Time Complexity

Lecture 21
Section 9.6

Robb T. Koether

Hampden-Sydney College

Wed, Mar 8, 2017

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 $n \geq n_0$.

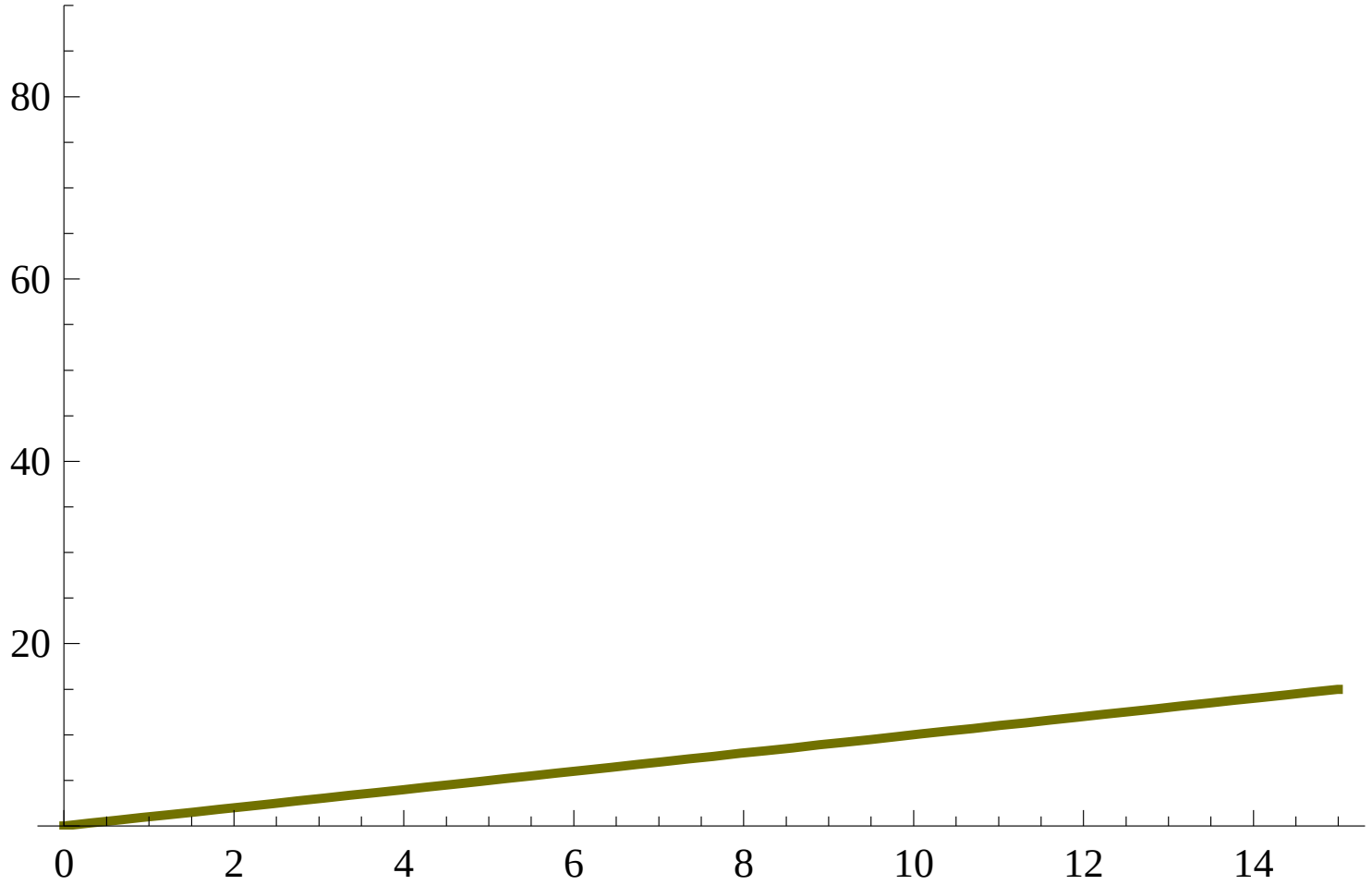
- In other words, $g(n)$ is bounded above by a constant multiple of $f(n)$ from some point on.

Examples

Example (Proving and Disproving Big-O)

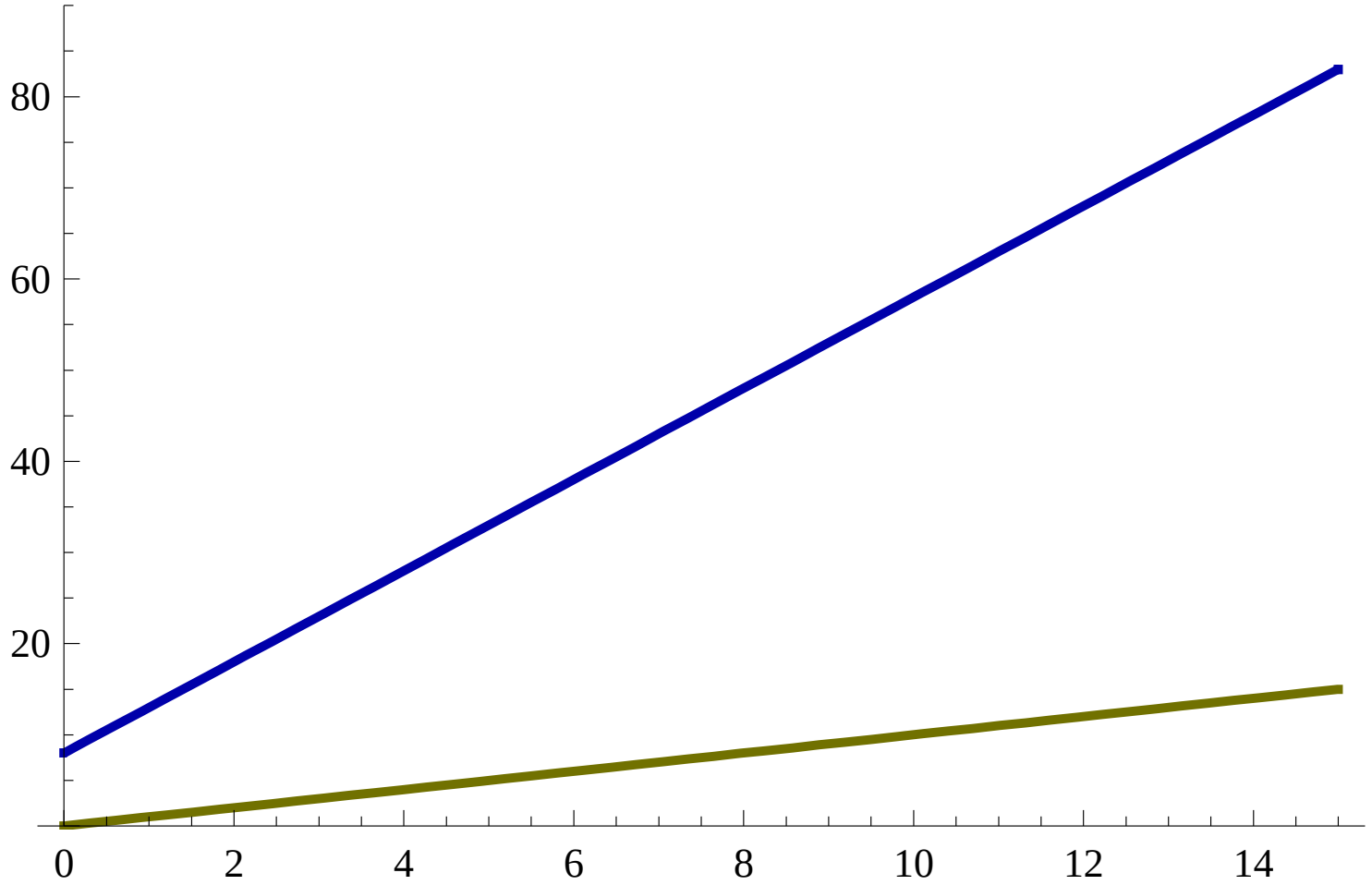
- Show that $5n + 8$ is $O(n)$.
- Show that $n^2 + 10n + 5$ is $O(n^2)$.
- Show that $n^2 + 10n + 5$ is *not* $O(n)$.

Examples



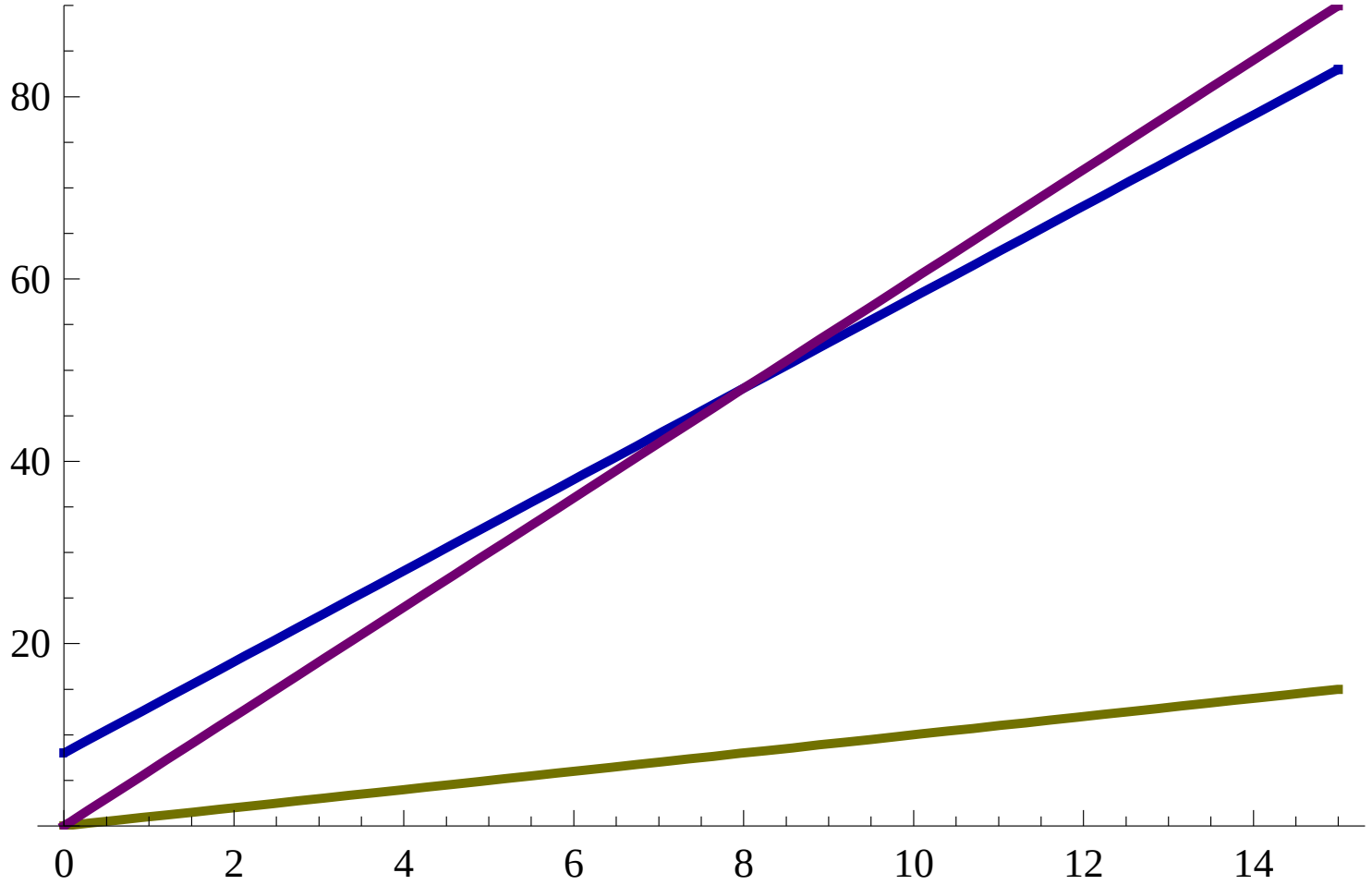
$f(n) = n$

Examples



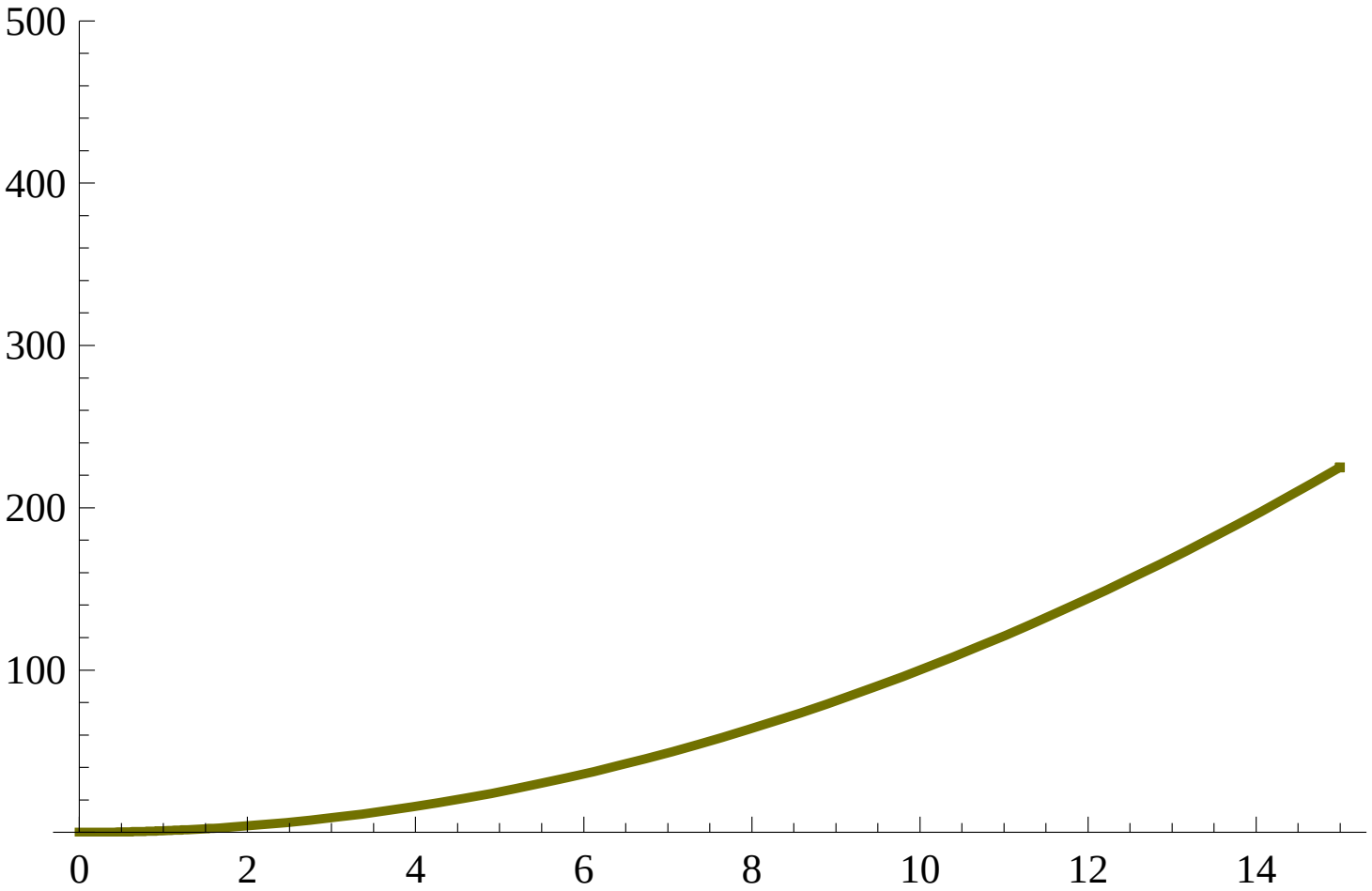
$f(n) = n$ and $g(n) = 5n + 8$

Examples



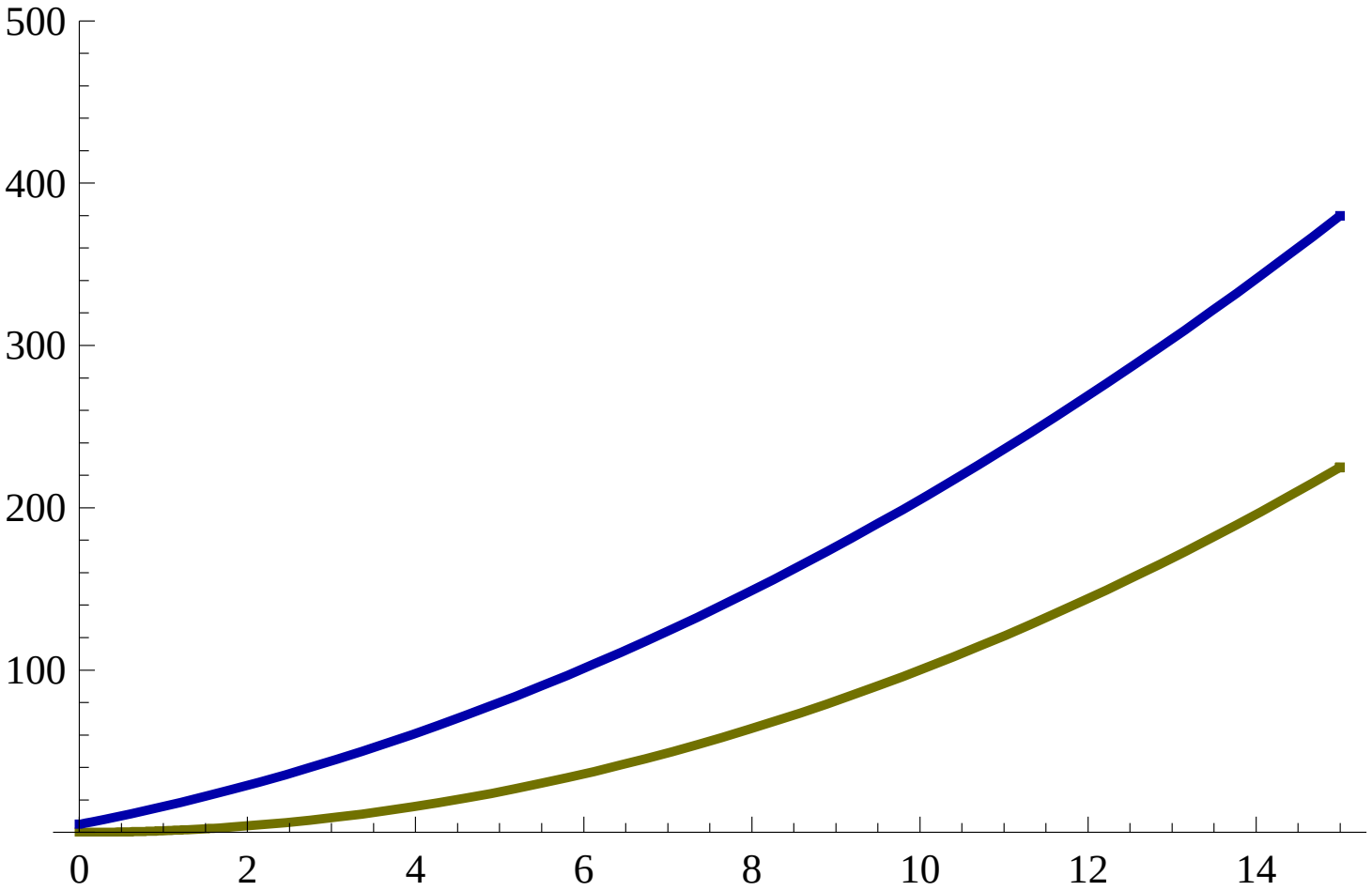
$f(n) = n$ and $g(n) = 5n + 8$ and $6f(n) = 6n$

Examples



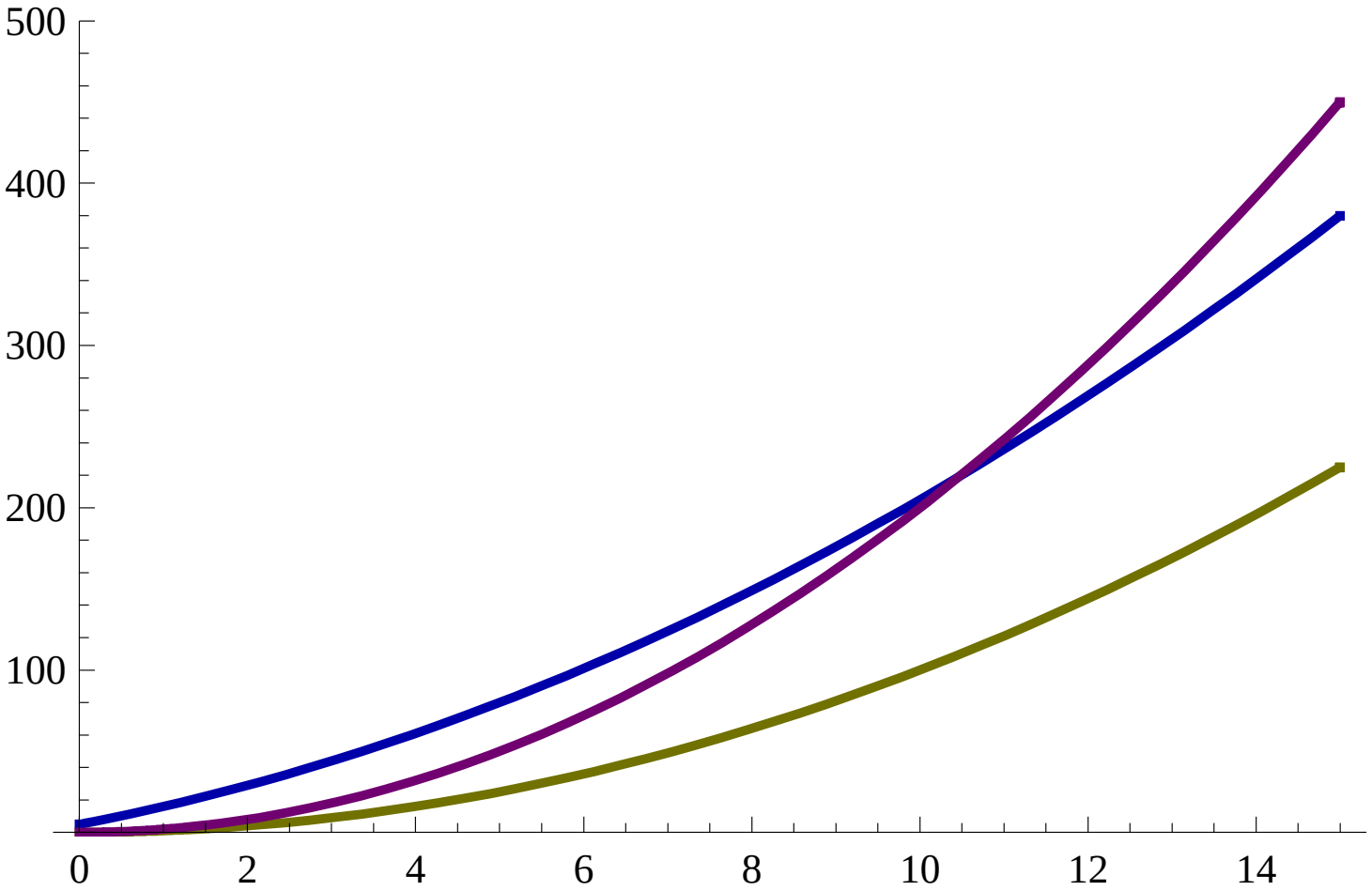
$f(n) = n^2$

Examples



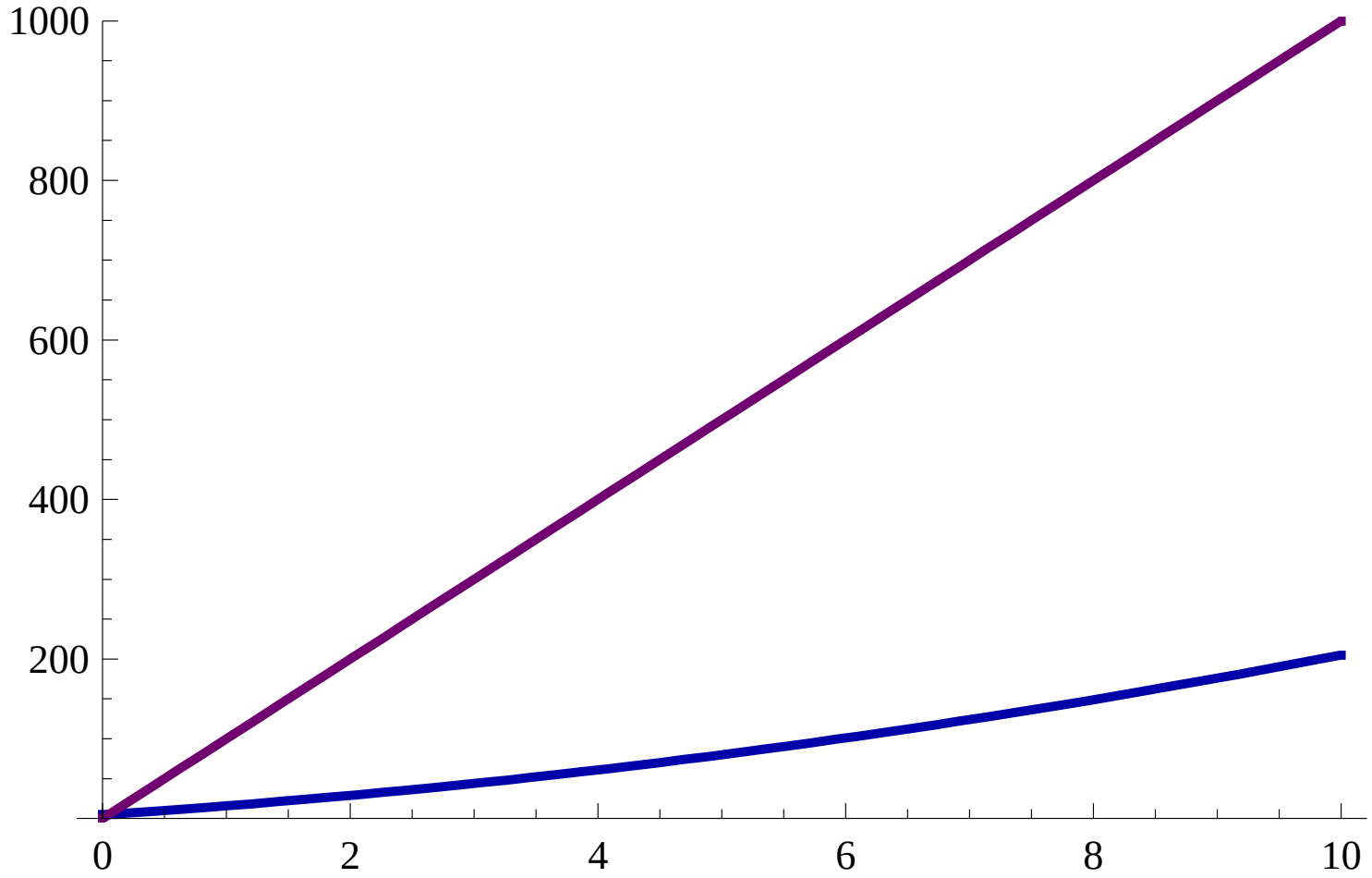
$f(n) = n^2$ and $g(n) = n^2 + 10n + 5$

Examples



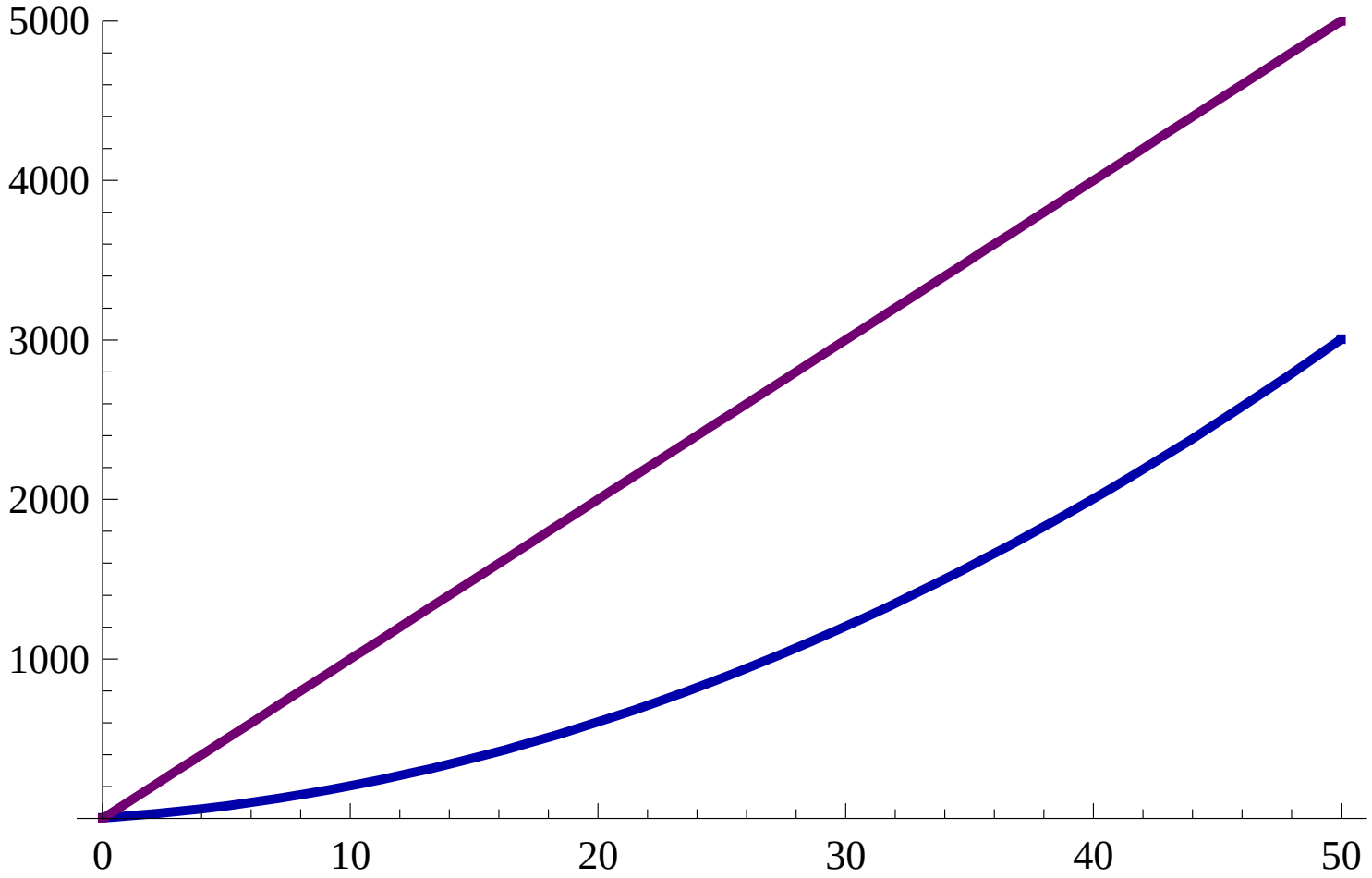
$f(n) = n^2$ and $g(n) = n^2 + 10n + 5$ and $2f(n) = 2n^2$

Examples



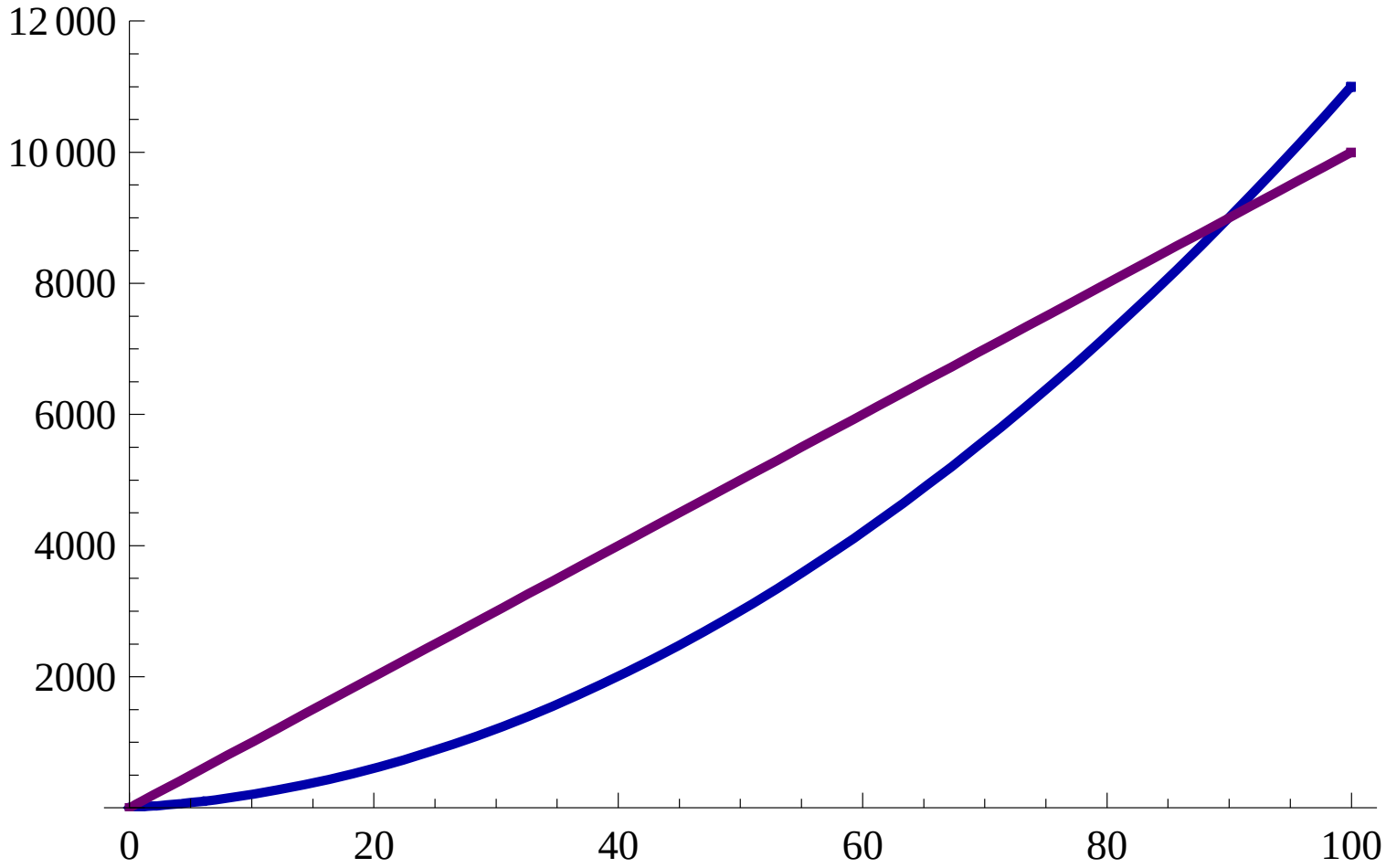
$100f(n) = 100n$ and $g(n) = n^2 + 10n + 5$, over $0 \leq n \leq 10$

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

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 $t_0 = 5$ μsec 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 $t_0 = 5$ μsec when the input size is $n_0 = 100$.
- Then

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

Example

Example (Estimating Run Times)

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

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

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

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

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 $t_0 = 5$ μsec 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 $t_0 = 5$ μsec when the input size is $n_0 = 100$.
- Then

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

Example

Example (Estimating Run Times)

- Suppose the run time of a program is

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

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

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

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

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

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 μsec 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 μsec	1 μsec	1 μsec	1 μsec	1 μsec

Comparison of Growth Rates

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

Comparison of Growth Rates

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

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μsec	1 μsec	1 μsec	1 μsec	1 μsec
10^3	1 μsec	1.5 μsec	10 μsec	15 μsec	100 μsec
10^4	1 μsec	2 μsec	100 μsec	200 μsec	10 msec
10^5	1 μsec	2.5 μsec	1 msec	2.5 msec	1 sec

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μsec	1 μsec	1 μsec	1 μsec	1 μsec
10^3	1 μsec	1.5 μsec	10 μsec	15 μsec	100 μsec
10^4	1 μsec	2 μsec	100 μsec	200 μsec	10 msec
10^5	1 μsec	2.5 μsec	1 msec	2.5 msec	1 sec
10^6	1 μsec	3 μsec	10 msec	30 msec	1.7 min

Comparison of Growth Rates

n	$\Theta(1)$	$\Theta(\log_2 n)$	$\Theta(n)$	$\Theta(n \log_2 n)$	$\Theta(n^2)$
10^2	1 μsec	1 μsec	1 μsec	1 μsec	1 μsec
10^3	1 μsec	1.5 μsec	10 μsec	15 μsec	100 μsec
10^4	1 μsec	2 μsec	100 μsec	200 μsec	10 msec
10^5	1 μsec	2.5 μsec	1 msec	2.5 msec	1 sec
10^6	1 μsec	3 μsec	10 msec	30 msec	1.7 min
10^7	1 μsec	3.5 μsec	100 msec	350 msec	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 μsec	1 μsec	1 μsec	1 μsec	1 μsec
10^3	1 μsec	1.5 μsec	10 μsec	15 μsec	100 μsec
10^4	1 μsec	2 μsec	100 μsec	200 μsec	10 msec
10^5	1 μsec	2.5 μsec	1 msec	2.5 msec	1 sec
10^6	1 μsec	3 μsec	10 msec	30 msec	1.7 min
10^7	1 μsec	3.5 μsec	100 msec	350 msec	2.8 hr
10^8	1 μsec	4 μsec	1 sec	4 sec	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 μsec 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 μsec	1 μsec

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

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μsec	1 μsec
110	1.2 μsec	1 msec

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

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μsec	1 μsec
110	1.2 μsec	1 msec
120	1.4 μsec	1 sec

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

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μsec	1 μsec
110	1.2 μsec	1 msec
120	1.4 μsec	1 sec
130	1.7 μsec	18 min

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

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μsec	1 μsec
110	1.2 μsec	1 msec
120	1.4 μsec	1 sec
130	1.7 μsec	18 min
140	2.0 μsec	13 d

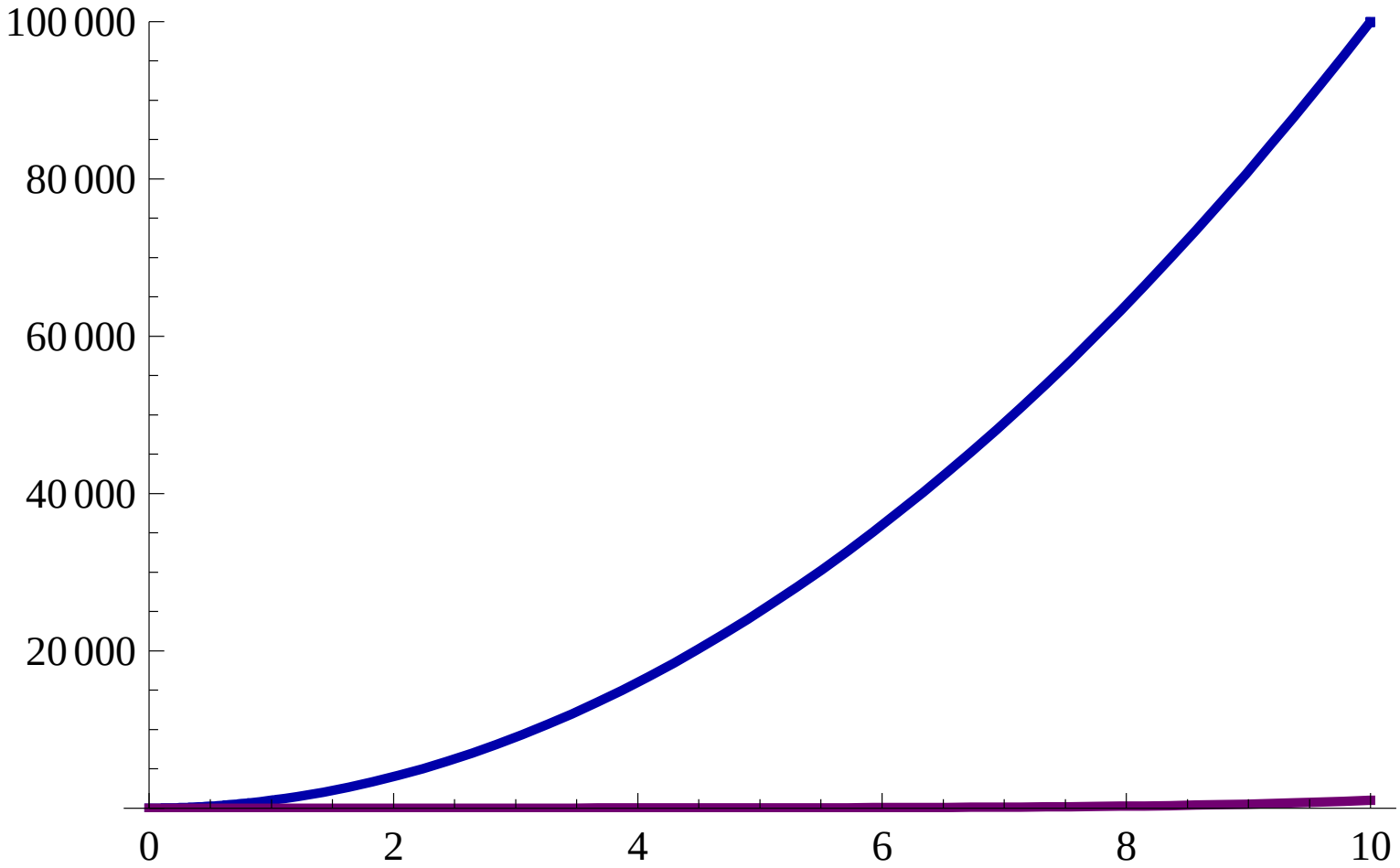
Comparison of $\Theta(n^2)$ and $\Theta(2^n)$

n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μsec	1 μsec
110	1.2 μsec	1 msec
120	1.4 μsec	1 sec
130	1.7 μsec	18 min
140	2.0 μsec	13 d
150	2.3 μsec	37 yr

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

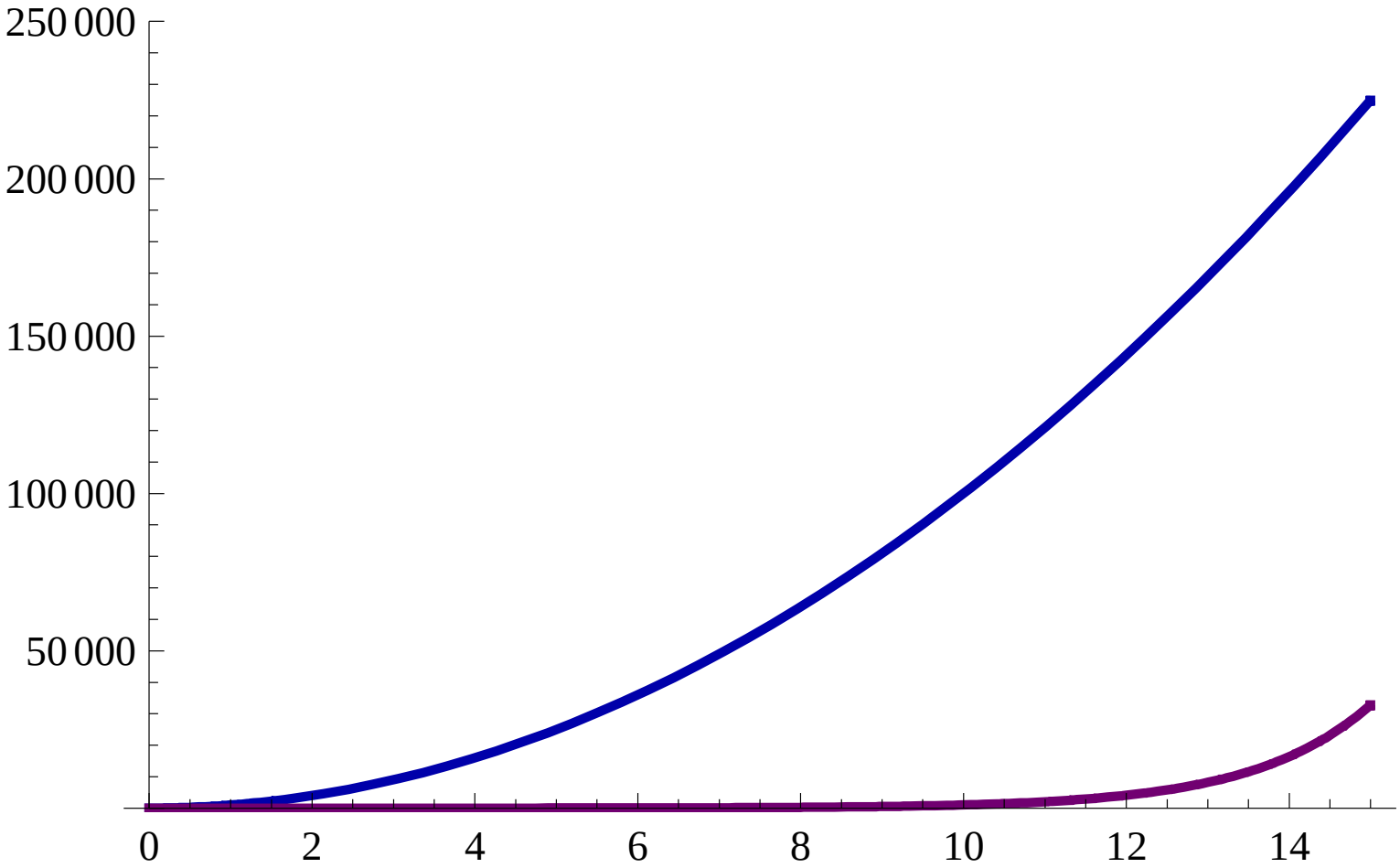
n	$\Theta(n^2)$	$\Theta(2^n)$
100	1 μsec	1 μsec
110	1.2 μsec	1 msec
120	1.4 μsec	1 sec
130	1.7 μsec	18 min
140	2.0 μsec	13 d
150	2.3 μsec	37 yr
160	2.6 μsec	37,000 yr

Examples



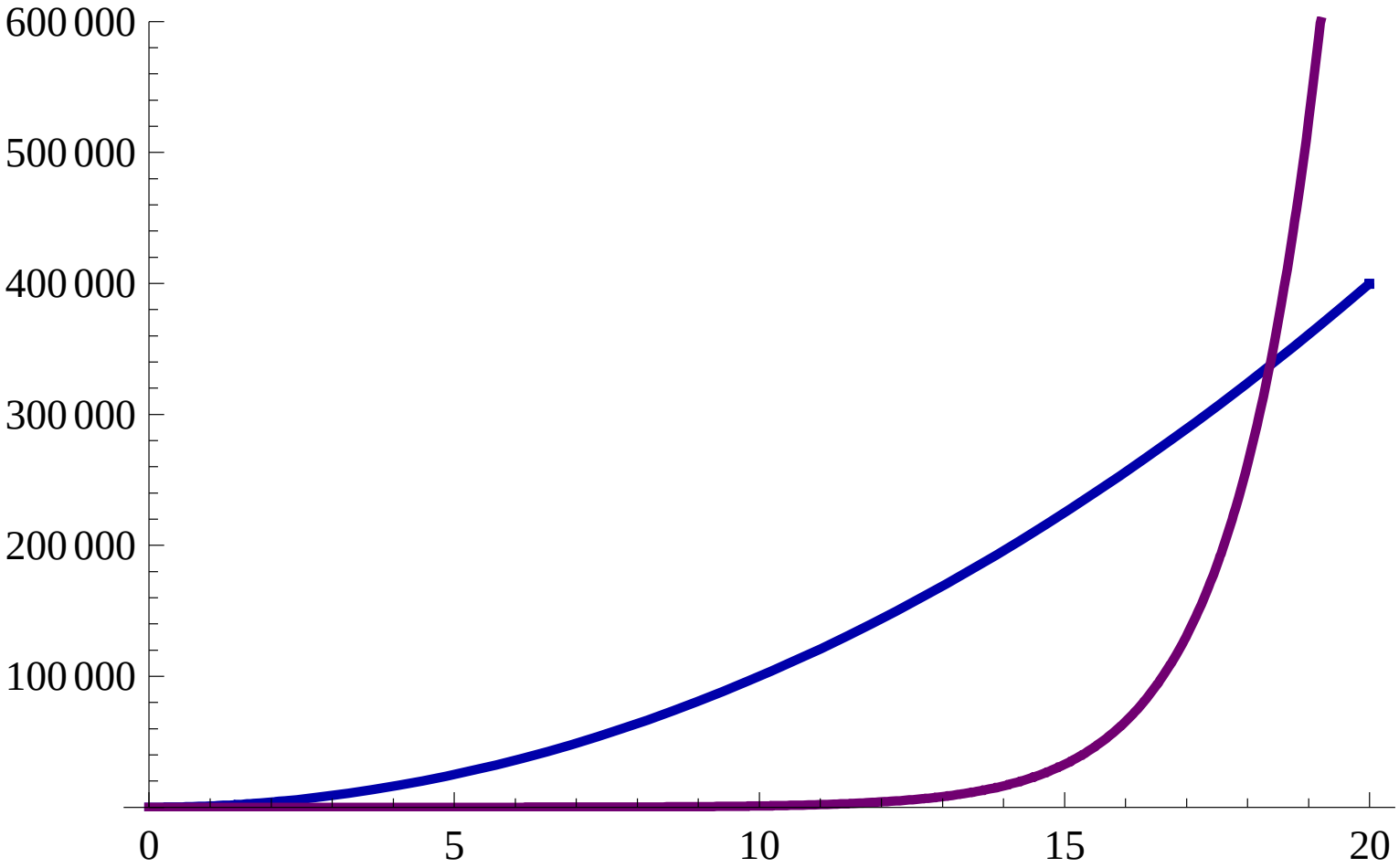
$g(n) = 2^n$ vs. $1000f(n) = 1000n^2$, $0 \leq n \leq 10$

Examples



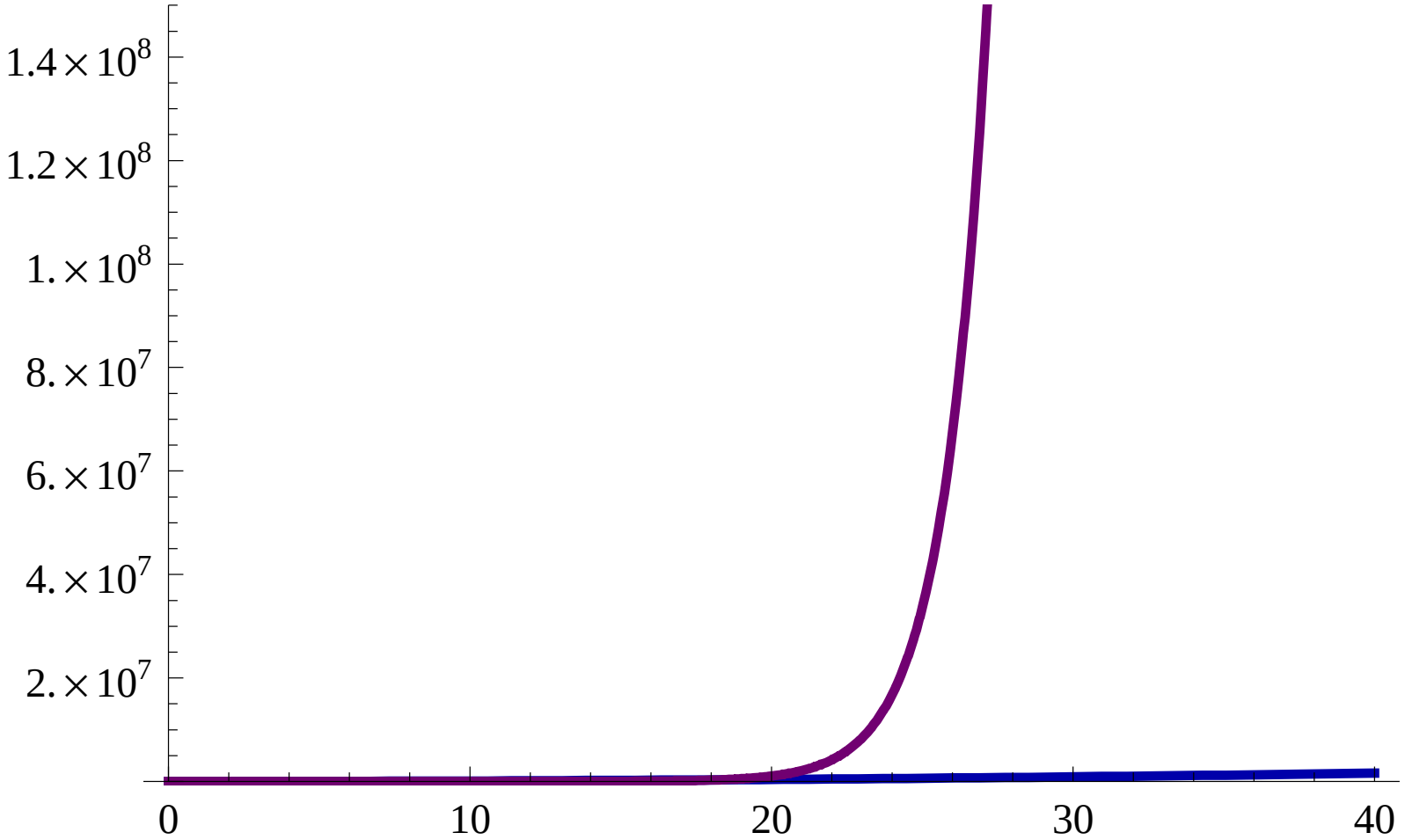
$g(n) = 2^n$ vs. $1000f(n) = 1000n^2$, $0 \leq n \leq 15$

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 μsec 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 μsec	1 μsec

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

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

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

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μsec	1 μsec
101	2 μsec	100 μsec
102	4 μsec	10 msec

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

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μsec	1 μsec
101	2 μsec	100 μsec
102	4 μsec	10 msec
103	8 μsec	1.1 sec

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

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μsec	1 μsec
101	2 μsec	100 μsec
102	4 μsec	10 msec
103	8 μsec	1.1 sec
104	16 μsec	1.8 min

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

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μsec	1 μsec
101	2 μsec	100 μsec
102	4 μsec	10 msec
103	8 μsec	1.1 sec
104	16 μsec	1.8 min
105	32 μsec	3.2 hr

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

n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μsec	1 μsec
101	2 μsec	100 μsec
102	4 μsec	10 msec
103	8 μsec	1.1 sec
104	16 μsec	1.8 min
105	32 μsec	3.2 hr
106	64 μsec	14 d

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

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

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

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

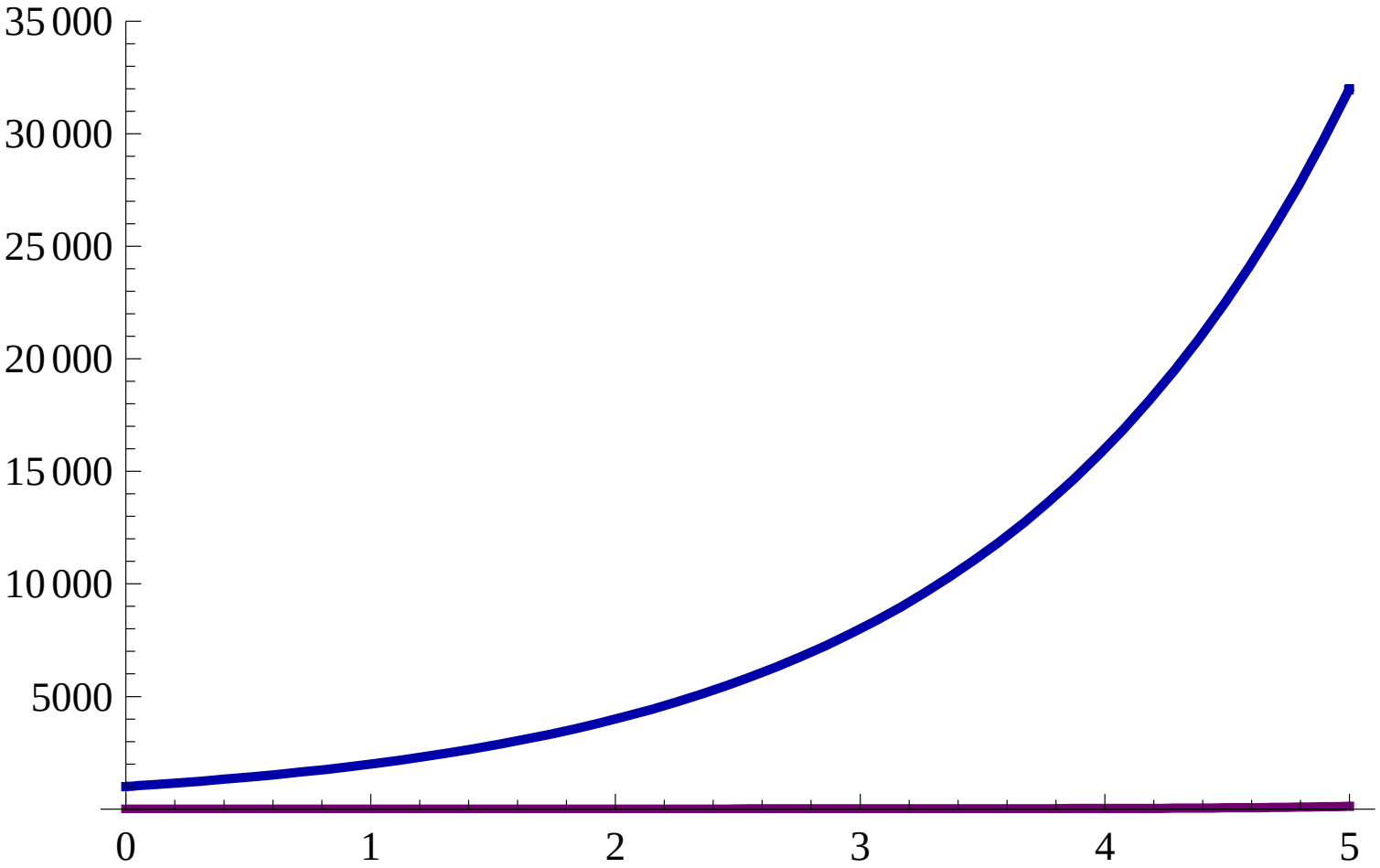
Comparison of $\Theta(2^n)$ and $\Theta(n!)$

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

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

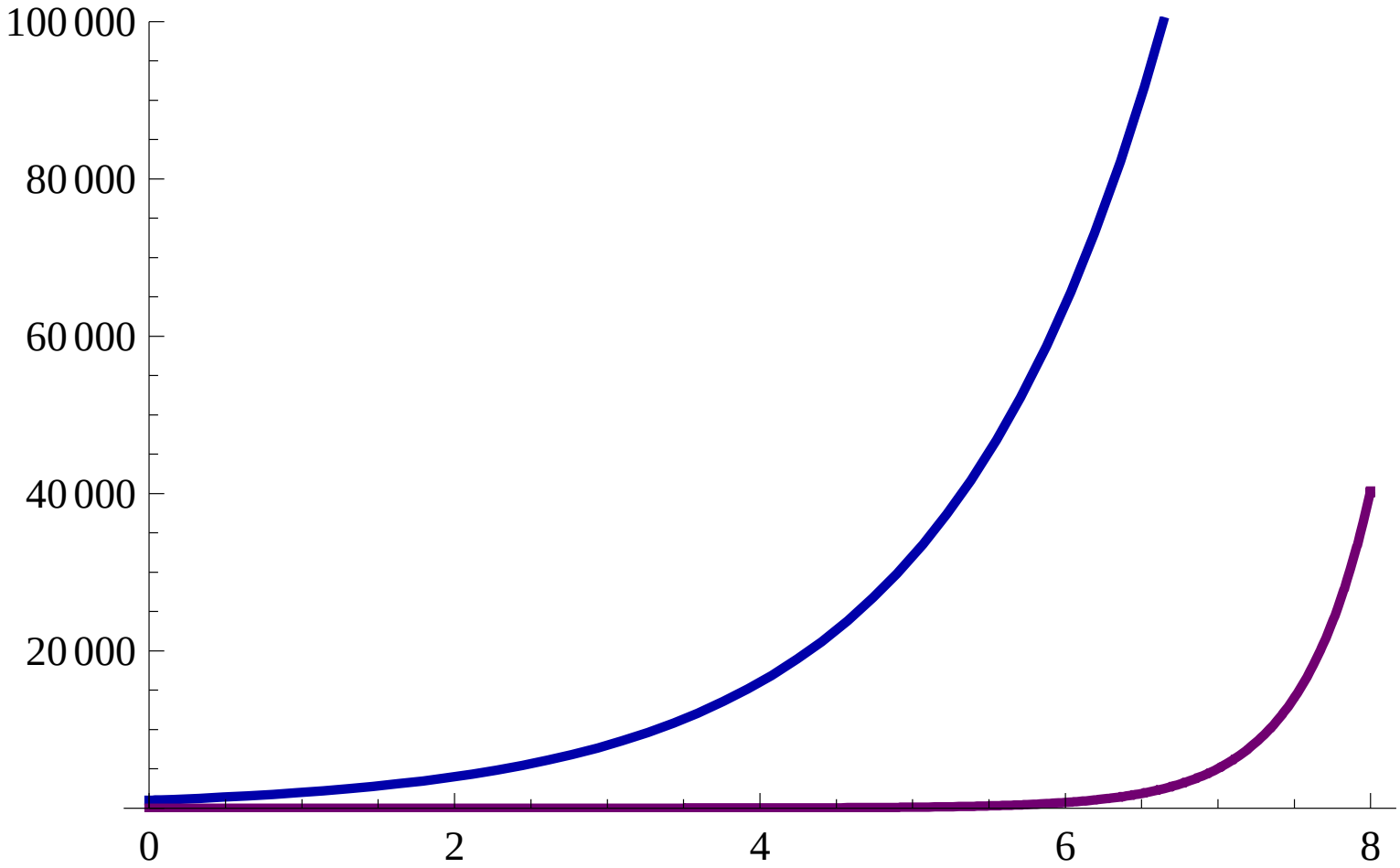
n	$\Theta(2^n)$	$\Theta(n!)$
100	1 μsec	1 μsec
101	2 μsec	100 μsec
102	4 μsec	10 msec
103	8 μsec	1.1 sec
104	16 μsec	1.8 min
105	32 μsec	3.2 hr
106	64 μsec	14 d
107	128 μsec	4.2 yr
108	256 μsec	450 yr
109	512 μsec	49,000 yr
110	1024 μsec	5,400 mill

Examples



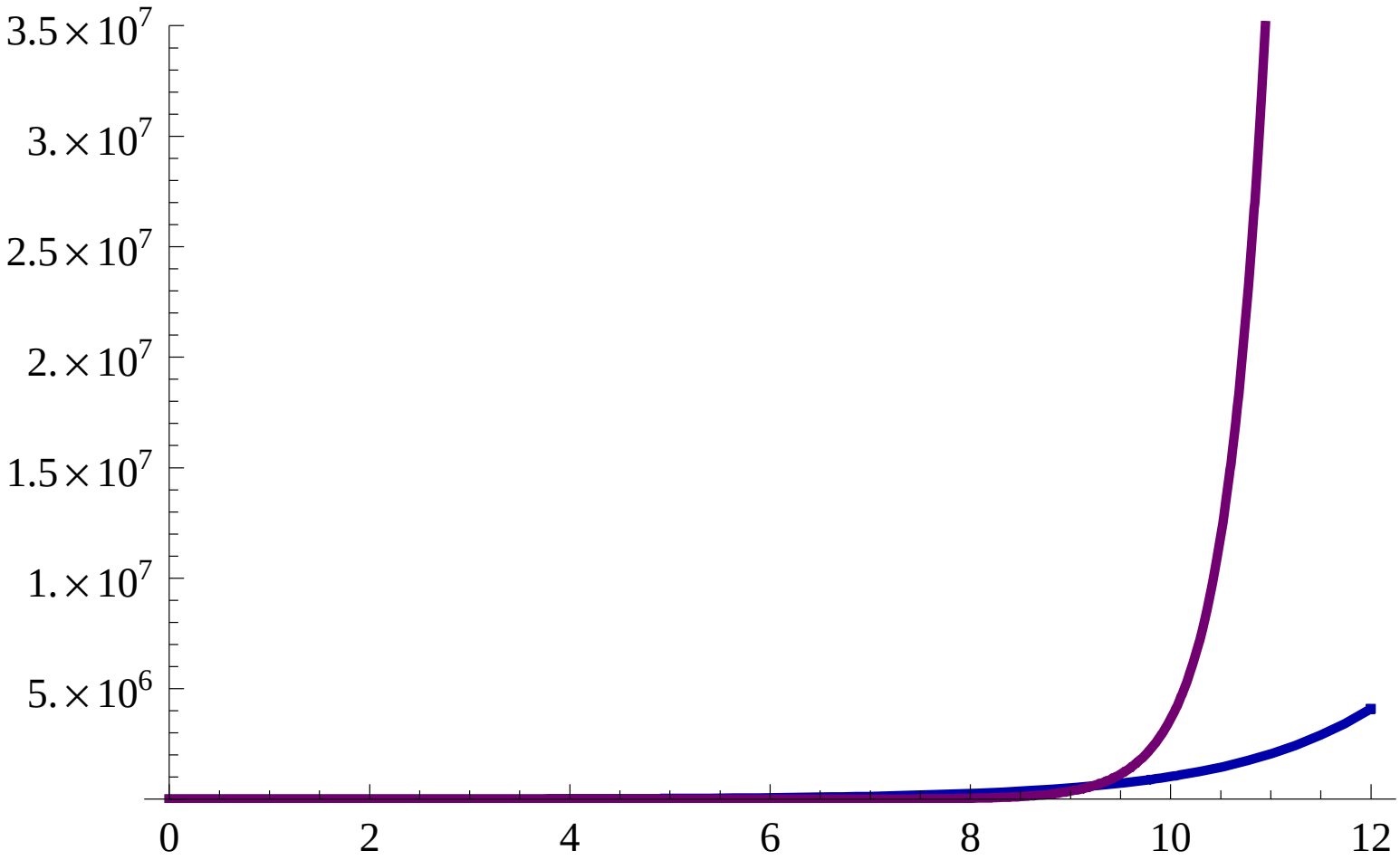
$g(n) = n!$ vs. $1000f(n) = 1000 \cdot 2^n$, $0 \leq n \leq 5$

Examples



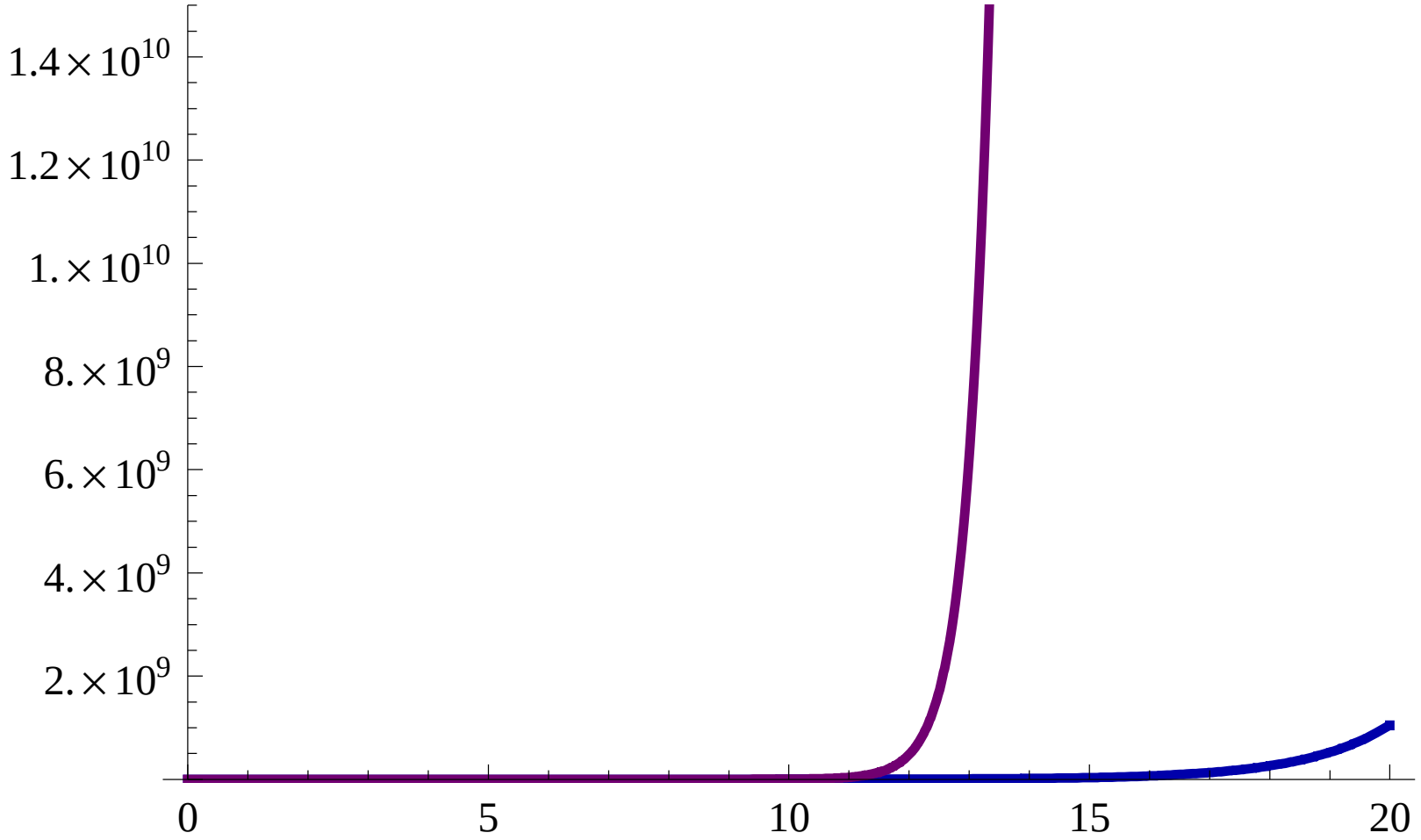
$g(n) = n!$ vs. $1000f(n) = 1000 \cdot 2^n$, $0 \leq n \leq 8$

Examples



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

Examples



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

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 `getElement()`, `insert()`, `remove()`, `pushFront()`, `popFront()`, `pushBack()`, and `popBack()`.

Evaluation of Lists

	operator[]	insert	remove
ArrayList	0.05 μsec	1.01 μsec	1.02 μsec
CircArrayList	0.09 μsec	1.14 μsec	1.14 μsec
LinkedList	0.33 μsec	1.76 μsec	1.82 μsec
LinkedListwTail	0.29 μsec	1.72 μsec	1.87 μsec
DoublyLinkedList	0.21 μsec	1.37 μsec	1.60 μsec
CircLinkedList	0.20 μsec	1.46 μsec	1.63 μsec

	pushFront	popFront	pushBack	popBack
ArrayList	1.56 μsec	1.49 μsec	0.31 μsec	0.38 μsec
CircArrayList	0.33 μsec	0.30 μsec	0.32 μsec	0.31 μsec
LinkedList	1.00 μsec	1.39 μsec	2.04 μsec	3.36 μsec
LinkedListwTail	0.97 μsec	1.26 μsec	0.85 μsec	3.23 μsec
DoublyLinkedList	1.05 μsec	1.44 μsec	1.06 μsec	1.45 μsec
CircLinkedList	1.12 μsec	1.45 μsec	1.11 μsec	1.45 μsec

List size = 100.

Evaluation of Lists

	operator[]	insert	remove
ArrayList	0.04 μsec	5.94 μsec	6.13 μsec
CircArrayList	0.08 μsec	3.68 μsec	3.73 μsec
LinkedList	3.29 μsec	10.7 μsec	10.5 μsec
LinkedListwTail	3.21 μsec	10.7 μsec	11.1 μsec
DoublyLinkedList	1.19 μsec	5.41 μsec	5.73 μsec
CircLinkedList	1.14 μsec	5.54 μsec	5.86 μsec

	pushFront	popFront	pushBack	popBack
ArrayList	11.7 μsec	11.5 μsec	0.20 μsec	0.28 μsec
CircArrayList	0.23 μsec	0.21 μsec	0.20 μsec	0.21 μsec
LinkedList	1.06 μsec	1.43 μsec	17.0 μsec	31.3 μsec
LinkedListwTail	1.13 μsec	1.41 μsec	0.96 μsec	31.3 μsec
DoublyLinkedList	1.17 μsec	1.54 μsec	1.15 μsec	1.53 μsec
CircLinkedList	1.24 μsec	1.54 μsec	1.20 μsec	1.52 μsec

List size = 1000.

Evaluation of Lists

	operator[]	insert	remove
ArrayList	0.04 μsec	67.4 μsec	61.5 μsec
CircArrayList	0.08 μsec	31.1 μsec	29.9 μsec
LinkedList	36.9 μsec	111. μsec	116. μsec
LinkedListwTail	36.5 μsec	112. μsec	116. μsec
DoublyLinkedList	18.6 μsec	58.3 μsec	61.0 μsec
CircLinkedList	18.2 μsec	59.8 μsec	60.6 μsec

	pushFront	popFront	pushBack	popBack
ArrayList	127. μsec	114. μsec	0.22 μsec	0.29 μsec
CircArrayList	0.24 μsec	0.21 μsec	0.22 μsec	0.22 μsec
LinkedList	1.14 μsec	1.44 μsec	165. μsec	314. μsec
LinkedListwTail	1.03 μsec	1.45 μsec	0.98 μsec	314. μsec
DoublyLinkedList	1.27 μsec	1.59 μsec	1.26 μsec	1.64 μsec
CircLinkedList	1.34 μsec	1.55 μsec	1.30 μsec	1.68 μsec

List size = 10000.

Evaluation of Lists

	operator[]	insert	remove
ArrayList	0.04 μsec	1.01 msec	977. μsec
CircArrayList	0.08 μsec	172. μsec	179. μsec
LinkedList	381. μsec	499. μsec	524. μsec
LinkedListwTail	381. μsec	495. μsec	516. μsec
DoublyLinkedList	188. μsec	422. μsec	427. μsec
CircLinkedList	183. μsec	409. μsec	420. μsec

	pushFront	popFront	pushBack	popBack
ArrayList	1.17 msec	1.14 msec	0.21 μsec	0.28 μsec
CircArrayList	0.23 μsec	0.21 μsec	0.21 μsec	0.22 μsec
LinkedList	1.09 μsec	1.47 μsec	2.86 msec	5.59 msec
LinkedListwTail	1.06 μsec	1.39 μsec	0.98 μsec	5.61 msec
DoublyLinkedList	1.18 μsec	1.61 μsec	1.19 μsec	1.61 μsec
CircLinkedList	1.21 μsec	1.62 μsec	1.20 μsec	1.63 μsec

List size = 100000.

Outline

Evaluation of Lists

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

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

List size = 100.

Evaluation of Lists

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

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

List size = 1000.

Evaluation of Lists

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

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

List size = 10000.

Outline

Assignment

Assignment

- Read Section 9.6.