LANDAU NOTATIONS
(a.k.a. Asymptotic notations)

upper bound

$$O(f(n)) = \{g(n) \mid \exists\, n_0, c > 0 \text{ such that } \forall\, n > n_0,\ g(n) \leq c \cdot f(n)\}$$

lower bound

$$\Omega(f(n)) = \{g(n) \mid \exists\, n_0, c > 0 \text{ such that } \forall\, n > n_0,\ g(n) \geq c \cdot f(n)\}$$

$$\Theta(f(n)) = \{g(n) \mid \exists\, n_0, c_1, c_2 > 0 \text{ such that } \forall\, n > n_0,\ c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)\}$$

exact bound

obviously,

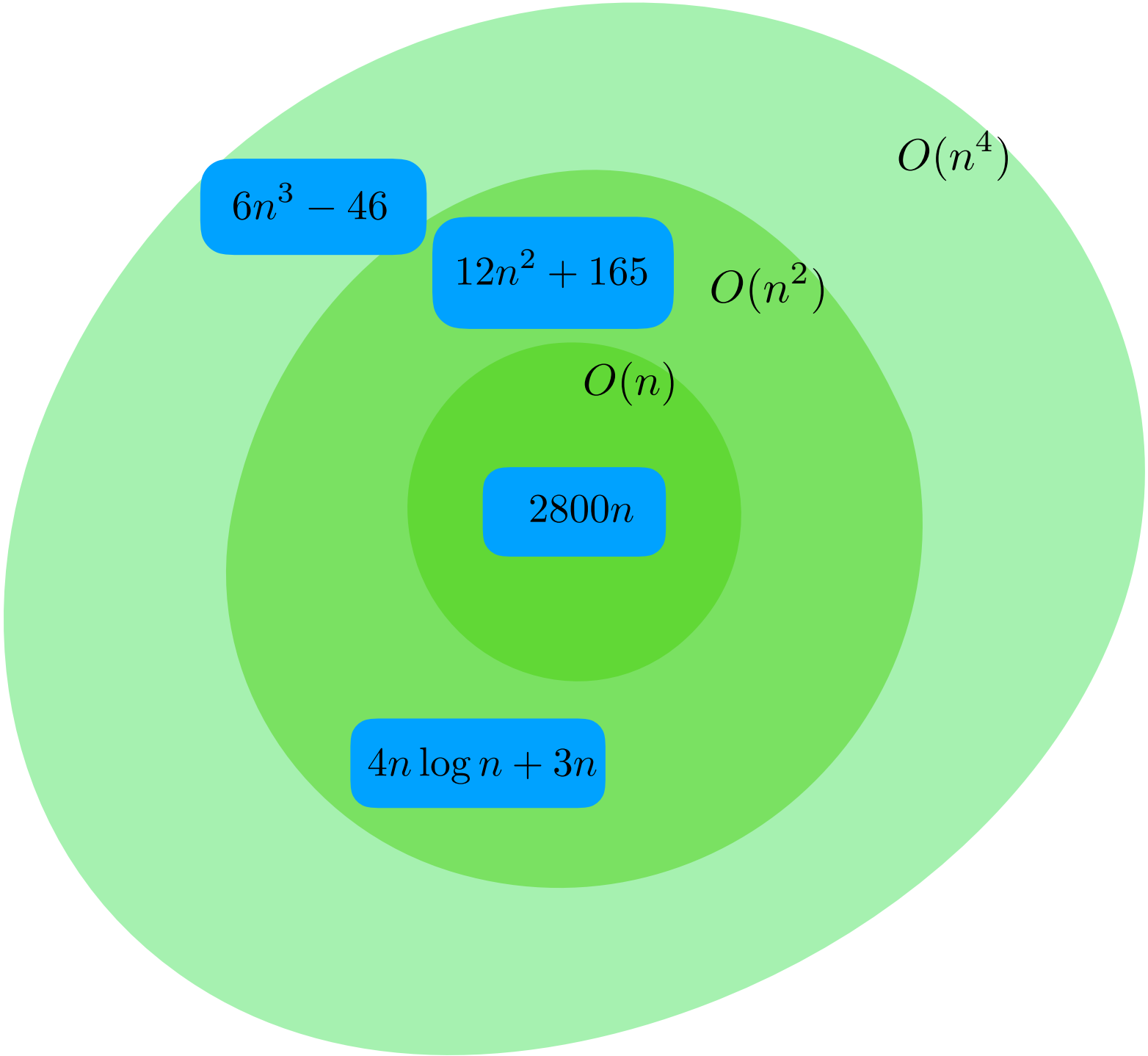
$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

234n^3 + 2 n^2 - 234 belongs to Theta(n^5) NO

234n^3 + 2 n^2 - 234 belongs to Theta(n^3) NO

234n^3 + 2 n^2 - 234 belongs to Theta(n^2) NO

WHERE ARE THESE FUNCTIONS?



TRUE OR FALSE?

(logarithms are in base 2, where meaningful)

$n\sqrt{n} \in O(2^{\log n})$

$n^3 \in O(n^{1.5})$

$2^{(\log n)^2} \in \Omega(2^n)$

COMPARE LOGS log^2(n) n

$\log^4 n \in \Omega(n)$

$n \in \Theta(4^{\frac{\log n}{2}})$

log(n) log(n)

$n \log n \in \Omega(n^{1.1})$

$n^3 \log n \in O(n^2)$

$n^7 \in \Omega(2^{5 \log n})$

2^((log n) 5) = (2^logn)^5 = n^5

SORT THESE CLASSES

log(…) = n $O(2^n)$

log(…..) = sqrt(n) + log(n) $O(2^{\sqrt{n}} n)$

log(…) = sqrt(n)/2 + 3 log(n) $O(2^{\frac{\sqrt{n}}{2}} n^3)$

log(…) = log^2(n) log(n) = log^3 n $O(n^{\log^2 n})$

log(…) = 4 log(n) $O(n^4)$

$O(n^3 \log^5 n)$

$O(n^3 \log n)$

$O(n^3)$

$O(n^2)$

$O\left(\frac{n^2}{\log n}\right)$

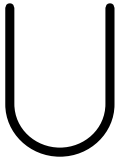
$O(n \log n)$

$O(n)$

$O(\sqrt{n})$

$O(\log^8 n)$

$O(\log n)$



ASYMPTOTIC COMPLEXITY vs COMPUTING TIME

assume one billion operations per s

set of n numbers: find a subset whose sum is S
TSP: travelling salesman

instance size	log n	n	n log n	n^2	n^4	2^n
10	3 ns	10 ns	33 ns	100 ns	10 microsec	1 microsec
20	4 ns	20 ns	86 ns	400 ns	160 microsec	1 millisec
50	6 ns	50 ns	282 ns	2.5 microsec	6 millisec	13 days
100	7 ns	100 ns	664 ns	10 microsec	0.1 sec	40,000 billions years
1000	10 ns	1 microsec	10 microsec	1 millisec	17 min	***
1,000,000	20 ns	1 millisec	20 millisec	17 min	31 millions years	
1,000,000,000	30 ns	1 sec	30 sec	31.7 years	31 billions billions years	

graph: find the shortest path

graph: find the longest path without passing two times the same place

COMPLEXITY OF SOME ALGORITHMS

binary search

O(log n)

sequential scan
(linear search, sum, avg, maximum, …)

VERY SMART sort

O(n) ???

merge sort

O(n log n)

selection sort

O(n^2)

primality testing
(n is the number value)

hamiltonian path

O(2^n)

COMPLEXITY OF A PROBLEM (intrinsic complexity)

we say the INTRINSIC COMPLEXITY of problem P is F() if ANY POSSIBLE ALGORITHMS that solves P has complexity at least F()

this is a lower bound for the complexity of any also for P

find the maximum REQUIRES Omega(n) operations

OPTIMAL ALGORITHM:

an algorithm for P is OPTIMAL if its complexity matches the INTRINSIC COMPLEXITY of P

look at on number at a time and compare with the current maximum performs O(n) operations

sort three numbers

```
1  def sort_three(a, b, c):
2      if a < b:
3          if b < c:
4              return [a, b, c]
5          else:
6              if a < c:
7                  return [a, c, b]
8              else:
9                  return [c, a, b]
10     else:
11         if a < c:
12             return [b, a, c]
13         else:
14             if b < c:
15                 return [b, c, a]
16             else:
17                 return [c, b, a]
```

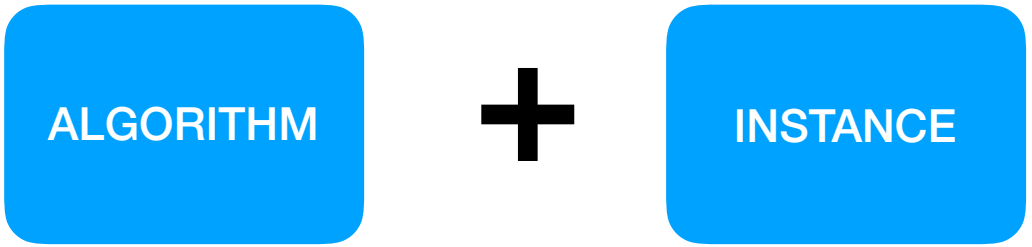
need 6 possible outcomes (3!) in order to answer correctly

worst case: 3 comparisons

is it possible with only 2 comparisons?

one comparison … two outcomes ….
one more comp. … 4 outcomes

COMPUTATION



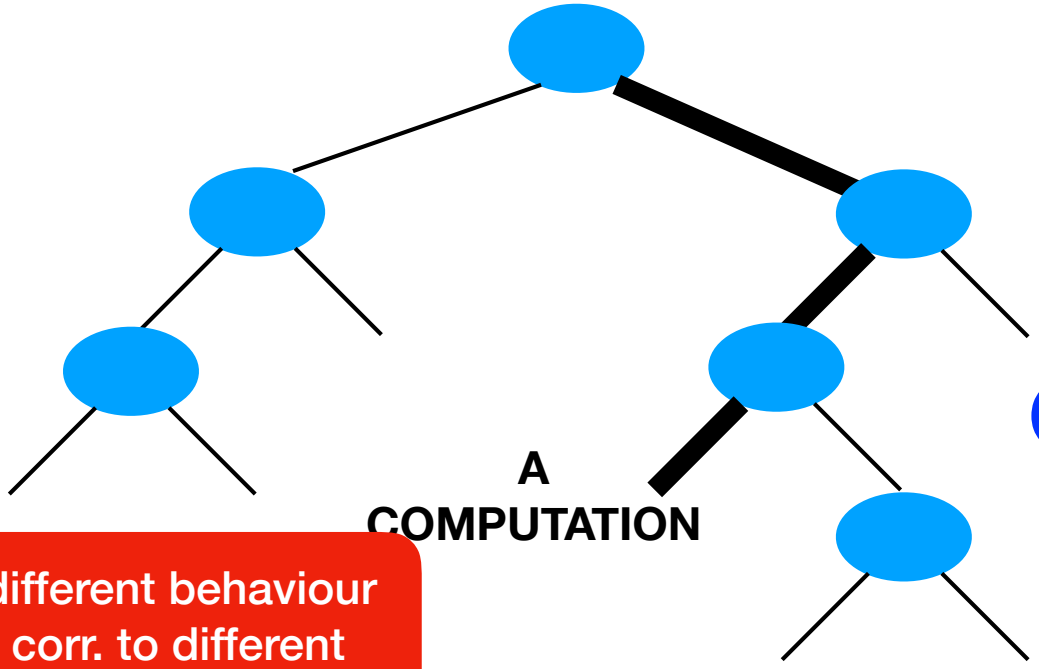
different instances give rise to different computations …
… worst case, best case, …
… longest computation, shortest computation (for instances of length n)

HOW CAN WE DERIVE THE INTRINSIC COMPLEXITY OF A PROBLEM?

(for comparison based algorithms)

**some operations, e.g. comparisons,
can differentiate the behaviour of the algorithm,
giving rise to different computations**

BRANCHES

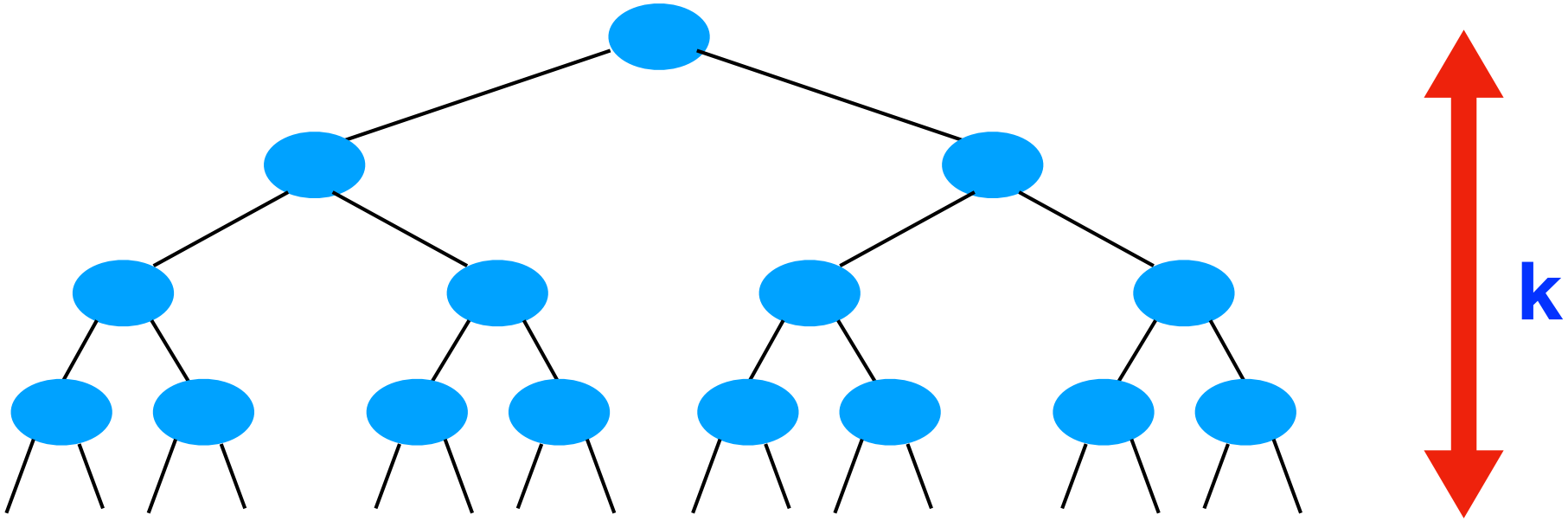


COMPUTATION TREE

different behaviour corr. to different leaves

NUMBER OF POSSIBLE COMPUTATIONS

if the longest computation has length k,
how many different computations are possible?

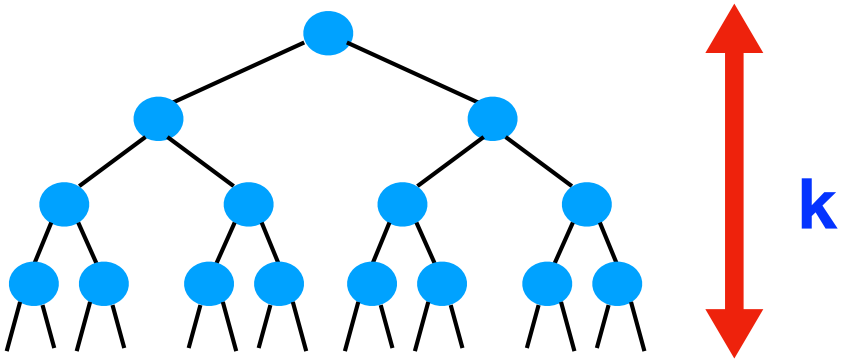


2^k different computations

INTRINSIC COMPLEXITY

in order to have at least S different results
we need at least S different computations

k (the depth of the computation tree)
must be at least log S

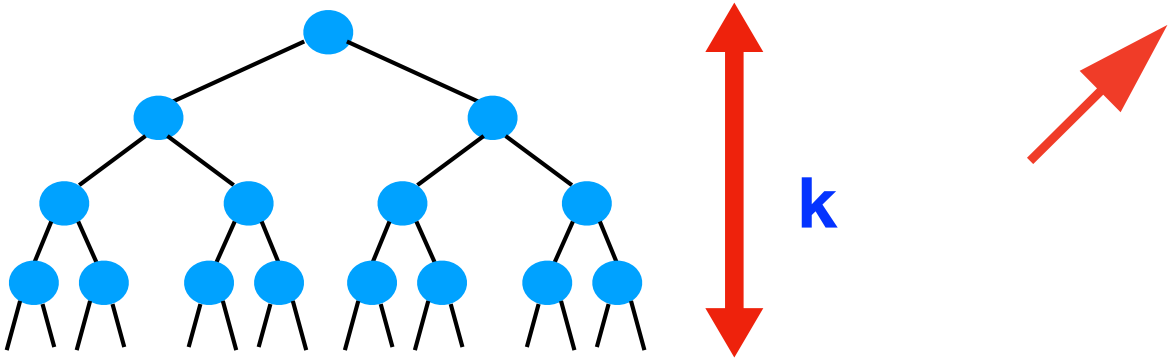


SORT 3 NUMBERS

in order to have at least 3! different results
we need at least 3! = 6 different computations

k (the depth of the computation tree)
must be at least log 6 (logs are in base 2)

2 < log 6 < 3 **2 comparisons are not enough**



intrinsic complexity for sorting n numbers

- **how many different results? n! ALL THE POSSIBLE PERMUTATIONS**
- **the intrinsic complexity is log (n!) NEEDED TO CHOOSE AMONG n! results**
- **just evaluate log(n!) IN AN ASYMPTOTIC WAY**

sort 5 numbers …… 5! =120 different possible results, ONE is correct

need a number of BRANCHES that is at least log(120) in base 2 = 6.xxx

6 comparisons are not enough, I need 7