Set 02: Tools for Asymptotic Analysis

CS240: Data Structures and Data Management

Jérémy Barbay

Outline

Comparison of Three Sorting Algorithms

How do we find out which of these algorithms is the best?

- Selection Sort
- Merge Sort
- Counting Sort

Running Time

- The **running time** $T_A(x)$ of an algorithm A for a particular input x is the time that the algorithm requires to solve the input x.
- The **worst-case running time** $T_A(n)$ of an algorithm A is a function of the size n of the input, where $T_A(n)$ is the *largest time* required to solve an input of size n:

$$T_A(n) = \max\{\, T_A(x) \mid |x| = n \}.$$

- The **average-case running time** $T_A^{(\mathrm{avg})}(n)$ of an algorithm A is a function of the size n of the input, where $T_A(n)$ is a *average of times* required to solve all inputs of size n:

$$T_A^{(\mathrm{avg})}(n) = \mathrm{avg}\{\, T_A(x) \mid |x| = n \}.$$

$\boxed{\text{How to measure time?}}$

Elementary operation

An *Elementary operation* is an operation whose time can be bounded by a constant.

Examples:

- arithmetic operation: elementary.
- maximum:
  - of an array: non elementary,
  - of two numbers: elementary.
- comparison: elementary.
- pattern matching: non elementary.
- program flow control: elementary.
- concatenation of strings: non elementary (in general).
- factorial: non elementary.

Random Access Machine (RAM) Model

Definition (RAM model)

- Memory consists of an unbounded number of cells.
- Each cell holds a number or a character.
- Basic operations take one unit of time.
- Other operations are described in terms of basic operations.
- Instructions are executed in sequence (no parallelism).

First Asymptotic Notation

Definition

A function $f(n)$ is in $O(g(n))$ iff
there exist $c > 0$ and $n_0 > 0$,
such that $\forall n > n_0, 0 \leq f(n) \leq cg(n)$.

Notation:
$f(n) \in O(g(n))$,
or $f(n) = O(g(n))$.

Selection sort

```
for i:=1 to n
| min:=i;
| for j:=i+1 to n
| | if a[min]>a[j] min:=j;
| tmp:=a[i]; a[i]:=a[min]; a[min]:=tmp;
```

$$\sum_{i=1}^{n}\left(1+\sum_{j=i+1}^{n} 1\right) = n + \sum_{i=1}^{n}(n-i) \leq n^2 + n$$

The running time of selection sort is $O(n^2)$.

Merge sort

```
function merge(from,mid,to)
| copy a[from..mid] to a new array b
| copy a[mid+1..to] to a new array c
| add infinity to both b and c as the last element
| k:=1; m:=1;
| for j:=from to to
| | if b[k]<c[m] then
| | | a[j]:=b[k]; k++;
| | else
| | | a[j]:=c[m]; m++;
```

The running time of the function merge is $O(\text{to} - \text{from})$.

Merge sort

```
function sort(from,to)
| if (from>to)
| | mid:=floor((from+to)/2);
| | sort(from,mid); sort(mid+1,to);
| | merge(from,mid,to);
```

The running time of merge sort is $O(n \log n)$.

Counting sort

```
clear array count[1..1000];
for i:=1 to n
| count[a[i]]++;

k:=1;
for i:=1 to 1000
| for j:=1 to count[i];
| | a[k]:=i; k++;
```

The running time of counting sort is $O(n)$. The algorithm assumes that the values in the array are in $\{1, \ldots, 1000\}$.

Comparison of Three Sorting Algorithms

How do we find out which of these algorithms is the best?

- Selection Sort
- Merge Sort
- Counting Sort

This analysis seems to suggest that:

- "counting sort" is better than "merge sort"
- which is better than "selection sort".

Does this analysis have any relationship to the actual running times on a real computer?

Experimental results

Size of array	**Counting sort**	**Merge sort**	**Selection sort**	**Exponential algorithm**
n	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(2^n)$
10	ε	ε	ε	ε
50	ε	ε	ε	2 weeks
100	ε	ε	ε	2800 univ.
1000	0.01s	ε	ε	—
10000	0.01s	0.03s	0.27s	—
100000	0.06s	0.15s	26.5s	—
1 mil.	0.74s	1.6s	44.2m	—
10 mil.	7.4s	16.5s	3.1d	—

(As measured on a Pentium 4 2Ghz computer.)

Summary

- Even with today's fast processors, better algorithms matter.
- Asymptotic analysis allows us to easily analyze and compare algorithms without considering details specific to a particular computer.
- For a single problem there can be several solutions with different complexities.

Outline

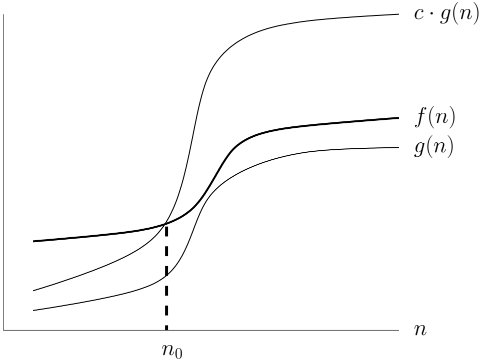
Asymptotic Notations

- Function $f(n)$ maps non-negative integers to real numbers.
- Group similarly growing functions together.
- Asymptotic analysis is not restricted to CS.
- In CS, n typically represents some size of the input.
- Characterize worst, best or average case for any of the algorithm comparison criteria.

Big-O formal Definition

- Used to express an upper bound

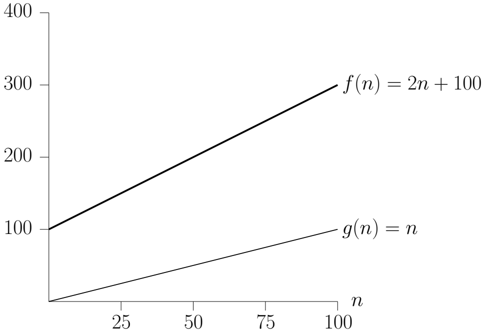
Definition: $f(n)$ is $O(g(n))$ iff $\exists$ a real $c > 0$ and an integer $n_0 > 0$ such that $\forall\ n \geq n_0$, $f(n) \leq cg(n)$



- f grows no faster than g.
- Want tight bounds and simple terms.

Example

- Show $2n + 100 \in O(n)$



- You **must** provide a valid c and n_0 to prove formally.
- **Example**: Prove $(n+1)^5 \in O(n^5)$

Informal Terms

constant	$O(1)$	ex: 1, 2, 6,...
logarithmic	$O(\log n)$	ex: $2\log n$, $1 + \log n$, ...
linear	$O(n)$	ex: $2n$, $5n + 4$, $n + \log n$, ...
quadratic	$O(n^2)$	$(4n)^2$, $n^2 + 4n$, $n^2 + logn$, ...
polynomial	$O(n^k)$	for $k \geq 0$
exponential	$O(a^n)$	for $a > 1$

Properties of $O()$

The following claims can be proven directly from the definition.

1. if $f(n) \in O(g(n))$ and $c > 0$ is a constant
   then $cf(n) \in O(g(n))$
2. **Maximum rule.** If $t(n) \in O(f(n) + g(n))$
   then $t(n) \in O(\max(f(n), g(n)))$
3. **Transitivity.** If $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$
   then $f(n) \in O(h(n))$
4. $n^k \in O(a^n)$ for all constants $k > 0$ and $a > 1$
5. Similarly: $\log^k n \in O(n^a)$ for all constants $k > 0$ and $a > 0$
6. $n^k \in O(n^\ell)$ for all constants $k > 0$ and $\ell \geq k$
7. if $f(n) \in O(f'(n))$ and $g(n) \in O(g'(n))$ then
   $f(n)g(n) \in O(f'(n)g'(n))$

Examples:

- $3.8n^2 + 2.6n^3 + 10n \log n \in O(n^3)$? true
- $10^{100} n \in O(n)$? true
- $(n+1)! \in O(n!)$? false
- $2^{2n} \in O(2^n)$? false
- $n \in O(n^{10})$? true

Properties

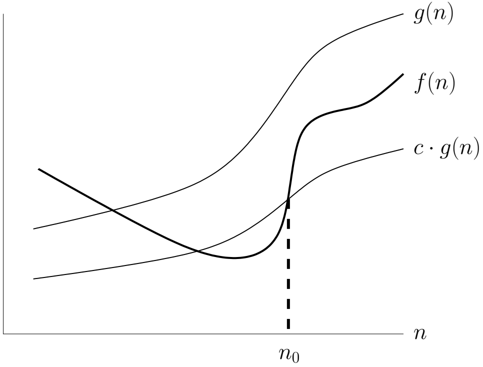
Prove these from the definition

1. $f(n) \in O(af(n))$, $a > 0$
2. if $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$ then $f(n) \in O(h(n))$
3. $[f(n) + g(n)] \in O\left(\textsc{Max}\left\{f(n),\ g(n)\right\}\right)$
4. $a_0 + a_1x^1 + \ldots + a_nx^n \in O(x^n)$ in x for n fixed , $a_n > 0$.
5. $n^x \in O(a^n)$, $x > 0$, $a > 1$
6. $log^x n \in O(n^y)$, $x > 0$, $y > 0$

Big-Ω Notation

- What if we want to express a lower bound?

Definition: $f(n)$ is $\Omega(g(n))$ iff $\exists$ a real $c > 0$ and an integer $n_0 > 0$ such that $\forall\ n \geq n_0$, $f(n) \geq cg(n)$

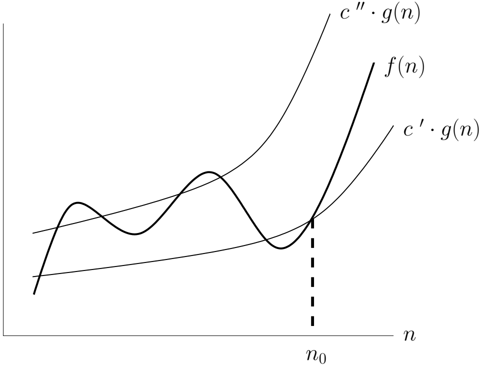


- f grows no slower than g
- **Example**: Prove $n^3 \lg n \in \Omega(n^3)$

Big-Θ Notation

- f grows at the same rate as g

Definition: $f(n)$ is $\Theta(g(n))$ iff $\exists$ reals $c', \; c'' > 0$ and an integer $n_0 > 0$ such that $\forall \; n \geq n_0$, $c'g(n) \leq f(n) \leq c''g(n)$



- g is an asymptotically tight bound
- $f(n) \in \Theta(g(n))$ iff $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$
- **Example**: Prove $3 \lg n + \lg \lg n \in \Theta(\log n)$

Little-o and ω Notation

- $f(n) \in o(g(n))$: f grows strictly slower than g
  - it is not an asymptotically tight upper bound

Definition: $f(n)$ is $o(g(n))$ iff $\forall$ real $c > 0$, $\exists$ an integer $n_0 > 0$ such that $\forall\ n \geq n_0$, $f(n) < cg(n)$

- Equivalently, $f(n) \in o(g(n))$ iff

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

- Reciprocal for ω as a non-tight lower bound
- **Example**: Prove $\ln n \in o(n)$

Overview

Notation	**Definition**
$f(n) \in O(g(n))$	There exists $c > 0$ and $n_0 > 0$ s.t. $(\forall n > n_0)(0 \leq f(n) \leq cg(n))$
$f(n) \in \Omega(g(n))$	There exists $c > 0$ and $n_0 > 0$ s.t. $(\forall n > n_0)(f(n) \geq cg(n) \geq 0)$
$f(n) \in \Theta(g(n))$	$f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$
$f(n) \in o(g(n))$	For any $c > 0$ there exists $n_0 > 0$ s.t. $(\forall n > n_0)(0 \leq f(n) < cg(n))$
$f(n) \in \omega(g(n))$	For any $c > 0$ there exists $n_0 > 0$ s.t. $(\forall n > n_0)(f(n) > cg(n) \geq 0)$

Exercise:

Can rules similar to the ones we mentioned in case of O notation be applied in case of the other notations?

- if $f(n) \in \omega(g(n))$ then $f(n) \notin O(g(n))$
- if $f(n) \in O(g(n))$ then $f(n) \notin \omega(g(n))$
- but there are functions where $f(n) \notin O(g(n))$ and $f(n) \notin \Omega(g(n))$

How to Prove Negative Results?

Definition

$$f(n) \in O(g(n))$$

$$\Leftrightarrow$$

$$\exists c > 0, \exists n_0 > 0, \forall n > n_0, 0 \le f(n) \le cg(n)$$

Negation:

$$f(n) \notin O(g(n))$$

$$\Leftrightarrow$$

$$\forall c > 0, \forall n_0 > 0, \exists n > n_0, f(n) < 0 \text{ or } f(n) > cg(n))$$

Example:

$$(n+1)! \notin O(n!)$$

We know $(n+1)! = (n+1) \cdot n!$.
For any $c > 0$ and $n_0 > 0$, take $n = \lceil c \rceil \lceil n_0 \rceil$.
Then:

$$(\lceil c \rceil \lceil n_0 \rceil + 1)(\lceil c \rceil \lceil n_0 \rceil)! > c(\lceil c \rceil \lceil n_0 \rceil)!$$

Summary

- f needs to be asymptotically non-negative
- What does it mean for an algorithm to be $O(f)$?
- Ω does **not** mean best case!
- Caution: Asymptotic analysis is not always the whole story.