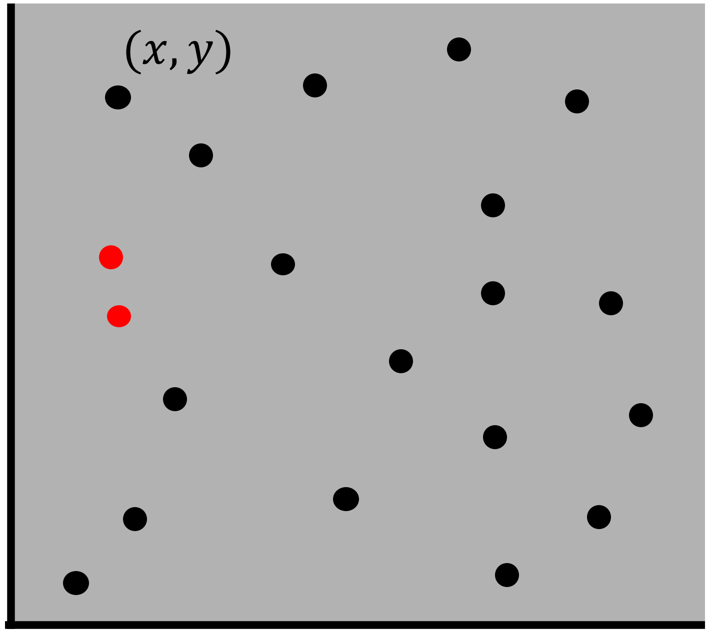
Introduction
CSCI 432

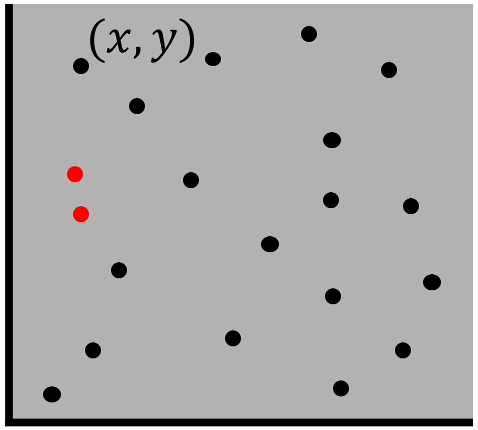
Closest Pair Problem



Given n points, find a pair of points with the smallest distance between them.

???

Closest Pair Problem

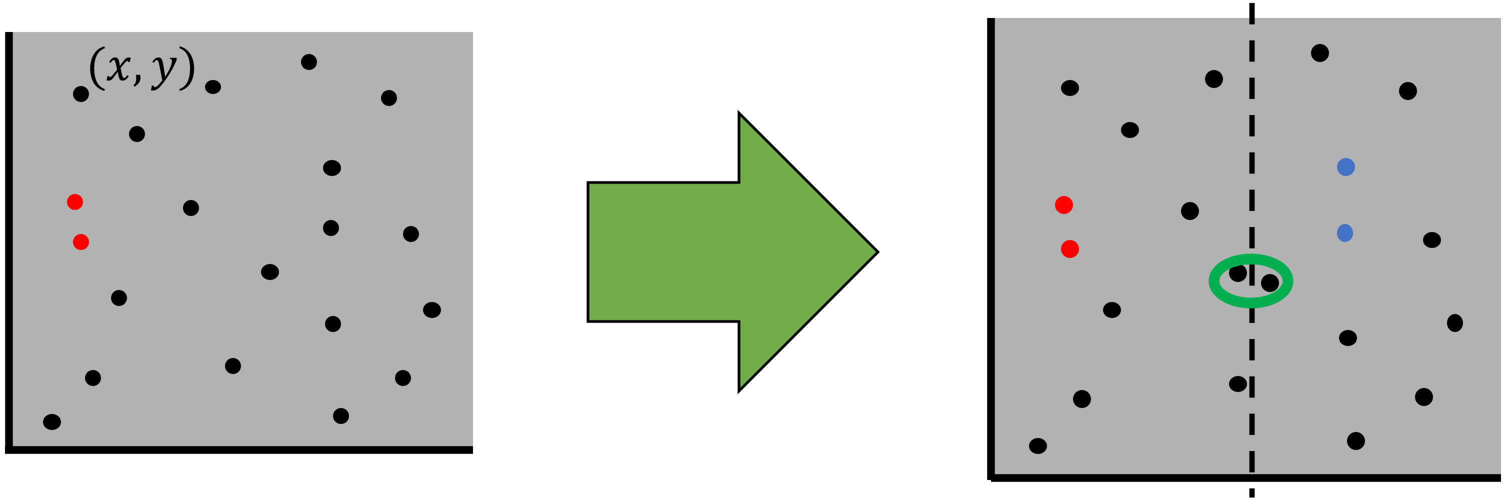


	P_1	P_2	...	P_n
P_1	/	$d_{1,2}$	...	$d_{1,n}$
P_2	$d_{2,1}$	/	...	$d_{2,n}$
...	...	...	...	...
P_n	$d_{n,1}$	$d_{n,2}$	...	/

Solution 1:

1. Compute distance for each pair.
2. Select smallest.

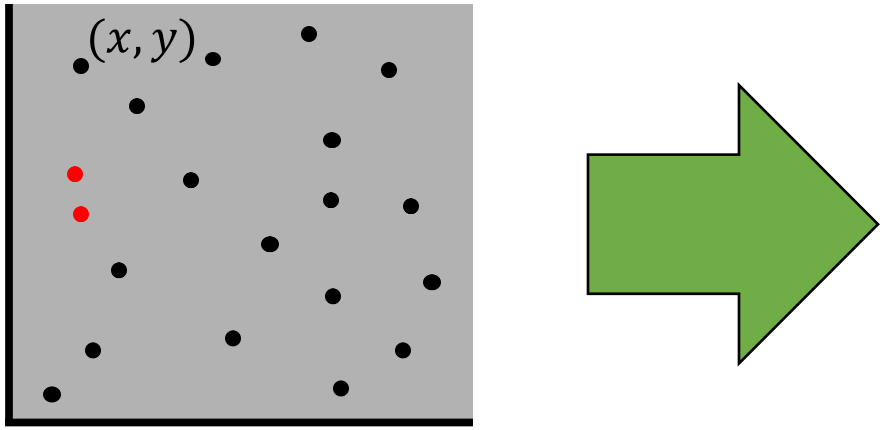
Closest Pair Problem



Solution 2:

1. Split in half.
2. Find closest in left and right sides (recursively).
3. Find closet straddling middle.
4. Select closest of all three.

Closest Pair Problem



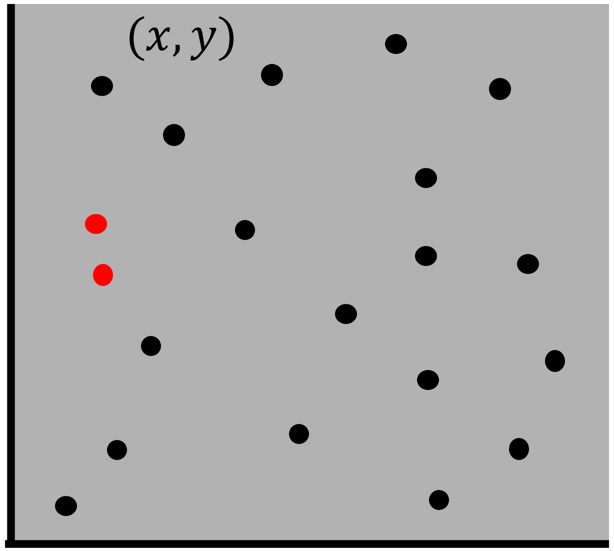
$$\{p_1, p_2, p_3, \ldots, p_n\}$$

$$\delta = d(p_1, p_2)$$

Solution 3:

1. Consider points in random order.
2. Let δ = closest pair found so far.
3. For each new point, check all "close" points for one $< \delta$.
4. If found update δ.

Closest Pair Problem



Solution 3:

1. Consider points in random order.
2. Let δ = closest pair found so far.
3. For each new point, check all "close" points for one $< \delta$.
4. If found update δ.

Solution 2:

1. Split in half.
2. Find closest in left and right sides.
3. Find closet straddling middle.
4. Select closest of all three.

Solution 1:

1. Compute distance for each pair.
2. Select smallest.

Closest Pair Problem

Solution 3:

1. Consider points in random order.

far.

What algorithm is best?

Soluti

1.
2. Select smallest.

ides.

.

4. Select closest of all three.

Closest Pair Problem

Solution 3:

1. Consider points in random order.

far.

(x, y)

What algorithm is best?

It depends.

Soluti

1.

2. Select smallest.

sides.

.

4. Select closest of all three.

Closest Pair Problem

Solution 3:

1. Consider points in random order.

far.

What algorithm is best?

It depends.

What does it mean for one algorithm to be "better" than another?

Soluti

1.

2. Select smallest.

ides.

.

4. Select closest of all three.

Closest Pair Problem

Solution 3:

1. Consider points in random order.

far.

(x, y)

What algorithm is best?

It depends.

What does it mean for one algorithm to be "better" than another?

Possible metrics: Running time, accuracy, resource requirements, simplicity, non-randomized.

Soluti

1.

2. Select smallest.

sides.

.

4. Select closest of all three.

Closest Pair Problem

Solution 3:

1. Consider points in random order.

far.

What algorithm is best?

It depends.

What does it mean for one algorithm to be “better” than another?

Possible metrics: Running time, accuracy, resource requirements, simplicity, non-randomized.

432 Goals:

Soluti

1.

2. Select smallest.

sides.

.

4. Select closest of all three.

Closest Pair Problem

Solution 3:

1. Consider points in random order.

far.

What algorithm is best?

It depends.

What does it mean for one algorithm to be "better" than another?

Possible metrics: Running time, accuracy, resource requirements, simplicity, non-randomized.

432 Goals:

Tools, tools, tools.

Soluti

1.

2. Select smallest.

sides.

.

4. Select closest of all three.

Closest Pair Problem

Solution 3:

1. Consider points in random order.

far.

What algorithm is best?

It depends.

What does it mean for one algorithm to be "better" than another?

Possible metrics: Running time, accuracy, resource requirements, simplicity, non-randomized.

432 Goals:

Tools, tools, tools.

Tools to build algorithms. Tools to analyze algorithms. Tools to compare algorithms. Tools to share algorithms.

Soluti

1.

2. Select smallest.

sides.

.

4. Select closest of all three.

What would you rather have?

Suppose that given a problem of size n…

Algorithm A runs in $O(n^2)$ time.

Algorithm B runs in $O(n)$ time.

What would you rather have?

Suppose that given a problem of size n…

Algorithm A runs in $n^2 \in O(n^2)$ time.

Algorithm B runs in $n + 10^{34} \in O(n)$ time.

What would you rather have?

Suppose that given a problem of size n...

Algorithm A runs in $n^2 \in O(n^2)$ time.

Algorithm B runs in $n + 10^{34} \in O(n)$ time.

~10^{17} seconds on fastest supercomputer today.

Age of universe ~10^{17} seconds.

What would you rather have?

Suppose that given a problem of size n…

Algorithm A runs in $O(2n^2)$ time.

Algorithm B runs in $O(n^2)$ time.

What would you rather have?

Suppose that given a problem of size n...

Algorithm A runs in $O(2^{2n})$ time.

Algorithm B runs in $O(2^n)$ time.

What would you rather have?

Suppose that given a problem of size n...

Algorithm A runs in $O(\log(n))$ time.

Algorithm B runs in $O\left(\frac{\log(n)}{\log(\log(n))}\right)$ time.

Big-O notation

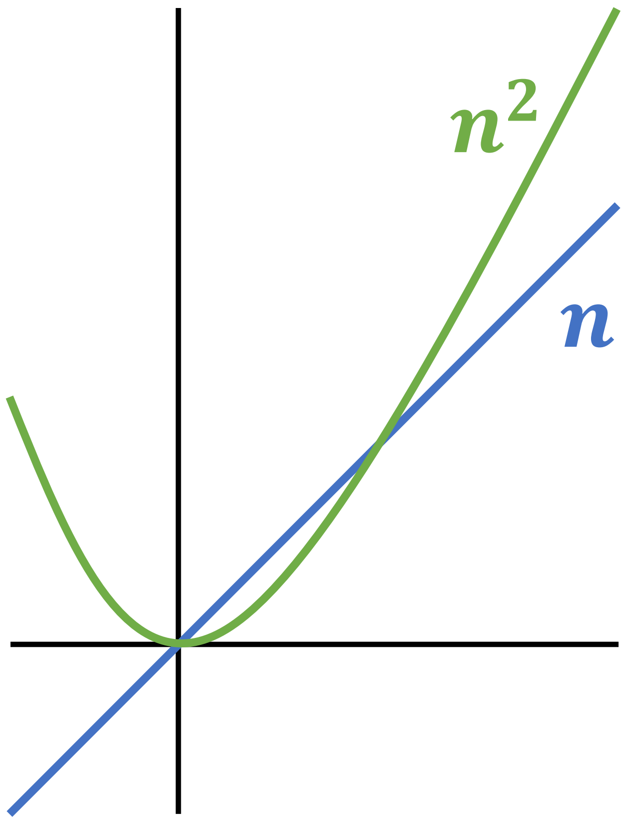
Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\text{: } \exists\ c, n_0 > 0 \text{ such that} \\ 0 \leq f(n) \leq cg(n)\ \forall\ n \geq n_0 \end{Bmatrix}$$

Big-O notation

Formal Definition:

$$O(g(n)) = \left\{ \begin{array}{l} f(n) \colon \exists\, c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le c g(n)\ \forall\, n \ge n_0 \end{array} \right\}$$

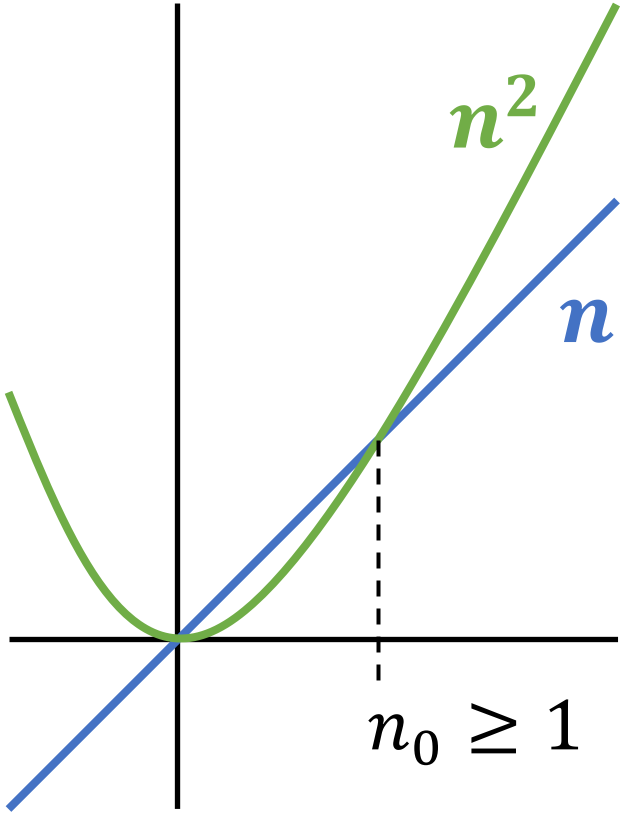


$$\Longrightarrow n \in O(n^2)$$

Big-O notation

Formal Definition:

$$O(g(n)) = \left\{ \begin{matrix} f(n):\ \exists\, c, n_0 > 0 \text{ such that } \\ 0 \leq f(n) \leq cg(n)\ \forall\, n \geq n_0 \end{matrix} \right\}$$



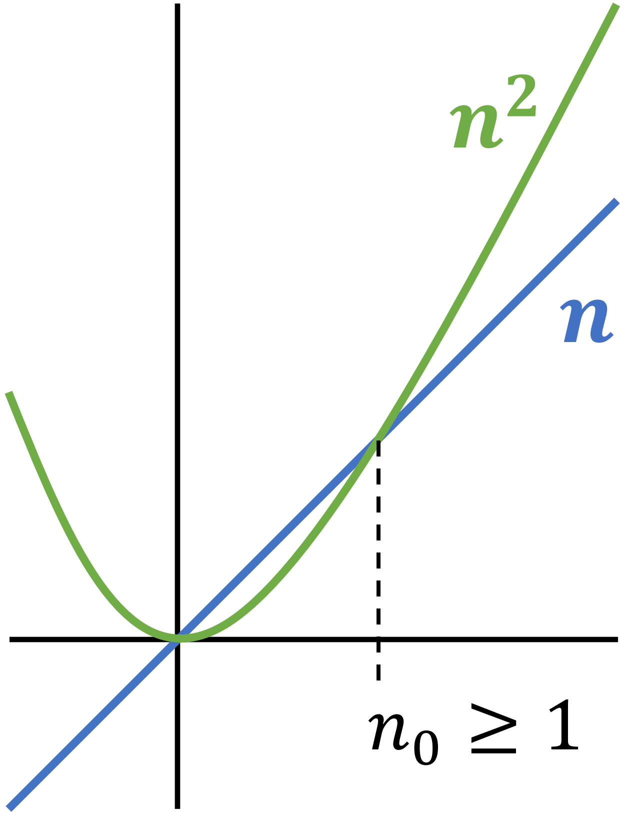
Let $c = 1$ and $n_0 = 1$.

$\Longrightarrow n \in O(n^2)$

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n) \colon \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\ n \ge n_0 \end{Bmatrix}$$

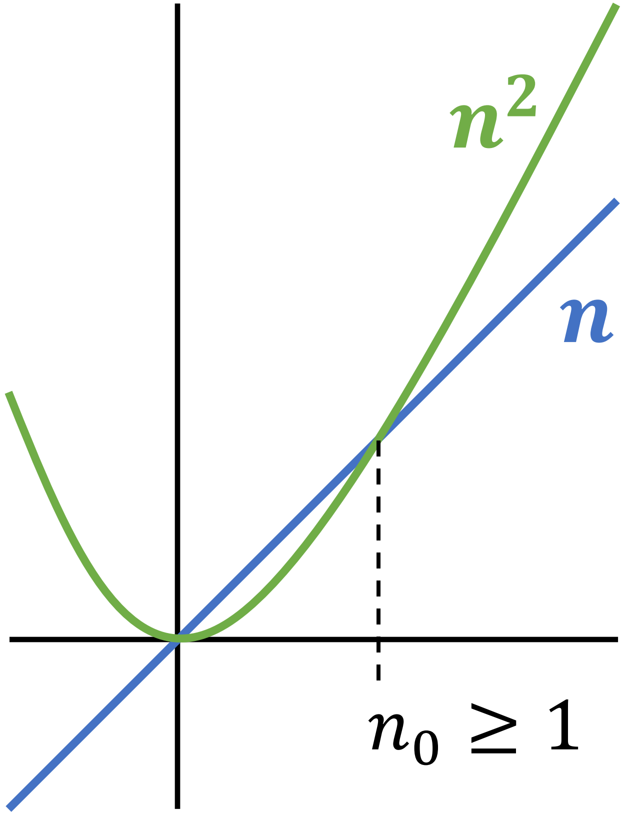


Let $c = 1$ and $n_0 = 1$.
Then, $n \le 1n^2\ \forall n \ge 1$
$\Longrightarrow n \in O(n^2)$

Big-O notation

Formal Definition:

$$O(g(n)) = \begin{Bmatrix} f(n)\colon \exists\, c, n_0 > 0 \text{ such that} \\ 0 \leq f(n) \leq cg(n)\ \forall\, n \geq n_0 \end{Bmatrix}$$



Let $c = 1$ and $n_0 = 1$.
Then, $n \leq 1n^2\ \forall n \geq 1$
$\Rightarrow n \in O(n^2)$

Note: Big-O notation provides an upper bound, but that upper bound need not be “tight”.

Big-O notation

Formal Definition:

$$O(g(n)) = \left\{ \begin{matrix} f(n)\colon \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\ n \ge n_0 \end{matrix} \right\}$$

Notes:

1. Big-O notation allows us to drop multiplicative constants and non-dominant factors.
2. Big-O notation allows us to broadly characterize algorithm efficiency.
3. Many (most) developers care greatly about multiplicative constants.

Big-O notation

Formal Definition:

$$O(g(n)) = \left\{ \begin{matrix} f(n)\colon \exists\, c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le c g(n)\ \forall\, n \ge n_0 \end{matrix} \right\}$$

Suppose that $f(n) \in O(g(n))$ and $k > 0$. Is $kf(n) \in O(g(n))$?

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\colon\ \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\ n \ge n_0 \end{Bmatrix}$$

Suppose that $f(n) \in O\big(g(n)\big)$ and $k > 0$. Is $kf(n) \in O\big(g(n)\big)$?

$f(n) \in O\big(g(n)\big) \Rightarrow$???

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\text{: } \exists\, c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\; \forall\, n \ge n_0 \end{Bmatrix}$$

Suppose that $f(n) \in O\big(g(n)\big)$ and $k > 0$. Is $kf(n) \in O\big(g(n)\big)$?

$f(n) \in O\big(g(n)\big) \Rightarrow \exists\, c, n_0 > 0$ such that $0 \le f(n) \le cg(n)\; \forall\, n \ge n_0$.

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\colon\ \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\ n \ge n_0 \end{Bmatrix}$$

Suppose that $f(n) \in O\big(g(n)\big)$ and $k > 0$. Is $kf(n) \in O\big(g(n)\big)$?

$f(n) \in O\big(g(n)\big) \Rightarrow \exists\ c, n_0 > 0$ such that $0 \le f(n) \le cg(n)\ \forall\ n \ge n_0$.

$k > 0 \Rightarrow 0 \le kf(n) \le ckg(n)\ \forall\ n \ge n_0$.

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\text{: } \exists\, c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\, n \ge n_0 \end{Bmatrix}$$

Suppose that $f(n) \in O\big(g(n)\big)$ and $k > 0$. Is $kf(n) \in O\big(g(n)\big)$?

$f(n) \in O\big(g(n)\big) \Rightarrow \exists\, c, n_0 > 0$ such that $0 \le f(n) \le cg(n)\ \forall\, n \ge n_0$.

$k > 0 \Rightarrow 0 \le kf(n) \le ckg(n)\ \forall\, n \ge n_0$.

So, $\exists\, m = ck, n_0 > 0$ such that $0 \le kf(n) \le mg(n)\ \forall\, n \ge n_0$

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\text{: } \exists\, c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\, n \ge n_0 \end{Bmatrix}$$

Suppose that $f(n) \in O\big(g(n)\big)$ and $k > 0$. Is $kf(n) \in O\big(g(n)\big)$?

$f(n) \in O\big(g(n)\big) \Rightarrow \exists\, c, n_0 > 0$ such that $0 \le f(n) \le cg(n)\ \forall\, n \ge n_0$.

$k > 0 \Rightarrow 0 \le kf(n) \le ckg(n)\ \forall\, n \ge n_0$.

So, $\exists\, m = ck, n_0 > 0$ such that $0 \le kf(n) \le mg(n)\ \forall\, n \ge n_0$
$\Rightarrow kf(n) \in O\big(g(n)\big)$.

Big-O notation

Formal Definition:

$$O(g(n)) = \left\{ \begin{matrix} f(n)\text{: } \exists\, c, n_0 > 0 \text{ such that} \\ 0 \leq f(n) \leq cg(n)\ \forall\, n \geq n_0 \end{matrix} \right\}$$

Suppose that $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$.

Is $f(n) \in O(h(n))$?

Big-O notation

Formal Definition:

$$O(g(n)) = \begin{Bmatrix} f(n)\text{: } \exists\, c, n_0 > 0 \text{ such that} \\ 0 \leq f(n) \leq cg(n)\ \forall\, n \geq n_0 \end{Bmatrix}$$

Suppose that $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$.
Is $f(n) \in O(h(n))$?

$f(n) \in O(g(n)) \Rightarrow \exists\, c, n_0 > 0$ such that $0 \leq f(n) \leq cg(n)\ \forall\, n \geq n_0$.
$g(n) \in O(h(n)) \Rightarrow \exists\, k, m_0 > 0$ such that $0 \leq g(n) \leq kh(n)\ \forall\, n \geq m_0$.

Big-O notation

Formal Definition:

$$O(g(n)) = \begin{Bmatrix} f(n)\text{: } \exists\, c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\, n \ge n_0 \end{Bmatrix}$$

Suppose that $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$.

Is $f(n) \in O(h(n))$?

$f(n) \in O(g(n)) \Rightarrow \exists\, c, n_0 > 0$ such that $0 \le f(n) \le cg(n)\ \forall\, n \ge n_0$.

$g(n) \in O(h(n)) \Rightarrow \exists\, k, m_0 > 0$ such that $0 \le g(n) \le kh(n)\ \forall\, n \ge m_0$.

$$c > 0 \Rightarrow 0 \le cg(n) \le ckh(n)$$

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \left\{ \begin{matrix} f(n):\ \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\ n \ge n_0 \end{matrix} \right\}$$

Suppose that $f(n) \in O\big(g(n)\big)$ and $g(n) \in O\big(h(n)\big)$.

Is $f(n) \in O\big(h(n)\big)$?

$f(n) \in O\big(g(n)\big) \Rightarrow \exists\ c, n_0 > 0$ such that $0 \le f(n) \le cg(n)\ \forall\ n \ge n_0$.

$g(n) \in O\big(h(n)\big) \Rightarrow \exists\ k, m_0 > 0$ such that $0 \le g(n) \le kh(n)\ \forall\ n \ge m_0$.

$c > 0 \Rightarrow 0 \le cg(n) \le ckh(n)$

$\forall\ \boldsymbol{n} \ge \boldsymbol{n_0}$

So, $0 \le f(n) \le cg(n)$

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\text{: } \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\ n \ge n_0 \end{Bmatrix}$$

Suppose that $f(n) \in O\big(g(n)\big)$ and $g(n) \in O\big(h(n)\big)$.
Is $f(n) \in O\big(h(n)\big)$?

$f(n) \in O\big(g(n)\big) \Rightarrow \exists\ c, n_0 > 0$ such that $0 \le f(n) \le cg(n)\ \forall\ n \ge n_0$.
$g(n) \in O\big(h(n)\big) \Rightarrow \exists\ k, m_0 > 0$ such that $0 \le g(n) \le kh(n)\ \forall\ n \ge m_0$.

$\forall\ n \ge n_0$ $\forall\ n \ge m_0$ $c > 0 \Rightarrow 0 \le cg(n) \le ckh(n)$

So, $0 \le f(n) \le cg(n) \le ckh(n)$

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\colon\ \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\ n \ge n_0 \end{Bmatrix}$$

Suppose that $f(n) \in O\big(g(n)\big)$ and $g(n) \in O\big(h(n)\big)$.

Is $f(n) \in O\big(h(n)\big)$?

$f(n) \in O\big(g(n)\big) \Rightarrow \exists\ c, n_0 > 0$ such that $0 \le f(n) \le cg(n)\ \forall\ n \ge n_0$.

$g(n) \in O\big(h(n)\big) \Rightarrow \exists\ k, m_0 > 0$ such that $0 \le g(n) \le kh(n)\ \forall\ n \ge m_0$.

$c > 0 \Rightarrow 0 \le cg(n) \le ckh(n)$

$\forall\ n \ge n_0$ $\quad$ $\forall\ n \ge m_0$

So, $0 \le f(n) \le cg(n) \le ckh(n)$

Thus, $0 \le f(n) \le ckh(n)\ \forall\ n \ge \max(n_0, m_0) \Rightarrow f(n) \in O\big(h(n)\big)$

Big-O notation

Formal Definition:

$$O(g(n)) = \left\{ \begin{matrix} f(n)\colon\ \exists\, c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le c g(n)\ \forall\, n \ge n_0 \end{matrix} \right\}$$

Prove or disprove: $2^{2n} \in O(2^n)$.

Big-O notation

Formal Definition:

$$O(g(n)) = \begin{Bmatrix} f(n)\text{: } \exists\ c, n_0 > 0 \text{ such that} \\ 0 \leq f(n) \leq cg(n)\ \forall\ n \geq n_0 \end{Bmatrix}$$

Prove or disprove: $2^{2n} \in O(2^n)$.

If so, $\exists\ c, n_0 > 0$ such that $0 \leq 2^{2n} \leq c2^n\ \forall\ n \geq n_0$.

Big-O notation

Formal Definition:

$$O(g(n)) = \left\{ \begin{matrix} f(n)\text{: } \exists\, c, n_0 > 0 \text{ such that} \\ 0 \leq f(n) \leq cg(n)\ \forall\, n \geq n_0 \end{matrix} \right\}$$

Prove or disprove: $2^{2n} \in O(2^n)$.

If so, $\exists\, c, n_0 > 0$ such that $0 \leq 2^{2n} \leq c2^n\ \forall\, n \geq n_0$.

…which means that, $2^{2n} = 2^n 2^n \leq c2^n$

Big-O notation

Formal Definition:

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\text{: } \exists\, c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\, n \ge n_0 \end{Bmatrix}$$

Prove or disprove: $2^{2n} \in O(2^n)$.

If so, $\exists\, c, n_0 > 0$ such that $0 \le 2^{2n} \le c2^n\ \forall\, n \ge n_0$.

…which means that, $2^{2n} = 2^n 2^n \le c2^n \Rightarrow 2^n \le c$.

Big-O notation

Formal Definition:

$$O(g(n)) = \begin{Bmatrix} f(n)\text{: } \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\ n \ge n_0 \end{Bmatrix}$$

Prove or disprove: $2^{2n} \in O(2^n)$.

If so, $\exists\ c, n_0 > 0$ such that $0 \le 2^{2n} \le c2^n\ \forall\ n \ge n_0$.

…which means that, $2^{2n} = 2^n 2^n \le c2^n \Rightarrow 2^n \le c$.

Contradiction, since c is a constant!

Big-O notation

Formal Definition:

$$O(g(n)) = \begin{Bmatrix} f(n)\text{: } \exists\, c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\, n \ge n_0 \end{Bmatrix}$$

Prove or disprove: $2^{2n} \in O(2^n)$.

If so, $\exists\, c, n_0 > 0$ such that $0 \le 2^{2n} \le c2^n\ \forall\, n \ge n_0$.

…which means that, $2^{2n} = 2^n 2^n \le c2^n \Rightarrow 2^n \le c$.

Contradiction, since c is a constant!

Thus, $2^{2n} \notin O(2^n)$.

Other Asymptotic Notation

Big-*O*

"Asymptotic upper bound"

$$O\big(g(n)\big) = \begin{Bmatrix} f(n)\colon \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le f(n) \le cg(n)\ \forall\ n \ge n_0 \end{Bmatrix}$$

Big-Theta

"Asymptotic tight bound"

$$\Theta\big(g(n)\big) = \begin{Bmatrix} f(n)\colon \exists\ c_1, c_2, n_0 > 0 \text{ such that} \\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n)\ \forall\ n \ge n_0 \end{Bmatrix}$$

Big-Omega

"Asymptotic lower bound"

$$\Omega\big(g(n)\big) = \begin{Bmatrix} f(n)\colon \exists\ c, n_0 > 0 \text{ such that} \\ 0 \le cg(n) \le f(n)\ \forall\ n \ge n_0 \end{Bmatrix}$$