ITMO UNIVERSITY

How to Win Coding Competitions: Secrets of Champions

Week 2: Computational complexity. Linear data structures
Lecture 1: Big O notation. Computational complexity

Pavel Krotkov
Saint Petersburg 2016

- consider two functions $f(x)$ and $g(x)$ defined on $\mathbb{R}$

- consider two functions $f(x)$ and $g(x)$ defined on $\mathbb{R}$
- how can we compare values of these functions?

- consider two functions $f(x)$ and $g(x)$ defined on $\mathbb{R}$
- how can we compare values of these functions?
- some obvious ideas
  - $\forall x : f(x) = g(x)$
  - $\forall x : f(x) < g(x)$
  - $\forall x : f(x) > 2 \times g(x)$

- consider two functions $f(x)$ and $g(x)$ defined on $\mathbb{R}$
- how can we compare values of these functions?
- some obvious ideas
  - $\forall x : f(x) = g(x)$
  - $\forall x : f(x) < g(x)$
  - $\forall x : f(x) > 2 \times g(x)$

We can't introduce equivalence classes for functions based on these comparisons.

Let's find a way to introduce equivalence classes for functions.

Let's find a way to introduce equivalence classes for functions.

$f(x) = \Theta(g(x))$

Let's find a way to introduce equivalence classes for functions.

$f(x) = \Theta(g(x))$

- $\exists x_0 : \begin{cases} \exists c_1 : \forall x > x_0 : f(x) < c_1 \times g(x) \\ \exists c_2 : \forall x > x_0 : f(x) > c_2 \times g(x) \end{cases}$

Let's find a way to introduce equivalence classes for functions.

$f(x) = \Theta(g(x))$

- $\exists x_0 : \begin{cases} \exists c_1 : \forall x > x_0 : f(x) < c_1 \times g(x) \\ \exists c_2 : \forall x > x_0 : f(x) > c_2 \times g(x) \end{cases}$

f is bounded *above* and *below* by g asymptotically

Now we can introduce two related notations.

Now we can introduce two related notations.

- $f(x) = O(g(x))$

Now we can introduce two related notations.

- $f(x) = O(g(x))$
  - $\exists x_0, c : \forall x > x_0 : f(x) < c \times g(x)$

Now we can introduce two related notations.

- $f(x) = O(g(x))$
  - $\exists x_0, c : \forall x > x_0 : f(x) < c \times g(x)$
  - f is bounded *above* by g assymptotically

Now we can introduce two related notations.

- $f(x) = O(g(x))$
  - $\exists x_0, c : \forall x > x_0 : f(x) < c \times g(x)$
  - f is bounded *above* by g assymptotically
- $f(x) = \Omega(g(x))$

Now we can introduce two related notations.

- $f(x) = O(g(x))$
  - $\exists x_0, c : \forall x > x_0 : f(x) < c \times g(x)$
  - f is bounded *above* by g assymptotically
- $f(x) = \Omega(g(x))$
  - $\exists x_0, c : \forall x > x_0 : f(x) > c \times g(x)$

Now we can introduce two related notations.

- $f(x) = O(g(x))$
  - $\exists x_0, c : \forall x > x_0 : f(x) < c \times g(x)$
  - f is bounded *above* by g assymptotically
- $f(x) = \Omega(g(x))$
  - $\exists x_0, c : \forall x > x_0 : f(x) > c \times g(x)$
  - f is bounded *below* by g assymptotically

Now we can introduce two related notations.

- $f(x) = O(g(x))$
  - $\exists x_0, c : \forall x > x_0 : f(x) < c \times g(x)$
  - f is bounded *above* by g assymptotically
- $f(x) = \Omega(g(x))$
  - $\exists x_0, c : \forall x > x_0 : f(x) > c \times g(x)$
  - f is bounded *below* by g assymptotically

$$f(x) = \Theta(g(x)) \Leftrightarrow \begin{cases} f(x) = O(g(x)) \\ f(x) = \Omega(g(x)) \end{cases}$$

Let $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$.

Let $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$.

- $f_1(x) + f_2(x) = O(\max(g_1(x), g_2(x)))$

Let $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$.

- $f_1(x) + f_2(x) = O(\max(g_1(x), g_2(x)))$
- $f_1(x) \times f_2(x) = O(g_1(x) \times g_2(x))$

Let $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$.

- $f_1(x) + f_2(x) = O(\max(g_1(x), g_2(x)))$
- $f_1(x) \times f_2(x) = O(g_1(x) \times g_2(x))$
- $c \times f_1(x) = O(g_1(x))$

Let $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$.

- $f_1(x) + f_2(x) = O(\max(g_1(x), g_2(x)))$
- $f_1(x) \times f_2(x) = O(g_1(x) \times g_2(x))$
- $c \times f_1(x) = O(g_1(x))$
- $f_1 + c = O(g_1(x))$

Let $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$.

- $f_1(x) + f_2(x) = O(\max(g_1(x), g_2(x)))$
- $f_1(x) \times f_2(x) = O(g_1(x) \times g_2(x))$
- $c \times f_1(x) = O(g_1(x))$
- $f_1 + c = O(g_1(x))$

Examples

Let $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$.

- $f_1(x) + f_2(x) = O(\max(g_1(x), g_2(x)))$
- $f_1(x) \times f_2(x) = O(g_1(x) \times g_2(x))$
- $c \times f_1(x) = O(g_1(x))$
- $f_1 + c = O(g_1(x))$

Examples

- $\log_2 x = O(x)$

Let $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$.

- $f_1(x) + f_2(x) = O(\max(g_1(x), g_2(x)))$
- $f_1(x) \times f_2(x) = O(g_1(x) \times g_2(x))$
- $c \times f_1(x) = O(g_1(x))$
- $f_1 + c = O(g_1(x))$

Examples

- $\log_2 x = O(x)$
- $3 \times x^2 - 5 \times x + 7 = \Theta(x^2)$

Let $f_1(x) = O(g_1(x))$ and $f_2(x) = O(g_2(x))$.

- $f_1(x) + f_2(x) = O(\max(g_1(x), g_2(x)))$
- $f_1(x) \times f_2(x) = O(g_1(x) \times g_2(x))$
- $c \times f_1(x) = O(g_1(x))$
- $f_1 + c = O(g_1(x))$

Examples

- $\log_2 x = O(x)$
- $3 \times x^2 - 5 \times x + 7 = \Theta(x^2)$
- $x \times (\ln x + \ln \ln x) \times \ln \ln x = O(x \times \ln x \times \ln \ln x)$

Consider a program

```
j = 1
for i = 1 to n
  j = j × i
```

Consider a program

```
j = 1
for i = 1 to n
    j = j × i
```

This cycle performs n iterations. Every iteration takes constant amount of operations.

Consider a program

```
j = 1
for i = 1 to n
  j = j × i
```

This cycle performs n iterations. Every iteration takes constant amount of operations.

Asymptotical complexity of this program is $O(n)$.

More complicated case

```
j = 1
for i = 1 to n
  for j = 1 to i
    print j
```

More complicated case

```
j = 1
for i = 1 to n
  for j = 1 to i
    print j
```

This cycle performs $\frac{n \times (n+1)}{2}$ iterations. Every iteration takes constant amount of operations.

More complicated case

```
j = 1
for i = 1 to n
  for j = 1 to i
    print j
```

This cycle performs $\frac{n \times (n+1)}{2}$ iterations. Every iteration takes constant amount of operations.

Asymptotical complexity of this program is $O(n^2)$. Constant doesn't matter when we are talking about asymptotical complexity.

Consider you have a program which takes one number n as an input. Let's try to estimate its run time from its complexity and value of n.

Consider you have a program which takes one number n as an input. Let's try to estimate its run time from its complexity and value of n.

complexity	$n = 10$	$n = 20$	$n = 100$	$n = 10^4$	$n = 10^5$	$n = 10^8$
$O(n!)$	$\approx 3 \times 10^5$	$\approx 2 \times 10^{18}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$
$O(2^n)$	1024	$\approx 10^6$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$
$O(n^2)$	100	400	10^4	10^8	10^{10}	10^{16}
$O(n \times \log_2 n)$	≈ 30	≈ 100	≈ 700	$\approx 3 \times 10^5$	$\approx 2 \times 10^6$	$\approx 3 \times 10^9$

Consider you have a program which takes one number n as an input. Let's try to estimate its run time from its complexity and value of n.

complexity	$n = 10$	$n = 20$	$n = 100$	$n = 10^4$	$n = 10^5$	$n = 10^8$
$O(n!)$	$\approx 3 \times 10^5$	$\approx 2 \times 10^{18}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$
$O(2^n)$	1024	$\approx 10^6$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$
$O(n^2)$	100	400	10^4	10^8	10^{10}	10^{16}
$O(n \times \log_2 n)$	≈ 30	≈ 100	≈ 700	$\approx 3 \times 10^5$	$\approx 2 \times 10^6$	$\approx 3 \times 10^9$

You can suppose that average amount of operations per second CPU can perform is $\approx 3 \times 10^8$. That is precise enough to check if yor program will pass Time Limit.

Consider you have a program which takes one number n as an input. Let's try to estimate its run time from its complexity and value of n.

complexity	$n = 10$	$n = 20$	$n = 100$	$n = 10^4$	$n = 10^5$	$n = 10^8$
$O(n!)$	$\approx 3 \times 10^5$	$\approx 2 \times 10^{18}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$
$O(2^n)$	1024	$\approx 10^6$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$
$O(n^2)$	100	400	10^4	10^8	10^{10}	10^{16}
$O(n \times \log_2 n)$	≈ 30	≈ 100	≈ 700	$\approx 3 \times 10^5$	$\approx 2 \times 10^6$	$\approx 3 \times 10^9$

You can suppose that average amount of operations per second CPU can perform is $\approx 3 \times 10^8$. That is precise enough to check if yor program will pass Time Limit.

Consider you have a program which takes one number n as an input. Let's try to estimate its run time from its complexity and value of n.

complexity	$n = 10$	$n = 20$	$n = 100$	$n = 10^4$	$n = 10^5$	$n = 10^8$
$O(n!)$	$\approx 3 \times 10^5$	$\approx 2 \times 10^{18}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$
$O(2^n)$	1024	$\approx 10^6$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$	$> 10^{20}$
$O(n^2)$	100	400	10^4	10^8	10^{10}	10^{16}
$O(n \times \log_2 n)$	≈ 30	≈ 100	≈ 700	$\approx 3 \times 10^5$	$\approx 2 \times 10^6$	$\approx 3 \times 10^9$

You can suppose that average amount of operations per second CPU can perform is $\approx 3 \times 10^8$. That is precise enough to check if yor program will pass Time Limit.

Thank you
for your attention!