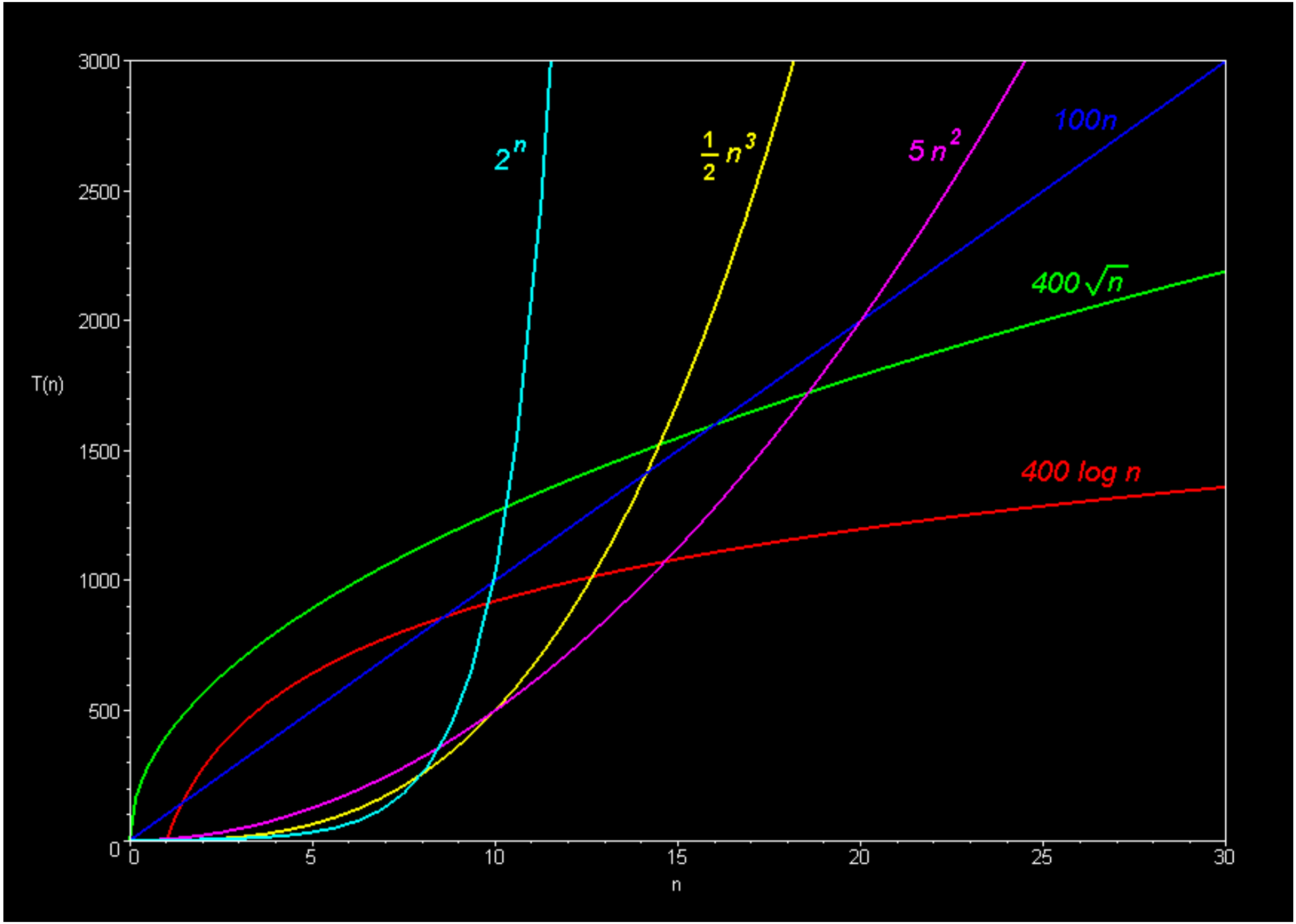
Comparing the Time Complexity of Algorithms

- Linear: $O(n)$ $\mathrm{T}(n) = 100n$
- Quadratic: $O(n^2)$ $\mathrm{T}(n) = 5n^2$
- Cubic: $O(n^3)$ $\mathrm{T}(n) = \frac{1}{2}\, n^3$
- Logarithmic: $O(\log n)$ $\mathrm{T}(n) = 400 \log n$
- Exponential: $O(2^n)$ $\mathrm{T}(n) = 2^n$
- Square root: $O(\sqrt{n})$ $\mathrm{T}(n) = 400 \sqrt{n}$

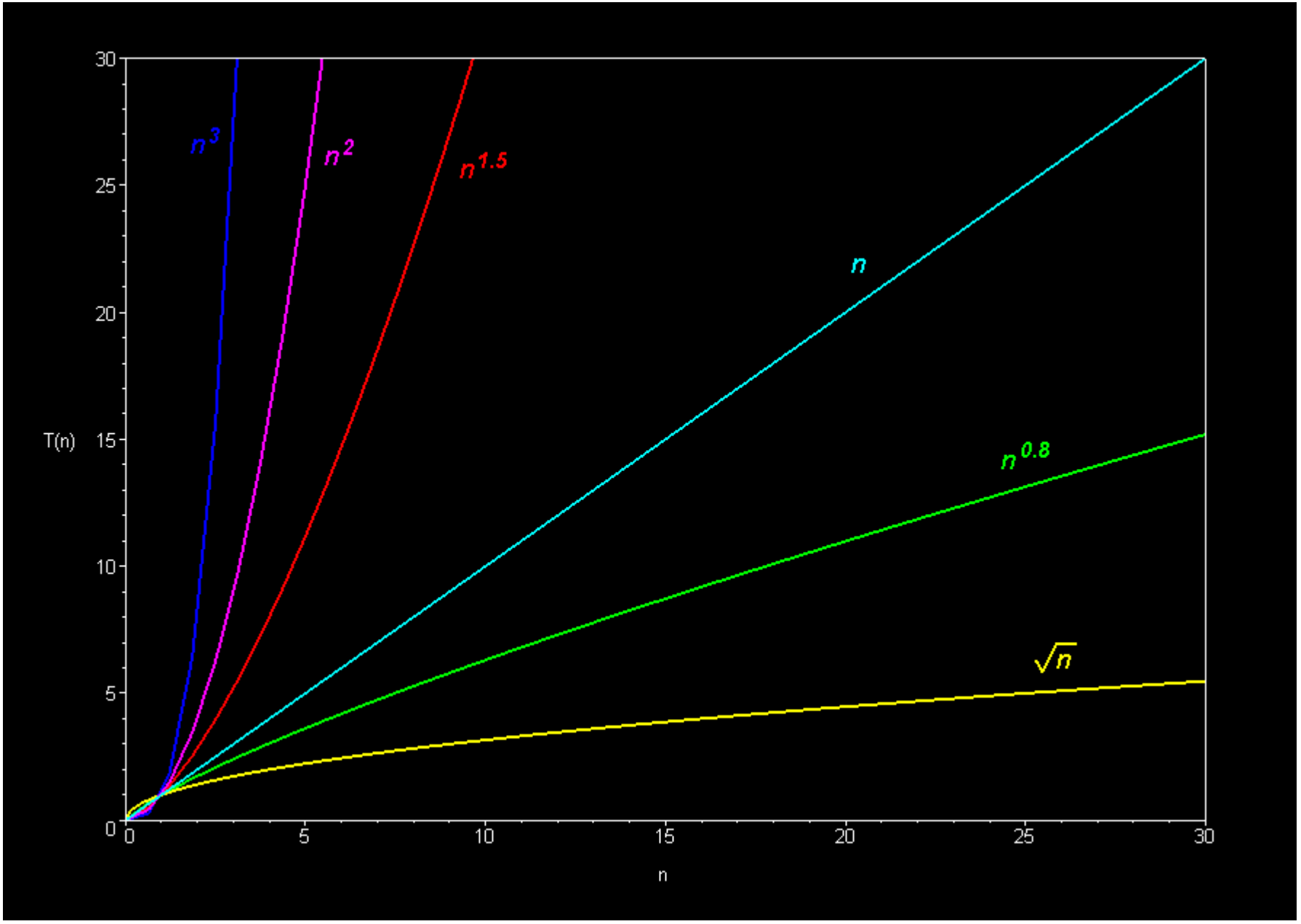

3000
2500
2000
1500
1000
500
0
T(n)
0
5
10
15
20
25
30
n
2^n
$\frac{1}{2}n^3$
$5n^2$
$100n$
$400\sqrt{n}$
$400 \log n$

Comparison of algorithms in terms of the maximum problem size they can handle:

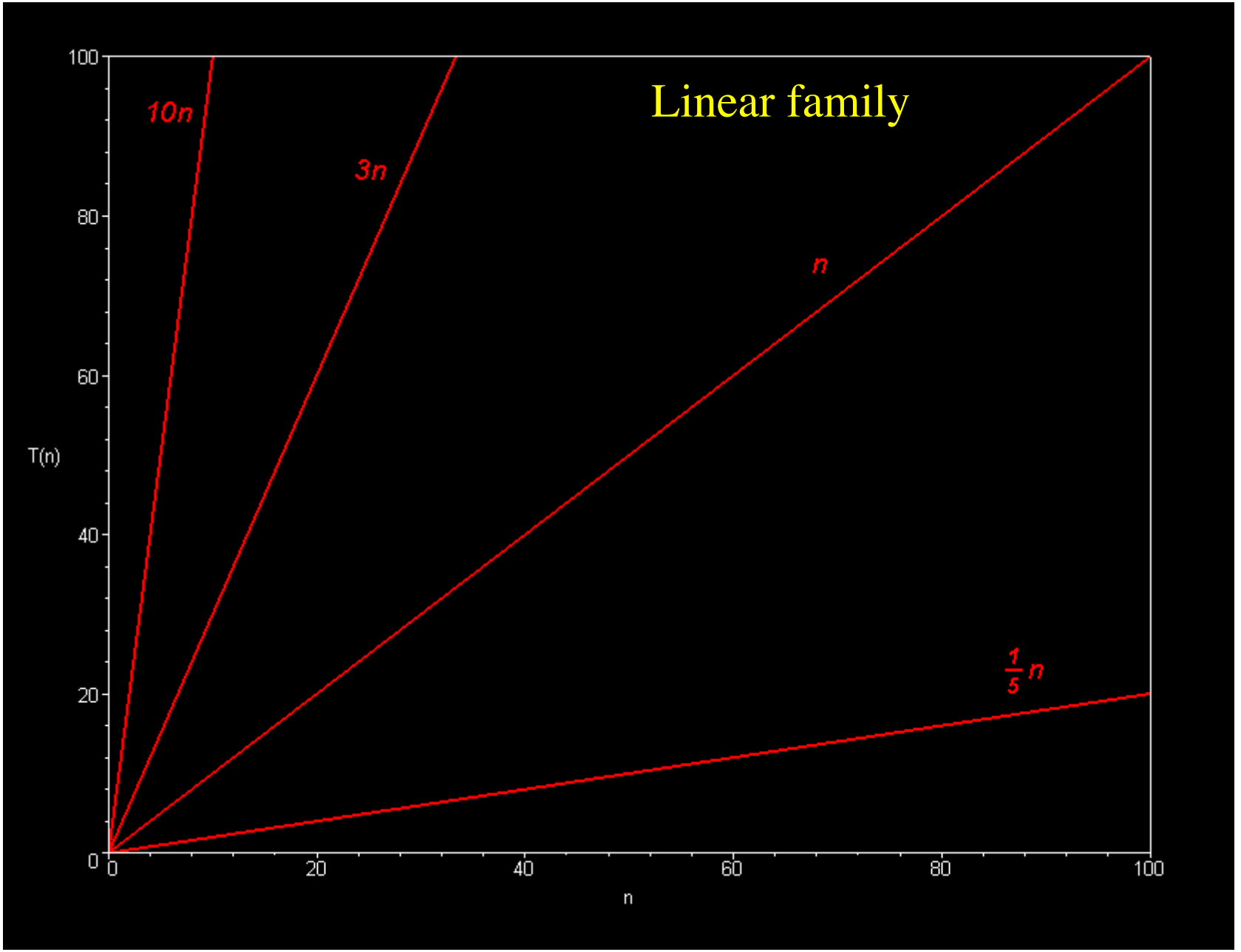
Algorithm Complexity	Running time T(n) (measured in seconds on an Apple Delicious computer)	Maximum problem size given 1000 seconds on an Apple Delicious	Computer Speed x 10 Maximum problem size given 1000 seconds on a Power Delicious (or 10,000 seconds on a classic Delicious)
O(n)	100n	n = 10	n = 100 *(x 10 increase)*
$O(n^2)$	$5n^2$	n = 14	n = 45 *(x 3)*
$O(n^3)$	$\frac{1}{2} n^3$	n = 12	n = 27 *(x 2)*
$O(2^n)$	2^n	n = 10	n = 13 *(x 1.3)*
O(sqrt n)	400 sqrt(n)	n = 6	n = 625 *(x 100)*
O(log n)	400 log(n)	n = 12	n = 72 billion *(x 6 billion)*

MORAL: Cheaper, faster computers mean bigger problems to solve. Bigger problems to solve mean efficiency is *more* important, not less!

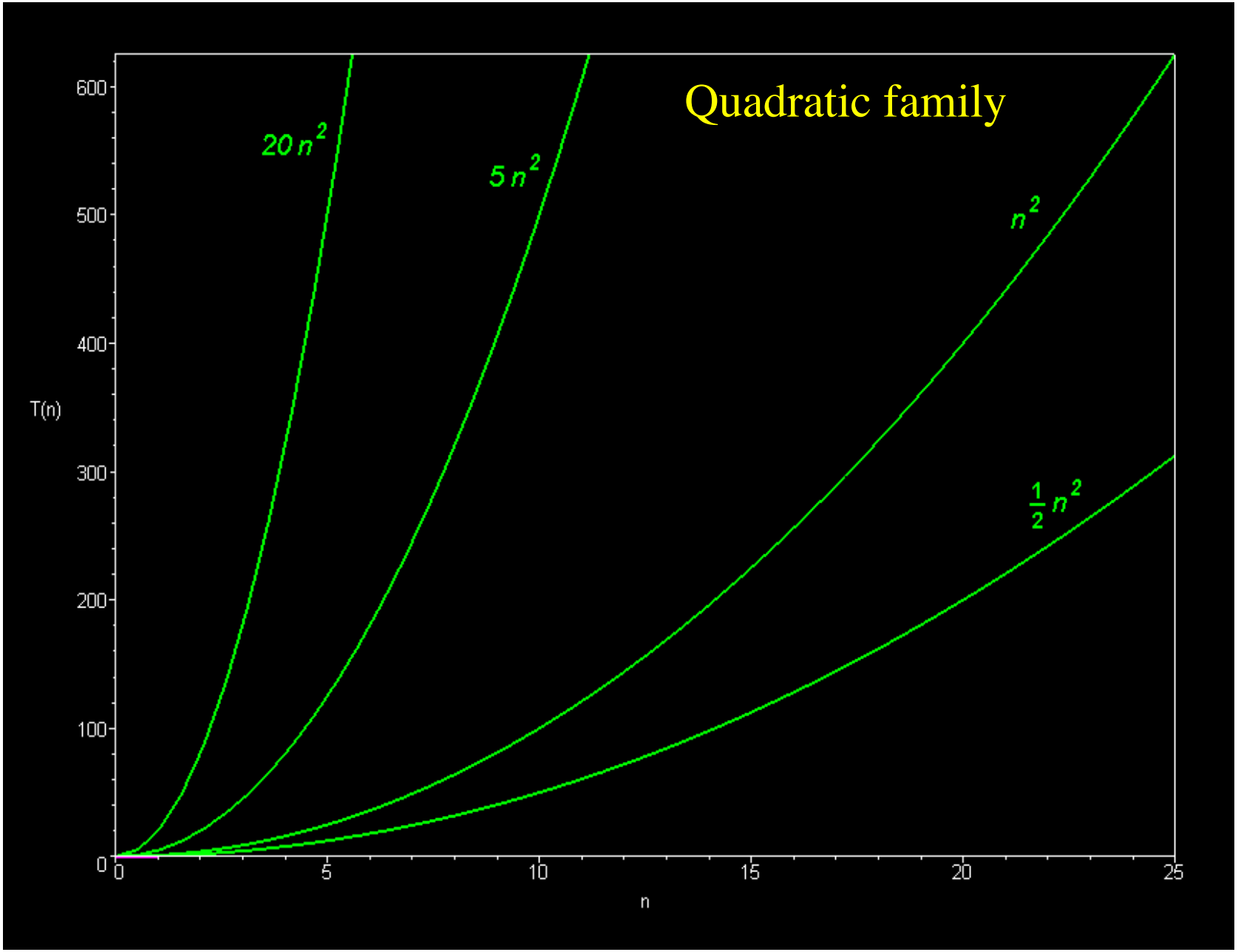
The basic shape of a polynomial function is determined by the highest valued exponent in the polynomial (called the *order* of the polynomial).



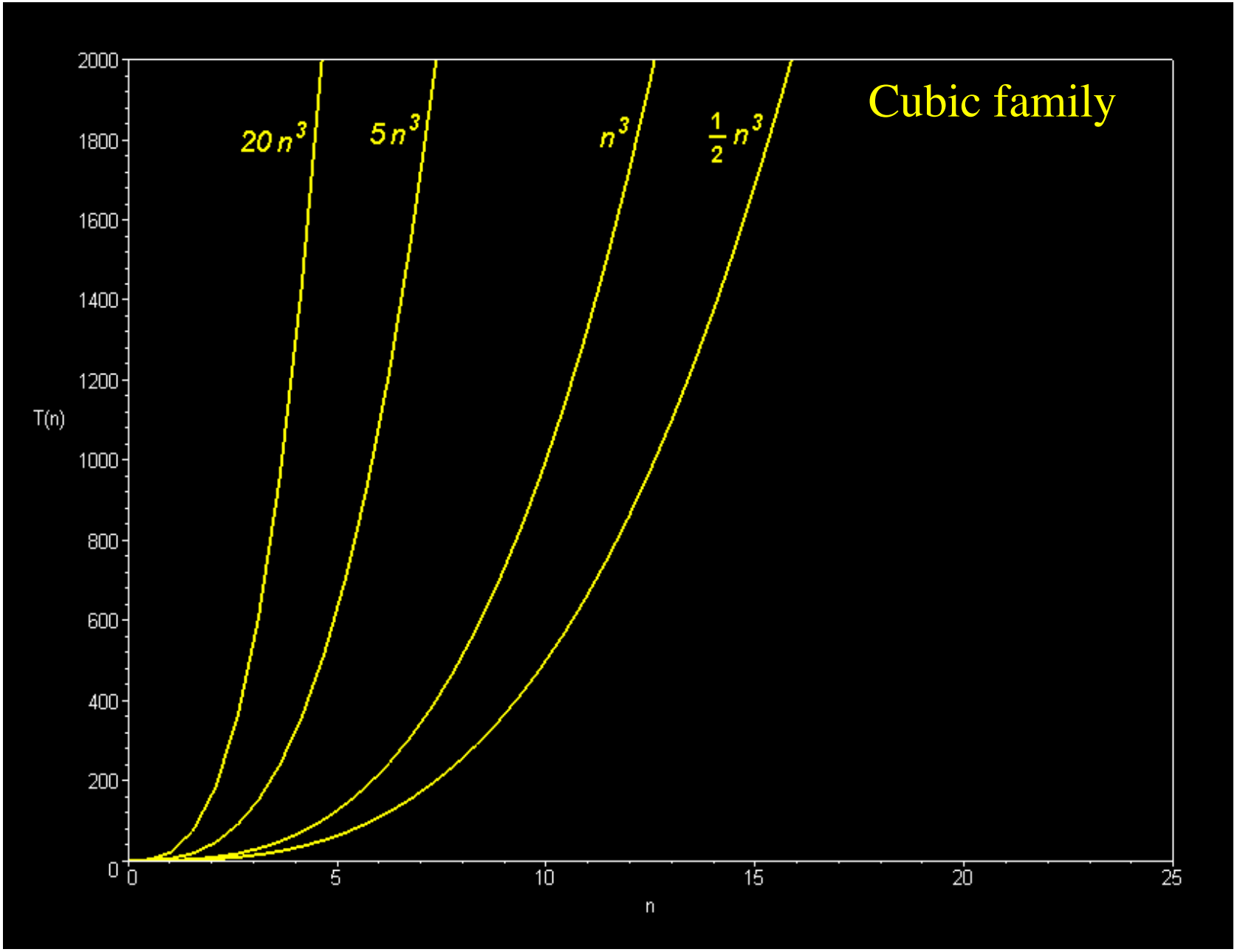
Multiplicative constants do not affect the fundamental shape of a curve. Only the steepness of the curve is affected.



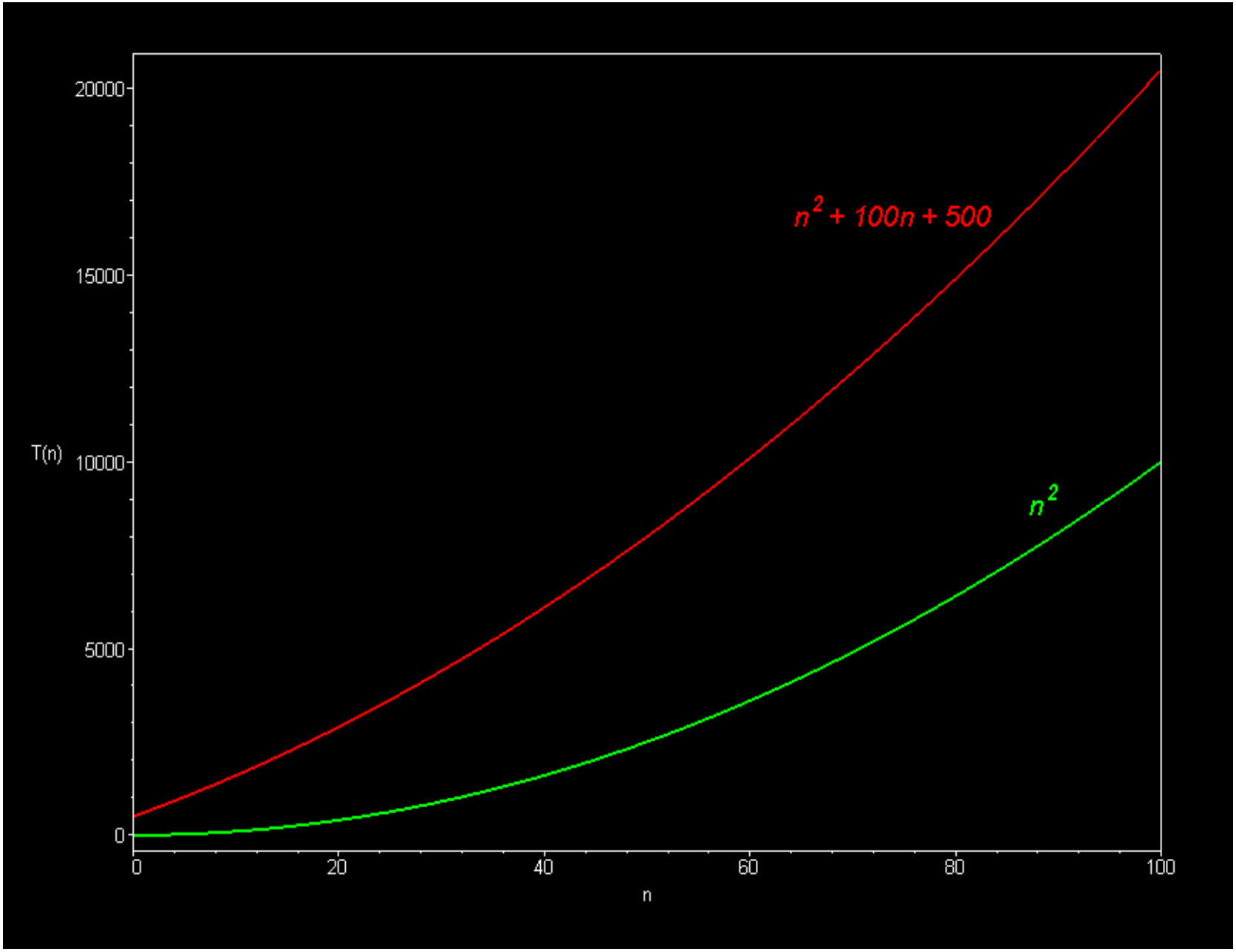
Multiplicative constants do not affect the fundamental shape of a curve. Only the steepness of the curve is affected.



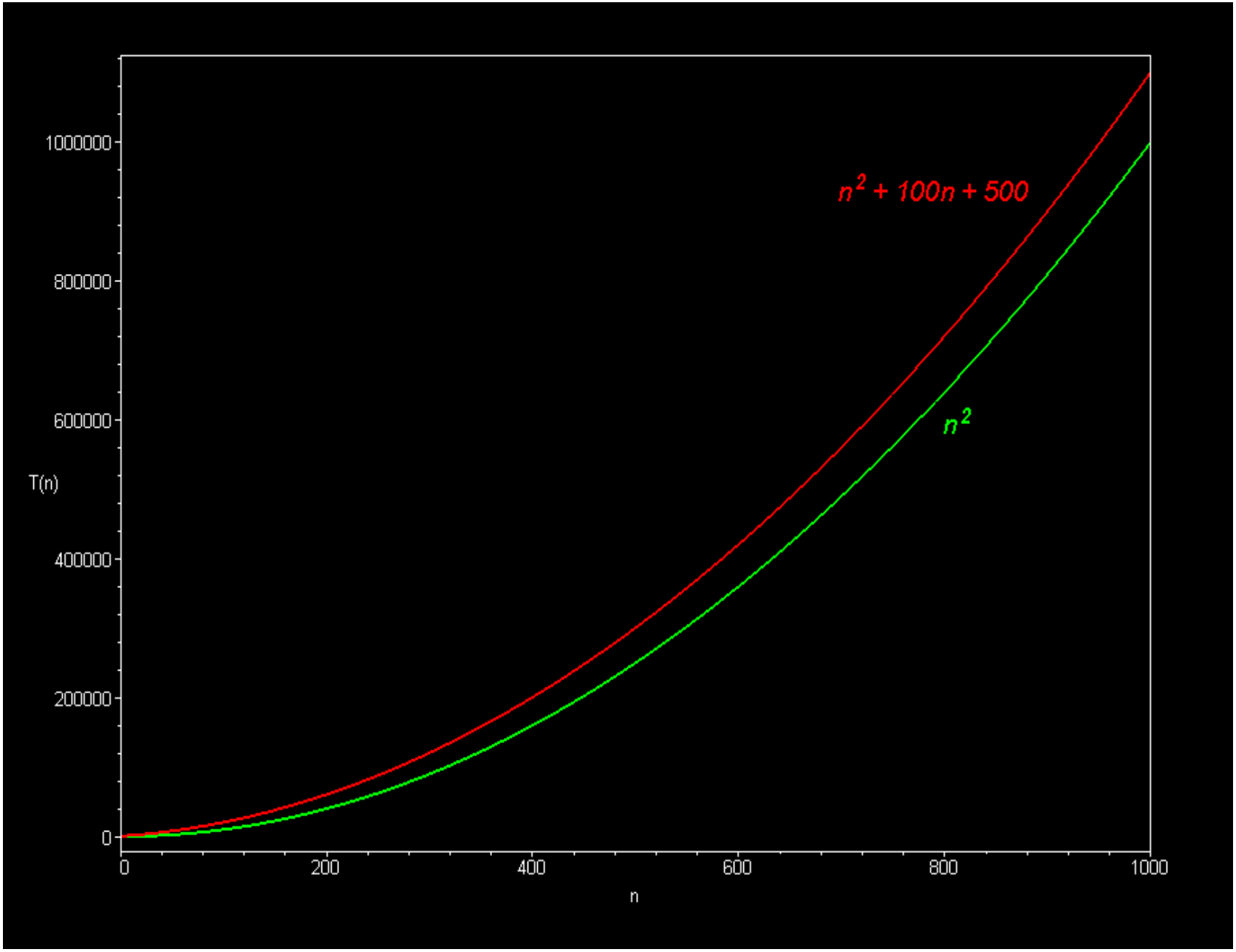
Multiplicative constants do not affect the fundamental shape of a curve. Only the steepness of the curve is affected.



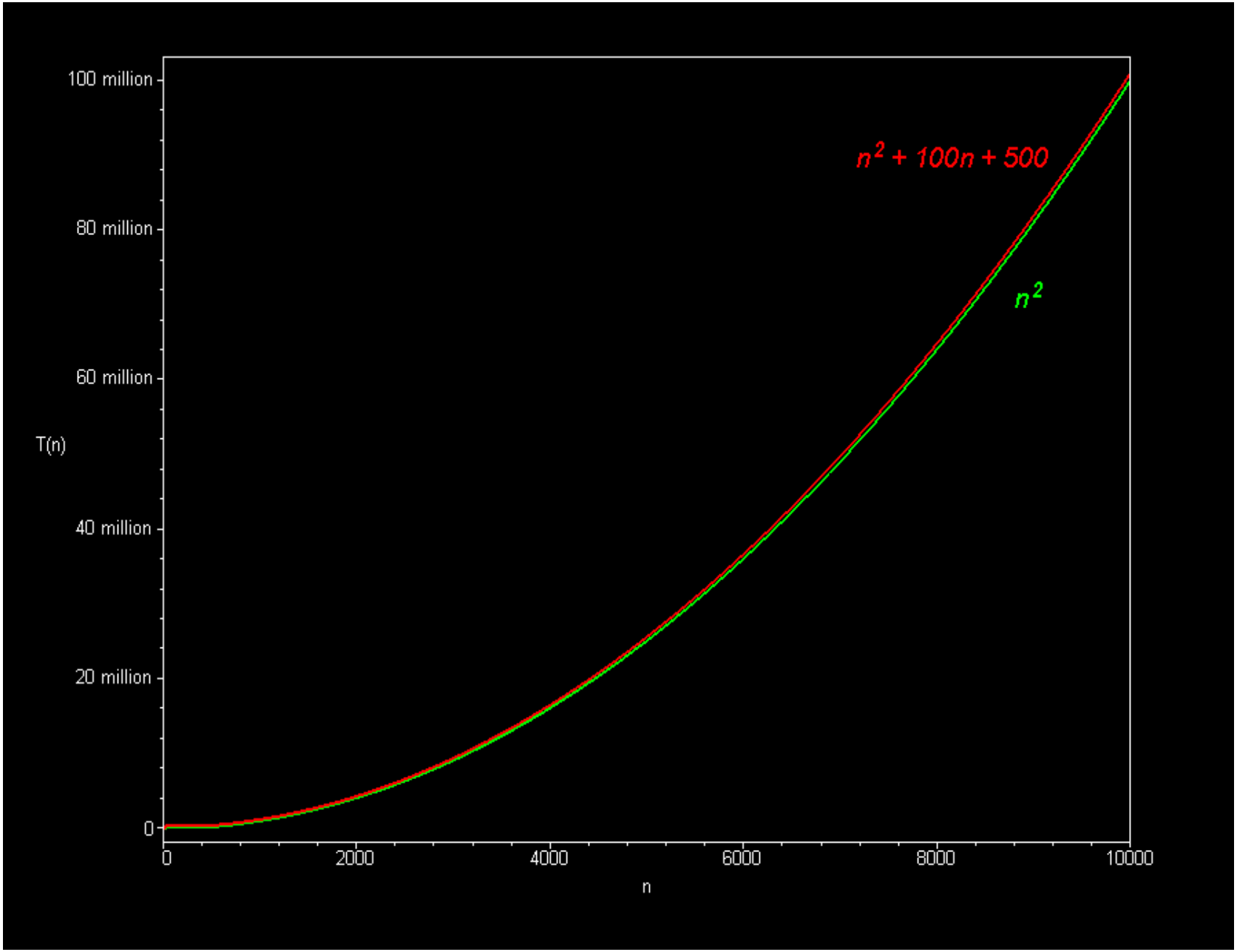
Only the *dominant terms* of a polynomial matter in the long run. Lower-order terms fade to insignificance as the problem size increases.



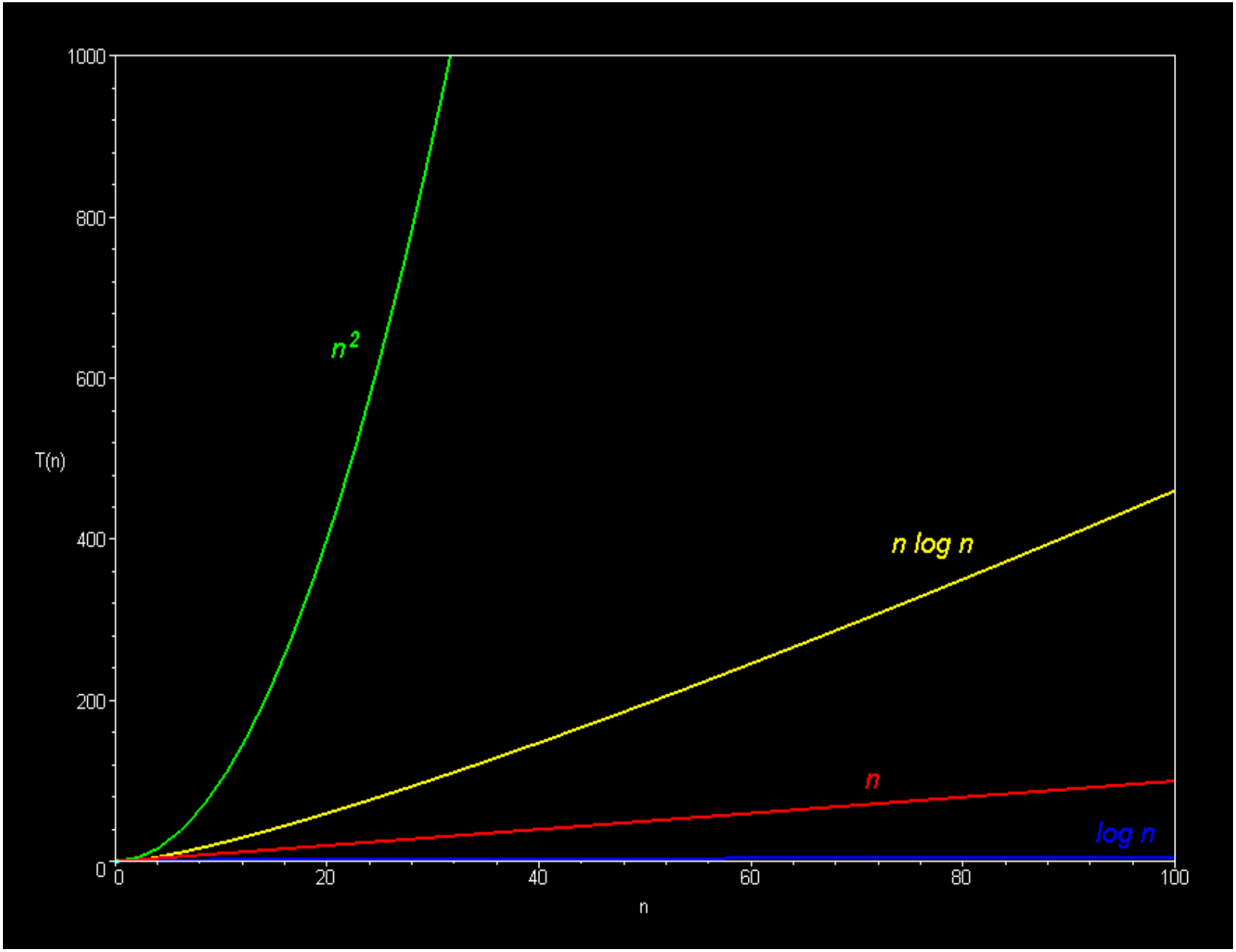
Only the *dominant terms* of a polynomial matter in the long run. Lower-order terms fade to insignificance as the problem size increases.



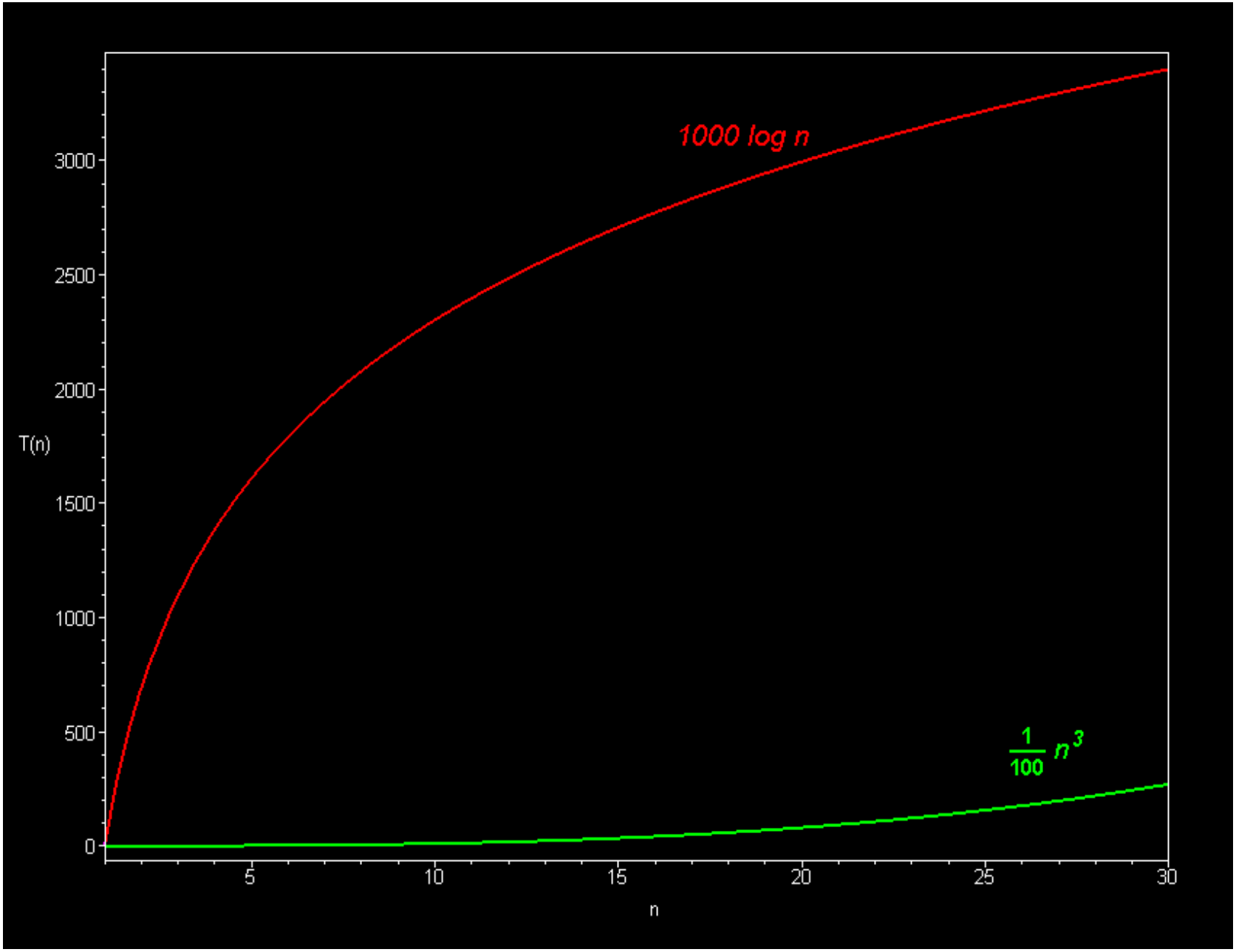
Only the *dominant terms* of a polynomial matter in the long run. Lower-order terms fade to insignificance as the problem size increases.



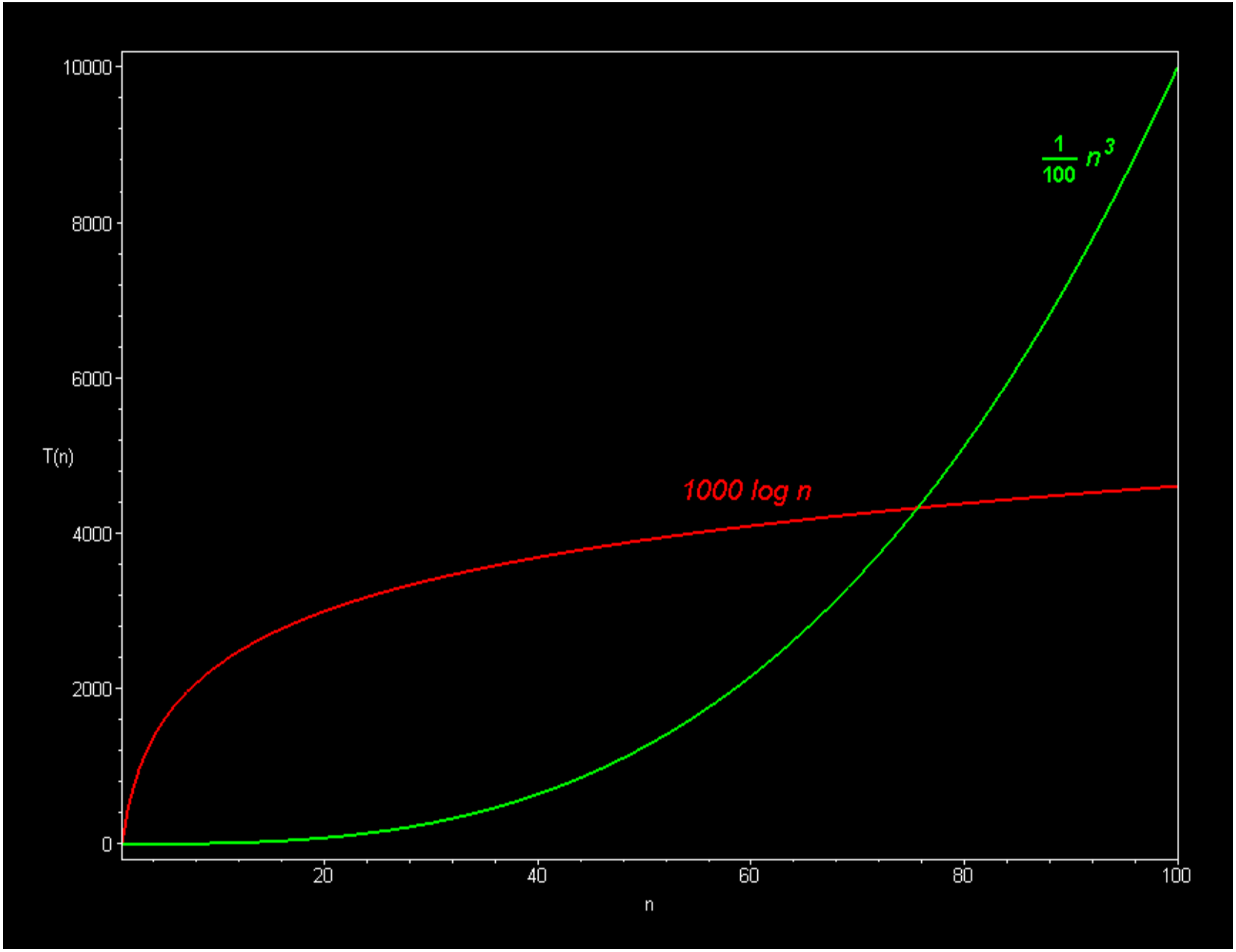
The best sorting algorithms (e.g., mergesort) run in O(n log n) time. Slower ones (bubble sort, selection sort, insertion sort), take O(n^2) time.



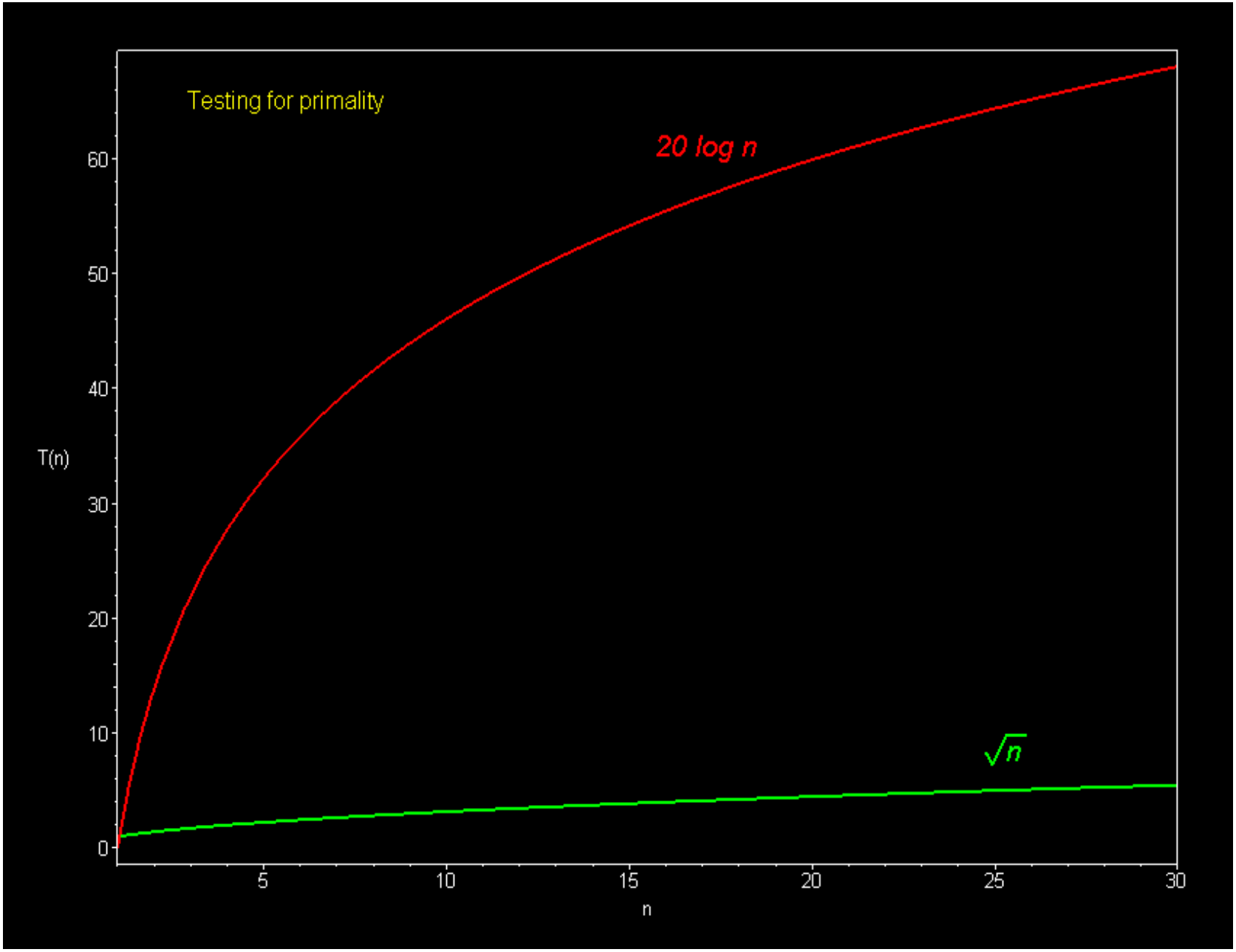
Polynomial curves will always overtake logarithmic curves eventually, when the problem size gets big enough, regardless of the multiplicative constants.



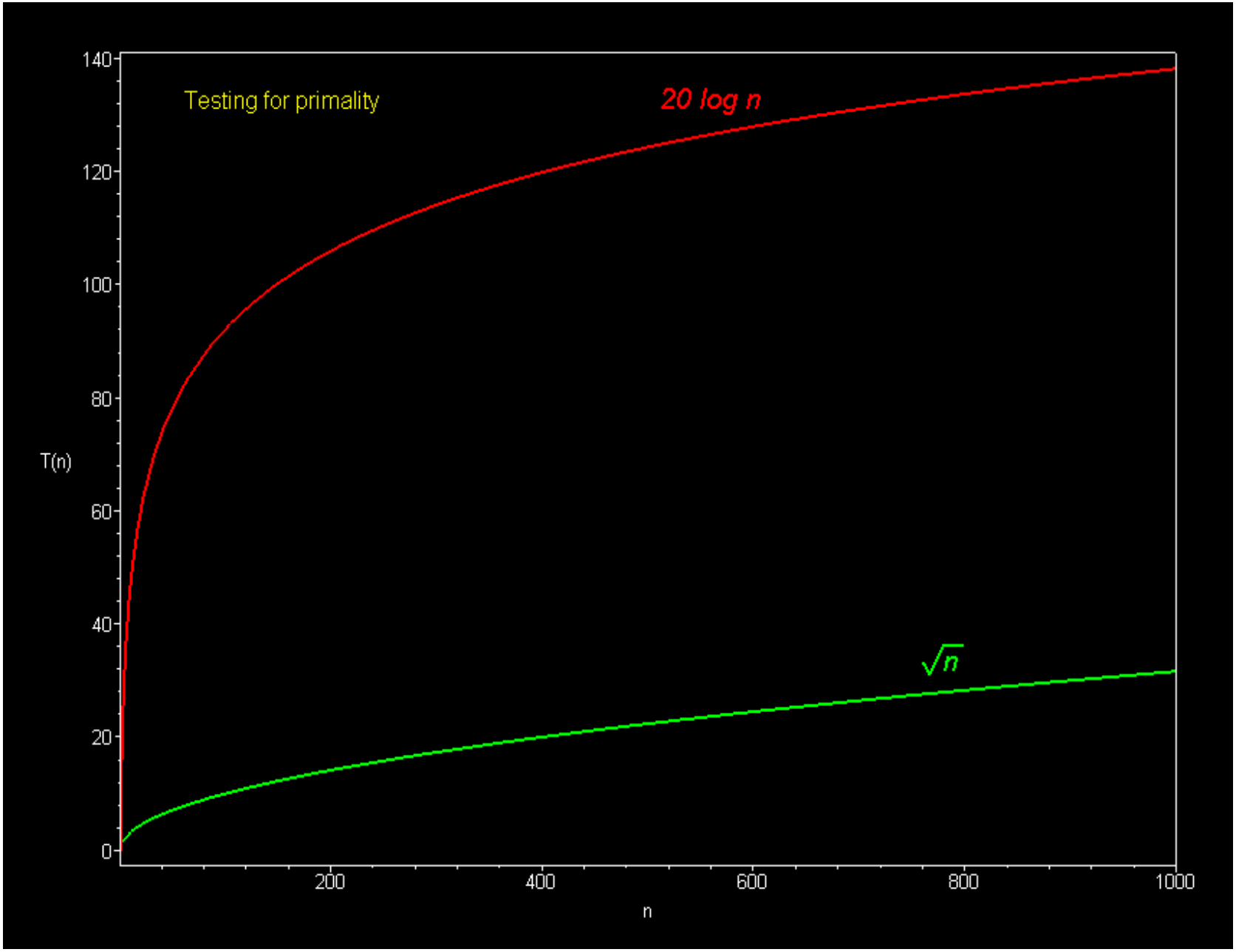
Polynomial curves will always overtake logarithmic curves eventually, when the problem size gets big enough, regardless of the multiplicative constants.



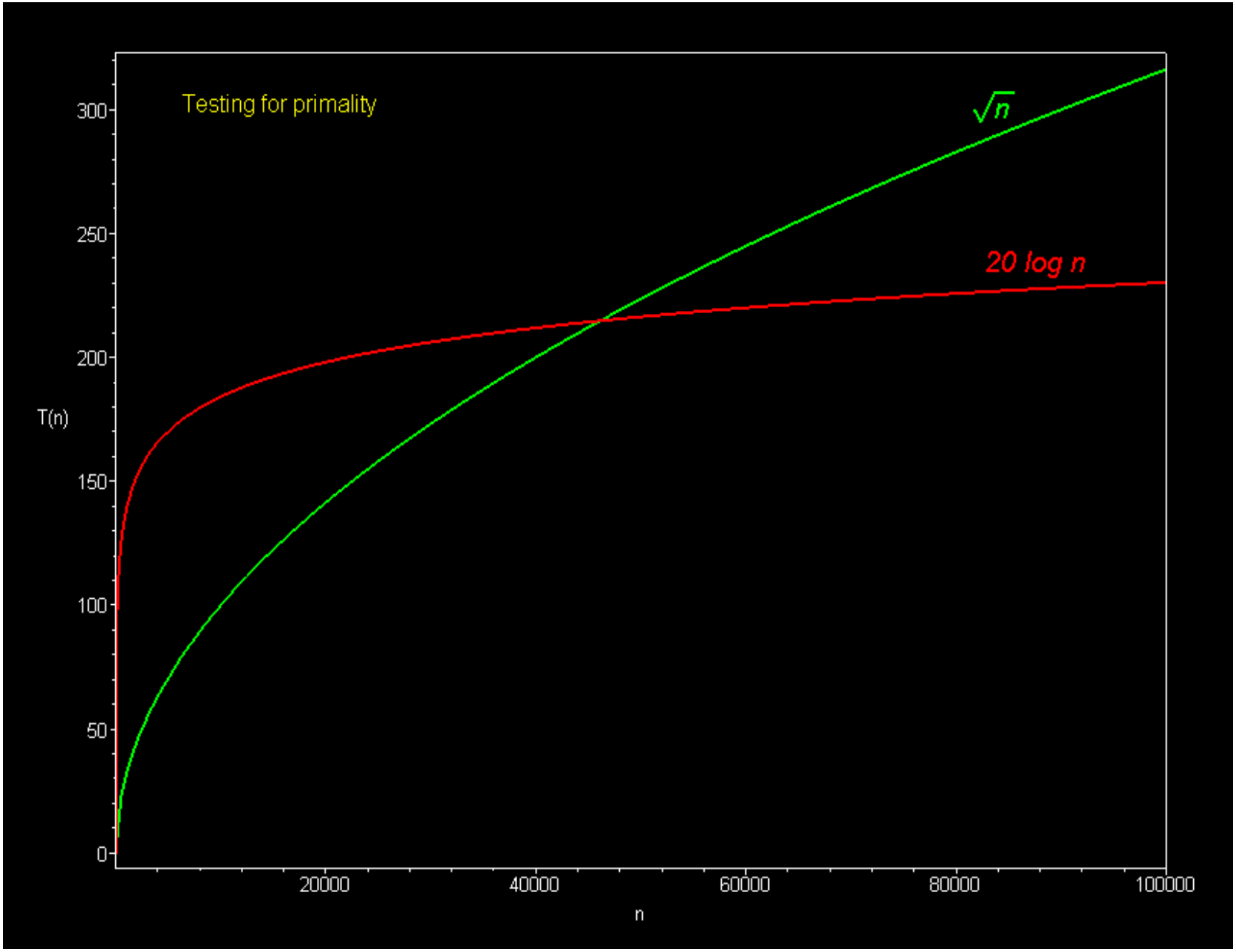
The superiority of the O(log *n*) Fermat prime test over the O(sqrt *n*) prime test becomes clear for really big integers.



The superiority of the O(log *n*) Fermat prime test over the O(sqrt *n*) prime test becomes clear for really big integers.



The superiority of the O(log *n*) Fermat prime test over the O(sqrt *n*) prime test becomes clear for really big integers.



The superiority of the O(log *n*) Fermat prime test over the O(sqrt *n*) prime test becomes clear for really big integers.