Complexity

Sorting
Counting Sort
Correctness
Complexity
Order Notation
O()
And Friends

Complexity

Complexity

Sorting
Counting Sort
Correctness
Complexity
Order Notation
O()
And Friends

Sorting

- Bubble sort
- Selection sort
- Insertion sort
- Shell sort
- Merge sort
- Heap sort
- Quick sort

How do we sort one million records?

How do we sort one billion **16-bit integers**?

How do we sort one trillion **4-bit integers**?

Sorting

- Bubble sort
- Selection sort
- Insertion sort
- Shell sort
- Merge sort
- Heap sort
- Quick sort

How do we sort one million records?
How do we sort one billion **16-bit integers**?
How do we sort one trillion **4-bit integers**?

Sorting

- Bubble sort
- Selection sort
- Insertion sort
- Shell sort
- Merge sort
- Heap sort
- Quick sort

How do we sort one million records?
How do we sort one billion **16-bit integers**?
How do we sort one trillion **4-bit integers**?

Counting Sort

For n numbers in the range 0 to k:

```
for x from 0 to k do
    count[x] ← 0
for all input number x do
    increment count[x]
for x from 0 to k do
    print x count[x] times
```

Correctness?

Complexity?

Counting Sort

For n numbers in the range 0 to k:

for x from 0 to k **do**
    count[x] $\leftarrow$ 0
for all input number x **do**
    increment count[x]
for x from 0 to k **do**
    print x count[x] times

Correctness?

Complexity?

Complexity
Sorting
Counting Sort
Correctness
Complexity
Order Notation
O()
And Friends

Correctness

property 1: output is in sorted order
proof sketch: loop in (5) increments x

property 2: output contains same numbers as input
invariant: for each value,
    remaining input + tally in count array = total
proof sketch:
initialized/established: before line 3
maintained: through lines 3-4
at termination: no remaining input
    each number printed *count* times
    therefore, output has same numbers as input

Complexity
Sorting
Counting Sort
Correctness
Complexity
Order Notation
O()
And Friends

Correctness

property 1: output is in sorted order
proof sketch: loop in (5) increments x

property 2: output contains same numbers as input
invariant:

invariant: for each value,
remaining input + tally in count array = total
proof sketch:
initialized/established: before line 3
maintained: through lines 3-4
at termination: no remaining input
each number printed *count* times
therefore, output has same numbers as input

Correctness

property 1: output is in sorted order
proof sketch: loop in (5) increments x

property 2: output contains same numbers as input
invariant: for each value,
remaining input + tally in count array = total
proof sketch:
initialized/established: before line 3
maintained: through lines 3-4
at termination: no remaining input
each number printed *count* times
therefore, output has same numbers as input

Correctness

Complexity
- Sorting
- Counting Sort
- Correctness
- Complexity
- Order Notation
- O()
- And Friends

property 1: output is in sorted order
proof sketch: loop in (5) increments x

property 2: output contains same numbers as input
invariant: for each value,
remaining input + tally in count array = total
proof sketch:
initialized/established: before line 3
maintained: through lines 3-4
at termination: no remaining input
each number printed *count* times
therefore, output has same numbers as input

Correctness

property 1: output is in sorted order
proof sketch: loop in (5) increments x

property 2: output contains same numbers as input
invariant: for each value,
remaining input + tally in count array = total
proof sketch:
initialized/established: before line 3
maintained: through lines 3-4
at termination: no remaining input
each number printed *count* times
therefore, output has same numbers as input

Correctness

property 1: output is in sorted order
proof sketch: loop in (5) increments x

property 2: output contains same numbers as input
invariant: for each value,
remaining input + tally in count array = total
proof sketch:
initialized/established: before line 3
maintained: through lines 3-4
at termination: no remaining input
each number printed *count* times
therefore, output has same numbers as input

Correctness

property 1: output is in sorted order
proof sketch: loop in (5) increments x

property 2: output contains same numbers as input
invariant: for each value,
remaining input + tally in count array = total
proof sketch:
initialized/established: before line 3
maintained: through lines 3-4
at termination: no remaining input
each number printed *count* times
therefore, output has same numbers as input

Complexity

For n numbers in the range 0 to k:

for x from 0 to k **do**	
count[x] $\leftarrow$ 0	$O(k)$
for all input number x **do**	
increment count[x]	$O(n)$
for x from 0 to k **do**	
print x count[x] times	$O(k+n)$

$$O(k+n+k+n) = O(2k+2n) = O(k+n) \neq O(n \lg n)$$

Complexity
Sorting
Counting Sort
Correctness
Complexity
Order Notation
O()
And Friends

Complexity

For n numbers in the range 0 to k:

for x from 0 to k **do**	
count[x] $\leftarrow$ 0	$O(k)$
for all input number x **do**	
increment count[x]	$O(n)$
for x from 0 to k **do**	
print x count[x] times	$O(k+n)$

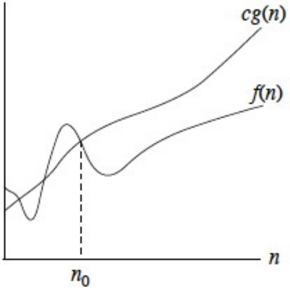
$$O(k+n+k+n) = O(2k+2n) = O(k+n) \neq O(n \lg n)$$

Order Notation

Rules of thumb:

- ignore constant factors
- ignore 'start-up' costs
- focus on upper bound (worst-case scenario)

$$f(n) = O(g(n))$$



eg, running time is $O(n) = n \lg n$

$O()$

Definition

$$O(g(n)) = \{f(n) : \exists \text{ positive constants } c, n_0 \text{ such that } f(n) \leq cg(n) \text{ for all } n \geq n_0\}$$

We can upper-bound (the tail of) f by scaling g by a consant

Example

Show that $O(n^2) = 3n^2 + 6n + 1$: pick an appropriate c and n_0.

1. $0.002n^2 - 35000n + 2^{80}$
2. $O(n^2)$ vs $O(n^3)$
3. $O(n \lg n)$ vs $O(n)$
4. $O(n^2)$ vs $O(n^6)$ vs $O(2^n)$

What does n signify?

$O()$

Definition

$$O(g(n)) = \{f(n) : \exists \text{ positive constants } c, n_0 \text{ such that } f(n) \leq cg(n) \text{ for all } n \geq n_0\}$$

We can upper-bound (the tail of) f by scaling g by a consant

Example

Show that $O(n^2) = 3n^2 + 6n + 1$: pick an appropriate c and n_0.

1. $0.002n^2 - 35000n + 2^{80}$
2. $O(n^2)$ vs $O(n^3)$
3. $O(n \lg n)$ vs $O(n)$
4. $O(n^2)$ vs $O(n^6)$ vs $O(2^n)$

What does n signify?

$O()$

Definition

$$O(g(n)) = \{f(n) : \exists \text{ positive constants } c, n_0 \text{ such that } f(n) \leq cg(n) \text{ for all } n \geq n_0\}$$

We can upper-bound (the tail of) f by scaling g by a consant

Example

Show that $O(n^2) = 3n^2 + 6n + 1$: pick an appropriate c and n_0.

1. $0.002n^2 - 35000n + 2^{80}$
2. $O(n^2)$ vs $O(n^3)$
3. $O(n \lg n)$ vs $O(n)$
4. $O(n^2)$ vs $O(n^6)$ vs $O(2^n)$

What does n signify?

And Friends

Upper bound('order of'):

$$O(g(n)) = \{f(n) : \exists \text{ positive constants } c, n_0 \\ \text{such that } f(n) \leq cg(n) \text{ for all } n \geq n_0\}$$

Lower bound:

$$\Omega(g(n)) = \{f(n) : \exists \text{ positive constants } c, n_0 \\ \text{such that } f(n) \geq cg(n) \text{ for all } n \geq n_0\}$$

Tight bound:

$$\Theta(g(n)) = \{f(n) : \exists \text{ positive constants } c_1, c_2, n_0 \\ \text{such that } c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}$$