2.2 Asymptotic Order of Growth

- definitions and notation (2.2)
- examples (2.4)
- properties (2.2)

Asymptotic Order of Growth

Upper bounds. $T(n)$ is $O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \leq c \cdot f(n)$.

Lower bounds. $T(n)$ is $\Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \geq c \cdot f(n)$.

Tight bounds. $T(n)$ is $\Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$.

Ex: $T(n) = 32n^2 + 17n + 32$.

Q. Is $T(n)$...

1) $O(n^2)$?
2) $\Omega(n^3)$?
3) $O(n^3)$?
4) $\Theta(n)$?
5) $O(n)$?
6) $\Omega(n^2)$?
7) $\Omega(n)$?
8) $\Theta(n^2)$?
9) $\Theta(n^3)$?

Asymptotic Order of Growth

Upper bounds. $T(n)$ is $O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \leq c \cdot f(n)$.

Lower bounds. $T(n)$ is $\Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ we have $T(n) \geq c \cdot f(n)$.

Tight bounds. $T(n)$ is $\Theta(f(n))$ if $T(n)$ is both $O(f(n))$ and $\Omega(f(n))$.

Ex: $T(n) = 32n^2 + 17n + 32$.

A.

$T(n)$ is $O(n^2)$, $O(n^3)$, $\Omega(n^2)$, $\Omega(n)$, and $\Theta(n^2)$.

$T(n)$ is not $O(n)$, $\Omega(n^3)$, $\Theta(n)$, or $\Theta(n^3)$.

Notation

Slight abuse of notation. $T(n) = O(f(n))$.

Asymmetric:

- $f(n) = 5n^3$; $g(n) = 3n^2$
- $f(n) = O(n^3) = g(n)$
- but $f(n) \neq g(n)$.

Better notation: $T(n) \in O(f(n))$.

Notation

Slight abuse of notation. $T(n) = O(f(n))$.

Asymmetric:

- $f(n) = 5n^3$; $g(n) = 3n^2$
- $f(n) = O(n^3) = g(n)$
- but $f(n) \neq g(n)$.

Better notation: $T(n) \in O(f(n))$.

Meaningless statement. Any comparison-based sorting algorithm requires at least $O(n \log n)$ comparisons. (Watch out!)

Q. What is wrong with this statement?

Notation

Slight abuse of notation. $T(n) = O(f(n))$.

Asymmetric:

- $f(n) = 5n^3$; $g(n) = 3n^2$
- $f(n) = O(n^3) = g(n)$
- but $f(n) \neq g(n)$.

Better notation: $T(n) \in O(f(n))$.

Meaningless statement. Any comparison-based sorting algorithm requires at least $O(n \log n)$ comparisons. (Watch out!)

Q. What is wrong with this statement?

A.

Statement doesn't "type-check”: $O(f(n))$ is for upper bounds

Use Ω for lower bounds.

2.4 A Survey of Common Running Times

Worst-case analysis

Q. What is the worst-case running time of the following algorithms on an array of length n? (1 min)

find the maximum value

insertion sort (p.113 of Bailey)

merge sort (p.115-118 of Bailey)

find in a binary search tree (p.331-)

Worst-case analysis

Q. What is the worst-case running time of the following algorithms on an array of length n? (1 min)

find the maximum value	$O(n)$
insertion sort (p.113 of Bailey)	$O(n^2)$
merge sort (p.115-118 of Bailey)	$O(n \log n)$
find in a binary search tree (p.331-)	$O(\log n)$

TU Delft

Linear Time: O(n)

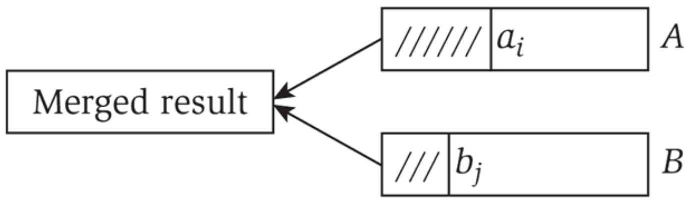
Linear time. Running time is at most a constant factor times the size of the input.

Computing the maximum. Compute maximum of n numbers $a_1, \ldots, a_n$.

```
max ← a1
for i = 2 to n {
   if (ai > max)
      max ← ai
}
```

Linear Time: O(n)

Merge. Combine two sorted lists $A = a_1, a_2, \ldots, a_n$ with $B = b_1, b_2, \ldots, b_n$ into sorted whole.




```
i = 1, j = 1
while (both lists are nonempty) {
   if (a_i ≤ b_j) append a_i to output list and increment i
   else           append b_j to output list and increment j
}
append remainder of nonempty list to output list
```

Claim. Merging two lists of size n takes O(n) time.

Pf. After each comparison, the length of output list increases by 1.

O(n log n) Time

O(n log n) time. Arises in divide-and-conquer algorithms.

also referred to as linearithmic time

Sorting. Mergesort and heapsort are sorting algorithms that perform O(n log n) comparisons.

Largest empty interval. Given n time-stamps $x_1, \ldots, x_n$ on which copies of a file arrive at a server, what is largest interval of time when no copies of the file arrive?

O(n log n) solution. Sort the time-stamps. Scan the sorted list in order, identifying the maximum gap between successive time-stamps.

Quadratic Time: $O(n^2)$

Quadratic time. Enumerate all pairs of elements.

Closest pair of points. Given a list of n points in the plane $(x_1, y_1), \ldots, (x_n, y_n)$, find the pair that is closest.

$O(n^2)$ solution. Try all pairs of points.

```
min ← (x1 - x2)² + (y1 - y2)²
for i = 1 to n {
   for j = i+1 to n {
      d ← (xi - xj)² + (yi - yj)²
      if (d < min)
         min ← d
   }
}
```

don't need to take square roots

Remark. $\Omega(n^2)$ seems inevitable, but this is just an illusion.

see chapter 5

Cubic Time: $O(n^3)$

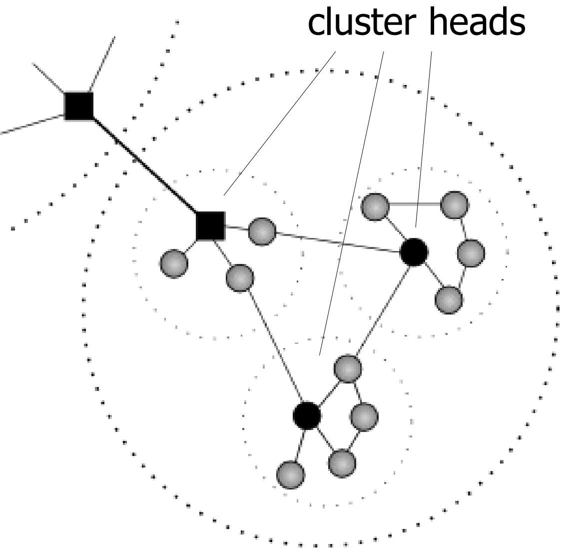
Cubic time. Enumerate all triples of elements.

Set disjointness. Given n sets $S_1, \ldots, S_n$ each of which is a subset of 1, 2, …, n, is there some pair of these which are disjoint?

$O(n^3)$ solution. For each pairs of sets, determine if they are disjoint.

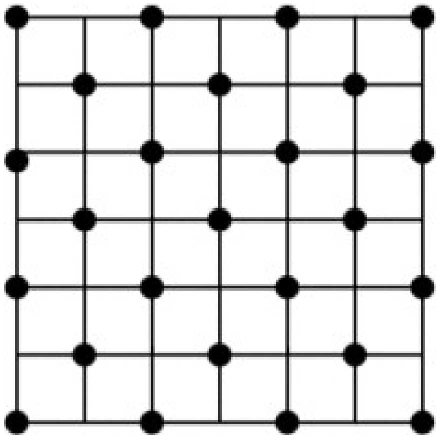
```
foreach set Si {
   foreach other set Sj {
      foreach element p of Si {
         determine whether p also belongs to Sj
      }
      if (no element of Si belongs to Sj)
         report that Si and Sj are disjoint
   }
}
```

Routing in ad-hoc wireless networks



- only some nodes can communicate directly with each other
- how to route messages to nodes that cannot be reached directly?
- common approach is to create a hierarchy

Goal. are there *k cluster heads that do not interfere* during simultaneous transmissions?



Similar to

Given. a set of potential locations for e.g. a Starbucks and connections if they interfere.

Goal. are there k locations that do not interfere?

Polynomial Time with a fixed parameter: $O(n^k)$ Time

Independent set of size k. Given a graph, are there k nodes such that no two are joined by an edge?

k is a constant

$O(n^k)$ solution. Enumerate all subsets of k nodes.

```
foreach subset S of k nodes {
   check whether S is an independent set
   if (S is an independent set)
      report S
   }
}
```

Check whether S is an independent set = $O(k^2)$.

Number of k element subsets = $\binom{n}{k} = \frac{n\,(n-1)\,(n-2)\,\cdots\,(n-k+1)}{k\,(k-1)\,(k-2)\,\cdots\,(2)\,(1)} \leq \frac{n^k}{k!}$

$O(k^2 n^k / k!) = O(n^k)$.

poly-time for k=17,
but not practical

Exponential Time

Independent set. Given a graph, what is maximum size of an independent set?

$O(n^2\, 2^n)$ solution. Enumerate all subsets.

```
S* ← ϕ
foreach subset S of nodes {
   check whether S is an independent set
   if (S is largest independent set seen so far)
      update S* ← S
   }
}
```

Advanced algorithms (using bounded search trees):
$O^*(1.3803^n)$ or even $O^*(1.2227^n)$

Properties

Transitivity.

If f is O(g) and g is O(h) then f is O(h).

If f is Ω(g) and g is Ω(h) then f is Ω(h).

If f is Θ(g) and g is Θ(h) then f is Θ(h).

Additivity.

If f is O(h) and g is O(h) then f + g is O(h).

If f is Ω(h) and g is Ω(h) then f + g is Ω(h).

If f is Θ(h) and g is Θ(h) then f + g is Θ(h).

Asymptotic Bounds for Some Common Functions

Polynomial. $a_0 + a_1 n + \ldots + a_d n^d$ is $\Theta($ f(n) $)$ if $a_d > 0$.

Q. What is the simplest f(n)? (where simplest is the least number of terms)

A.

Q. For every $x > 0$, log n is $O(n^x)$? (or n^x is O(log n) ?)

A.

Q. For every $r > 1$ and every $d > 0$ is r^n is $O(n^d)$? (or n^d is $O(r^n)$?)

A.

Asymptotic Bounds for Some Common Functions

Polynomial. $a_0 + a_1 n + \ldots + a_d n^d$ is $\Theta($ f(n) $)$ if $a_d > 0$.

Q. What is the simplest f(n)? (where simplest is the least number of terms)

A. $f(n) = n^d$

Polynomial time. Running time is $O(n^d)$ for some constant d independent of the input size n.

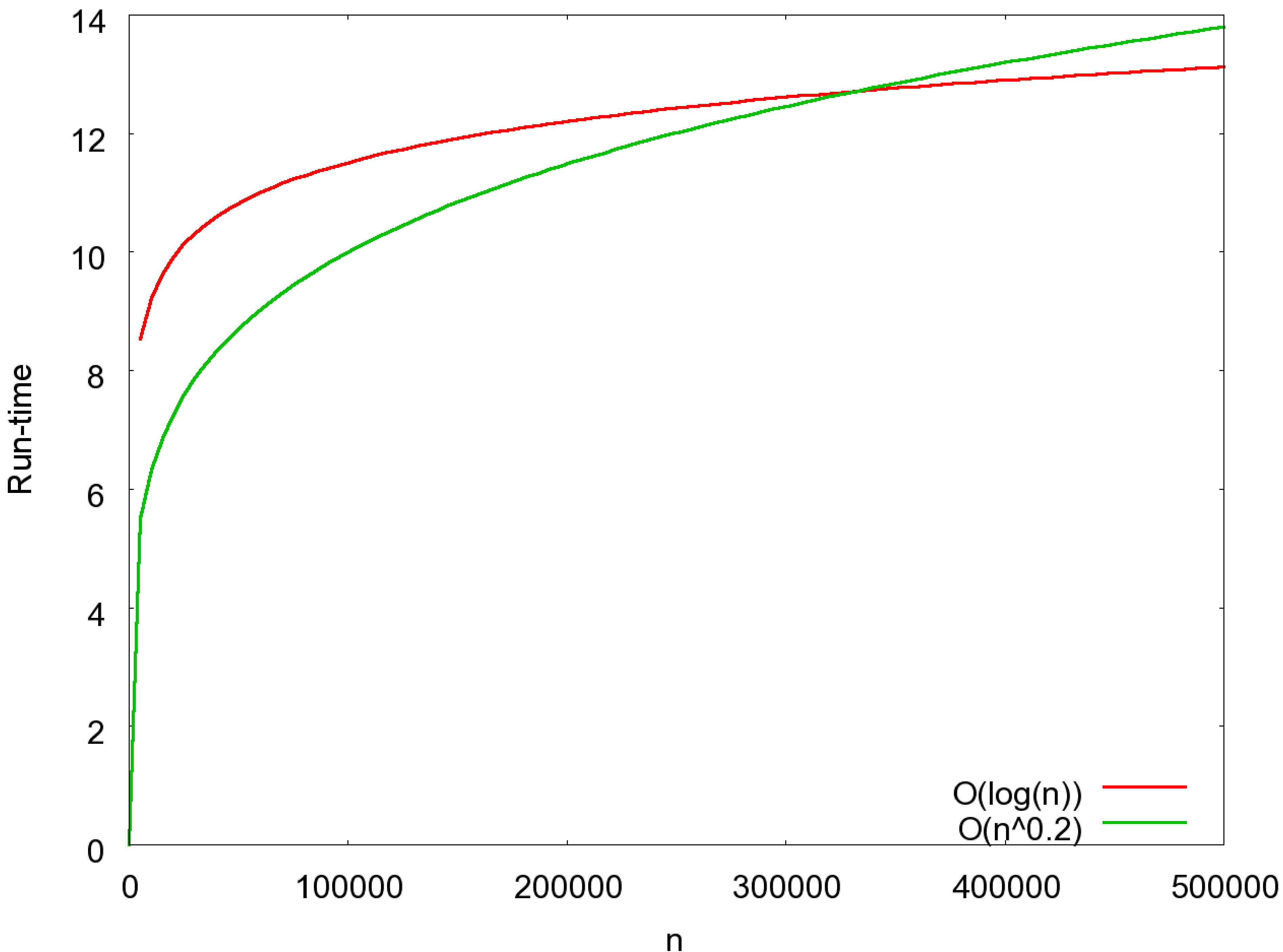
Q. For every $x > 0$, log n is $O(n^x)$? (or n^x is $O(\log n)$?)

A.

Q. For every $r > 1$ and every $d > 0$ is r^n is $O(n^d)$? (or n^d is $O(r^n)$?)

A.

TU Delft



Run-time
n
0
2
4
6
8
10
12
14
0
100000
200000
300000
400000
500000
O(log(n))
O(n^0.2)

Asymptotic Bounds for Some Common Functions

Polynomial. $a_0 + a_1 n + \ldots + a_d n^d$ is $\Theta($ f(n)) if $a_d > 0$.

Q. What is the simplest f(n)?

A. $f(n) = n^d$

Polynomial time. Running time is $O(n^d)$ for some constant d independent of the input size n.

Q. For every $x > 0$, log n is $O(n^x)$? (or n^x is O(log n) ?)

A. Yes. log grows slower than *every* polynomial.

Q. For every $r > 1$ and every $d > 0$ is r^n is $O(n^d)$? (or n^d is $O(r^n)$?)

A.

TU Delft

Asymptotic Bounds for Some Common Functions

Polynomial. $a_0 + a_1 n + \ldots + a_d n^d$ is $\Theta($ f(n)) if $a_d > 0$.

Q. What is the simplest f(n)?

A. $f(n) = n^d$

Polynomial time. Running time is $O(n^d)$ for some constant d independent of the input size n.

Q. For every $x > 0$, log n is $O(n^x)$? (or n^x is O(log n) ?)

A. Yes. log grows slower than *every* polynomial.

Q. For every $r > 1$ and every $d > 0$ is r^n is $O(n^d)$? (or n^d is $O(r^n)$?)

A. No. Every exponential grows faster than every polynomial.

Logarithms. $O(\log_a n) = O(\log_b n)$ for any constants a, b > 0.

TU Delft