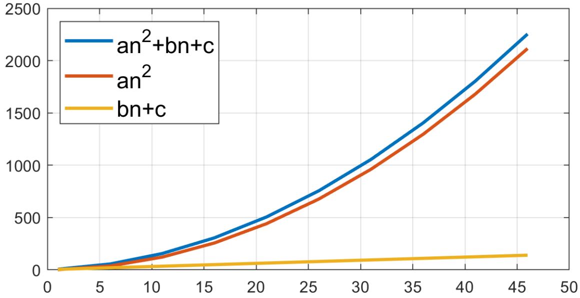
In our analysis, we will often concentrate on studying the worst case, since it gives us a guaranteed upper bound on the total cost.

Order of growth



an^2 dominates the lower-order terms.

Order of growth



an^2 dominates the lower-order terms.

We say that insertion sort has worst-case running time of $\Theta(n^2)$.

Exercises

Solve the following exercises:

1. Write an algorithm in pseudocode to perform linear search.

   Given a sequence of numbers $A = (a_1, \ldots, a_n)$ and a number v, find an index i such that $v = A[i]$. Return a special number if v can not be found in the sequence.

   The search must be done by simple linear scanning through the sequence.

2. How many elements must be checked on average, and in the best and worst cases?

3. What are the average-case, best-case, and worst-case running times of linear search in Θ-notation?

Suggested reading

"Introduction to Algorithms – 2nd Ed.", Cormen et al.

- Chapter 2.1, skipping the "Loop invariants" and "Pseudocode conventions" paragraphs
- Chapter 2.2