Time Complexity

Lecture 22
Section 9.6

Robb T. Koether

Hampden-Sydney College

Wed, Mar 14, 2018

Outline

“Big-O” Notation: O(f)

Definition (Big-O)

- Let $f : \mathbb{R} \to \mathbb{R}$ be a function.
- A function $g : \mathbb{R} \to \mathbb{R}$ is O(f) (“big-O of f”) if there exist constants $c \in \mathbb{R}$, $n_0 \in \mathbb{N}$ such that

$$g(n) \leq cf(n)$$

for all $n \geq n_0$.

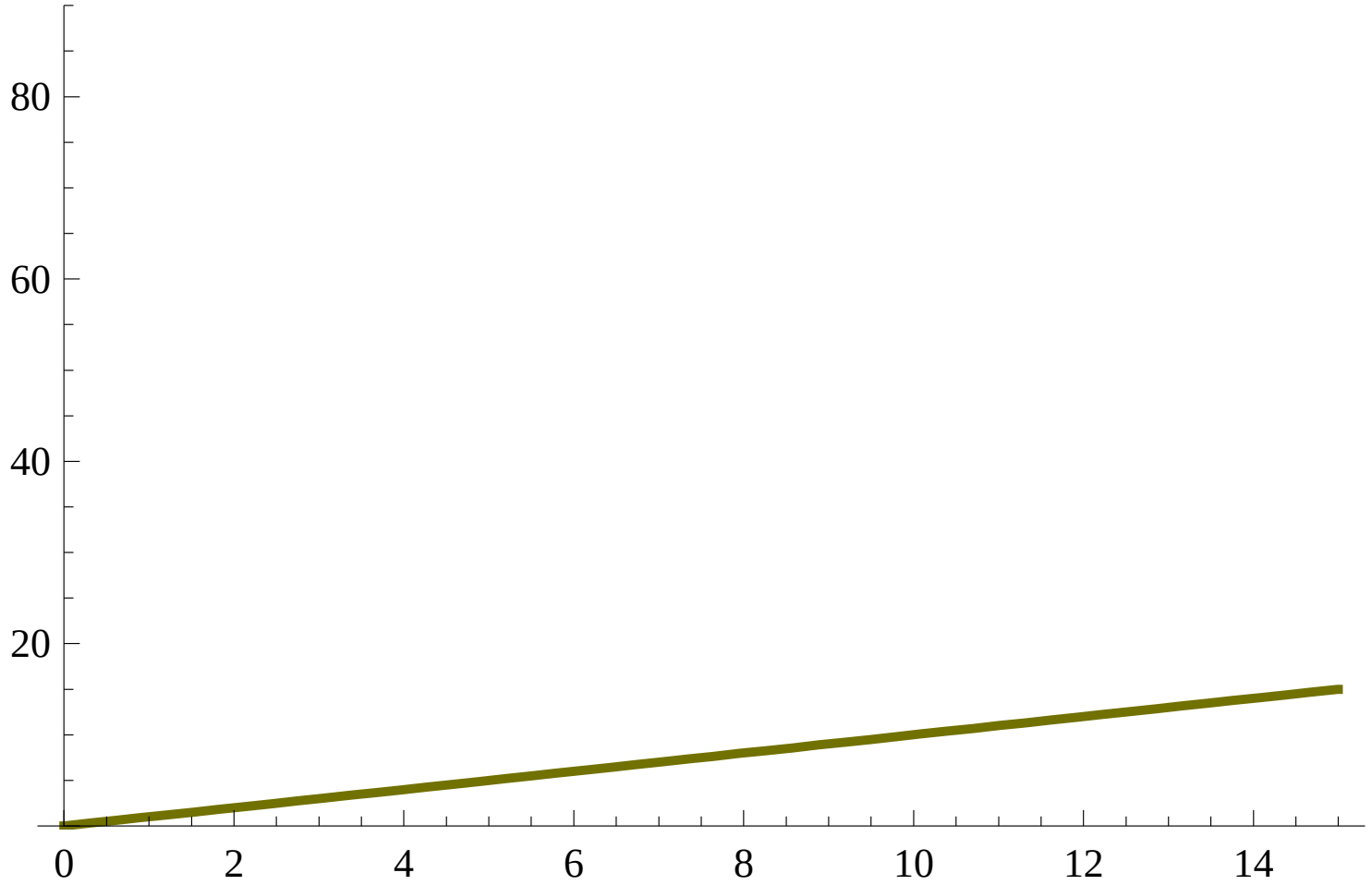
- In other words, $g(n)$ is bounded above by a constant multiple of $f(n)$ from some point on.

Examples

Example (Proving and Disproving Big-O)

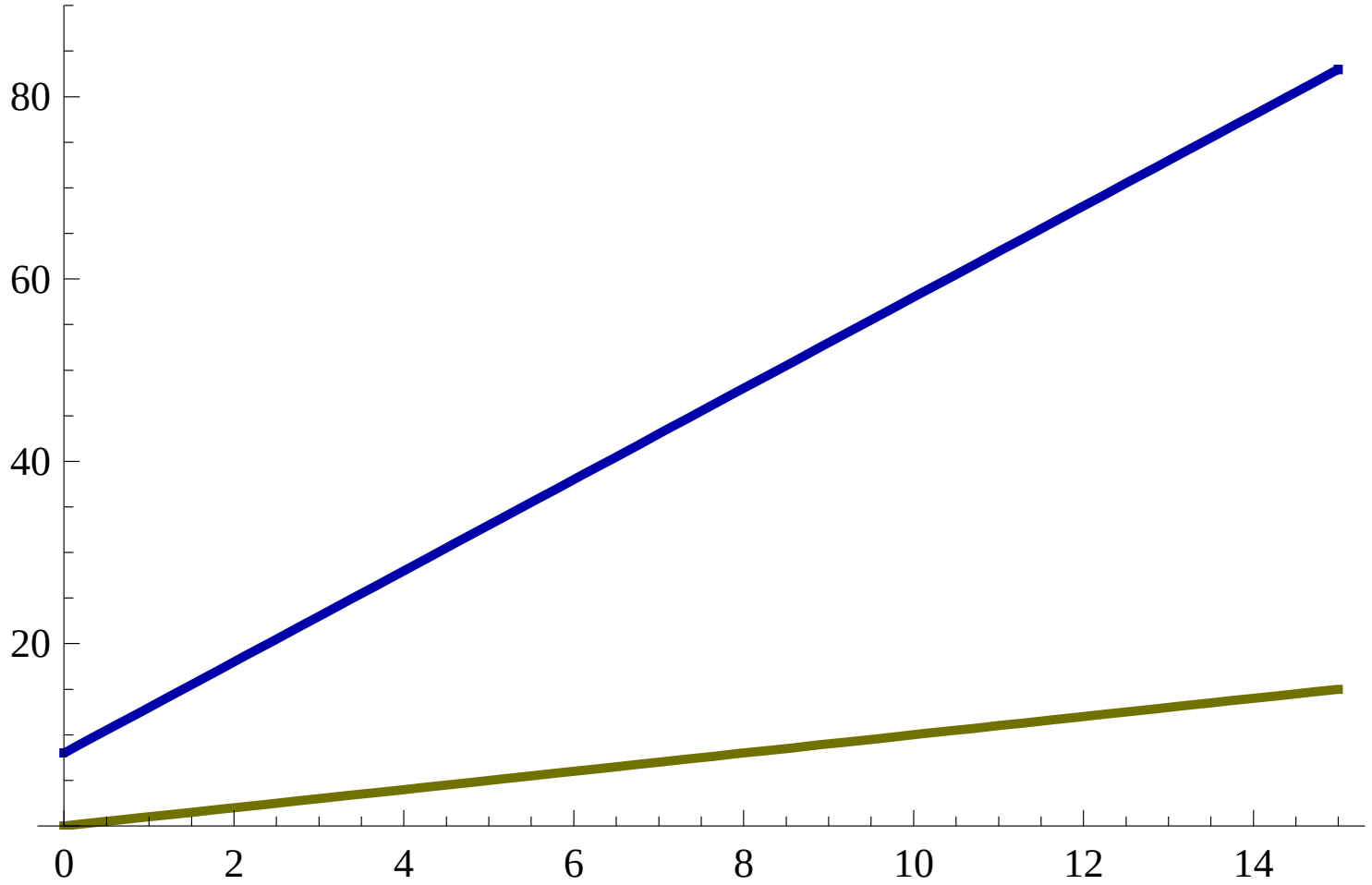
- Show that $5n + 8$ is $O(n)$.
- Show that $n^2 + 10n + 5$ is $O(n^2)$.
- Show that $n^2 + 10n + 5$ is *not* $O(n)$.

Examples



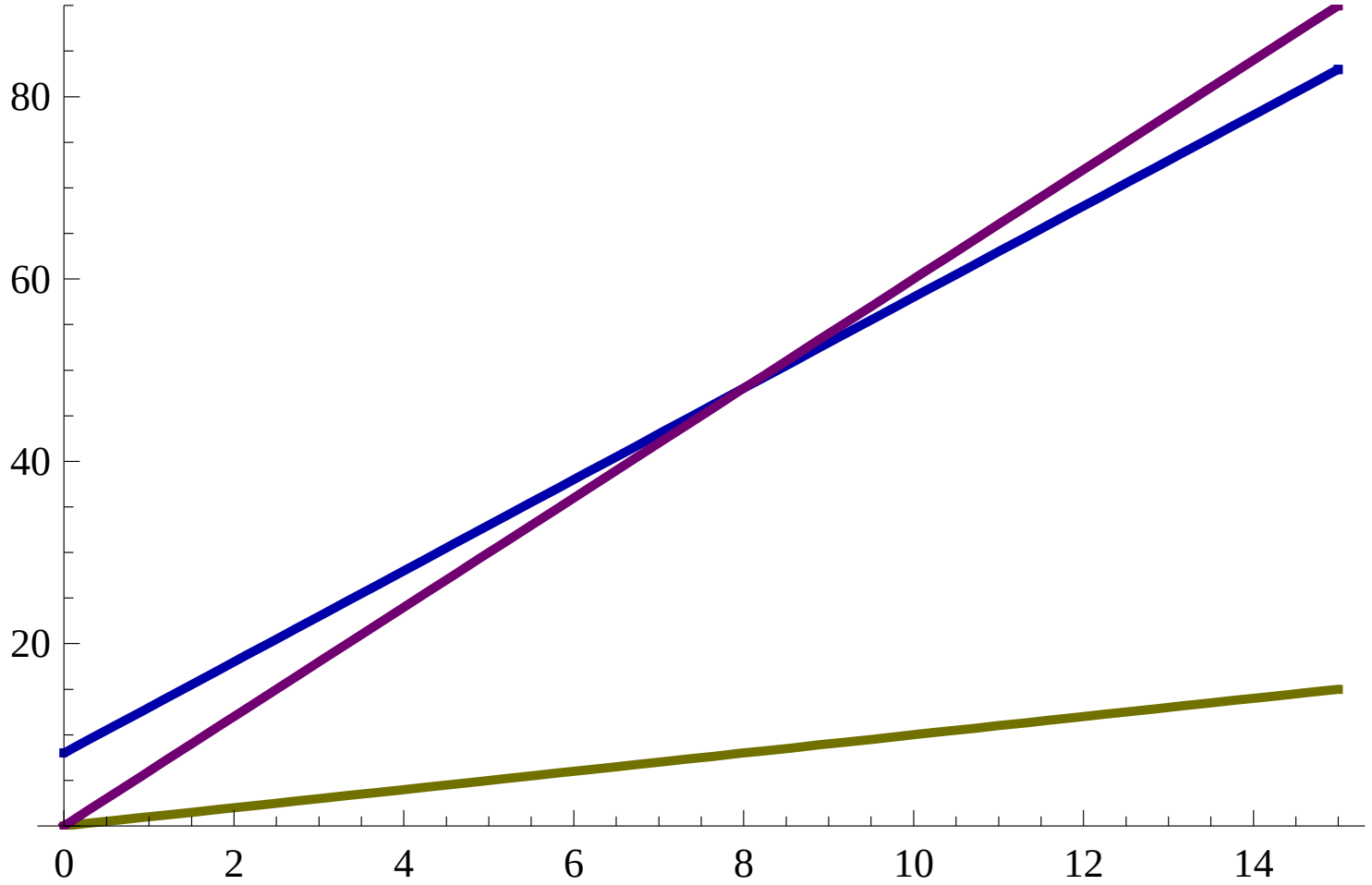
$f(n) = n$

Examples



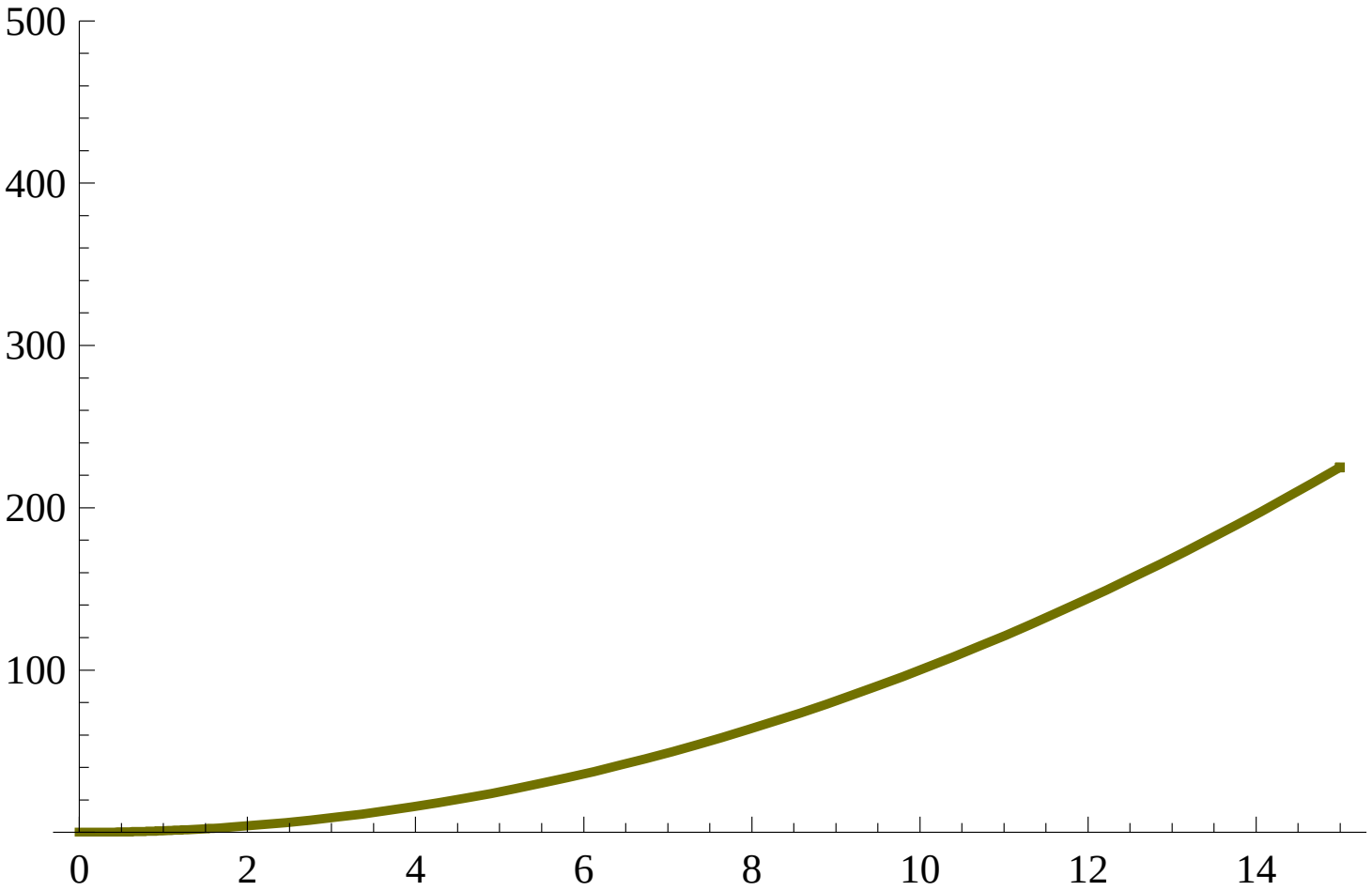
$f(n) = n$ and $g(n) = 5n + 8$

Examples



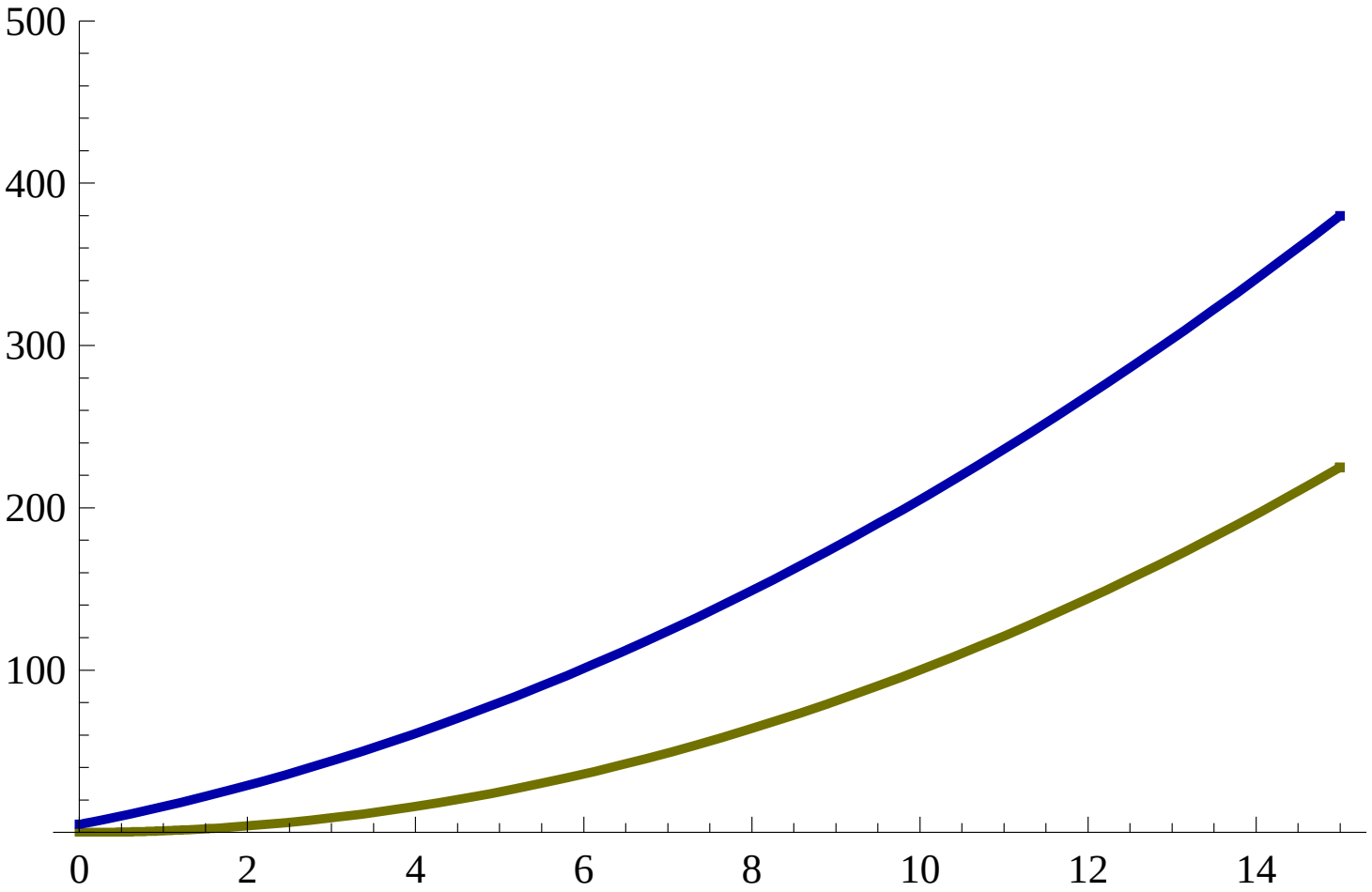
$f(n) = n$ and $g(n) = 5n + 8$ and $6f(n) = 6n$

Examples



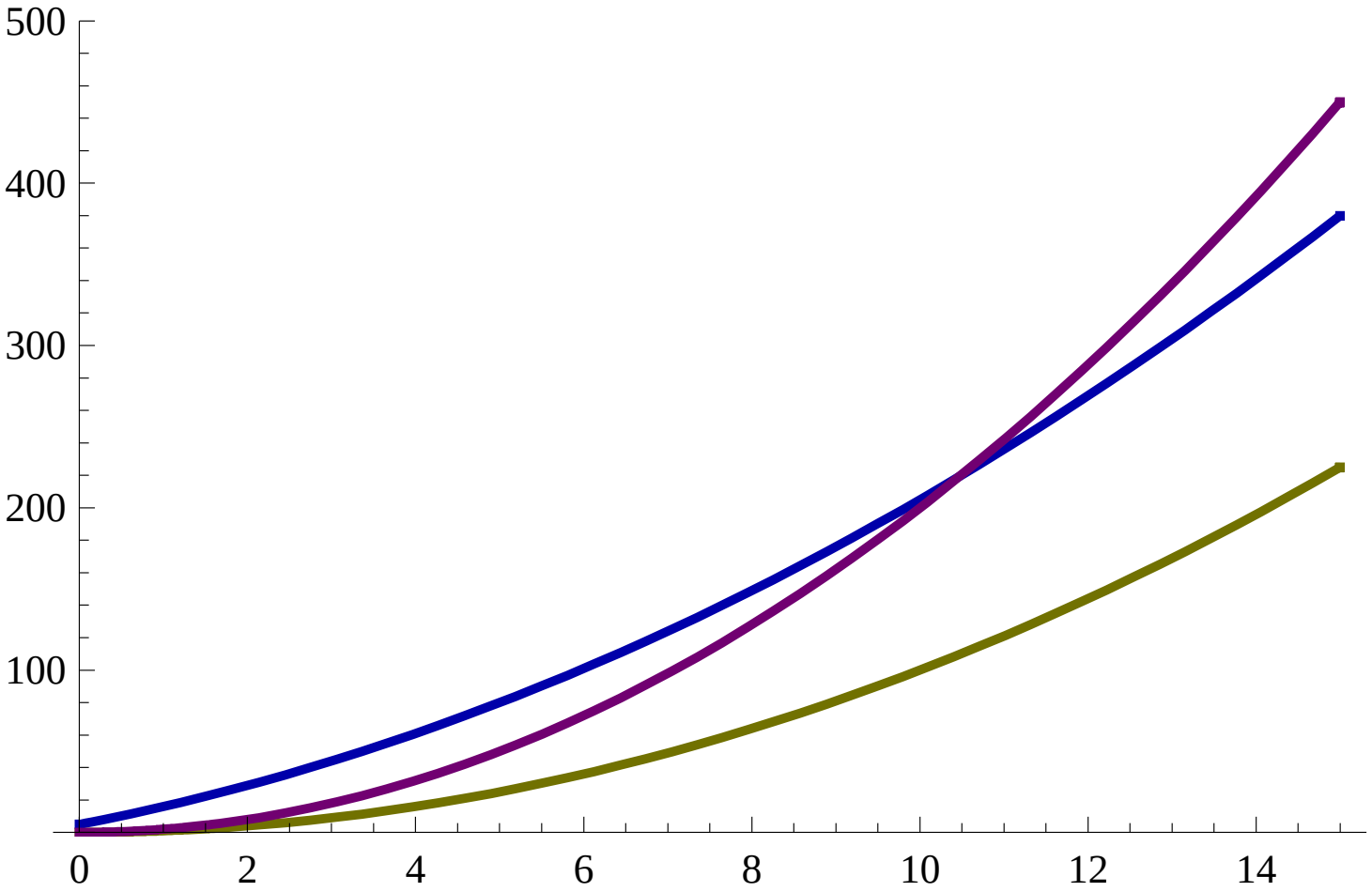
$f(n) = n^2$

Examples



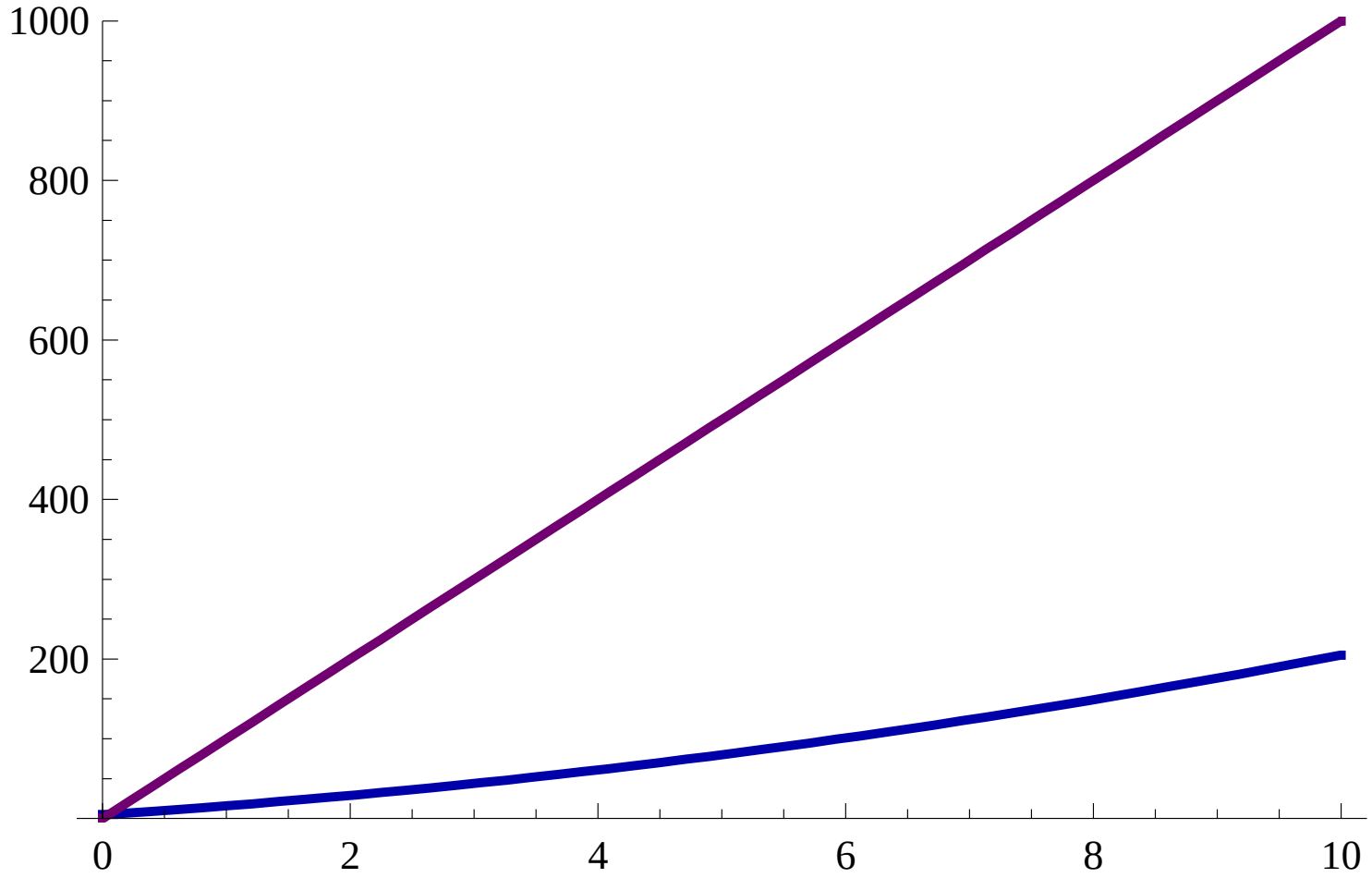
$f(n) = n^2$ and $g(n) = n^2 + 10n + 5$

Examples



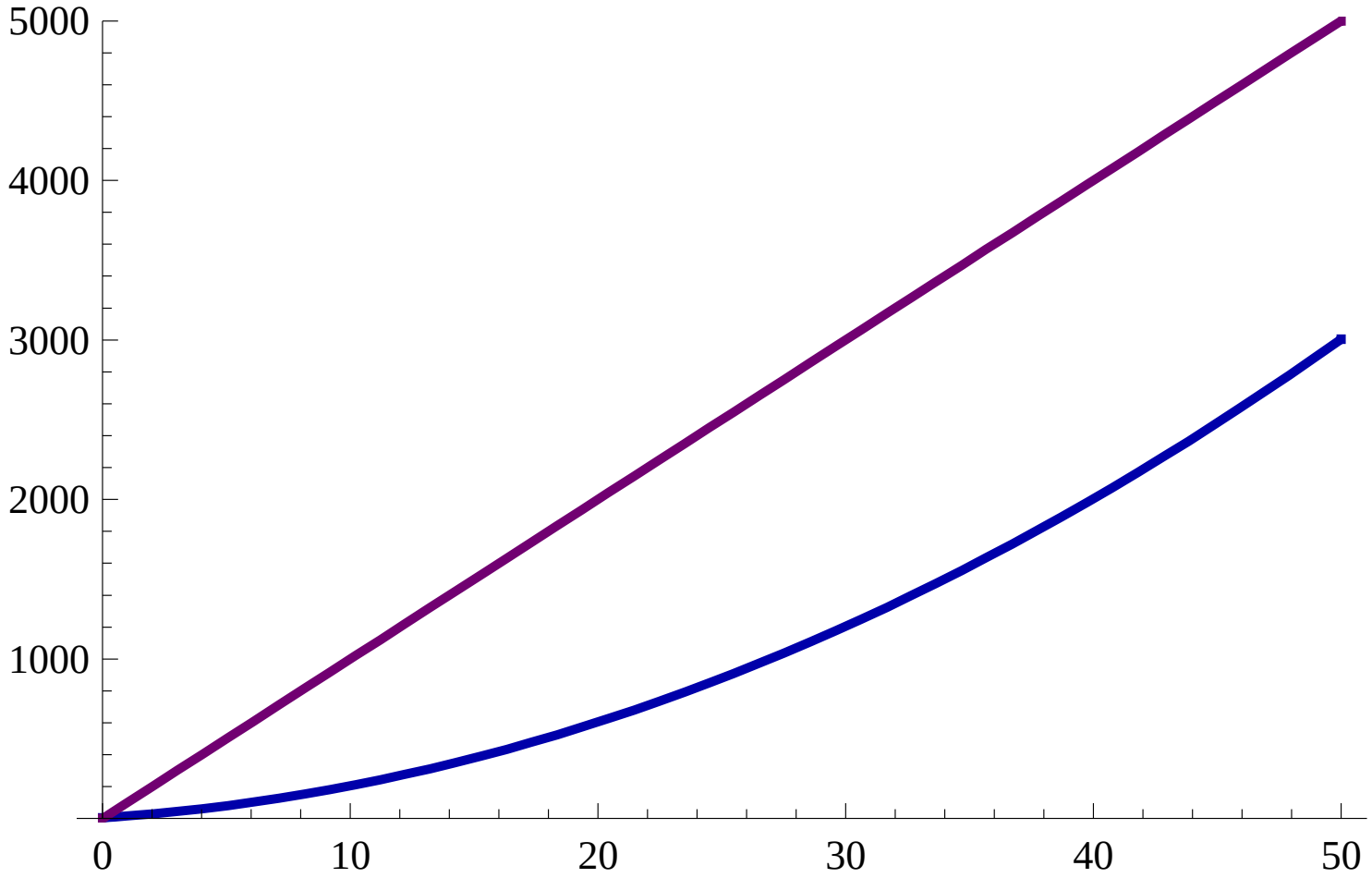
$f(n) = n^2$ and $g(n) = n^2 + 10n + 5$ and $2f(n) = 2n^2$

Examples



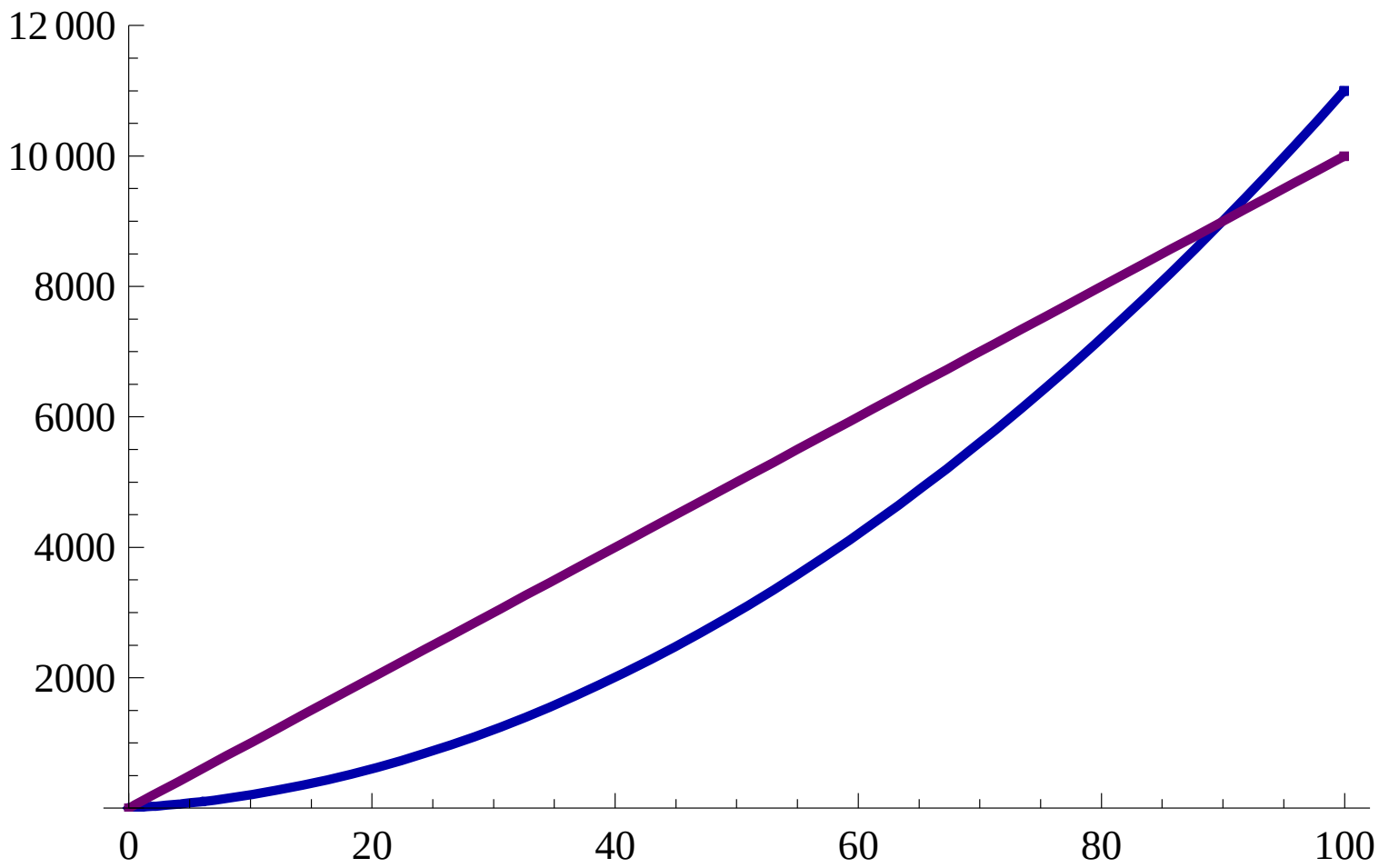
$100f(n) = 100n$ and $g(n) = n^2 + 10n + 5$, over $0 \leq n \leq 10$

Examples



$100f(n) = 100n$ and $g(n) = n^2 + 10n + 5$, over $0 \leq n \leq 50$

Examples



$100f(n) = 100n$ and $g(n) = n^2 + 10n + 5$, over $0 \leq n \leq 100$

Growth Rates

- If g is $O(f)$, then the growth rate of $g(n)$ is no greater than the growth rate of $f(n)$.
- If g is $O(f)$ and f is $O(g)$, then $f(n)$ and $g(n)$ have the same growth rate.
- In this case, we say that g is $\Theta(f)$ (big-theta).

Example

Example (Equal Growth Rates)

- Show that $5n + 8$ is $\Theta(n)$.

Equal Growth Rates

Theorem (Equal Growth Rates)

- $an + b$ *is* $\Theta(n)$ *for all* a *and* b*, with* $a > 0$*.*

Equal Growth Rates

Theorem (Equal Growth Rates)

- $an + b$ *is* $\Theta(n)$ *for all* a *and* b*, with* $a > 0$*.*
- $an^2 + bn + c$ *is* $\Theta(n^2)$ *for all* a*,* b*, and* c*, with* $a > 0$*.*

Equal Growth Rates

Theorem (Equal Growth Rates)

- $an + b$ *is* $\Theta(n)$ *for all* a *and* b*, with* $a > 0$.
- $an^2 + bn + c$ *is* $\Theta(n^2)$ *for all* a*,* b*, and* c*, with* $a > 0$.
- $an^3 + bn^2 + cn + d$ *is* $\Theta(n^3)$ *for all* a*,* b*,* c*, and* d*, with* $a > 0$.

Equal Growth Rates

Theorem (Equal Growth Rates)

- $an + b$ *is* $\Theta(n)$ *for all* a *and* b*, with* $a > 0$.
- $an^2 + bn + c$ *is* $\Theta(n^2)$ *for all* a*,* b*, and* c*, with* $a > 0$.
- $an^3 + bn^2 + cn + d$ *is* $\Theta(n^3)$ *for all* a*,* b*,* c*, and* d*, with* $a > 0$.
- *And so on.*

Common Growth Rates

Growth Rate	Example
$\Theta(1)$	
$\Theta(\log_2 n)$	
$\Theta(n)$	
$\Theta(n \log_2 n)$	
$\Theta(n^2)$	
$\Theta(2^n)$	
$\Theta(n!)$	

Common Growth Rates

Growth Rate	Example
$\Theta(1)$	Access an array element
$\Theta(\log_2 n)$	
$\Theta(n)$	
$\Theta(n \log_2 n)$	
$\Theta(n^2)$	
$\Theta(2^n)$	
$\Theta(n!)$	

Common Growth Rates

Growth Rate	Example
$\Theta(1)$	Access an array element
$\Theta(\log_2 n)$	Binary search
$\Theta(n)$	
$\Theta(n \log_2 n)$	
$\Theta(n^2)$	
$\Theta(2^n)$	
$\Theta(n!)$	

Common Growth Rates

Growth Rate	Example
$\Theta(1)$	Access an array element
$\Theta(\log_2 n)$	Binary search
$\Theta(n)$	Sequential search
$\Theta(n \log_2 n)$	
$\Theta(n^2)$	
$\Theta(2^n)$	
$\Theta(n!)$	

Common Growth Rates

Growth Rate	Example
$\Theta(1)$	Access an array element
$\Theta(\log_2 n)$	Binary search
$\Theta(n)$	Sequential search
$\Theta(n \log_2 n)$	Merge sort
$\Theta(n^2)$	
$\Theta(2^n)$	
$\Theta(n!)$	

Common Growth Rates

Growth Rate	Example
$\Theta(1)$	Access an array element
$\Theta(\log_2 n)$	Binary search
$\Theta(n)$	Sequential search
$\Theta(n \log_2 n)$	Merge sort
$\Theta(n^2)$	Selection sort
$\Theta(2^n)$	
$\Theta(n!)$	

Common Growth Rates

Growth Rate	Example
$\Theta(1)$	Access an array element
$\Theta(\log_2 n)$	Binary search
$\Theta(n)$	Sequential search
$\Theta(n \log_2 n)$	Merge sort
$\Theta(n^2)$	Selection sort
$\Theta(2^n)$	Factor an integer
$\Theta(n!)$	

Common Growth Rates

Growth Rate	Example
$\Theta(1)$	Access an array element
$\Theta(\log_2 n)$	Binary search
$\Theta(n)$	Sequential search
$\Theta(n \log_2 n)$	Merge sort
$\Theta(n^2)$	Selection sort
$\Theta(2^n)$	Factor an integer
$\Theta(n!)$	Traveling salesman problem

Outline

Estimating Run Times

- Suppose the run time of a program is $\Theta(f)$, for a specific function $f(n)$.
- Then the run time $t(n)$ satisfies

$$c_1 f(n) \leq t(n) \leq c_2 f(n)$$

  for some real numbers c_1 and c_2.
- We typically assume that $t(n) = cf(n)$ for some real number c.

Estimating Run Times

- Suppose program runs in t_0 seconds when the input size is n_0. Then
$$c = \frac{t_0}{f(n_0)}.$$
- Thus,
$$t(n) = t_0 \left(\frac{f(n)}{f(n_0)} \right).$$

Example

Example (Estimating Run Times)

- Suppose the run time of a program is $\Theta(n^2)$.

Example

Example (Estimating Run Times)

- Suppose the run time of a program is $\Theta(n^2)$.
- Suppose further that the program runs in $t_0 = 5$ μs when the input size is $n_0 = 100$.

Example

Example (Estimating Run Times)

- Suppose the run time of a program is $\Theta(n^2)$.
- Suppose further that the program runs in $t_0 = 5$ μs when the input size is $n_0 = 100$.
- Then

$$t(n) = 5\left(\frac{n^2}{100^2}\right) \ \mu\text{s}.$$

Example

Example (Estimating Run Times)

- Suppose the run time of a program is $\Theta(n^2)$.
- Suppose further that the program runs in $t_0 = 5$ μs when the input size is $n_0 = 100$.
- Then

$$t(n) = 5\left(\frac{n^2}{100^2}\right) \ \mu\text{s}.$$

- Thus, if the input size is 1000, then the run time is

$$t(1000) = 5\left(\frac{1000^2}{100^2}\right) = 500 \ \mu\text{s}.$$

Example

Example (Estimating Run Times)

- Suppose the run time of a program is

$$\Theta(n \log n).$$

Example

Example (Estimating Run Times)

- Suppose the run time of a program is

$$\Theta(n \log n).$$

- Suppose further that the program runs in $t_0 = 5$ μs when the input size is $n_0 = 100$.

Example

Example (Estimating Run Times)

- Suppose the run time of a program is

$$\Theta(n \log n).$$

- Suppose further that the program runs in $t_0 = 5$ μs when the input size is $n_0 = 100$.
- Then

$$t(n) = 5 \left(\frac{n \log n}{100 \log 100} \right) \ \mu\text{s}.$$

Example

Example (Estimating Run Times)

- Suppose the run time of a program is

$$\Theta(n \log n).$$

- Suppose further that the program runs in $t_0 = 5$ μs when the input size is $n_0 = 100$.
- Then

$$t(n) = 5 \left(\frac{n \log n}{100 \log 100} \right) \ \mu\text{s}.$$

- Thus, if the input size is 1000, then the run time is

$$t(n) = 5 \left(\frac{1000 \log 1000}{100 \log 100} \right) = 75 \ \mu\text{s}.$$

Outline

Comparison of Growth Rates

- Consider programs with run times that are $\Theta(1)$, $\Theta(\log n)$, $\Theta(n)$, $\Theta(n \log n)$, and $\Theta(n^2)$.
- Assume that each program runs in 1 μs when the input size is 100.
- Calculate the run times for input sizes 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, and 10^8.

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μs	1 μs	1 μs	1 μs	1 μs

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μs	1 μs	1 μs	1 μs	1 μs
10^3	1 μs	1.5 μs	10 μs	15 μs	100 μs

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μs	1 μs	1 μs	1 μs	1 μs
10^3	1 μs	1.5 μs	10 μs	15 μs	100 μs
10^4	1 μs	2 μs	100 μs	200 μs	10 ms

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μs	1 μs	1 μs	1 μs	1 μs
10^3	1 μs	1.5 μs	10 μs	15 μs	100 μs
10^4	1 μs	2 μs	100 μs	200 μs	10 ms
10^5	1 μs	2.5 μs	1 ms	2.5 ms	1 s

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μs	1 μs	1 μs	1 μs	1 μs
10^3	1 μs	1.5 μs	10 μs	15 μs	100 μs
10^4	1 μs	2 μs	100 μs	200 μs	10 ms
10^5	1 μs	2.5 μs	1 ms	2.5 ms	1 s
10^6	1 μs	3 μs	10 ms	30 ms	1.7 m

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μs	1 μs	1 μs	1 μs	1 μs
10^3	1 μs	1.5 μs	10 μs	15 μs	100 μs
10^4	1 μs	2 μs	100 μs	200 μs	10 ms
10^5	1 μs	2.5 μs	1 ms	2.5 ms	1 s
10^6	1 μs	3 μs	10 ms	30 ms	1.7 m
10^7	1 μs	3.5 μs	100 ms	350 ms	2.8 hr

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μs	1 μs	1 μs	1 μs	1 μs
10^3	1 μs	1.5 μs	10 μs	15 μs	100 μs
10^4	1 μs	2 μs	100 μs	200 μs	10 ms
10^5	1 μs	2.5 μs	1 ms	2.5 ms	1 s
10^6	1 μs	3 μs	10 ms	30 ms	1.7 m
10^7	1 μs	3.5 μs	100 ms	350 ms	2.8 hr
10^8	1 μs	4 μs	1 s	4 s	11.7 d

Comparison of $\Theta(n^2)$ and $\Theta(2^n)$

- Now consider a program with growth rate $\Theta(2^n)$.
- Assume that the program runs in 1 μs when the input size is 100.
- Calculate the run times for input sizes 100, 110, 120, 130, 140, 150, and 160.
- Note that we will not try to go as far as 10^3. How come?

Comparison of $\Theta(n^2)$ and $\Theta(2^n)$

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μs	1 μs

Comparison of $\Theta(n^2)$ and $\Theta(2^n)$

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μs	1 μs
110	1.2 μs	1 ms

Comparison of $\Theta(n^2)$ and $\Theta(2^n)$

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μs	1 μs
110	1.2 μs	1 ms
120	1.4 μs	1 s

Comparison of $\Theta(n^2)$ and $\Theta(2^n)$

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μs	1 μs
110	1.2 μs	1 ms
120	1.4 μs	1 s
130	1.7 μs	18 m

Comparison of $\Theta(n^2)$ and $\Theta(2^n)$

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μs	1 μs
110	1.2 μs	1 ms
120	1.4 μs	1 s
130	1.7 μs	18 m
140	2.0 μs	13 d

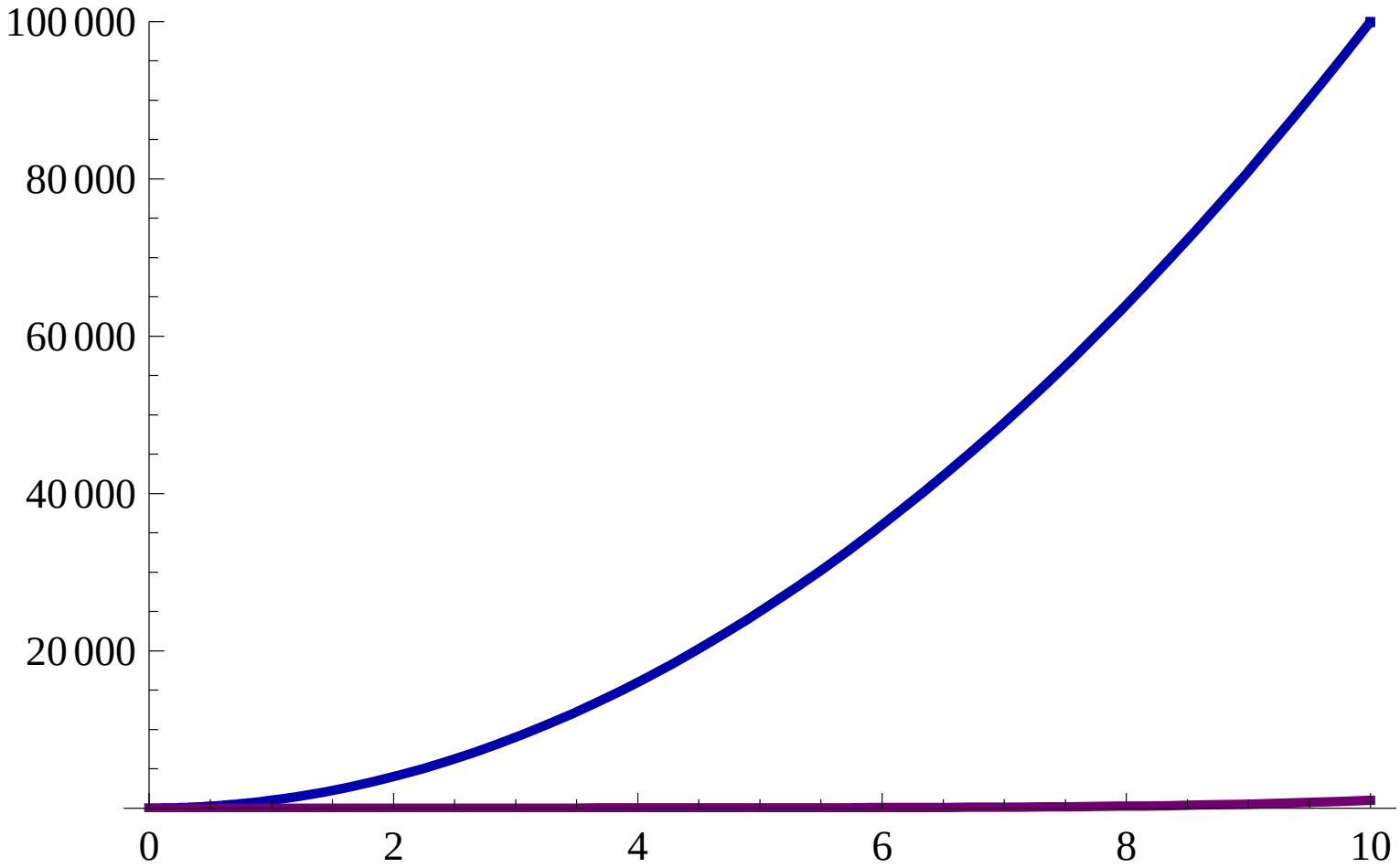
Comparison of $\Theta(n^2)$ and $\Theta(2^n)$

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μs	1 μs
110	1.2 μs	1 ms
120	1.4 μs	1 s
130	1.7 μs	18 m
140	2.0 μs	13 d
150	2.3 μs	37 yr

Comparison of $\Theta(n^2)$ and $\Theta(2^n)$

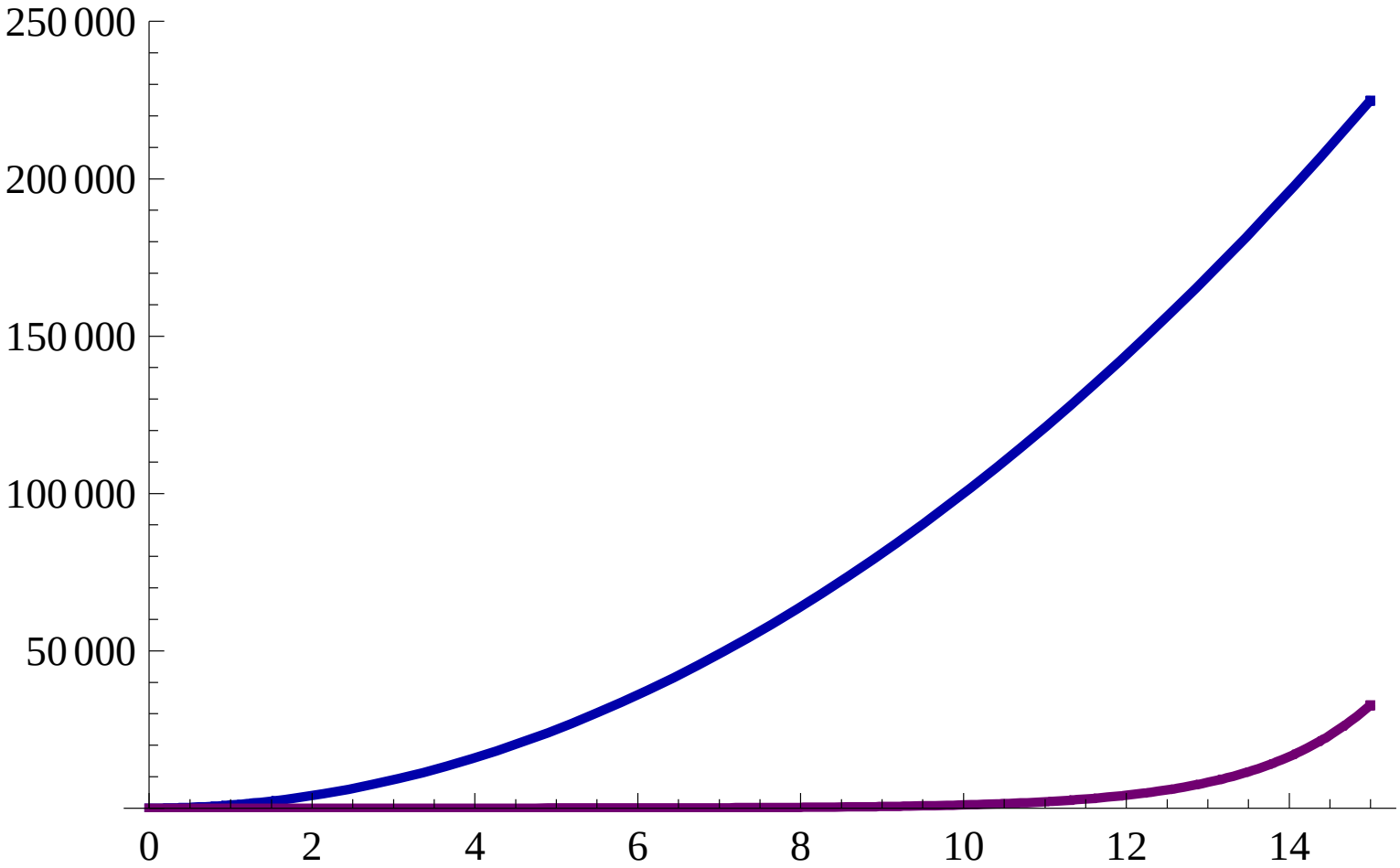
n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μs	1 μs
110	1.2 μs	1 ms
120	1.4 μs	1 s
130	1.7 μs	18 m
140	2.0 μs	13 d
150	2.3 μs	37 yr
160	2.6 μs	37,000 yr

Examples



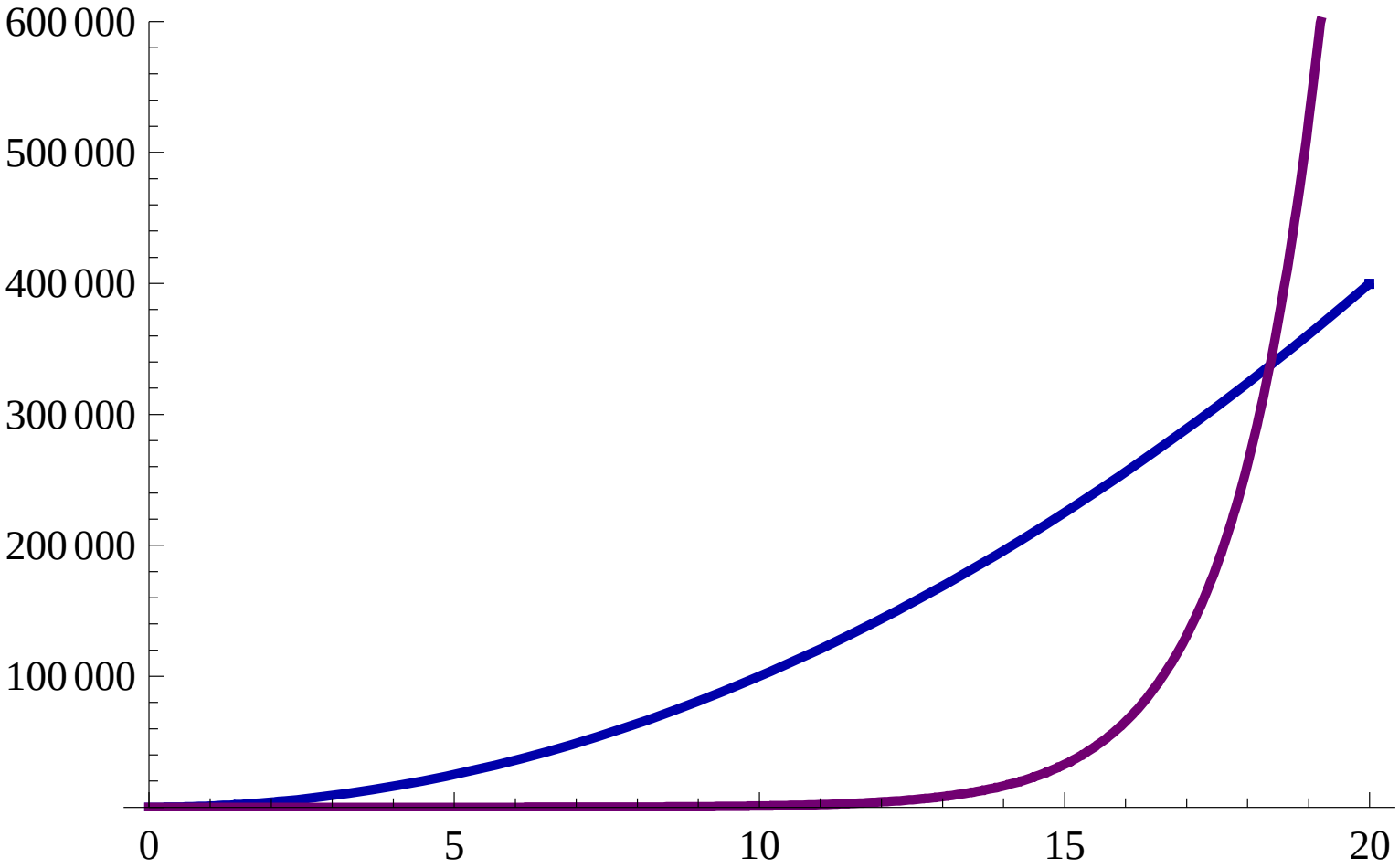
$g(n) = 2^n$ vs. $1000f(n) = 1000n^2$, $0 \leq n \leq 10$

Examples



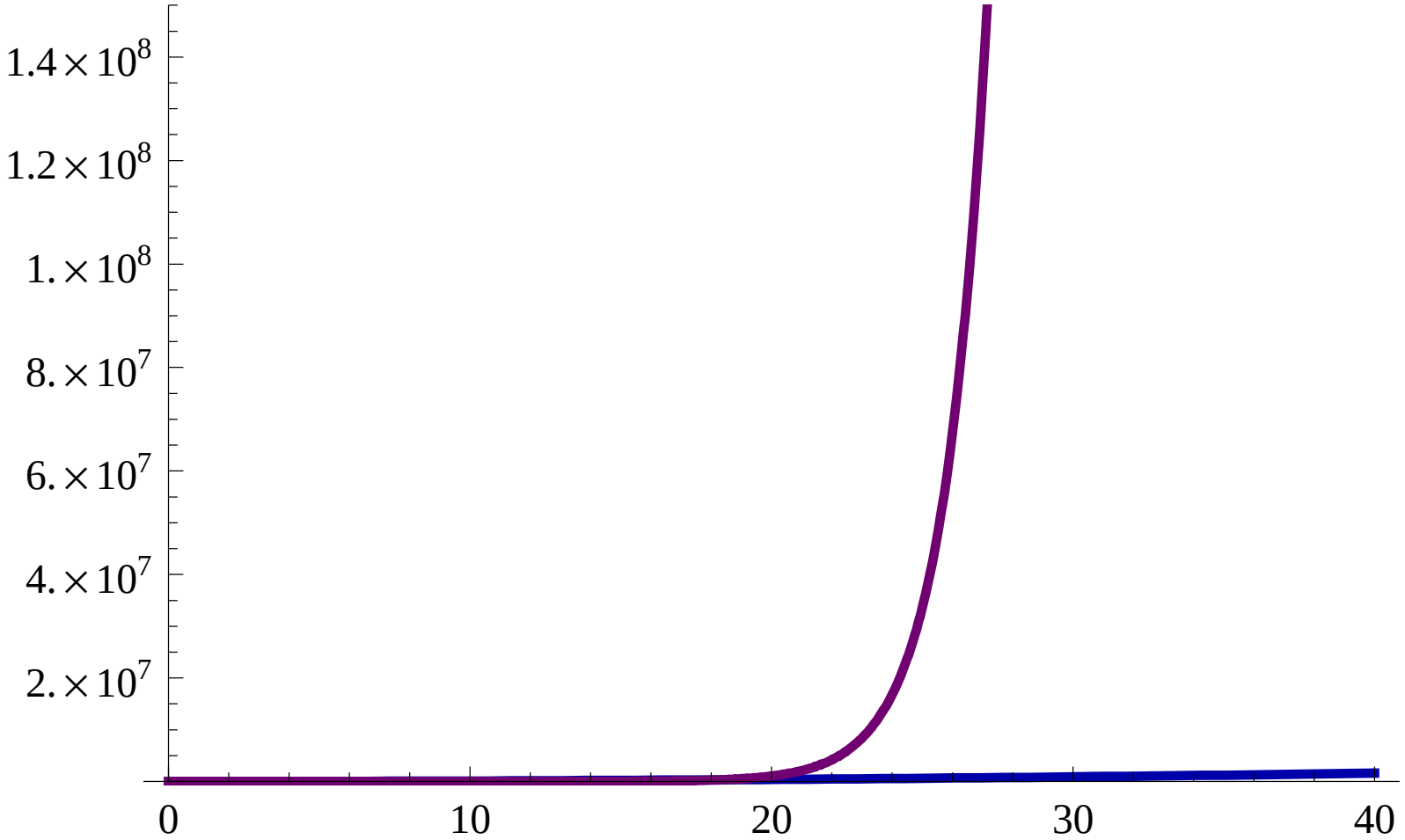
$g(n) = 2^n$ vs. $1000f(n) = 1000n^2$, $0 \leq n \leq 15$

Examples



$g(n) = 2^n$ vs. $1000f(n) = 1000n^2$, $0 \leq n \leq 20$

Examples



$g(n) = 2^n$ vs. $1000f(n) = 1000n^2$, $0 \leq n \leq 40$

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

- Now consider a program with growth rate $\Theta(n!)$.
- Assume that the program runs in 1 μs when the input size is 100.
- Calculate the run times for input sizes $100, 101, 102, \ldots, 110$.
- Note that we will not try to go as far as 120. How come?

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs
102	4 μs	10 ms

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs
102	4 μs	10 ms
103	8 μs	1.1 s

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs
102	4 μs	10 ms
103	8 μs	1.1 s
104	16 μs	1.8 m

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs
102	4 μs	10 ms
103	8 μs	1.1 s
104	16 μs	1.8 m
105	32 μs	3.2 hr

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs
102	4 μs	10 ms
103	8 μs	1.1 s
104	16 μs	1.8 m
105	32 μs	3.2 hr
106	64 μs	14 d

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs
102	4 μs	10 ms
103	8 μs	1.1 s
104	16 μs	1.8 m
105	32 μs	3.2 hr
106	64 μs	14 d
107	128 μs	4.2 yr

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs
102	4 μs	10 ms
103	8 μs	1.1 s
104	16 μs	1.8 m
105	32 μs	3.2 hr
106	64 μs	14 d
107	128 μs	4.2 yr
108	256 μs	450 yr

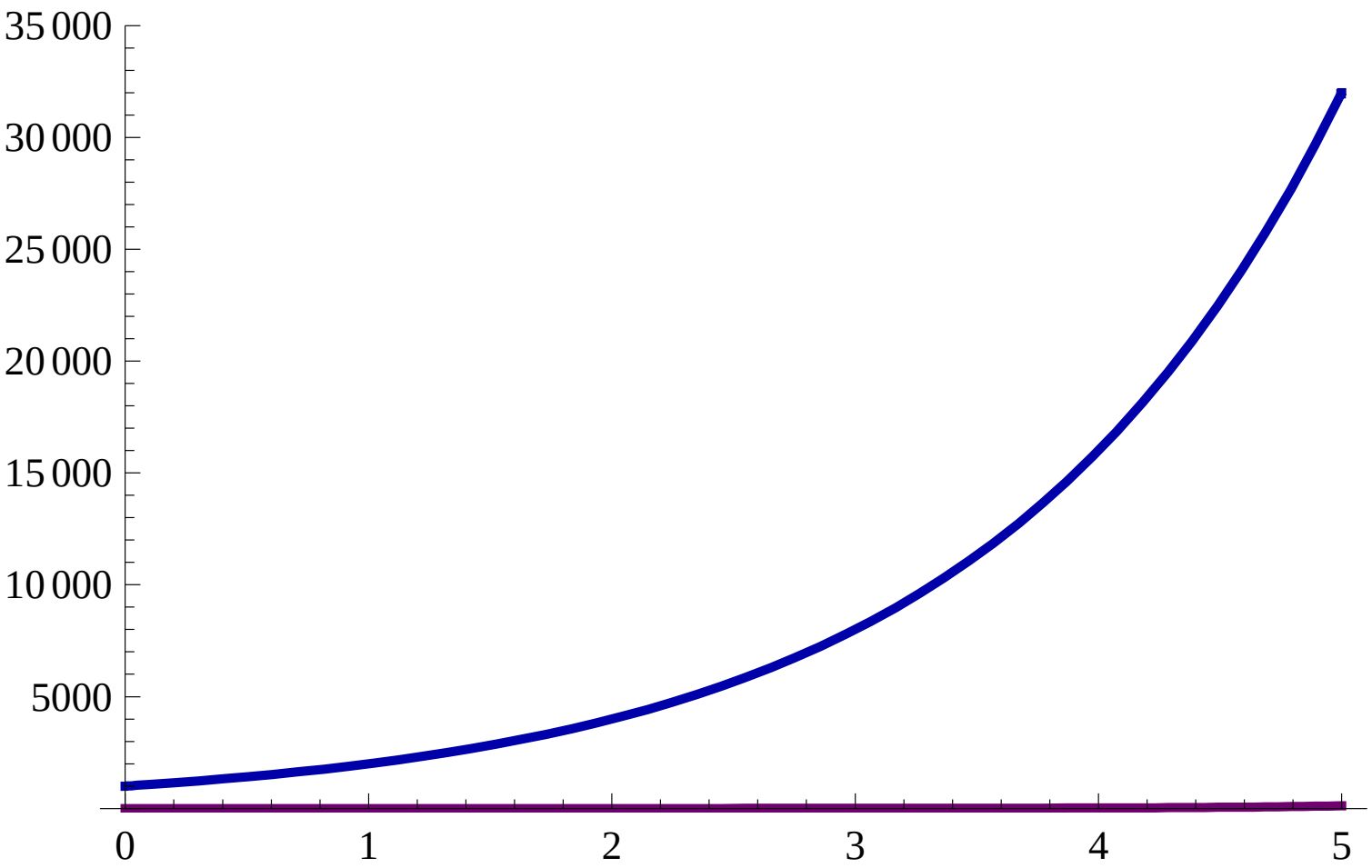
Comparison of $\Theta(2^n)$ and $\Theta(n!)$

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs
102	4 μs	10 ms
103	8 μs	1.1 s
104	16 μs	1.8 m
105	32 μs	3.2 hr
106	64 μs	14 d
107	128 μs	4.2 yr
108	256 μs	450 yr
109	512 μs	49,000 yr

Comparison of $\Theta(2^n)$ and $\Theta(n!)$

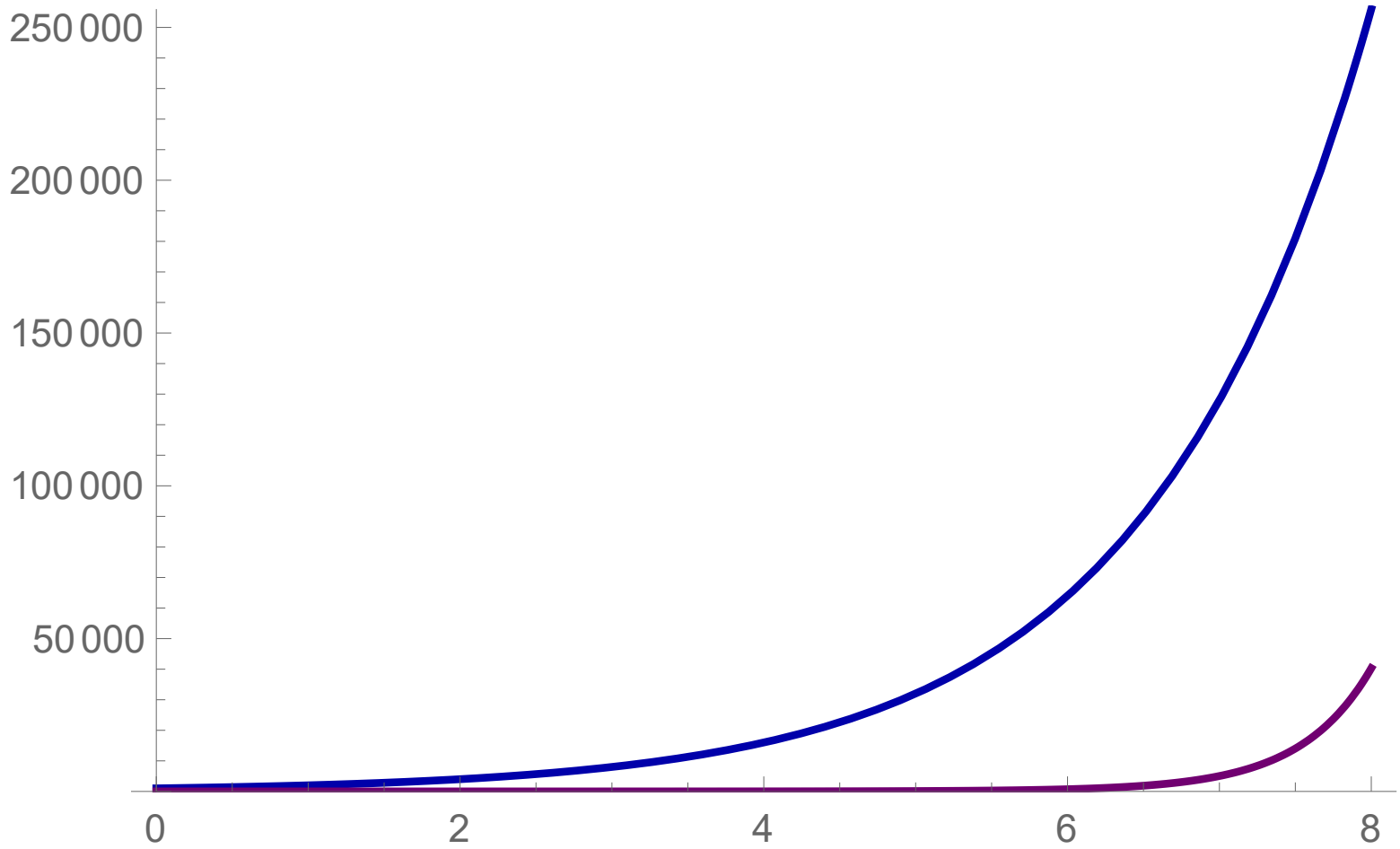
n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μs	1 μs
101	2 μs	100 μs
102	4 μs	10 ms
103	8 μs	1.1 s
104	16 μs	1.8 m
105	32 μs	3.2 hr
106	64 μs	14 d
107	128 μs	4.2 yr
108	256 μs	450 yr
109	512 μs	49,000 yr
110	1024 μs	5,400 mill

Examples



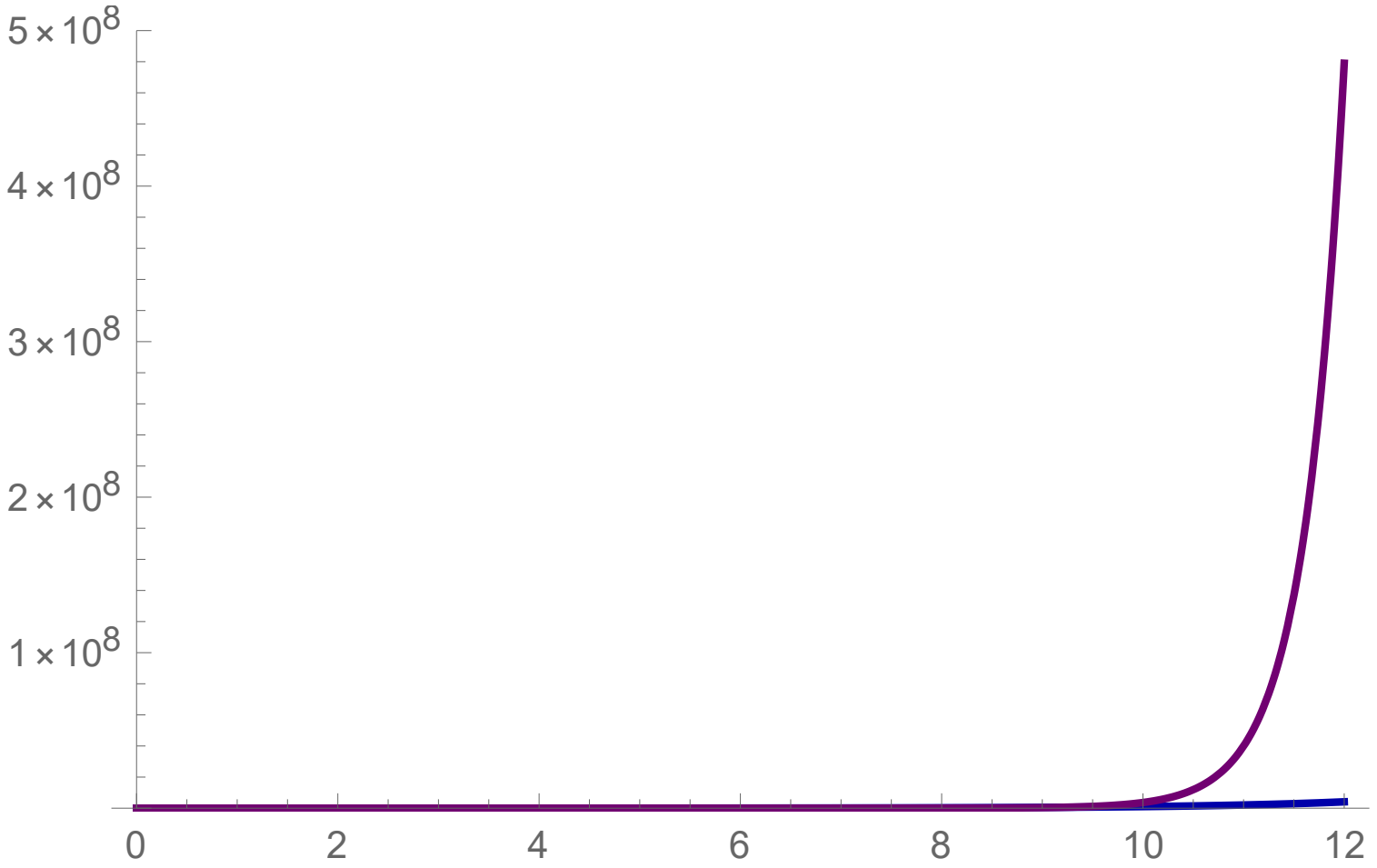
$g(n) = n!$ vs. $1000f(n) = 1000 \cdot 2^n$, $0 \leq n \leq 5$

Examples



$g(n) = n!$ vs. $1000f(n) = 1000 \cdot 2^n$, $0 \leq n \leq 8$

Examples



$g(n) = n!$ vs. $1000f(n) = 1000 \cdot 2^n$, $0 \leq n \leq 12$

Example

Example (Traveling Salesman)

- `TravelingSalesman.cpp`.

Feasible vs. Infeasible

Definition (Feasible)

An algorithm is considered to be feasible if it is $O(n^k)$ for some integer k. Otherwise, it is considered to be infeasible.

- Of course, nearly every algorithm is "feasible" for small input sizes.
- It is feasible to factor small integers ($<$ 50 digits).
- It is infeasible to factor large integers ($>$ 200 digits).

Outline

Outline

Basic Member Functions

- I tested the functions `element()`, `insert()`, `remove()`, `pushFront()`, `popFront()`, `pushBack()`, and `popBack()`.

Evaluation of Lists

	element	insert	remove	pushFront	pushBack	popBack
ArrayList	0.139 μs	0.174 μs	0.148 μs	0.128 μs	0.025 μs	0.023 μs
CircArrayList	0.000 μs	0.174 μs	0.131 μs	0.034 μs	0.025 μs	0.025 μs
LinkedList	0.037 μs	0.299 μs	0.225 μs	0.162 μs	0.339 μs	0.365 μs
LinkedListwTail	0.037 μs	0.296 μs	0.233 μs	0.182 μs	0.148 μs	0.288 μs
DoublyLinkedList	0.125 μs	0.365 μs	0.214 μs	0.196 μs	0.122 μs	0.060 μs
CircLinkedList	0.017 μs	0.265 μs	0.182 μs	0.162 μs	0.137 μs	0.062 μs

List size = 100.

Evaluation of Lists

	element	insert	remove	pushFront	pushBack	popBack
ArrayList	0.015 μs	0.564 μs	0.555 μs	0.874 μs	0.010 μs	0.009 μs
CircArrayList	0.001 μs	0.366 μs	0.313 μs	0.012 μs	0.008 μs	0.011 μs
LinkedList	0.534 μs	1.352 μs	1.349 μs	0.095 μs	2.514 μs	2.406 μs
LinkedListwTail	0.532 μs	1.344 μs	1.341 μs	0.095 μs	0.091 μs	2.263 μs
DoublyLinkedList	0.323 μs	0.933 μs	0.903 μs	0.107 μs	0.099 μs	0.049 μs
CircLinkedList	0.245 μs	0.857 μs	0.907 μs	0.098 μs	0.092 μs	0.053 μs

List size = 1000.

Evaluation of Lists

	element	insert	remove	pushFront	pushBack	popBack
ArrayList	0.002 μs	4.786 μs	4.810 μs	8.488 μs	0.014 μs	0.009 μs
CircArrayList	0.001 μs	2.310 μs	2.234 μs	0.017 μs	0.013 μs	0.013 μs
LinkedList	7.866 μs	31.30 μs	43.30 μs	0.093 μs	32.89 μs	32.38 μs
LinkedListwTail	7.883 μs	31.11 μs	42.10 μs	0.091 μs	0.122 μs	32.78 μs
DoublyLinkedList	4.435 μs	25.73 μs	33.05 μs	0.099 μs	0.098 μs	0.052 μs
CircLinkedList	4.513 μs	25.00 μs	32.45 μs	0.101 μs	0.093 μs	0.058 μs

List size = 10000.

Evaluation of Lists

	element	insert	remove	pushFront	pushBack	popBack
ArrayList	0.000 μs	72.96 μs	77.51 μs	86.04 μs	0.012 μs	0.007 μs
CircArrayList	0.001 μs	12.34 μs	12.07 μs	0.014 μs	0.010 μs	0.010 μs
LinkedList	87.85 μs	277.1 μs	300.9 μs	0.086 μs	337.5 μs	343.6 μs
LinkedListwTail	88.13 μs	292.0 μs	321.3 μs	0.087 μs	0.087 μs	345.3 μs
DoublyLinkedList	53.68 μs	300.5 μs	311.8 μs	0.098 μs	0.103 μs	0.062 μs
CircLinkedList	54.55 μs	292.7 μs	340.4 μs	0.104 μs	0.100 μs	0.072 μs

List size = 100000.

Outline

Evaluation of Lists

	Seq Search	seqSearch	Bin Search	binSearch
ArrayList	616 μs	31.3 μs	94.4 μs	13.5 μs
CircArrayList	810 μs	36.5 μs	127 μs	16.7 μs
LinkedList	1.66 ms	39.5 μs	351 μs	63.6 μs
DoublyLinkedList	1.45 ms	39.3 μs	258 μs	64.0 μs
CircLinkedList	1.44 ms	40.3 μs	249 μs	62.9 μs

	Sel Sort	selSort	Merge Sort	mergeSort
ArrayList	675 μs	31.1 μs	644 μs	268 μs
CircArrayList	895 μs	37.6 μs	799 μs	322 μs
LinkedList	2.70 ms	35.9 μs	1.77 ms	641 μs
DoublyLinkedList	1.59 ms	36.5 μs	1.44 ms	646 μs
CircLinkedList	1.59 ms	36.7 μs	1.53 ms	756 μs

List size = 100.

Evaluation of Lists

	Seq Search	seqSearch	Bin Search	binSearch
ArrayList	61.5 ms	2.56 ms	1.34 ms	174 μs
CircArrayList	79.6 ms	2.61 ms	1.85 ms	173 μs
LinkedList	1.01 s	3.83 ms	45.5 ms	5.87 ms
DoublyLinkedList	660 ms	3.81 ms	20.5 ms	5.93 ms
CircLinkedList	626 ms	4.03 ms	20.0 ms	5.82 ms

	Sel Sort	selSort	Merge Sort	mergeSort
ArrayList	62.1 ms	2.37 ms	7.88 ms	3.37 ms
CircArrayList	81.0 ms	3.10 ms	10.0 ms	4.15 ms
LinkedList	2.38 s	3.74 ms	154 ms	9.44 ms
DoublyLinkedList	645 ms	3.60 ms	62.5 ms	9.67 ms
CircLinkedList	616 ms	3.83 ms	62.4 ms	10.6 ms

List size = 1000.

Evaluation of Lists

	Seq Search	seqSearch	Bin Search	binSearch
ArrayList	6.01 s	252. ms	17.4 ms	2.13 ms
CircArrayList	7.96 s	253. ms	25.6 ms	2.10 ms
LinkedList	20.7 m	417. ms	7.21 s	735. ms
DoublyLinkedList	15.7 m	405. ms	3.88 s	726. ms
CircLinkedList	15.1 m	434. ms	3.81 s	721. ms

	Sel Sort	selSort	Merge Sort	mergeSort
ArrayList	6.02 s	230. ms	102. ms	41.6 ms
CircArrayList	7.97 s	320. ms	130. ms	52.1 ms
LinkedList	41.3 m	402. ms	20.5 s	134. ms
DoublyLinkedList	15.5 m	392. ms	9.93 s	136. ms
CircLinkedList	15.7 m	423. ms	9.92 s	147. ms

List size = 10000.

Outline

Assignment

Assignment

- Read Section 9.6.