# CS-218 DATA STRUCTURES AND ALGORITHMS

LECTURE 4

BY UROOJ AINUDDIN

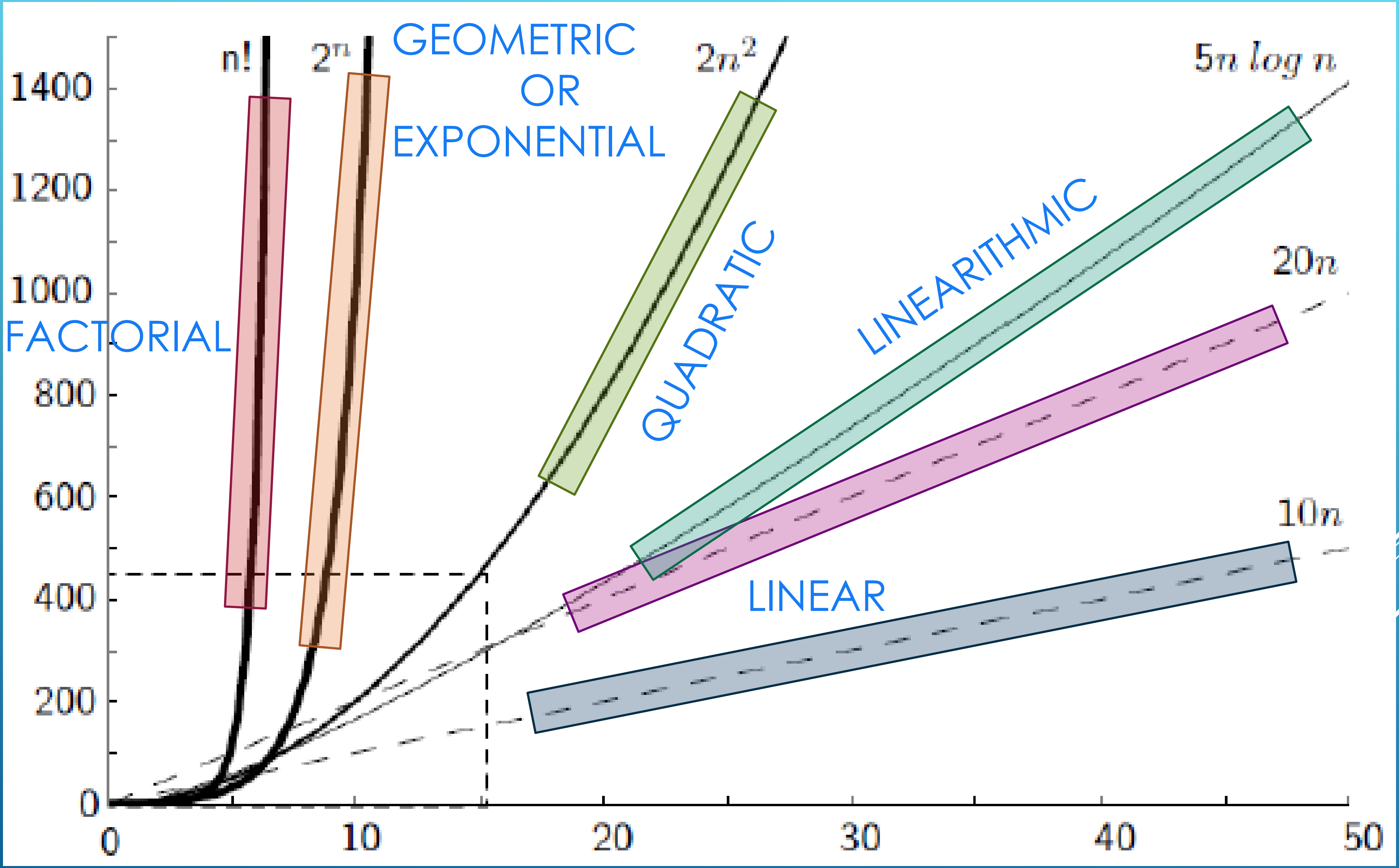
## IN THE LAST LECTURE…

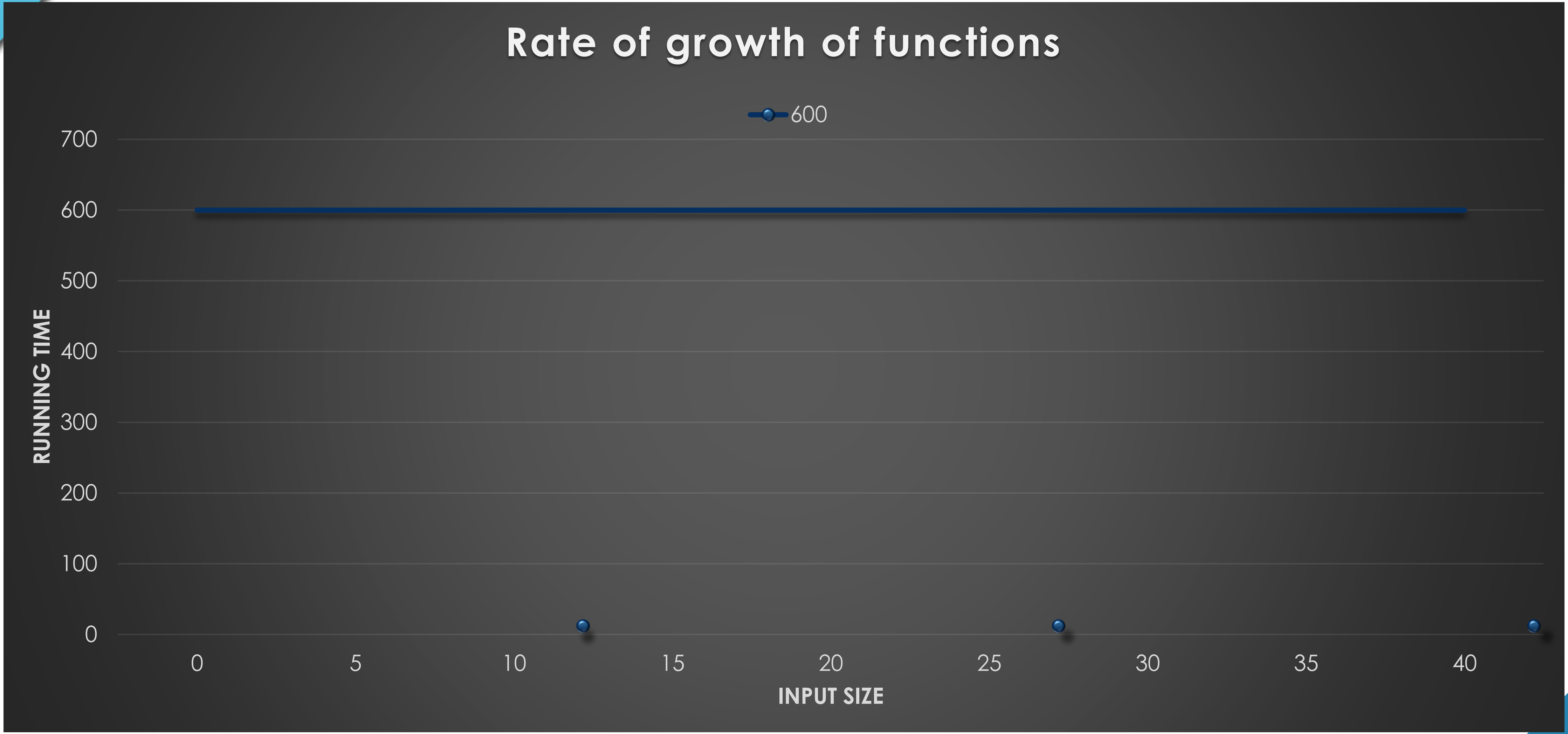
- We analyzed some more algorithms.
- We realized that there is no need to find absolute running time of an algorithm.
- We realized that we analyze algorithms typically for large input sizes.
- We were introduced to asymptotic analysis.
- We conducted asymptotic analysis of algorithms.

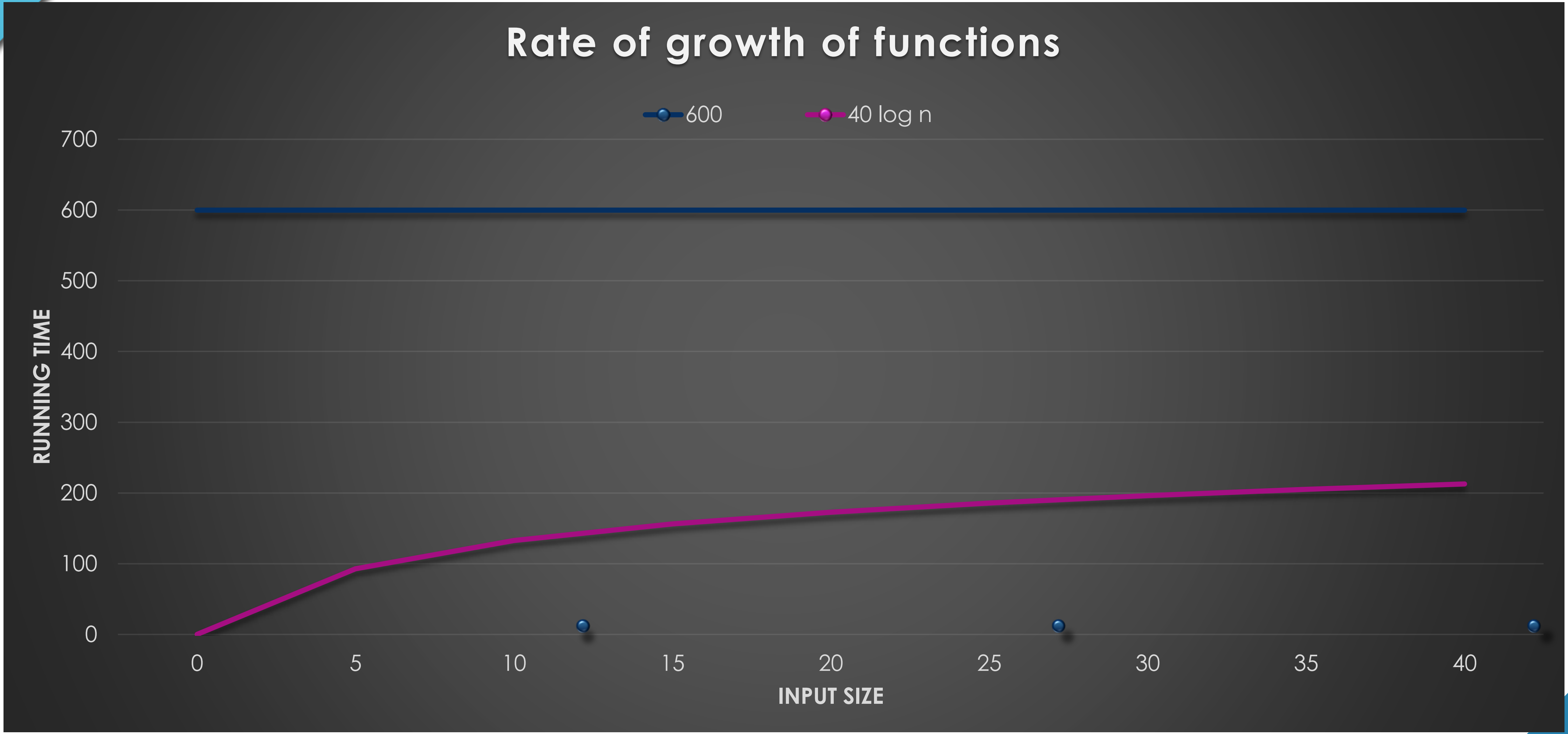
# ANALYSIS OF ALGORITHMS (ASYMPTOTIC ANALYSIS)

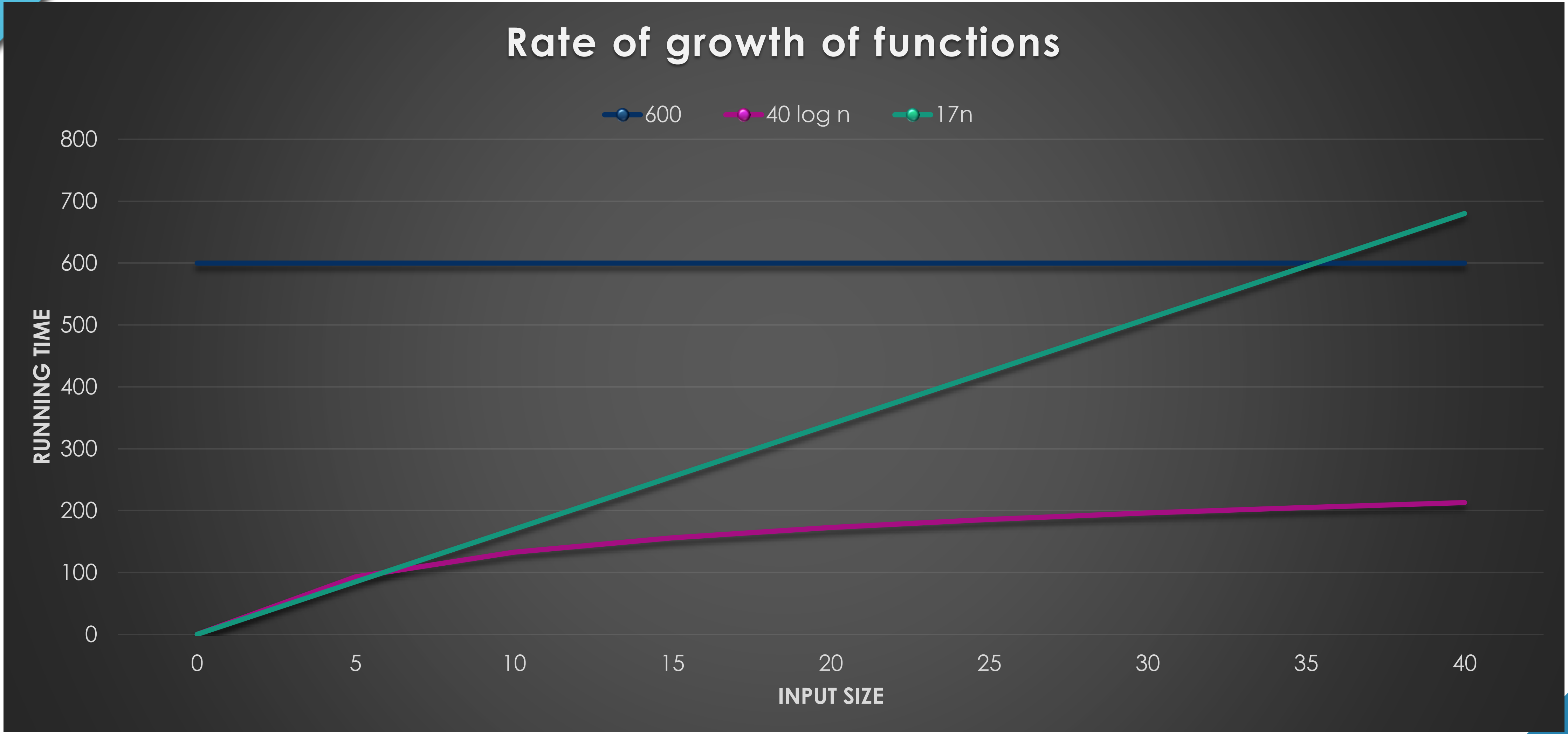
BOOK 1 CHAPTER 4

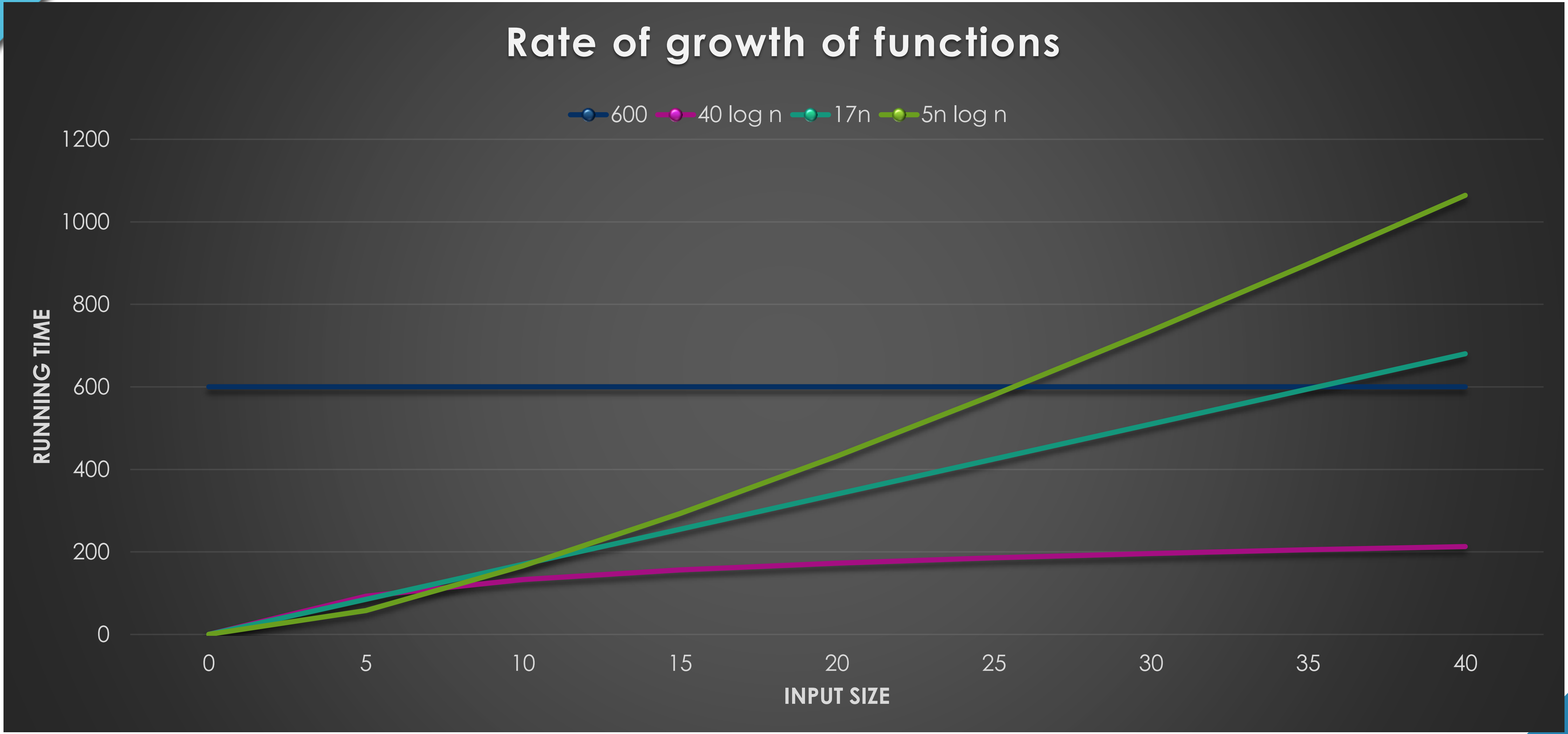
BOOK 2 CHAPTER 2

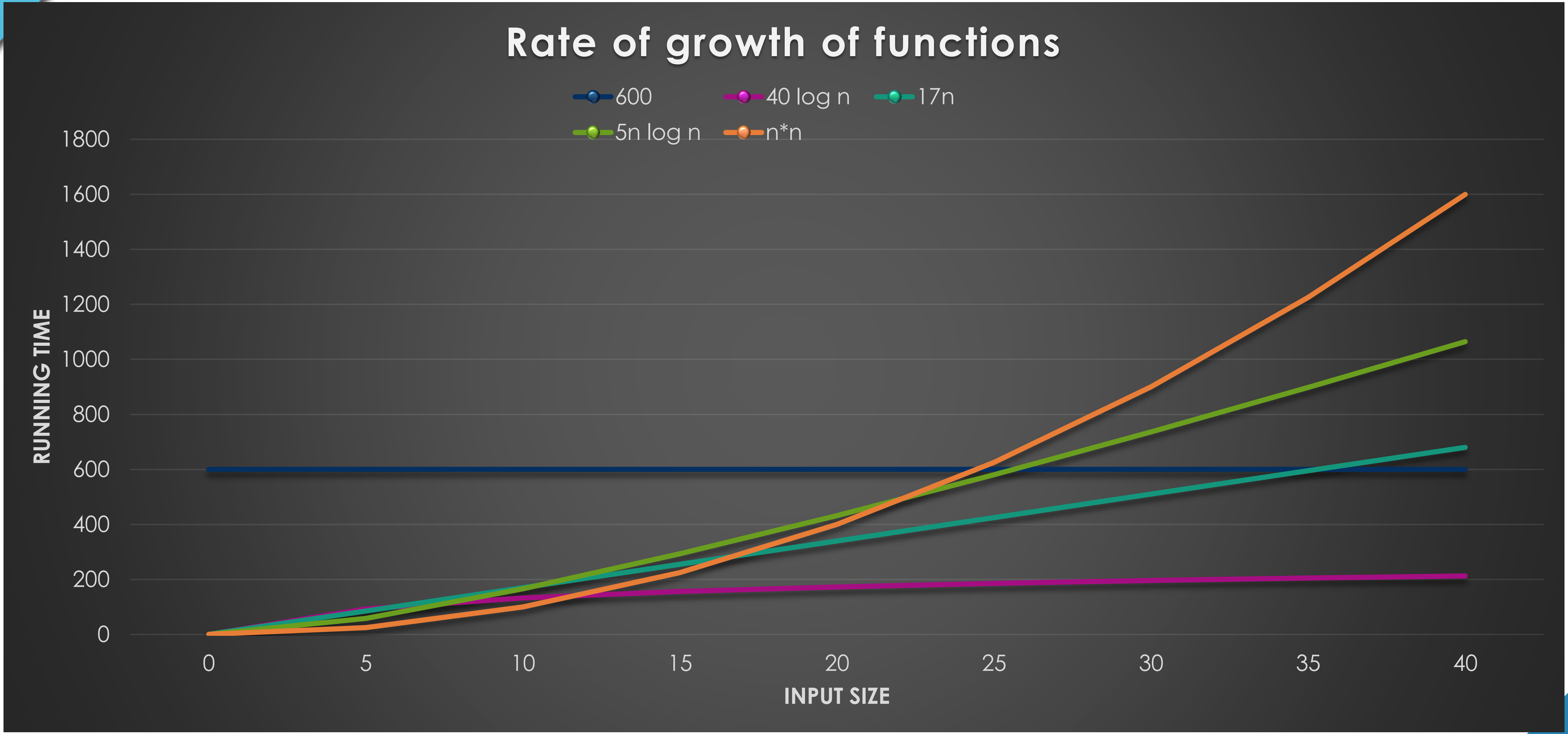

n!
$2^n$
GEOMETRIC OR EXPONENTIAL
$2n^2$
$5n \log n$
QUADRATIC
LINEARITHMIC
$20n$
$10n$
LINEAR
FACTORIAL
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50


Rate of growth of functions
600
RUNNING TIME
700
600
500
400
300
200
100
0
0
5
10
15
20
25
30
35
40
INPUT SIZE


Rate of growth of functions
600
40 log n
700
600
500
400
300
200
100
0
RUNNING TIME
0
5
10
15
20
25
30
35
40
INPUT SIZE


Rate of growth of functions
600
40 log n
17n
RUNNING TIME
800
700
600
500
400
300
200
100
0
0
5
10
15
20
25
30
35
40
INPUT SIZE


Rate of growth of functions
600
40 log n
17n
5n log n
1200
1000
800
600
400
200
0
RUNNING TIME
0
5
10
15
20
25
30
35
40
INPUT SIZE



Rate of growth of functions
600
40 log n
17n
5n log n
n*n
RUNNING TIME
1800
1600
1400
1200
1000
800
600
400
200
0
0
5
10
15
20
25
30
35
40
INPUT SIZE

# FUNCTIONS ARRANGED WITH RESPECT TO RATE OF GROWTH

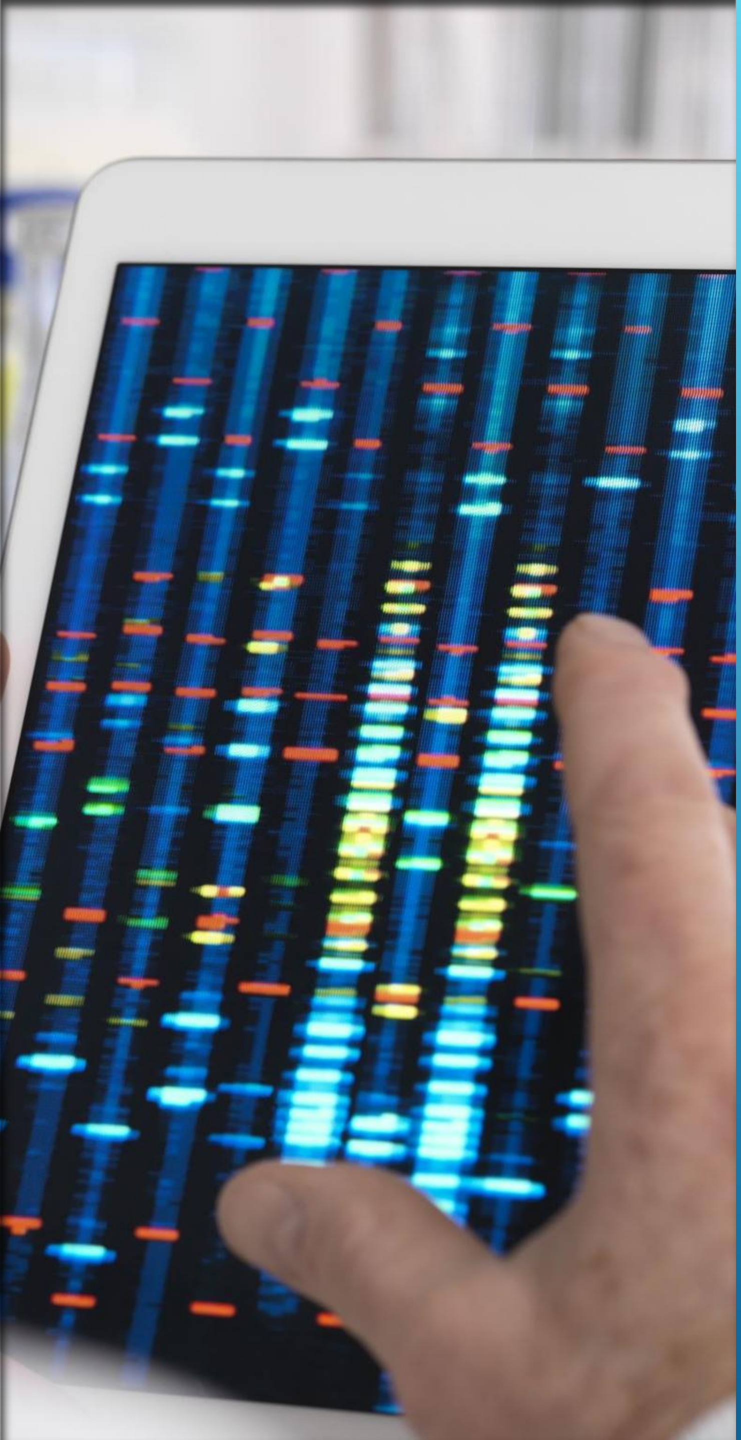
$$n^n = n(n)(n)\ldots(n)(n)$$
$$n! = n(n-1)(n-2)\ldots(2)(1)$$

RATE OF GROWTH

| | |
|---|---|
| $c$ | constant function |
| $\log\log n$ | |
| $\log n$ | logarithmic function |
| $log^a\, n, a > 1, a \in \mathbb{R}$ | polylogarithmic function |
| $n$ | linear function |
| $n \log n$ | linearithmic function |
| $n^a, a > 1, a \in \mathbb{Z}$ | polynomial function |
| $a^n, a > 1, a \in \mathbb{R}$ | geometric/exponential function |
| $n!$ | factorial function |
| $n^n$ | |

## FORGET THE ALGORITHM, GET A FASTER COMPUTER!

- Some people may argue that analyzing algorithms takes effort and a relatively easier solution is to upgrade the machine that you have. This will automatically speed up execution of the algorithm.
- We want to understand if this is a good approach.
- Suppose we have the budget to get a faster machine.
- Let us see if a faster computer will speed up our algorithms.



- Our current machine executes *n* data items per hour.
- Suppose we purchase a new computer that is 10 times faster than the one we have.
- Assume that the new machine executes *m* data items per hour.
- We are hoping that the new machine, being faster, will be able to execute our algorithm on more than *n* data items in an hour, i.e. $m > n$.

$$\frac{time\ expended\ on\ larger\ data\ set}{time\ expended\ on\ smaller\ data\ set} = Speed\ up$$

$$\frac{T(m)}{T(n)} = 10$$

## MOVING TO A FASTER MACHINE

- $T(n) = 2\log n$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{2\log m}{2\log n} = 10$
- $\log m = 10\log n$
- $\log m = \log n^{10}$
- $2^{\log m} = 2^{\log n^{10}}$
- $m = n^{10}$

- $T(n) = 5\log n$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{5\log m}{5\log n} = 10$
- $\log m = 10\log n$
- $\log m = \log n^{10}$
- $2^{\log m} = 2^{\log n^{10}}$
- $m = n^{10}$

| T(n) | m | If n=1000, m= |
|---|---|---|
| $2\log n$ | $n^{10}$ | $1000^{10}$ |
| $5\log n$ | $n^{10}$ | $1000^{10}$ |

## ALGORITHMS OF LOGARITHMIC GROWTH RATE

- $T(n) = 10n$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{10m}{10n} = 10$
- $m = 10n$

- $T(n) = 50n$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{50m}{50n} = 10$
- $m = 10n$

| T(n) | m | If n=1000, m= |
|---|---|---|
| $2\log n$ | $n^{10}$ | $1000^{10}$ |
| $5\log n$ | $n^{10}$ | $1000^{10}$ |
| $10n$ | $10n$ | $10000$ |
| $50n$ | $10n$ | $10000$ |

## ALGORITHMS OF LINEAR GROWTH RATE

- $T(n) = 20n^2$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{20m^2}{20n^2} = 10$
- $m^2 = 10n^2$
- $m = \sqrt{10}n$

- $T(n) = 75n^2$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{75m^2}{75n^2} = 10$
- $m^2 = 10n^2$
- $m = \sqrt{10}n$

| T(n) | m | If n=1000, m= |
|---|---|---|
| $2\log n$ | $n^{10}$ | $1000^{10}$ |
| $5\log n$ | $n^{10}$ | $1000^{10}$ |
| $10n$ | $10n$ | $10000$ |
| $50n$ | $10n$ | $10000$ |
| $20n^2$ | $\sqrt{10}n$ | $3162$ |
| $75n^2$ | $\sqrt{10}n$ | $3162$ |

## ALGORITHMS OF QUADRATIC GROWTH RATE

- $T(n) = 3n^3$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{3m^3}{3n^3} = 10$
- $m^3 = 10n^3$
- $m = \sqrt[3]{10n}$

- $T(n) = 8n^3$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{8m^3}{8n^3} = 10$
- $m^3 = 10n^3$
- $m = \sqrt[3]{10n}$

| T(n) | m | If n=1000, m= |
|---|---|---|
| $2 \log n$ | $n^{10}$ | $1000^{10}$ |
| $5 \log n$ | $n^{10}$ | $1000^{10}$ |
| $10n$ | $10n$ | $10000$ |
| $50n$ | $10n$ | $10000$ |
| $20n^2$ | $\sqrt{10n}$ | $3162$ |
| $75n^2$ | $\sqrt{10n}$ | $3162$ |
| $3n^3$ | $\sqrt[3]{10n}$ | $2154$ |
| $8n^3$ | $\sqrt[3]{10n}$ | $2154$ |

# ALGORITHMS OF CUBIC GROWTH RATE

# ALGORITHMS OF GEOMETRIC GROWTH RATE

- $T(n) = 2^n$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{2^m}{2^n} = 10$
- $2^m = 10(2^n)$
- $\log_2 2^m = \log_2 10(2^n)$
- $m = \log_2 10 + \log_2 2^n$
- $m = n + \log_2 10$

- $T(n) = 5^n$
- $\frac{T(m)}{T(n)} = 10$
- $\frac{5^m}{5^n} = 10$
- $5^m = 10(5^n)$
- $\log_5 5^m = \log_5 10(5^n)$
- $m = \log_5 10 + \log_5 5^n$
- $m = n + \log_5 10$

| T(n) | m | If n=1000, m= |
|---|---|---|
| $2 \log n$ | $n^{10}$ | $1000^{10}$ |
| $5 \log n$ | $n^{10}$ | $1000^{10}$ |
| $10n$ | $10n$ | $10000$ |
| $50n$ | $10n$ | $10000$ |
| $20n^2$ | $\sqrt{10}n$ | $3162$ |
| $75n^2$ | $\sqrt{10}n$ | $3162$ |
| $3n^3$ | $\sqrt[3]{10}n$ | $2154$ |
| $8n^3$ | $\sqrt[3]{10}n$ | $2154$ |
| $2^n$ | $n + \log_2 10$ | $1003$ |
| $5^n$ | $n + \log_5 10$ | $1001$ |

## POINTS TO NOTE

- **The value of the coefficient does not affect the improvement in problem size gained by a faster computer.**
- The improvement is data size only depends on the growth rate of the function describing the running time of the algorithm.

| T(n) | m | If n=1000, m= |
|---|---|---|
| $2\log n$ | $n^{10}$ | $1000^{10}$ |
| $5\log n$ | $n^{10}$ | $1000^{10}$ |
| $10n$ | $10n$ | $10000$ |
| $50n$ | $10n$ | $10000$ |
| $20n^2$ | $\sqrt{10}n$ | $3162$ |
| $75n^2$ | $\sqrt{10}n$ | $3162$ |
| $3n^3$ | $\sqrt[3]{10}n$ | $2154$ |
| $8n^3$ | $\sqrt[3]{10}n$ | $2154$ |
| $2^n$ | $n + \log_2 10$ | $1003$ |
| $5^n$ | $n + \log_5 10$ | $1001$ |

## POINTS TO NOTE

- **The increase in problem size for an algorithm with exponential growth rate is by a constant addition, not by a multiplicative factor.**
- Exponential growth rate is radically different than polynomial growth rates.

| T(n) | m | If n=1000, m= |
|---|---|---|
| $2\log n$ | $n^{10}$ | $1000^{10}$ |
| $5\log n$ | $n^{10}$ | $1000^{10}$ |
| $10n$ | $10n$ | $10000$ |
| $50n$ | $10n$ | $10000$ |
| $20n^2$ | $\sqrt{10}n$ | $3162$ |
| $75n^2$ | $\sqrt{10}n$ | $3162$ |
| $3n^3$ | $\sqrt[3]{10}n$ | $2154$ |
| $8n^3$ | $\sqrt[3]{10}n$ | $2154$ |
| $2^n$ | $n + \log_2 10$ | $1003$ |
| $5^n$ | $n + \log_5 10$ | $1001$ |

## POINTS TO NOTE

- **The improvement in problem size for an algorithm with slower growth rate is much larger than that for a faster growth rate.**
- A bad algorithm will not let you take advantage of a fast machine, no matter how fast the computer is.

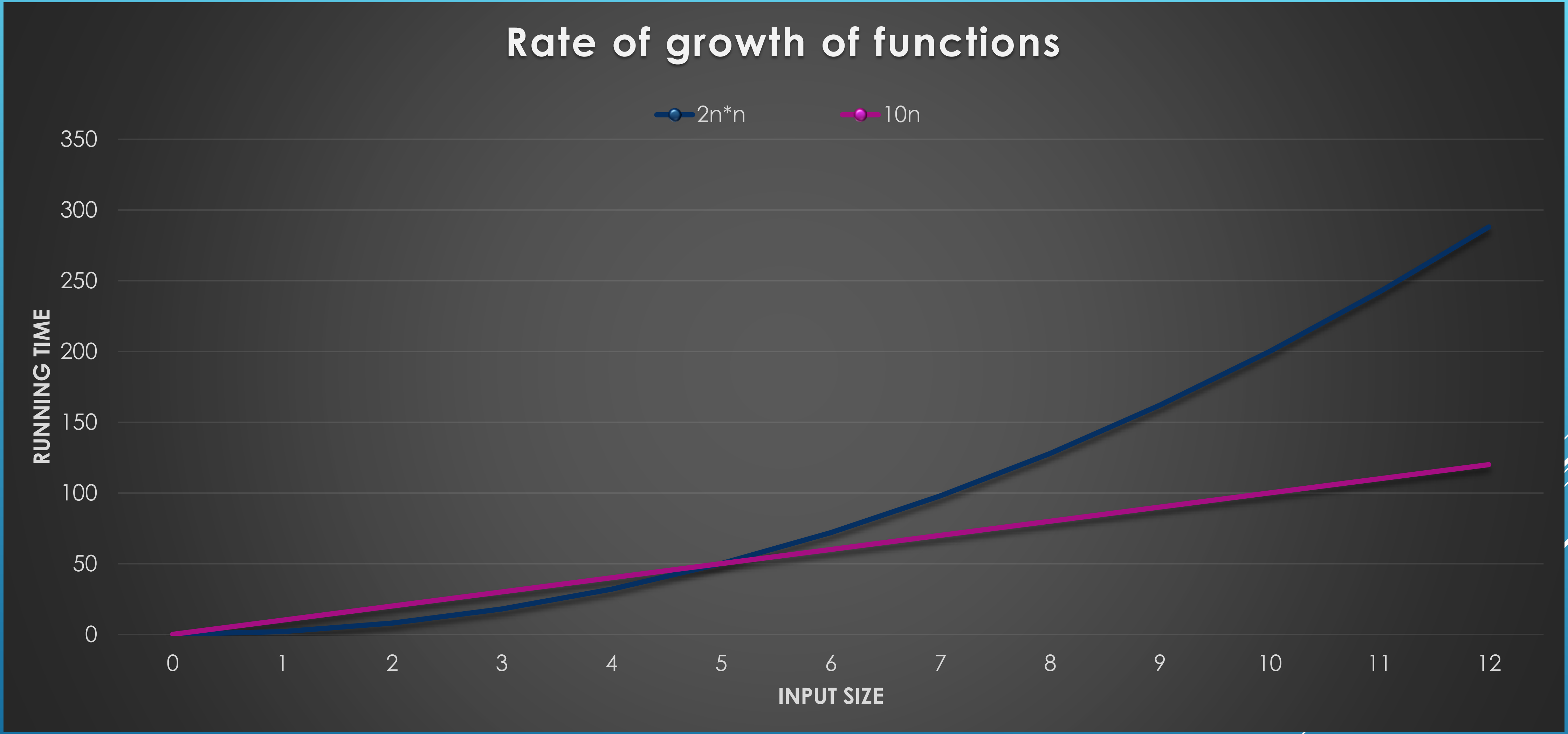
| T(n) | m | If n=1000, m= |
|---|---|---|
| $2\log n$ | $n^{10}$ | $1000^{10}$ |
| $5\log n$ | $n^{10}$ | $1000^{10}$ |
| $10n$ | $10n$ | $10000$ |
| $50n$ | $10n$ | $10000$ |
| $20n^2$ | $\sqrt{10}n$ | $3162$ |
| $75n^2$ | $\sqrt{10}n$ | $3162$ |
| $3n^3$ | $\sqrt[3]{10}n$ | $2154$ |
| $8n^3$ | $\sqrt[3]{10}n$ | $2154$ |
| $2^n$ | $n + \log_2 10$ | $1003$ |
| $5^n$ | $n + \log_5 10$ | $1001$ |

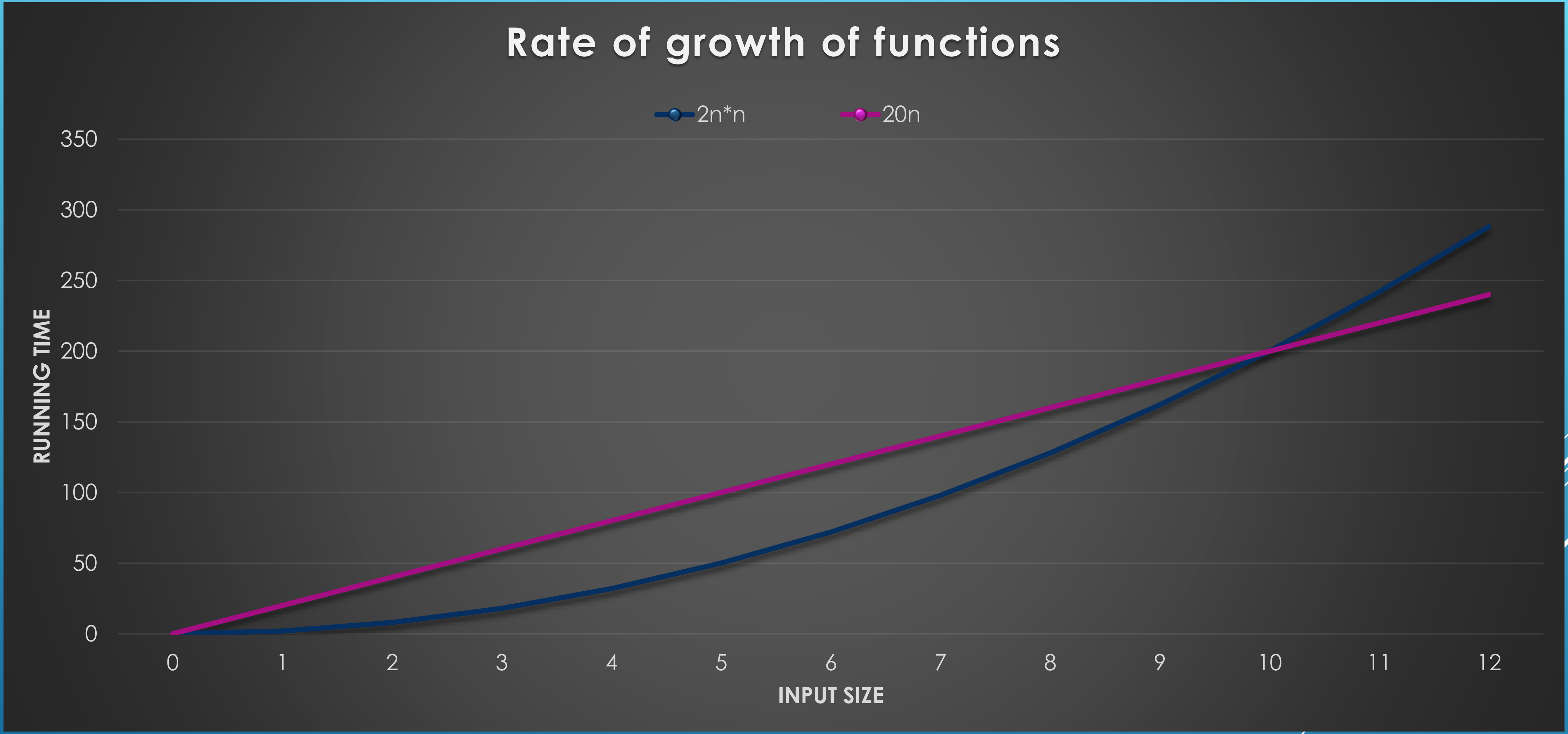
- The current algorithm *A* is a quadratic algorithm, $T_A(n) = n^2$.
- For $n = 2048$, $T_A = 4194304$.
- A friend proposes another algorithm *B*, which is linearithmic, $T_B(n) = n \log n$.
- For $n = 2048$, $T_B = 22528$.
- $\frac{4194304}{22528} \approx 186$
- Execution improves by a factor of 186 without changing the hardware!

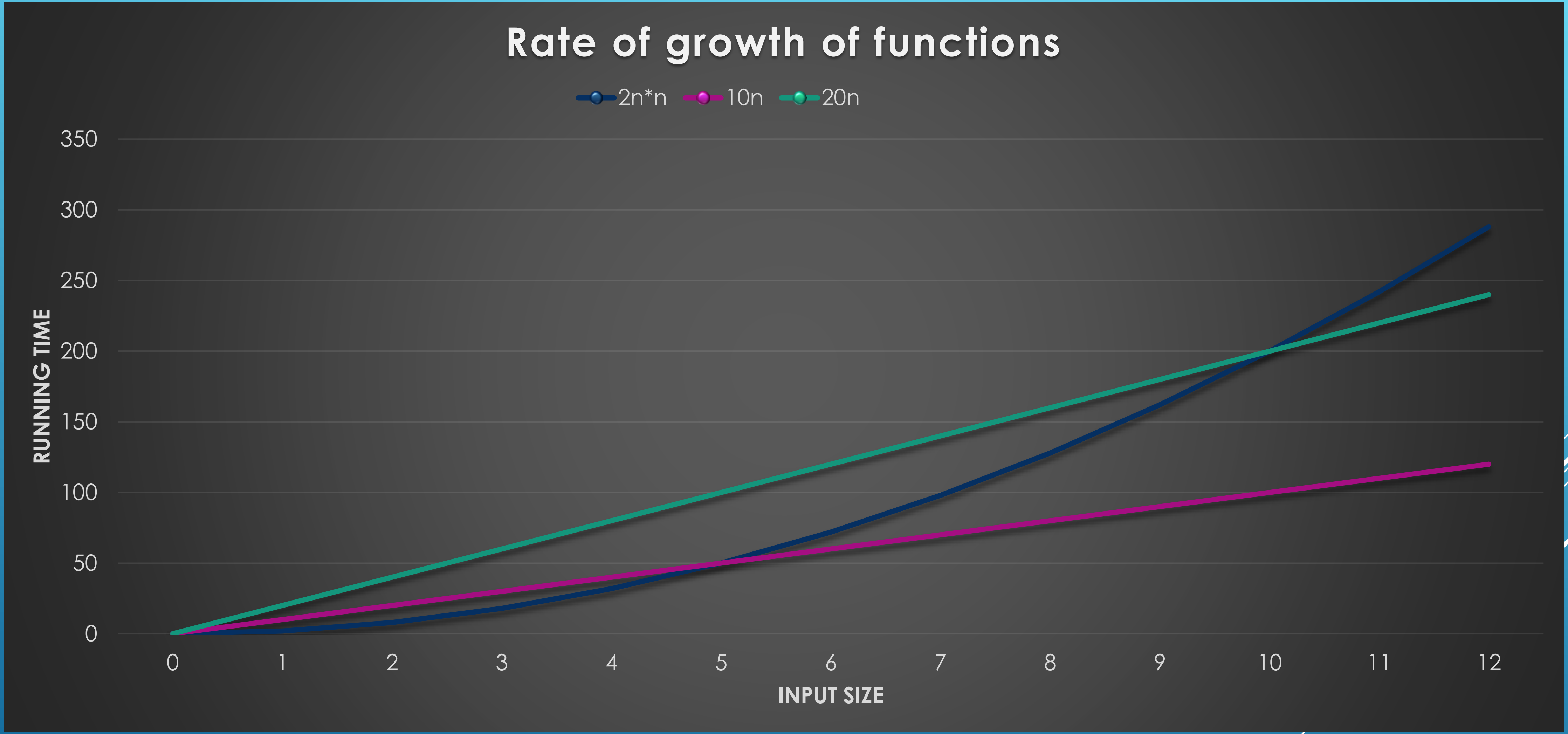
## DON'T CHANGE THE COMPUTER, CHANGE THE ALGORITHM!

- The notion that the effects of a bad algorithm can be mitigated by a good machine is FALSE.
- To reap full benefits of your investment in hardware, you must find a slick algorithm.
- You must learn Data Structures and Algorithms!!

## TO SUM IT UP…


Rate of growth of functions
2n*n
10n
350
300
250
200
150
100
50
0
RUNNING TIME
0
1
2
3
4
5
6
7
8
9
10
11
12
INPUT SIZE


Rate of growth of functions
2n*n
20n
RUNNING TIME
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8
9
10
11
12
INPUT SIZE


Rate of growth of functions
2n*n
10n
20n
RUNNING TIME
INPUT SIZE
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8
9
10
11
12

## ARE COEFFICIENTS IMPORTANT?

- Suppose the old algorithm was cubic and the new algorithm is quadratic in growth rate.
- The coefficients in the running times of the two algorithms:
  - Will not affect the improvement in problem size,
  - Will only affect the point of intersection of the two curves.
- After the point of intersection, moving towards $+\infty$, the curve with the larger growth rate will shoot away from the other curve.
- This leads us to ignore the coefficients of the function in asymptotic analysis, i.e. the analysis for large input sizes.

## WHEN SHOULD WE NOT IGNORE COEFFICIENTS?

- When we are comparing algorithms for small input sizes, the coefficients cannot be ignored.

  $T_A(n) = 10n, n = 4, T_A(4) = 40\ seconds$

  $T_B(n) = 20n, n = 4, T_B(4) = 80\ seconds$

  $T_c(n) = 2n^2, n = 4, T_C(4) = 32\ seconds$

- You can see that the quadratic algorithm, with the higher growth rate, gives a better running time than both linear algorithms, just because it comes with a smaller coefficient.
- We must remember to consider the coefficient when comparison involves small input sizes.

1. For each of the following running time expressions, how will the running time scale up as the data size *n* is increased eight-fold?

   a) $32 \log n$

   b) $\sqrt{n}$

   c) $n + \log n$

   d) $8n^2$

   e) $n^3$

   f) $16n \log n$

   g) $4^n$

## HOMEWORK

2. The running time of an algorithm is given as *T(n)*. How long will it take to execute on a 3.2GHz machine for the given input size, considering that each primitive operation takes 10 clock cycles?
   a) $T(n) = 5n^2 + 2n\sqrt{n}, n = 1000000$
   b) $T(n) = 3nlogn + 2n, n = 1000000$
   c) $T(n) = 2^n - 1, n = 100$

## HOMEWORK

3. The running time of an algorithm is given as *T(n)*. How large a problem can it solve in 10 minutes time on a processor capable of executing a basic operation in 5ns?

   a) $T(n) = 0.5\, n^3 + 20$

   b) $T(n) = 10 \log n + 100$

## HOMEWORK

## SO WHAT DID WE LEARN TODAY?

- We arranged functions in order of their growth rates.
- We realized that algorithms with slower growth rates are better than algorithms with faster growth rates.
- We learned that the coefficients can be ignored in asymptotic analysis.
- We agreed that for fast execution, we need better algorithms more than we need faster machines.

## THINGS TO DO

Read

- the book!

Note

- your questions and put them up in the relevant online session.

Email

- suggestions on content or quality of this lecture at uroojain@neduet.edu.pk