Analysis
CS 240

Alex Vondrak

ajvondrak@csupomona.edu

Winter 2012

Algorithms

Definition (Algorithm)

A precise step-by-step plan for a computational procedure that begins with an input value and yields an output value in a finite number of steps

Example (The Searching Problem)

Input: Any array of **int**s, plus a single **int** to search for.

Output: The value **true** if the **int** is an element of the array, or the value **false** if it is not.

```
boolean search(int needle, int[] haystack) {
   for (int element : haystack)
      if (element == needle) return true;
   return false;
}
```

Comparing Algorithms

- Correctness

Example (The Searching Problem)

```
boolean incorrect(int needle, int[] haystack) {
  return (haystack[0] == needle);
}
```

- Speed

Problems

How do we measure speed?

- `System.currentTimeMillis()`
- `$ time ...`

Counting Seconds vs Counting Steps

Counting Seconds

Literal running times vary. . .

- More/less sophisticated hardware (e.g., processor)
- More/less sophisticated software (e.g., OS, compiler, etc.)
- Random chance (e.g., what your OS is doing at the moment)

Counting Steps

To normalize our units of comparison, we can agree to count the total number of $\underbrace{\text{“primitive” operations}}_{\text{But what is “primitive”?}}$ an algorithm performs.

Counting Steps—Literally

Definition (Eiffel Tower Problem)

- You and a friend are at the top of the Eiffel Tower
- You want to count how many steps there are to the bottom (2689)
- What are the primitive operations?
  - Stepping on a single stair step
  - Marking a single character on a piece of paper

Multiple Choice Question

Algorithm 1

1. Take the paper, and go down the stairs
2. Every time you take a step, put a tally mark on the paper
3. At the bottom, climb back to the top and give your friend the paper

How many primitive operations do you perform?

(A) 2689

(B) 2689×2

(C) 2689×3

(D) 2689×4

Multiple Choice Question

Algorithm 2

1. Take one step down, place your hat upon it
2. Go back to the top and tell your friend to mark a tally
3. Go down to your hat, place it on the next step, take one more step, and repeat the process

How many primitive operations do you perform?

(A) 2689×2

(B) $1 + 2 + 3 + \cdots + 2689$

(C) $2 \times (1 + 2 + 3 + \cdots + 2689)$

(D) $2 \times (1 + 2 + 3 + \cdots + 2689) + 2689$

Multiple Choice Question

Algorithm 3

1. You see another friend at the bottom of the staircase
2. He shows you a sign with the number of steps in decimal
3. You write down each digit on the piece of paper

How many primitive operations do you perform?

(A) 2689

(B) 4

(C) 0

(D) 1

Counting Code Steps

Primitive operations on most modern processors include:

- Arithmetic (e.g., `+`, `-`, `*`, `/`)
- Conditionals (e.g., **`if`**, `==`)
- Variable assignment

Multiple Choice Question

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Times
`for (int element : haystack)`	c_1	?
`   if (element == needle)`	c_2	?
`      return true;`	c_3	?
`return false;`	c_4	?

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1 or 0

(D) Depends on c_i

Multiple Choice Question

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Times
`for (int element : haystack)`	c_1	$\sum_{i=0}^{n-1} t_i$
`   if (element == needle)`	c_2	?
`      return true;`	c_3	?
`return false;`	c_4	?

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1 or 0

(D) Depends on c_i

Multiple Choice Question

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Times
`for (int element : haystack)`	c_1	$\sum_{i=0}^{n-1} t_i$
`  if (element == needle)`	c_2	$\sum_{i=0}^{n-1} t_i$
`    return true;`	c_3	?
`return false;`	c_4	?

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1 or 0

(D) Depends on c_i

Multiple Choice Question

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Times
`for (int element : haystack)`	c_1	$\sum_{i=0}^{n-1} t_i$
`if (element == needle)`	c_2	$\sum_{i=0}^{n-1} t_i$
`return true;`	c_3	0 or 1
`return false;`	c_4	?

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1 or 0

(D) Depends on c_i

Multiple Choice Question

- Let $t_i = \#$ times **`for`** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Times
`for (int element : haystack)`	c_1	$\sum_{i=0}^{n-1} t_i$
`if (element == needle)`	c_2	$\sum_{i=0}^{n-1} t_i$
`return true;`	c_3	0 or 1
`return false;`	c_4	0 or 1

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1 or 0

(D) Depends on c_i

Multiple Choice Question

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Worst
`for (int element : haystack)`	c_1	?
`   if (element == needle)`	c_2	?
`      return true;`	c_3	?
`return false;`	c_4	?

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1

(D) Depends on c_i

Multiple Choice Question

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Worst
`for (int element : haystack)`	c_1	n
`    if (element == needle)`	c_2	?
`        return true;`	c_3	?
`return false;`	c_4	?

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1

(D) Depends on c_i

Multiple Choice Question

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Worst
`for (int element : haystack)`	c_1	n
`   if (element == needle)`	c_2	n
`      return true;`	c_3	?
`return false;`	c_4	?

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1

(D) Depends on c_i

Multiple Choice Question

- Let $t_i = \#$ times **for** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Worst
`for (int element : haystack)`	c_1	n
`    if (element == needle)`	c_2	n
`        return true;`	c_3	1
`return false;`	c_4	?

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1

(D) Depends on c_i

Multiple Choice Question

- Let $t_i = \#$ times **`for`** gets executed at element i.
- Let $n =$ `haystack.length`.

	Cost	Worst
`for (int element : haystack)`	c_1	n
`    if (element == needle)`	c_2	n
`        return true;`	c_3	1
`return false;`	c_4	1

(A) $\sum_{i=0}^{n-1} t_i$

(B) n

(C) 1

(D) Depends on c_i

Big-O Notation

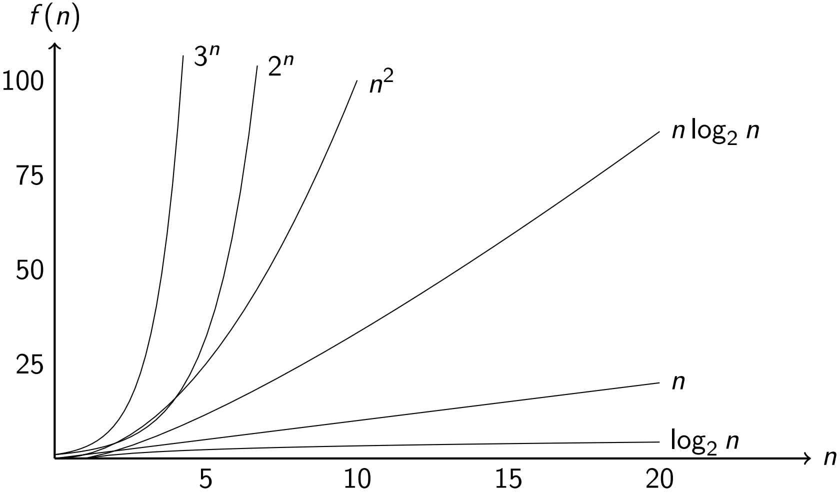
Idea: express number of steps a program takes as a function of the size of the input, n.

Definition

To consider the order of growth of a function f, we classify it as O ("big-oh") of another function g:

$$\begin{aligned} f \in O(g) \iff & \exists c > 0 \\ & \exists n_0 \geq 0 \\ & \forall n \geq n_0, \quad f(n) \leq cg(n) \end{aligned}$$

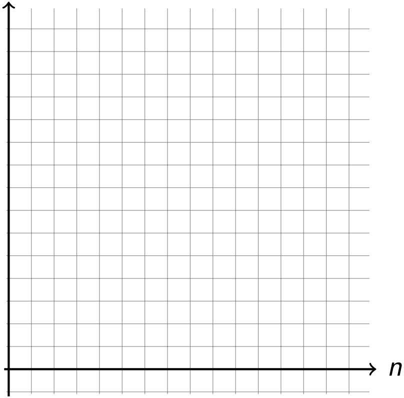
Common Functions



Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5 \log_2 n$ **(C)** 10 **(D)** $2n + 5$



$$\exists c > 0$$

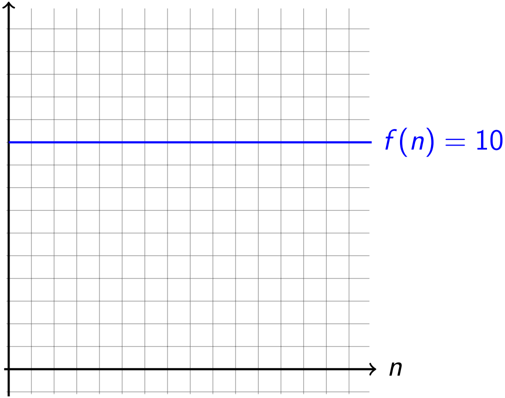
$$\exists n_0 \geq 0$$

$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5\log_2 n$ **(C)** 10 **(D)** $2n + 5$

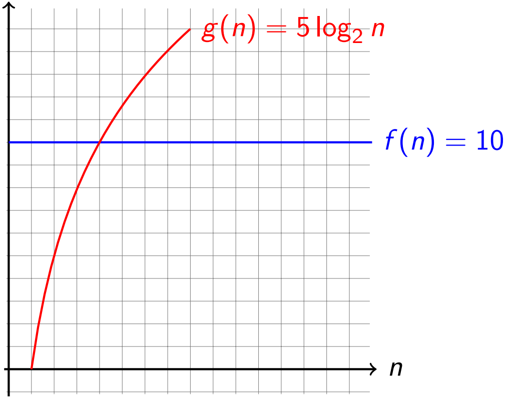


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5 \log_2 n$ **(C)** 10 **(D)** $2n + 5$



$$\exists c > 0$$

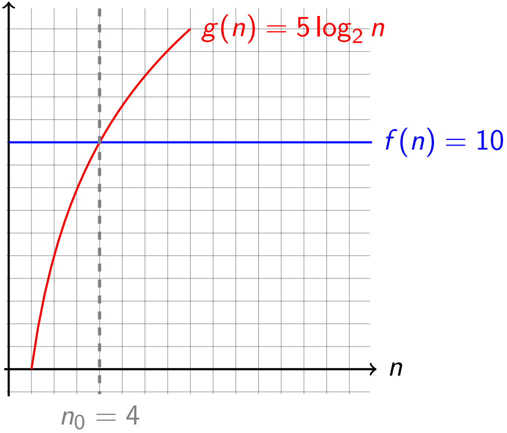
$$\exists n_0 \geq 0$$

$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5 \log_2 n$ **(C)** 10 **(D)** $2n + 5$

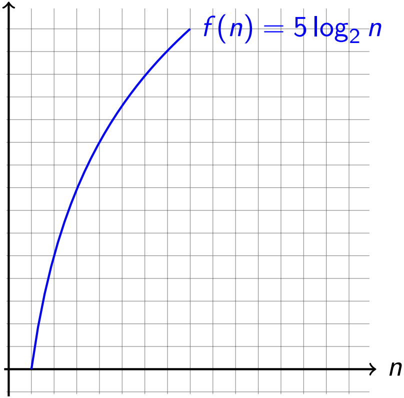


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5 \log_2 n$ **(C)** 10 **(D)** $2n + 5$

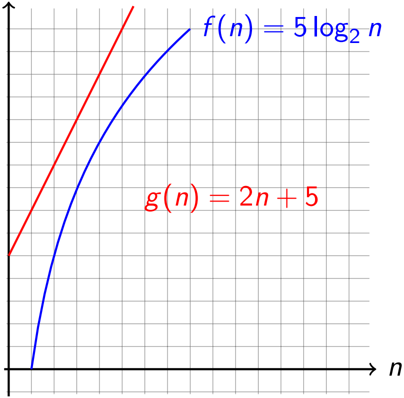


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5\log_2 n$ **(C)** 10 **(D)** $2n + 5$

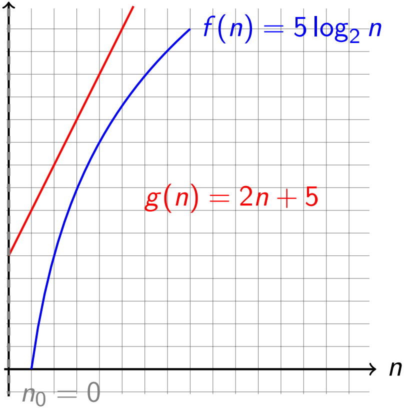


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5 \log_2 n$ **(C)** 10 **(D)** $2n + 5$

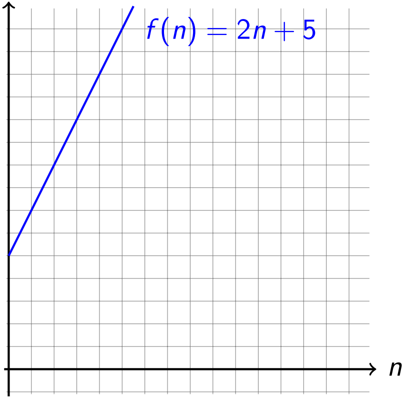


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq c g(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5 \log_2 n$ **(C)** 10 **(D)** $2n + 5$

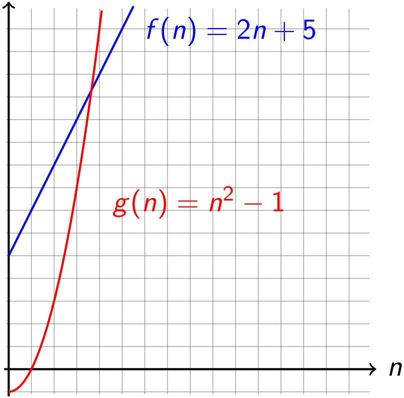


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5 \log_2 n$ **(C)** 10 **(D)** $2n + 5$

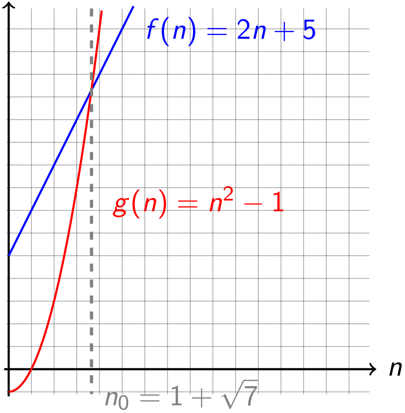


$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5 \log_2 n$ **(C)** 10 **(D)** $2n + 5$



$$\exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

Multiple Choice Question

Rearrange the following functions so that each is O of the next

(A) $n^2 - 1$ **(B)** $5\log_2 n$ **(C)** 10 **(D)** $2n + 5$

So,

$$\begin{aligned} 10 &\in O(5\log_2 n) \\ 5\log_2 n &\in O(2n+5) \\ 2n+5 &\in O(n^2-1) \end{aligned}$$

and the proper arrangement is

10 $5\log_2 n$ $2n + 5$ $n^2 - 1$

Multiple Choice Question

$$
\begin{aligned}
f \in O(g) \iff & \exists c > 0 \\
& \exists n_0 \geq 0 \\
& \forall n \geq n_0, \quad f(n) \leq cg(n)
\end{aligned}
$$

$10^{100} \overset{?}{\in} O(1)$

(A) Yes

(B) No

Multiple Choice Question

$$
\begin{aligned}
f \in O(g) \iff & \exists c > 0 \\
& \exists n_0 \geq 0 \\
& \forall n \geq n_0, \quad f(n) \leq cg(n)
\end{aligned}
$$

$10^{100} \overset{?}{\in} O(n^2)$

(A) Yes

(B) No

Multiple Choice Question

$$
\begin{aligned}
f \in O(g) \iff & \exists c > 0 \\
& \exists n_0 \geq 0 \\
& \forall n \geq n_0, \quad f(n) \leq cg(n)
\end{aligned}
$$

$n^2 \overset{?}{\in} O(1)$

(A) Yes

(B) No

Multiple Choice Question

$$
\begin{aligned}
f \in O(g) \iff & \exists c > 0 \\
& \exists n_0 \geq 0 \\
& \forall n \geq n_0, \quad f(n) \leq cg(n)
\end{aligned}
$$

$n^2 \overset{?}{\in} O(n^2)$

(A) Yes

(B) No

Multiple Choice Question

$$\begin{aligned} f \in O(g) \iff & \exists c > 0 \\ & \exists n_0 \geq 0 \\ & \forall n \geq n_0, \quad f(n) \leq cg(n) \end{aligned}$$

$2n^3 \overset{?}{\in} O(1)$

(A) Yes

(B) No

Multiple Choice Question

$$\begin{aligned} f \in O(g) \iff & \exists c > 0 \\ & \exists n_0 \geq 0 \\ & \forall n \geq n_0, \quad f(n) \leq cg(n) \end{aligned}$$

$2n^3 \overset{?}{\in} O(n^2)$

(A) Yes

(B) No

Multiple Choice Question

$$f \in O(g) \iff \exists c > 0$$
$$\exists n_0 \geq 0$$
$$\forall n \geq n_0, \quad f(n) \leq cg(n)$$

$2n^3 \overset{?}{\in} O(n^3)$

(A) Yes

(B) No

Multiple Choice Question

Let $f(n) = 1000n^5 - 400$. What function makes $f \in O(g)$ true?

(A) $g(n) = 10^{8675309}$

(B) $g(n) = 4000n^4 + 1000$

(C) $g(n) = n^{100}$

(D) None of the above

Multiple Choice Question

Let $f(n) = 867 \times 2^n + n^2 - n$. What function makes $f \in O(g)$ true?

(A) $g(n) = 2^n$

(B) $g(n) = n^2$

(C) $g(n) = n$

(D) None of the above

O proofs

We prove that O is reflexive:

$$\forall f, f \in O(f)$$

Proof.

. . . □

How do we begin?

(A) We don't; it's obvious

(B) Assume it's true, and show that the conclusion can't be false

(C) Let f be an arbitrary function

(D) Come up with an example function, f

O proofs

Proof.

Let f be an arbitrary function.
By the definition of O, $f \in O(f)$ would mean that
...

□

What does the definition of O tell us here?

(A) $\forall n, f(n) = f(n)$

(B) $\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) = f(n)$

(C) $\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) \leq f(n)$

(D) $\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) \leq cf(n)$

O proofs

Proof.

Let f be an arbitrary function.
By the definition of O, $f \in O(f)$ would mean that

$$\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) \leq cf(n)$$

We know that $f(n) = f(n)$ for every possible n. Thus

. . .

□

What can we say about $f(n) \overset{?}{\leq} f(n)$?

(A) $f(n) \leq f(n)$ for every possible n

(B) $f(n) \not\leq f(n)$ for every possible n

(C) $f(n) = f(n)$ for every possible n

(D) $\exists n, f(n) \leq f(n)$

O proofs

Proof.

Let f be an arbitrary function.
By the definition of O, $f \in O(f)$ would mean that

$$\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) \leq cf(n)$$

We know that $f(n) = f(n)$ for every possible n. Thus $f(n) \leq f(n)$ for all n.
This satisfies $f \in O(f)$ because . . . □

Why does this satisfy $f \in O(f)$?

(A) We can let $c = 1$

(B) We can let $c = 1$, $n_0 = 0$

(C) It's the very definition of O

(D) We don't know what c or n_0 could be, but they exist

O proofs

O is reflexive:

$$\forall f, f \in O(f)$$

Proof.

Let f be an arbitrary function.
By the definition of O, $f \in O(f)$ would mean that

$$\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) \leq cf(n)$$

We know that $f(n) = f(n)$ for every possible n. Thus $f(n) \leq f(n)$ for all n.
This satisfies $f \in O(f)$ because we can let $c = 1$ and $n_0 = 0$, making

$$\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) \leq cf(n)$$

true. □

O proofs

Show that

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

Proof.

. . . □

How do we proceed?

(A) Assume $f \in O(g)$ is true, show that $f(n) + g(n) \in O(g)$ must be true

(B) Assume $f(n) + g(n) \in O(g)$ is true, show that $f \in O(g)$ must be true

(C) Show that the property holds when f and g are particular functions

(D) Write a truth table

O proofs

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

Proof.

Assume f and g are arbitrary functions such that $f \in O(g)$.
By the definition of O, . . . □

What does the definition of O tell us here?

(A) $\forall n, f(n) = f(n)$

(B) $\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) = f(n)$

(C) $\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) \leq g(n)$

(D) $\exists c > 0, \exists n_0 \geq 0, \forall n \geq n_0, f(n) \leq cg(n)$

O proofs

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

Proof.

Assume f and g are arbitrary functions such that $f \in O(g)$.
By the definition of O, $f(n) \leq cg(n)$ for some $c > 0$ and for all $n >$ some $n_0 \geq 0$.

$$f(n) \leq cg(n)$$

□

How do we get an inequality involving $f(n) + g(n)$?

(A) Expand the O definition

(B) Add $g(n)$ to both sides

(C) Subtract $g(n)$ from both sides

(D) Solve for c

O proofs

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

Proof.

Assume f and g are arbitrary functions such that $f \in O(g)$.
By the definition of O, $f(n) \leq cg(n)$ for some $c > 0$ and for all $n >$ some $n_0 \geq 0$.

$$\begin{gathered} f(n) \leq cg(n) \\ f(n) + g(n) \leq cg(n) + g(n) \end{gathered}$$

□

How can we simplify this inequality?

(A) Subtract $g(n)$ from both sides

(B) Factor $g(n)$ out of the right side

(C) Solve for c

(D) Substitute $f(n)$ for $cg(n)$

O proofs

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

Proof.

Assume f and g are arbitrary functions such that $f \in O(g)$.
By the definition of O, $f(n) \leq cg(n)$ for some $c > 0$ and for all $n >$ some $n_0 \geq 0$.

$$\begin{aligned} f(n) &\leq cg(n) \\ f(n) + g(n) &\leq cg(n) + g(n) \\ f(n) + g(n) &\leq (c+1)g(n) \end{aligned}$$

□

Does this satisfy $f(n) + g(n) \in O(g)$?

(A) No: we have $c + 1$ instead of just c

(B) Yes: $f(n) + g(n)$ is less than or equal to a constant multiple of $g(n)$ for all n greater than a non-negative cuttoff

O proofs

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

Proof.

Assume f and g are arbitrary functions such that $f \in O(g)$.
By the definition of O, $f(n) \leq cg(n)$ for some $c > 0$ and for all $n >$ some $n_0 \geq 0$.

$$\begin{aligned} f(n) &\leq cg(n) \\ f(n) + g(n) &\leq cg(n) + g(n) \\ f(n) + g(n) &\leq (c+1)g(n) \end{aligned}$$

$\therefore f(n) + g(n) \in O(g)$ □

Multiple Choice Question

```
boolean search(int needle, int[] haystack) {
    for (int element : haystack)
        if (element == needle) return true;
    return false;
}
```

What is the running time of this algorithm in terms of O of a function of the size of the haystack, n?

(A) $O(1)$

(B) $O(n)$

(C) $O(n^2)$

(D) $O(2^n)$

Multiple Choice Question

```
for (int r = 0; i < n; i++) {
    for (int c = 0; j < n; j++) {
        // Some O(1) operations...
    }
}
```

What is the running time of this code?

(A) $O(1)$

(B) $O(n)$

(C) $O(n^2)$

(D) $O(2^n)$

More Analysis Tools

Definition (Big Omega)

$$
\begin{aligned}
f \in \Omega(g) \iff & \exists c > 0, \\
& \exists n_0 \geq 0, \\
& \forall n > n_0, \quad f(n) \geq cg(n)
\end{aligned}
$$

Definition (Big Theta)

$$
\begin{aligned}
f \in \Theta(g) \iff & \exists c_1 > 0, \\
& \exists c_2 > 0, \\
& \exists n_0 \geq 0, \\
& \forall n > n_0, \quad c_1 g(n) \leq f(n) \leq c_2 g(n)
\end{aligned}
$$

More Analysis Tools

Intuitively

$$\begin{aligned} f \in O(g) &\quad \text{is like} \quad f \leq g \\ f \in \Omega(g) &\quad \text{is like} \quad f \geq g \\ f \in \Theta(g) &\quad \text{is like} \quad f \approx g \end{aligned}$$