Algorithmen, KI und Data Science 1 (AKIDS 1): **Complexity and Sorting**

Herbst- / Wintersemester 2023/24

Prof. Dr. Goran Glavaš

Benedikt Ebing

Fabian David Schmidt

Exercise 1.1

Recap: Difference between O, Ω, and Θ

O-Notation:

$f \in$ O(g): For some **C** $\in R^{>0}, there\ exists\ a\ \boldsymbol{n_0} \in N, such\ that\ for\ all\ n \in N\ with\ n \geq n_0\ it\ holds:$

$$0 \leq \boldsymbol{f(n)} \leq \boldsymbol{C} \times \boldsymbol{g(n)}$$

Ω-Notation:

$f \in$ Ω(g): For some **C** $\in R^{>0}, there\ exists\ a\ \boldsymbol{n_0} \in N, such\ that\ for\ all\ n \in N\ with\ n \geq n_0\ it\ holds:$

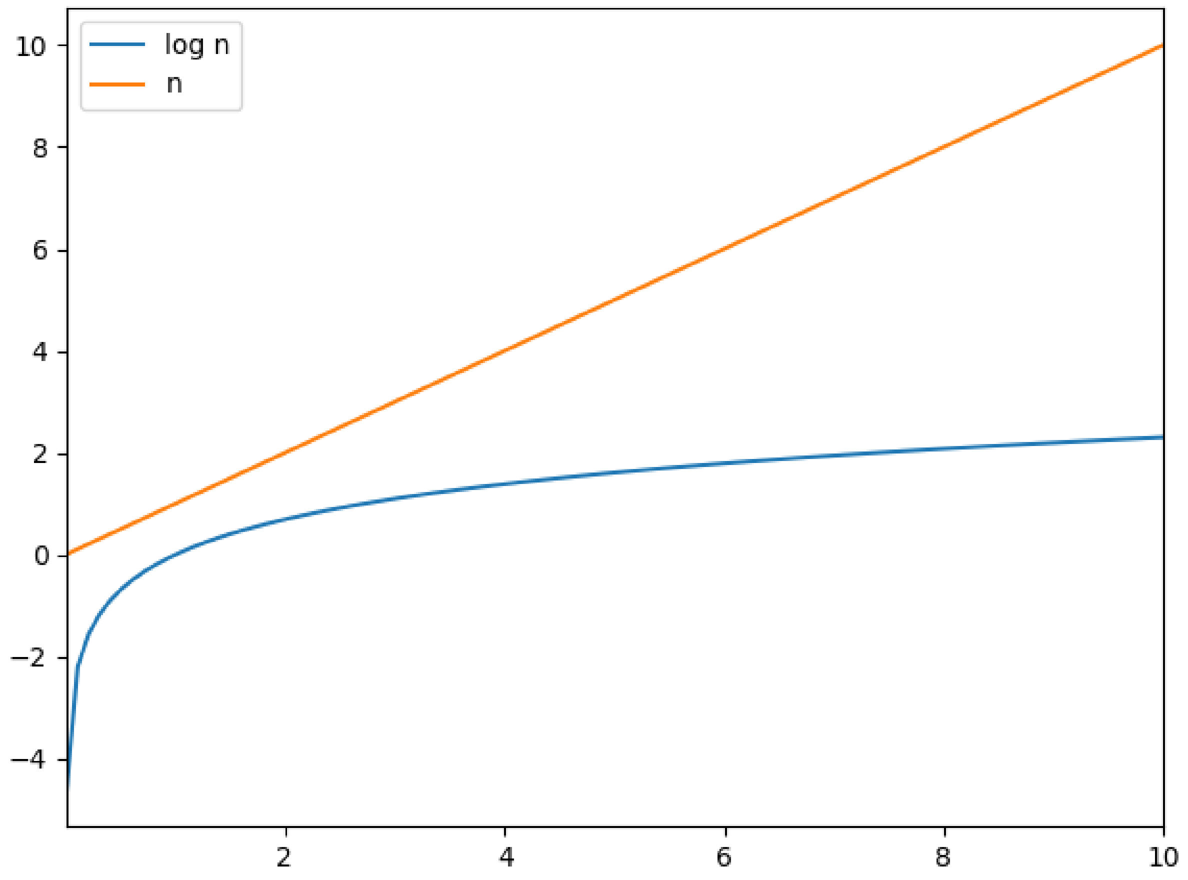
$$0 \leq \boldsymbol{C} \times \boldsymbol{g(n)} \leq \boldsymbol{f(n)}$$

Θ-Notation:

$f \in$ Θ(g): For some $\boldsymbol{C_1}, \boldsymbol{C_2} \in R^{>0}, there\ exists\ a\ \boldsymbol{n_0} \in N, such\ that\ for\ all\ n \in N\ with\ n \geq n_0\ it\ holds:$

$$\boldsymbol{0} \leq \boldsymbol{C_1} \times \boldsymbol{g(n)} \leq \boldsymbol{f(n)} \leq \boldsymbol{C_2} \times \boldsymbol{g(n)}$$

Example: $\log n \in O(n)$



Transitivity and Reflexivity

Transitivity

$f(n) \in \Theta(g(n))$ *and* $g(n) \in \Theta(h(n))$ *imply* $f(n) \in \Theta(h(n))$
$f(n) \in O(g(n))$ *and* $g(n) \in O(h(n))$ *imply* $f(n) \in O(h(n))$
$f(n) \in \Omega(g(n))$ *and* $g(n) \in \Omega(h(n))$ *imply* $f(n) \in \Omega(h(n))$

Reflexivity

- $f(n) \in \Theta(f(n))$
- $f(n) \in O(f(n))$
- $f(n) \in \Omega(f(n))$

Symmetry

Symmetry

- $f(n) \in \Theta\big(g(n)\big)\ if\ and\ only\ if\ g(n) \in \Theta(f(n))$

Transpose Symmetry

- $f(n) \in O\big(g(n)\big)\ if\ and\ only\ if\ g(n) \in \Omega(f(n))$

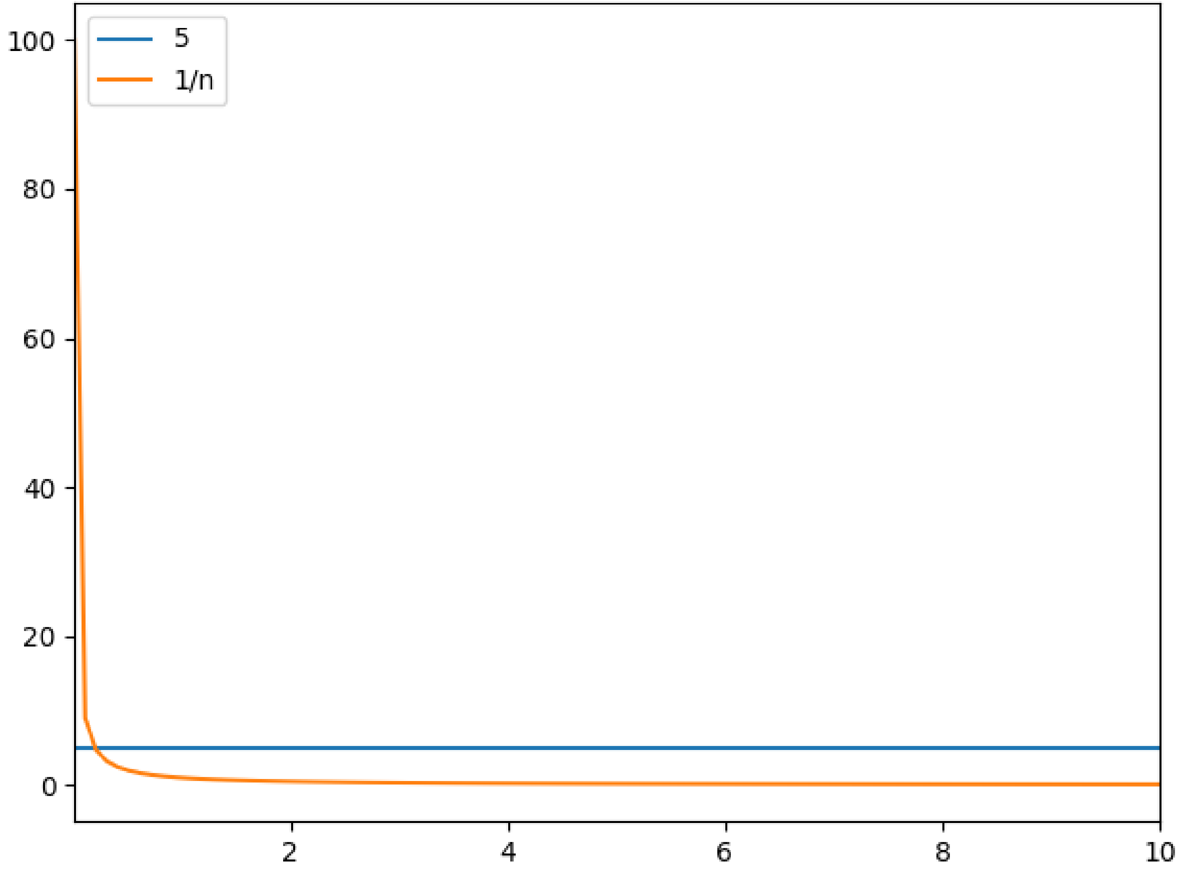
Complete the following table with symbols O, Ω, Θ.

	$\log n$	$2^{n/2}$	$\sqrt{n}$	5	2^n	$1/n$	n	e^n	n^2
$\log n$							O		
$2^{n/2}$									
$\sqrt{n}$									
5									
2^n									
$1/n$									
n									
e^n									
n^2									

Using reflexivity

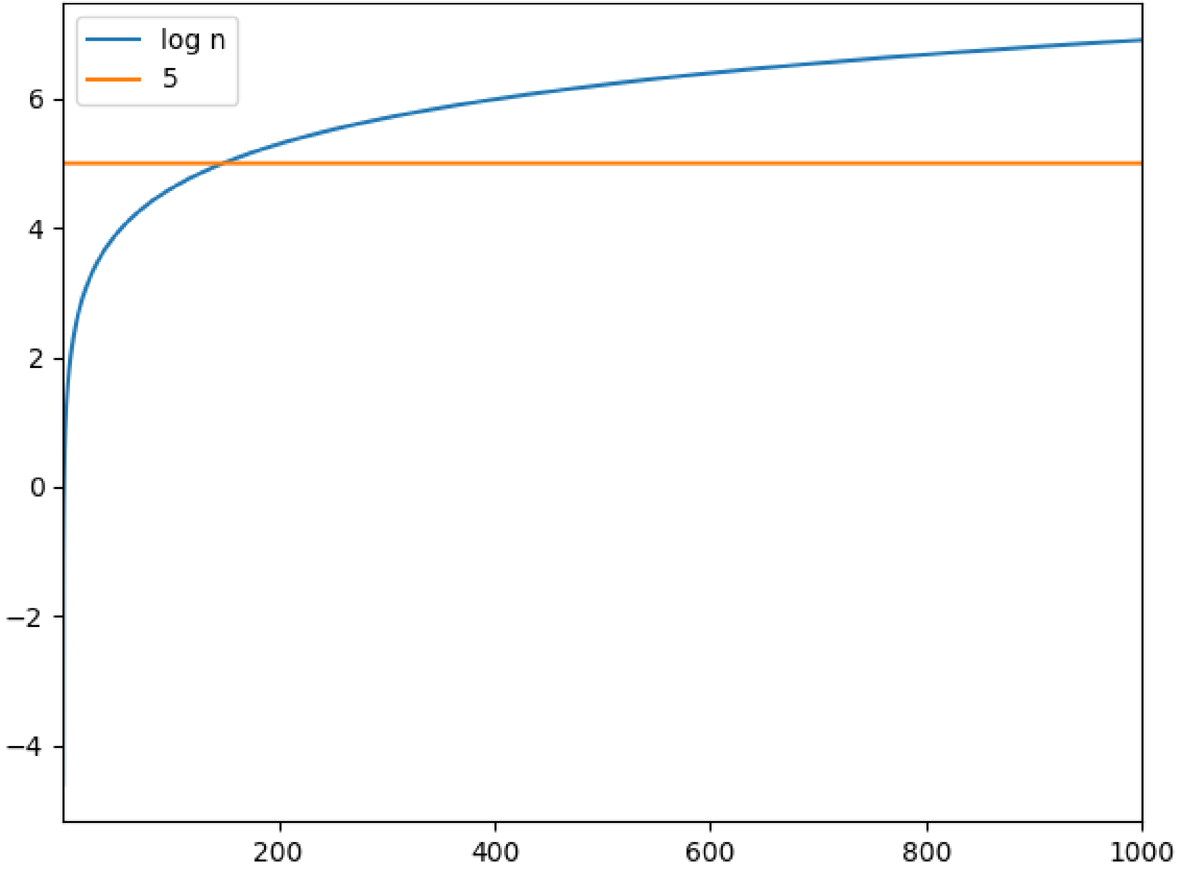
	$\log n$	$2^{n/2}$	$\sqrt{n}$	5	2^n	$1/n$	n	e^n	n^2
$\log n$	Θ						O		
$2^{n/2}$		Θ							
$\sqrt{n}$			Θ						
5				Θ					
2^n					Θ				
$1/n$						Θ			
n							Θ		
e^n								Θ	
n^2									Θ

$$5\ vs. \frac{1}{n}$$



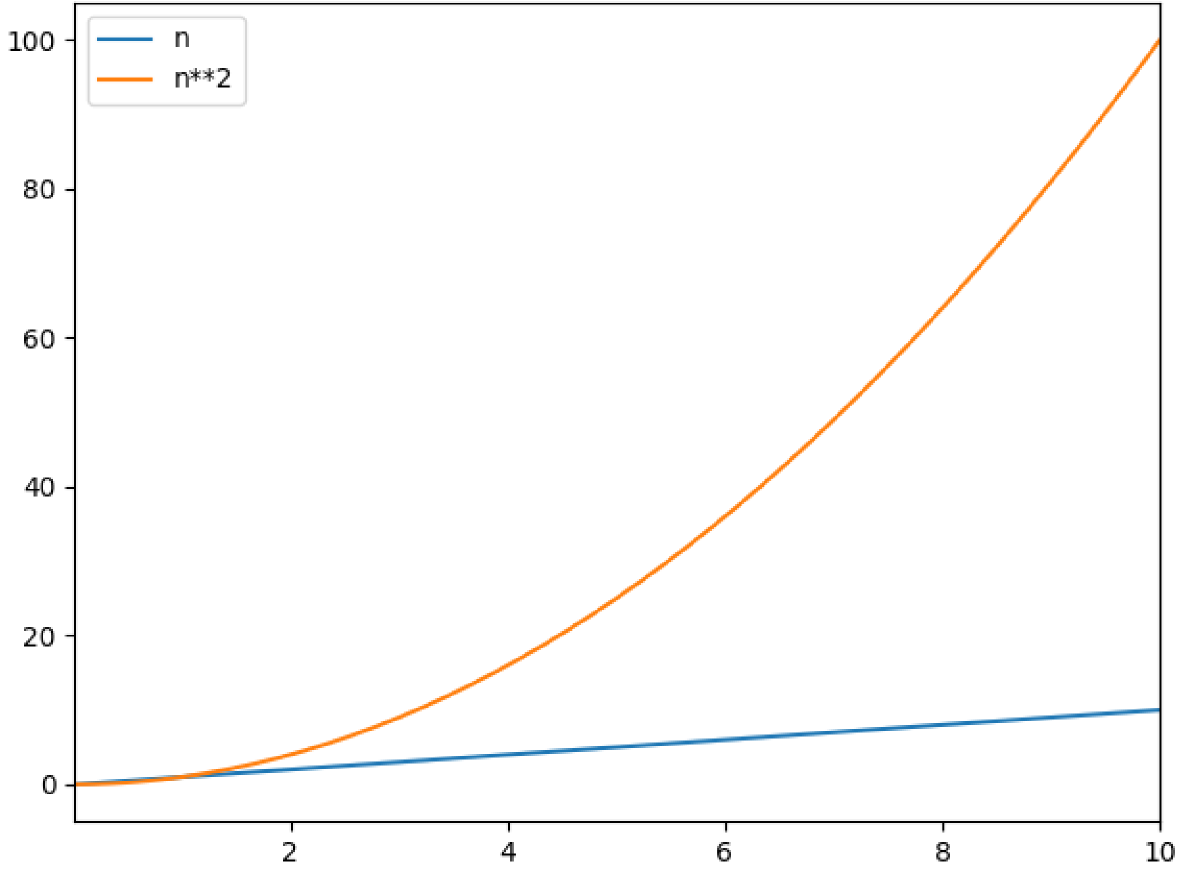
➔ $5 \in \Omega(\frac{1}{n}),\ \frac{1}{n} \in O(5)$

$\log n \;\; vs. 5$



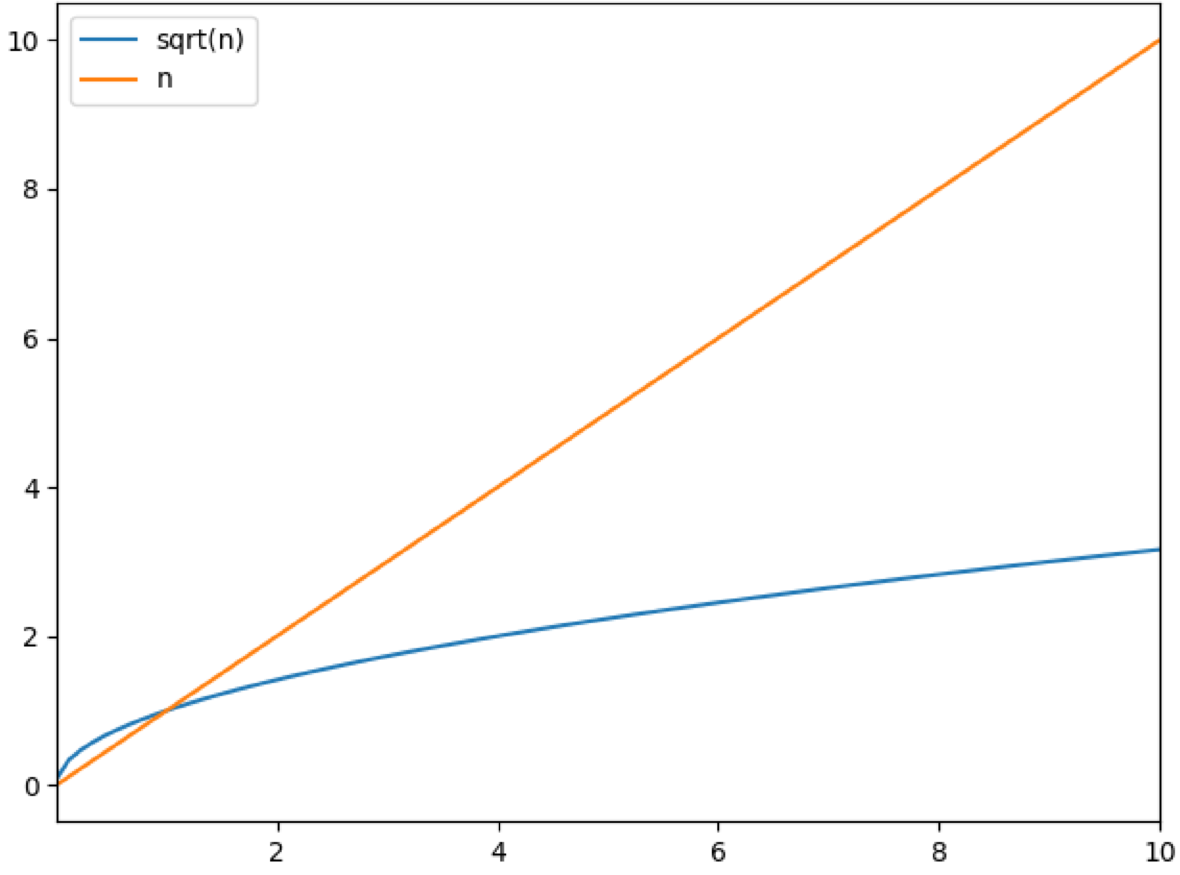
➔ $\log n \in \Omega(5), \;\; 5 \in O(\log n)$

$n\ vs.\ n^2$



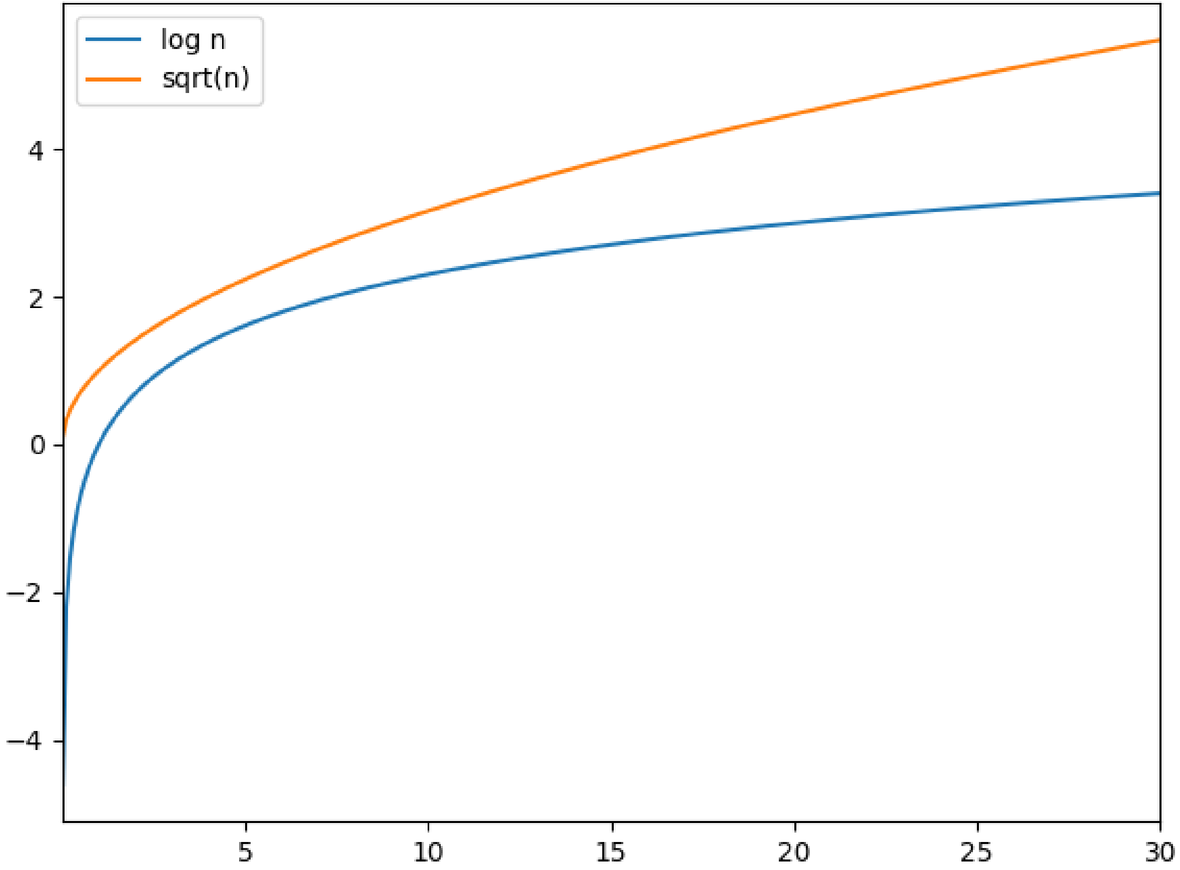
➔ $\mathrm{n} \in O(n^2), n^2 \in \Omega(n)$

$\sqrt{n}\ vs.\ n$



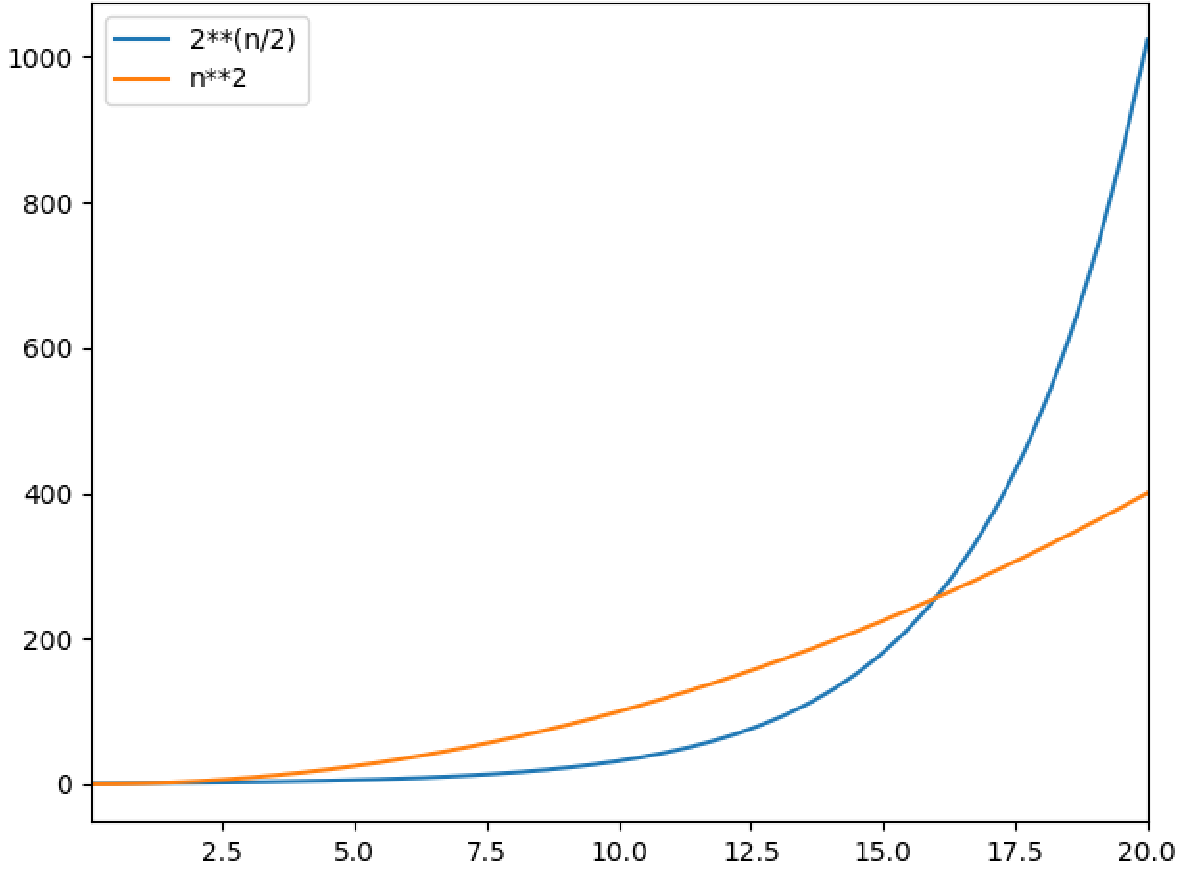
➔ $\sqrt{n} \in O(n), n \in \Omega(\sqrt{n})$

$\log n \;\; vs. \sqrt{n}$



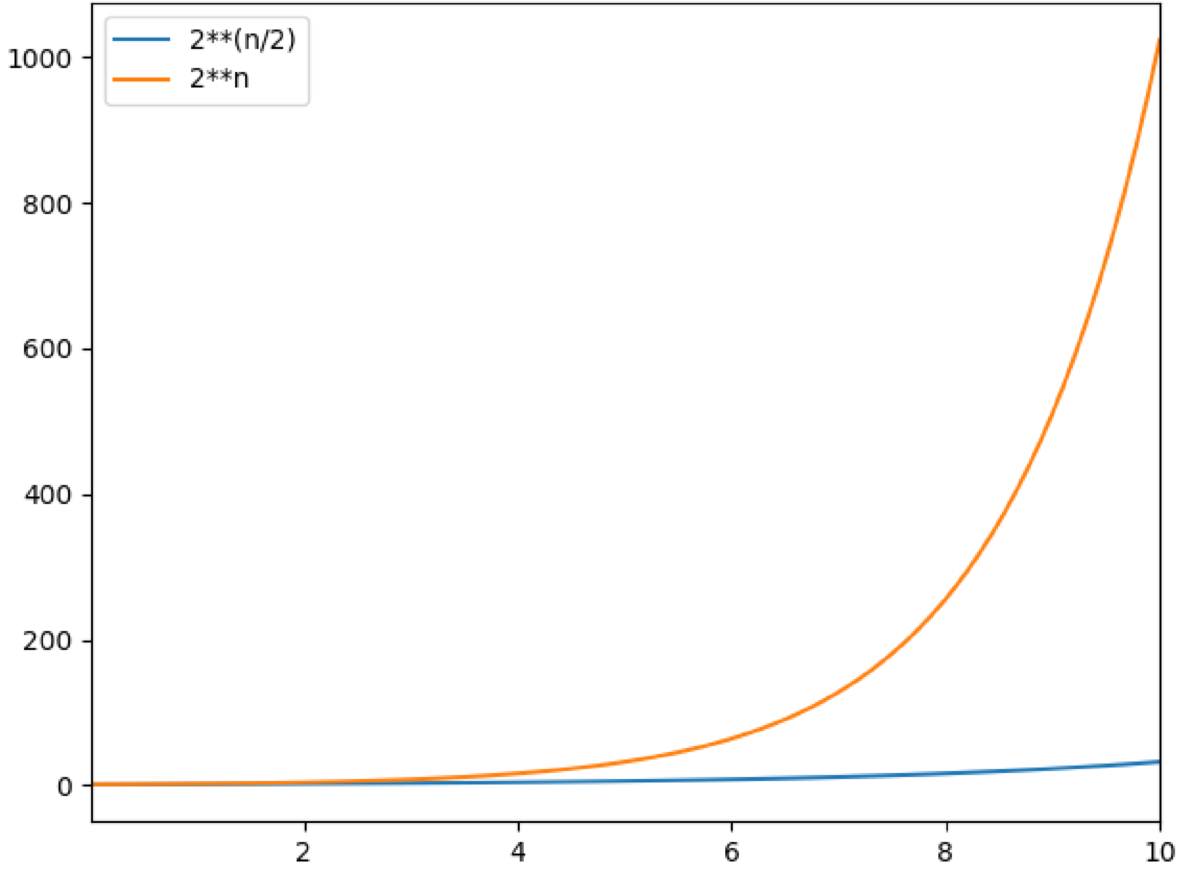
➔ $\log n \in O(\sqrt{n}), \sqrt{n} \in \Omega(\log n)$

$2^{\frac{n}{2}}$ vs. n^2



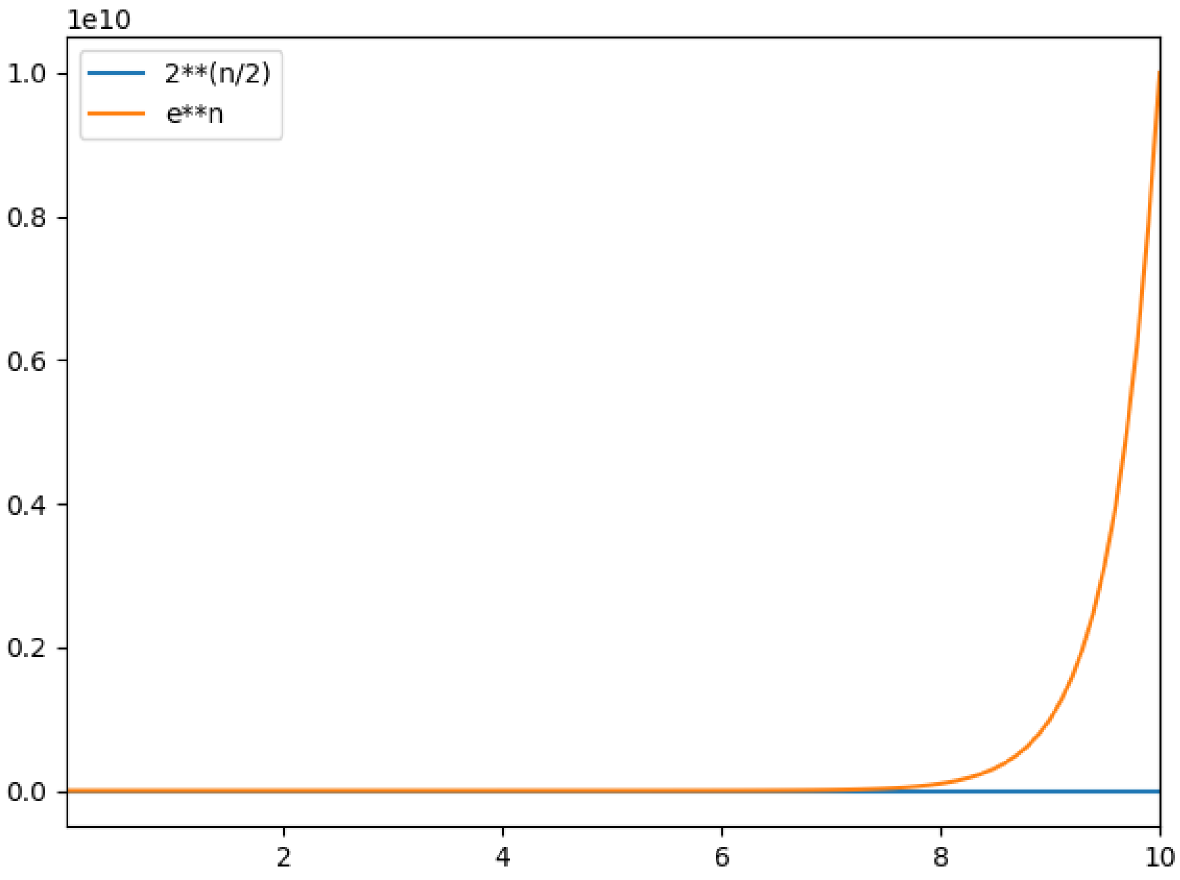
➔ $2^{\frac{n}{2}} \in \Omega(n^2), n^2 \in O(2^{\frac{n}{2}})$

$2^{\frac{n}{2}}$ vs. 2^n



➔ $2^{\frac{n}{2}} \in O(2^n), 2^n \in \Omega(2^{\frac{n}{2}})$

$2^{\frac{n}{2}}\ vs.\ e^n$



➔ $2^{\frac{n}{2}} \in O(e^n), e^n \in \Omega(2^{\frac{n}{2}})$

Summary of observations

- $\frac{1}{n} \in O(5)$
- $5 \in O(\log n)$
- $\log n \in O(\sqrt{n})$
- $\sqrt{n} \in O(n)$
- $n \in O(\mathrm{n}^2)$
- $n^2 \in O(2^{\frac{n}{2}})$
- $2^{\frac{n}{2}} \in O(2^{\mathrm{n}})$
- $2^{\mathrm{n}} \in O(e^{\mathrm{n}})$

➡ **transitivity** and **transpose symmetry**

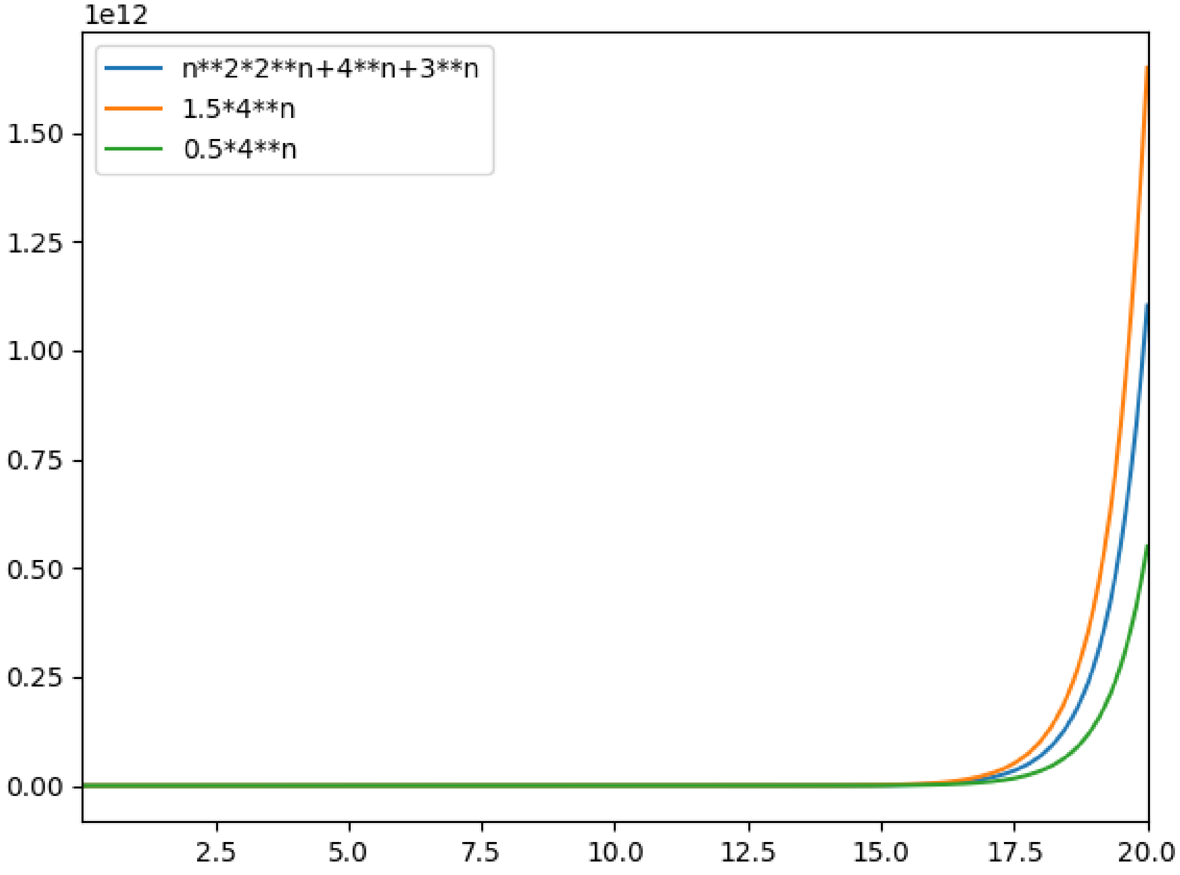
Solution

	$\log n$	$2^{n/2}$	$\sqrt{n}$	5	2^n	$1/n$	n	e^n	n^2
$\log n$	Θ	O	O	Ω	O	Ω	O	O	O
$2^{n/2}$	Ω	Θ	Ω	Ω	O	Ω	Ω	O	Ω
$\sqrt{n}$	Ω	O	Θ	Ω	O	Ω	O	O	O
5	O	O	O	Θ	O	Ω	O	O	O
2^n	Ω	Ω	Ω	Ω	Θ	Ω	Ω	O	Ω
$1/n$	O	O	O	O	O	Θ	O	O	O
n	Ω	O	Ω	Ω	O	Ω	Θ	O	O
e^n	Ω	Ω	Ω	Ω	Ω	Ω	Ω	Θ	Ω
n^2	Ω	O	Ω	Ω	O	Ω	Ω	O	Θ

For each of the following functions f_i, provide a function g_i having as few terms as possible and satisfying $f_i \in \Theta(g_i)$.

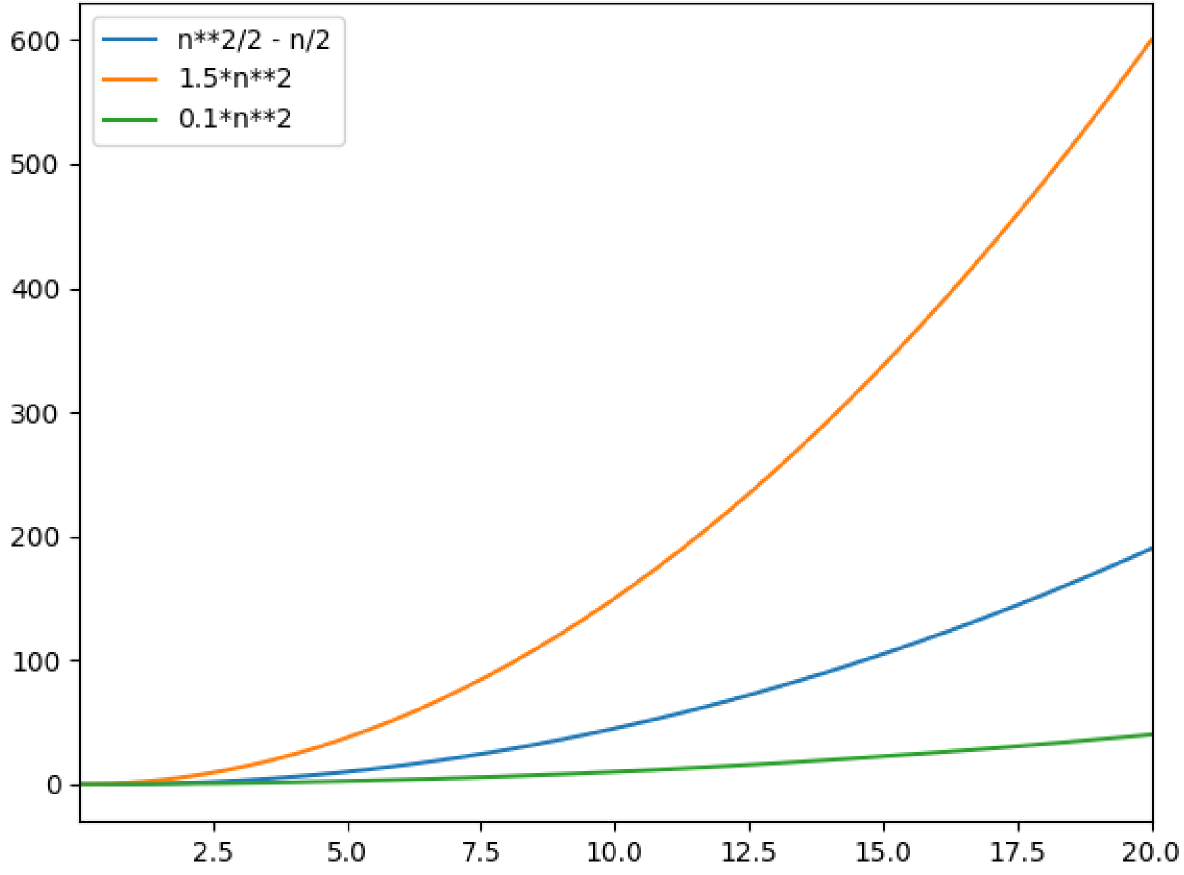
- $Example$: $f_0(n) = 3n^2 + 3 \in \Theta(n^2)$
- $f_1(n) = n^2 2^n + 4^n + 3^n$
- $f_2(n) = \frac{n(n-1)}{2}$
- $f_3(n) = \log(n^{70})$
- $f_4(n) = 9n\log(n) + 30n(\log(n))^2 + n$

$$f_1(n) = n^2 2^n + 4^n + 3^n$$



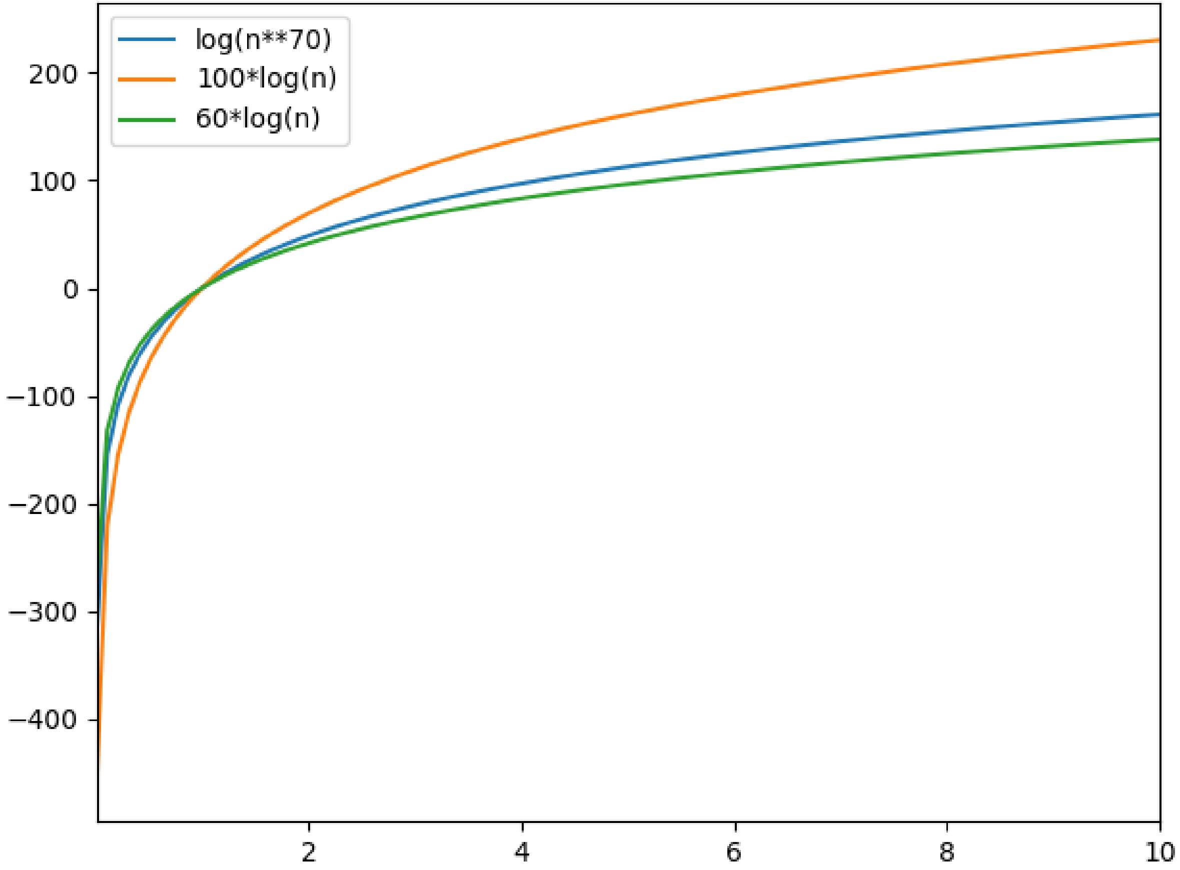
➔ $f_1 \in \Theta(4^n)$

$$f_2(n) = \frac{n(n-1)}{2} = \frac{\mathrm{n}^2}{2} - \frac{\mathrm{n}}{2}$$



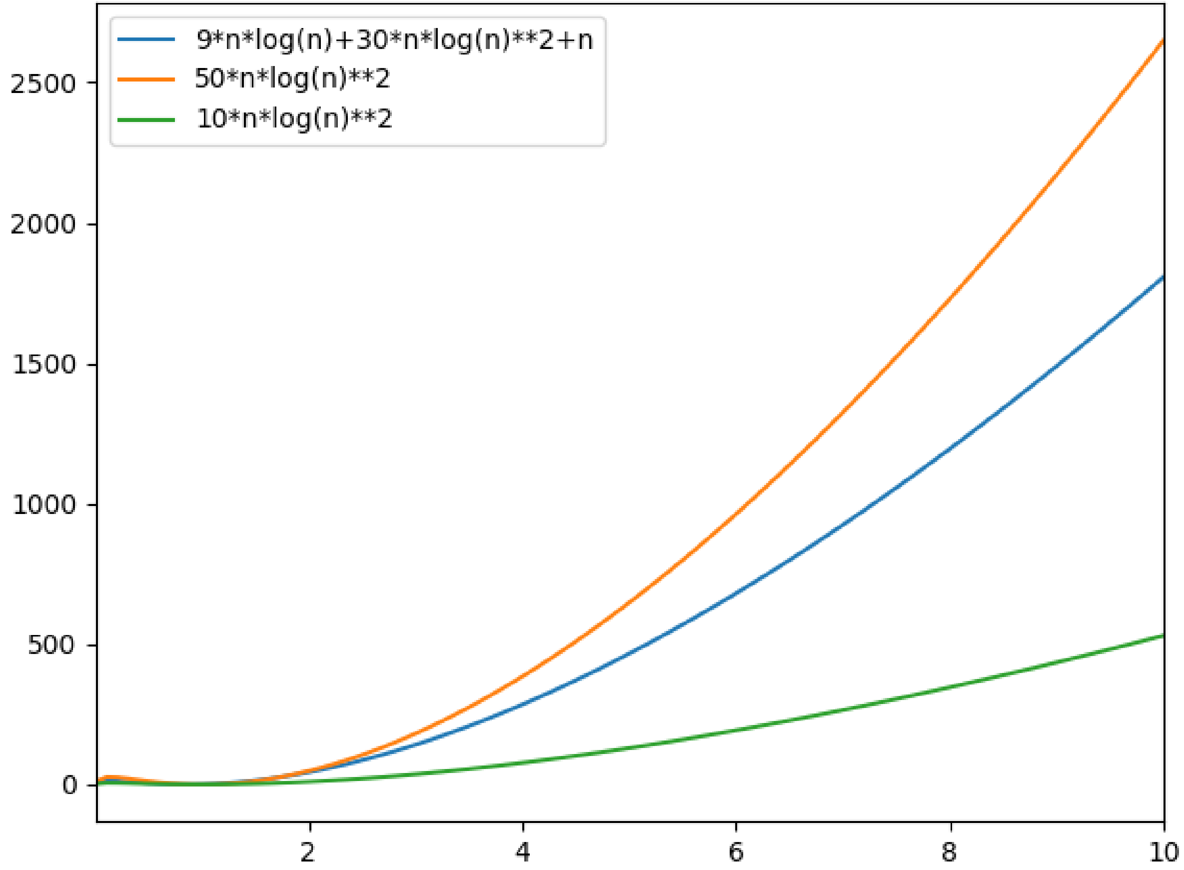
➔ $f_2 \in \Theta(\mathrm{n}^2)$

$$f_3(n) = \log(n^{70}) = 70 \log(n)$$



➔ $f_3 \in \Theta(\log(n))$

$$f_4(n) = 9n\log(n) + 30n(\log(n))^2 + n$$



➔ $f_3 \in \Theta(\mathrm{n}\log(\mathrm{n})^2)$

Exercise 1.3

Exercise 1.3

- Given Algorithm 1, explain in your own words, what the algorithm does.
- Determine its worst-case time complexity.
- Assuming that $nums$ is sorted. Implement the algorithm $algo2(nums, v)$ in the provided .ipynb that solves the problem in $\mathrm{O}(\log(n))$, where n is the length of $nums$.

Solution 1.3

Algorithm 1 algo1(nums, v)

for $i = 0$ **to** $nums.length - 1$ **do**
  if $nums[i] == v$ **then**
    return i
  end if
end for
return NIL

- Search for v in nums
- Return index of v, if found, otherwise NIL
- Complexity: $algo1 \in O(\mathrm{nums})$

Solution 1.3

- Algorithm we are looking for: „binary search“
- Idea (divide and conquer):
  - Compute median of the array (half the array)
  - Check whether v is found?
  - Check whether v is lower or higher than median
  - Proceed with the corresponding half

Exercise 2.1

Illustrate each step of merge sort on the following sequence: <3,9,1,2,7,3,9,6>.

3	9	1	2	7	3	9	6

3	9	1	2

7	3	9	6

3	9

1	2

7	3

9	6

3

9

1

2

7

3

9

6

3	9

1	2

3	7

6	9

1	2	3	9

3	6	7	9

1	2	3	3	6	7	9	9

Exercise 2.2

What value does partition return when all elements in the subarray $A[p:r]$ have the same value?

```
partition(A, p, r)
   pivot = A[r]
   s = p - 1 # index of the
   for i = p to r - 1:
     if A[i] ≤ pivot
       s = s + 1
       exchange(A[i], A[s])
   exchange(A[s+1], A[r])
   return s + 1
```

- $A[i] \leq pivot$ is always true
- ➔ method returns r

```
quick_sort(A, p, r)
   q = partition(A, p, r)
   quick_sort(A, p, q - 1)
   quick_sort(A, q + 1, r)
```

Exercise 2.3

Give a brief argument that the running time of partition on a subarray of size n is $O(n)$

```
partition(A, p, r)
   pivot = A[r]
   s = p - 1 # index of the
   for i = p to r - 1:
     if A[i] ≤ pivot
       s = s + 1
       exchange(A[i], A[s])
   exchange(A[s+1], A[r])
   return s + 1
```

- p – (r - 1) iterations in the loop
  - Each takes constant time

➔p – r = n

Exercise 2.4

Recap: insersort

```
insert_sort(L) # L is a list of numbers
  for i = 1 to L.length - 1 # 0-indexing
    key = L[i]
    j = i-1
    while j > -1 and L[j] > key
      L[j+1] = L[j]
      j = j - 1
    L[j+1] = key
```