Discrete Structures & Algorithms

Asymptotic complexity
and some sequences & summations

EECE 320 // UBC

Asymptotic complexity

- Running time of an algorithm as a function of input size n **for large n**.
- Expressed using only the **highest-order term** in the expression for the exact running time.
  - Instead of exact running time, say $\Theta(n^2)$.
- Describes behavior of function in the limit.
- Written using ***asymptotic notation.***

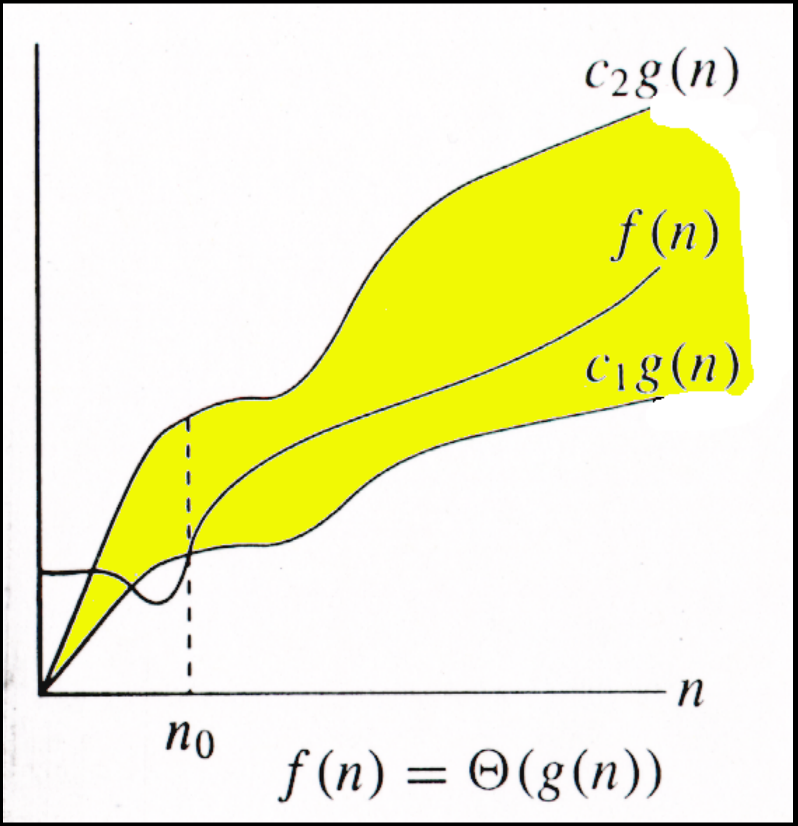
Asymptotic notation

- Θ, O, Ω, o, ω
- Defined for functions over the natural numbers.
  - **Example:** $f(n) = \Theta(n^2)$.
  - Describes how $f(n)$ grows in comparison to n^2.
- Define a **set** of functions; in practice used to compare two function sizes.
- The notations describe different rate-of-growth relations between the defining function and the defined set of functions.

Θ-notation

For function g(n), we define Θ(g(n)), big-Theta of n, as the set:

Θ(g(n)) = {f(n) :
∃ positive constants c_1, c_2, and n_0,
such that $\forall n \geq n_0$,
we have $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$
}



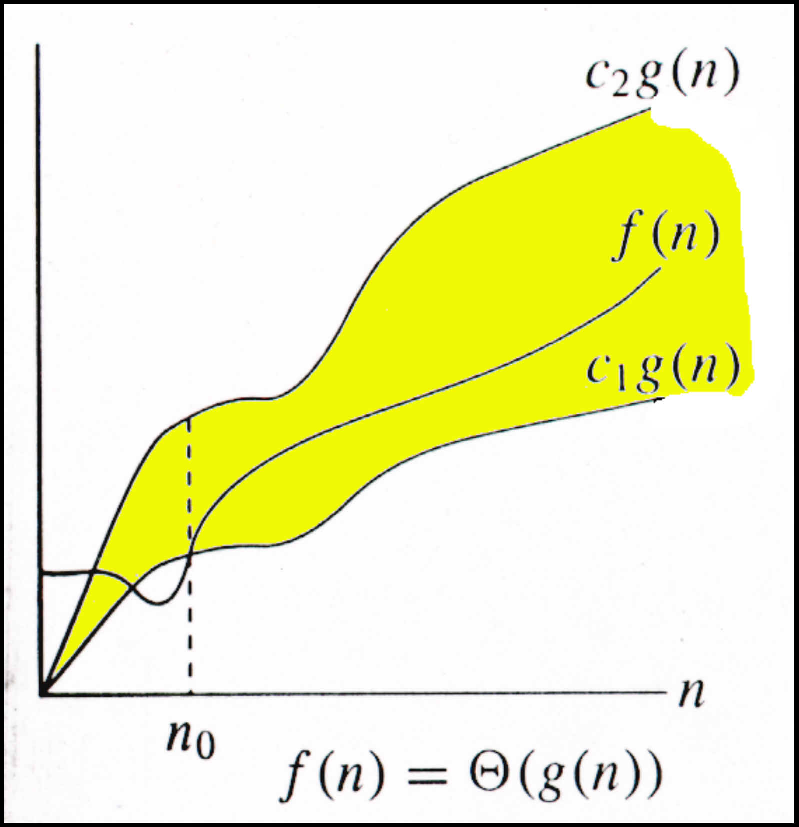
Intuitively: Set of all functions thathave the same *rate of growth* as g(n).

g(n) is an asymptotically tight bound for f(n).

Θ-notation

For function g(n), we define Θ(g(n)), big-Theta of n, as the set:

Θ(g(n)) = {f(n) :
∃ positive constants c_1, c_2, and n_0,
such that $\forall n \geq n_0$,
we have $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$
}



Technically, f(n) ∈ Θ(g(n)).
Older usage, f(n) = Θ(g(n)).
I'll accept either…

f(n) and g(n) are nonnegative, for large n.

Example

$\Theta(g(n)) = \{f(n) : \exists$ positive constants c_1, c_2, and n_0, such that $\forall n \geq n_0$, $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$

- $10n^2 - 3n = \Theta(n^2)$
- What constants for n_0, c_1, and c_2 will work?
- Make c_1 a little smaller than the leading coefficient, and c_2 a little bigger.
- **To compare orders of growth, look at the leading term.**
- Exercise: Prove that $n^2/2-3n = \Theta(n^2)$

Example

$\Theta(g(n)) = \{f(n) : \exists$ positive constants c_1, c_2, and n_0, such that $\forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$

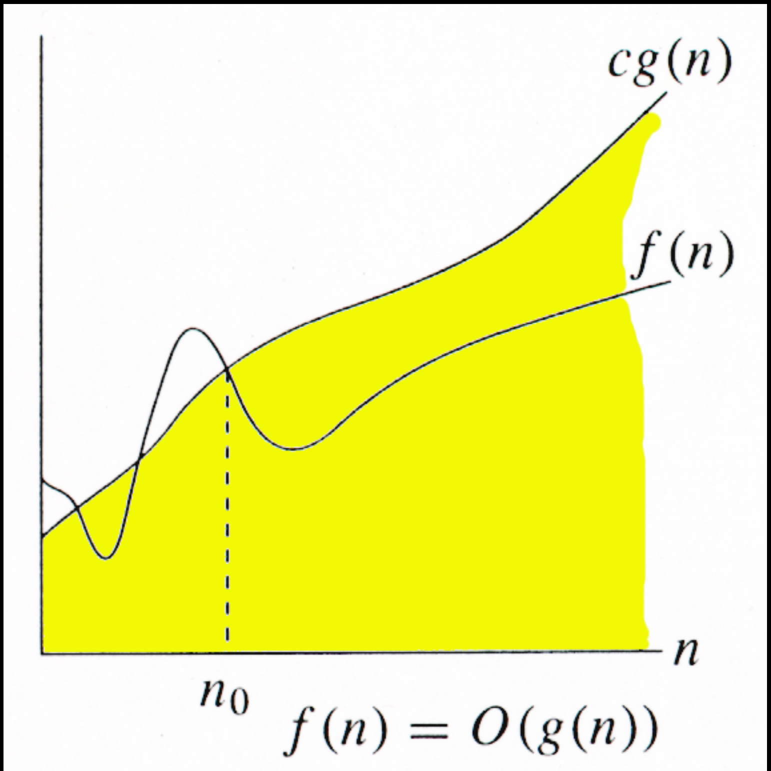
- Is $3n^3 \in \Theta(n^4)$?
- How about $2^{2n} \in \Theta(2^n)$?

O-notation

$f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$.
$\Theta(g(n)) \subset O(g(n))$.

For function $g(n)$, we define $O(g(n))$, big-O of n, as the set:

$O(g(n)) = \{f(n):$
$\exists$ positive constants c and n_0, such that $\forall n \geq n_0$,
we have $0 \leq f(n) \leq cg(n)$ $\}$



Intuitively: Set of all functions whose *rate of growth* is the same as or lower than that of $g(n)$.

g(n) is an asymptotic upper bound for *f*(*n*).

Examples

$O(g(n)) = \{f(n) : \exists$ positive constants c and n_0, such that $\forall n \geq n_0$, we have $0 \leq f(n) \leq cg(n)$ \}

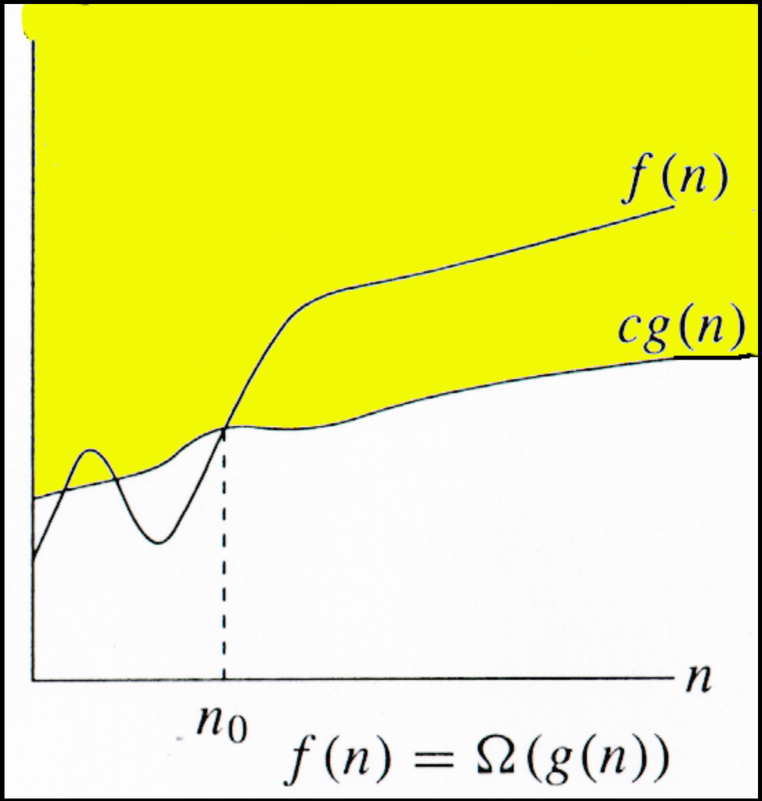
- Any linear *function* $an + b$ is in $O(n^2)$. **How?**
- Show that $3n^3 = O(n^4)$ for appropriate c and n_0.

Ω -notation

$f(n) = \Theta(g(n)) \Rightarrow f(n) = \Omega(g(n))$.
$\Theta(g(n)) \subset \Omega(g(n))$.

For function g(n), we define $\Omega(g(n))$, big-Omega of n, as the set:

$\Omega(g(n))$ = {$f(n)$:
$\exists$ positive constants c and n_0, such that $\forall n \geq n_0$,
we have $0 \leq cg(n) \leq f(n)$}



Intuitively: Set of all functions whose *rate of growth* is the same as or higher than that of g(n).

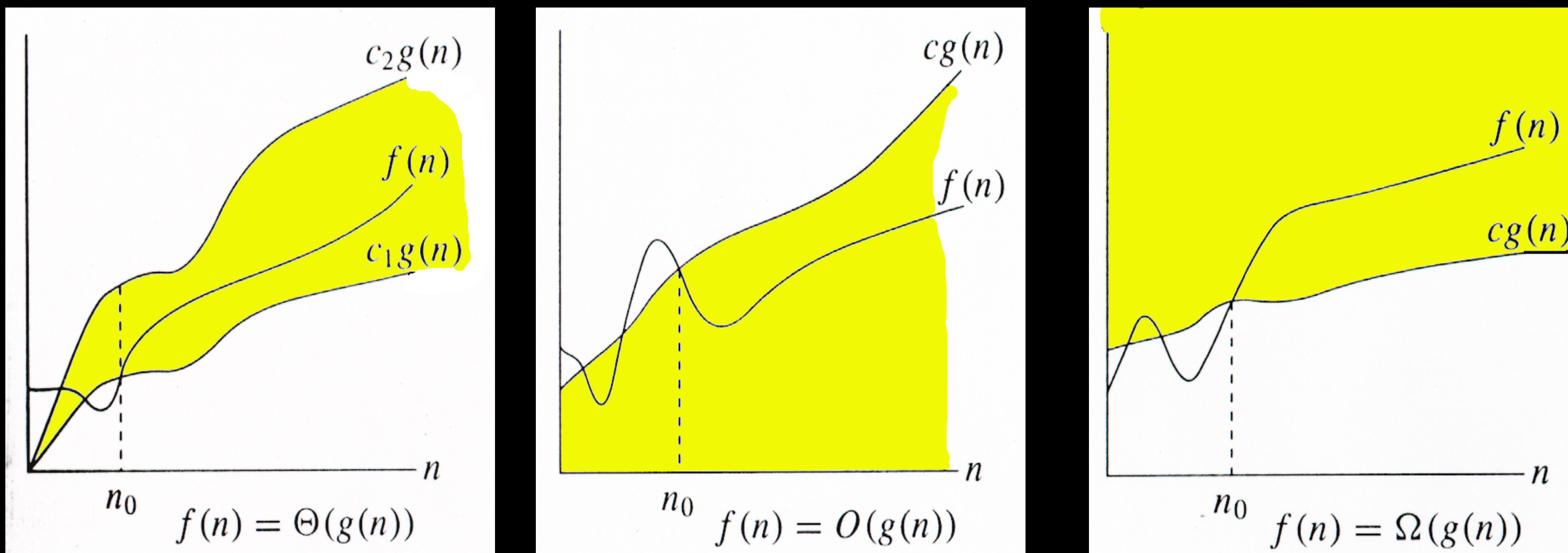
g(n) is an asymptotic lower bound for f(n).

Example

$\Omega(g(n)) = \{f(n) : \exists$ positive constants c and n_0, such that $\forall n \geq n_0$, we have $0 \leq cg(n) \leq f(n)\}$

- $\sqrt{n} = \Omega(\lg n)$. Choose c and n_0.

Relations between Θ, *O*, Ω



Relations between Θ, Ω, *O*

Theorem: For any two functions g(n) and f(n),
$f(n) = \Theta(g(n))$ iff
$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

- $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$
- In practice, asymptotically tight bounds are obtained from asymptotic upper and lower bounds.

Running times

- "Running time is $O(f(n))$" $\Rightarrow$ Worst case is $O(f(n))$
- $O(f(n))$ bound on the worst-case running time $\Rightarrow$ $O(f(n))$ bound on the running time of every input.
- $\Theta(f(n))$ bound on the worst-case running time $\not\Rightarrow$ $\Theta(f(n))$ bound on the running time of every input.
- "Running time is $\Omega(f(n))$" $\Rightarrow$ Best case is $\Omega(f(n))$
- Can still say "Worst-case running time is $\Omega(f(n))$"
  - Means worst-case running time is given by some unspecified function $g(n) \in \Omega(f(n))$.

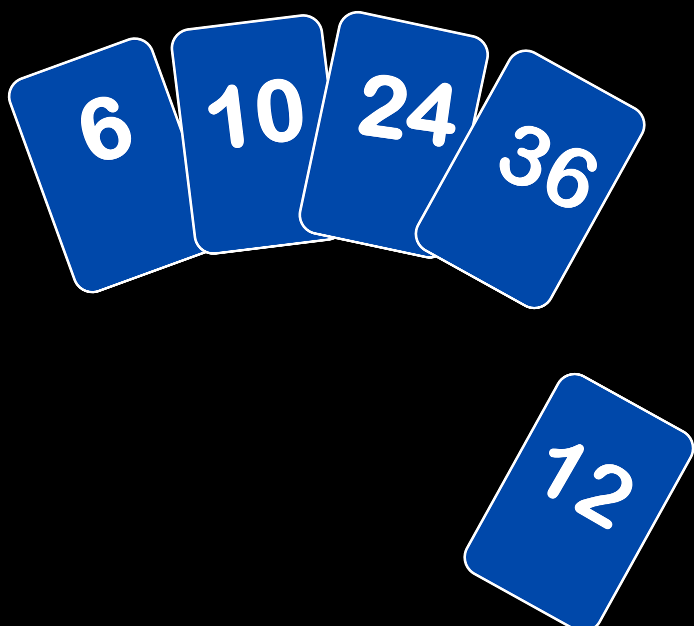
Example

- **Insertion sort** takes $\Theta(n^2)$ in the worst case, so sorting (as a *problem*) is $O(n^2)$. **Why?**

- Any sort algorithm must look at each item, so sorting is $\Omega(n)$.

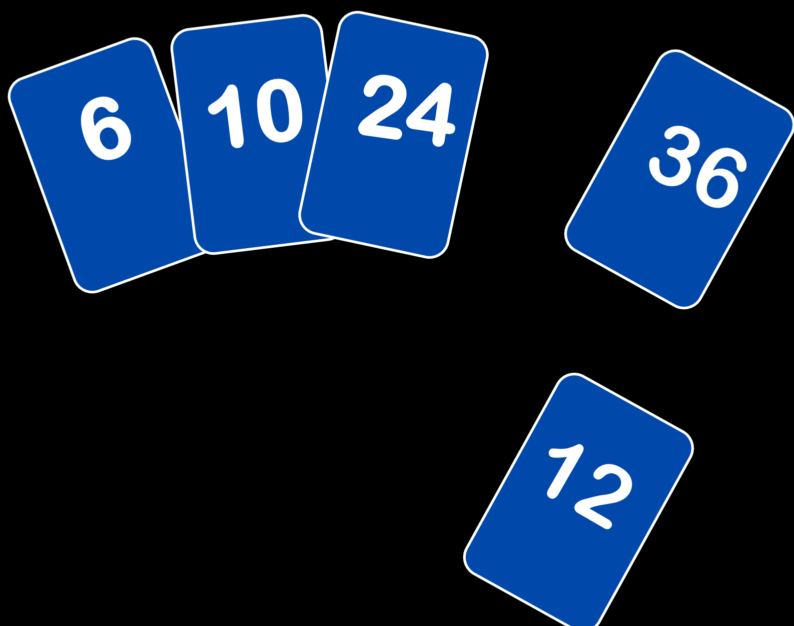
- In fact, using (e.g.) merge sort, sorting is $\Theta(n \lg n)$ in the worst case.
  - No comparison sort to do better in the worst case. [We may not see a proof of this result in this course.]

Insertion sort

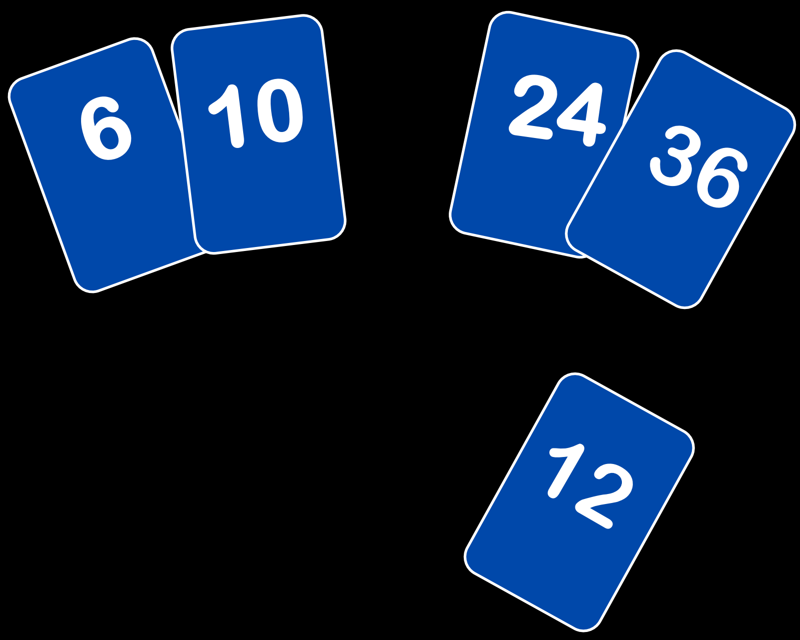


To insert 12, we need to make room for it by moving first 36 and then 24.

Insertion sort



Insertion sort



Insertion sort

input array

5 2 4 6 1 3

At each iteration, the array is divided in two sub-arrays:

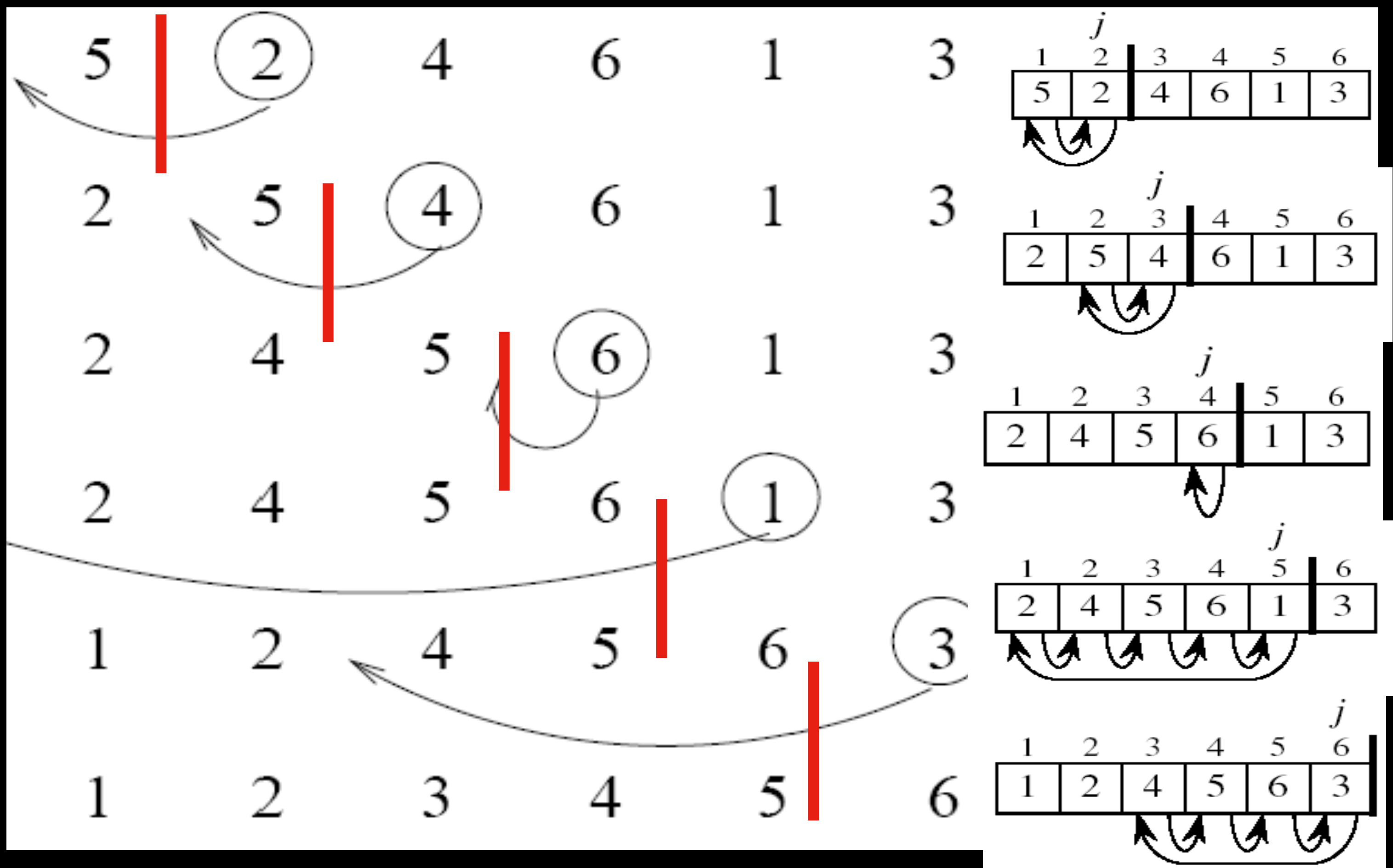
left sub-array

right sub-array

sorted

unsorted

Insertion Sort



Insertion sort

Algorithm: INSERTION-SORT*(A)*

```
for j ← 2 to n
      do key ← A[ j ]
         Comment: Insert A[ j ] into the sorted sequence A[1 . . j -1]
         i ← j - 1
         while i > 0 and A[i] > key
               do A[i + 1] ← A[i]
                  i ← i – 1
         A[i + 1] ← key
```

1	2	3	4	5	6	7	8
a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8

key

Asymptotic notation in equations

- Can use asymptotic notation in equations to replace expressions containing lower-order terms.
- For example,

  $4n^3 + 3n^2 + 2n + 1 = 4n^3 + 3n^2 + \Theta(n)$

  $= 4n^3 + \Theta(n^2) = \Theta(n^3)$. **How do we interpret this?**
- In equations, $\Theta(f(n))$ always stands for an **anonymous function** $g(n) \in \Theta(f(n))$
  - In the example above, $\Theta(n^2)$ stands for $3n^2 + 2n + 1$.

o-notation

For a given function g(n), the set little-*o*:

$o(g(n)) = \{f(n): \forall\ c > 0, \exists\ n_0 > 0$ such that
$\forall\ n \geq n_0$, we have $0 \leq f(n) < cg(n)\}$.

f(n) becomes insignificant relative to g(n) as n approaches infinity:

$$\lim_{n \to \infty} [f(n) / g(n)] = 0.$$

g(n) is an **upper bound** for f(n) that is not asymptotically tight.

Observe the difference in this definition from previous ones. **Why?**

ω-notation

For a given function g(n), the set little-omega:

$\omega(g(n)) = \{f(n): \forall\, c > 0, \exists\, n_0 > 0$ such that $\forall\, n \geq n_0$, we have $0 \leq cg(n) < f(n)\}$.

f(n) becomes arbitrarily large relative to g(n) as *n* approaches infinity:

$$\lim_{n \to \infty} [f(n) / g(n)] = \infty.$$

g(n) is a **lower bound** for *f*(*n*) that is not asymptotically tight.

Limits

- $\lim_{n\to\infty} [f(n) / g(n)] = 0 \Rightarrow f(n) \in o(g(n))$
- $\lim_{n\to\infty} [f(n) / g(n)] < \infty \Rightarrow f(n) \in O(g(n))$
- $0 < \lim_{n\to\infty} [f(n) / g(n)] < \infty \Rightarrow f(n) \in \Theta(g(n))$
- $0 < \lim_{n\to\infty} [f(n) / g(n)] \Rightarrow f(n) \in \Omega(g(n))$
- $\lim_{n\to\infty} [f(n) / g(n)] = \infty \Rightarrow f(n) \in \omega(g(n))$
- $\lim_{n\to\infty} [f(n) / g(n)]$ undefined $\Rightarrow$ can not say

Properties

- **Transitivity**
  - $f(n) = \Theta(g(n))$ & $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$
  - $f(n) = O(g(n))$ & $g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$
  - $f(n) = \Omega(g(n))$ & $g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$
  - $f(n) = o(g(n))$ & $g(n) = o(h(n)) \Rightarrow f(n) = o(h(n))$
  - $f(n) = \omega(g(n))$ & $g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$

- **Reflexivity**
  - $f(n) = \Theta(f(n))$
  - $f(n) = O(f(n))$
  - $f(n) = \Omega(f(n))$

Properties

- **Symmetry**

  $f(n) = \Theta(g(n))$ iff $g(n) = \Theta(f(n))$

- **Complementarity**

  $f(n) = O(g(n))$ iff $g(n) = \Omega(f(n))$

  $f(n) = o(g(n))$ iff $g(n) = \omega((f(n))$

Common functions

Monotonicity

- $f(n)$ is
  - **monotonically increasing** if $m \leq n \Rightarrow f(m) \leq f(n)$.
  - **monotonically decreasing** if $m \geq n \Rightarrow f(m) \geq f(n)$.
  - **strictly increasing** if $m < n \Rightarrow f(m) < f(n)$.
  - **strictly decreasing** if $m > n \Rightarrow f(m) > f(n)$.

Exponentials

- **Useful (and elementary) Identities:**

$$a^{-1} = \frac{1}{a}$$

$$(a^m)^n = a^{mn}$$

$$a^m a^n = a^{m+n}$$

- **Exponentials and polynomials**

$$\lim_{n \to \infty} \frac{n^b}{a^n} = 0$$

$$\Rightarrow n^b = o(a^n)$$

Logarithms

$x = \log_b a$ is the exponent for $a = b^x$.

Natural log: $\ln a = \log_e a$

Binary log: $\lg a = \log_2 a$

$\lg^2 a = (\lg a)^2$

$\lg \lg a = \lg (\lg a)$

$$a = b^{\log_b a}$$

$$\log_c (ab) = \log_c a + \log_c b$$

$$\log_b a^n = n \log_b a$$

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$\log_b (1/a) = -\log_b a$$

$$\log_b a = \frac{1}{\log_a b}$$

$$a^{\log_b c} = c^{\log_b a}$$

Logarithms and exponentials – Bases

- If the base of a logarithm is changed from one constant to another, the value is altered by a constant factor.
  - **Example:** $\log_{10} n * \log_2 10 = \log_2 n$.
  - Base of logarithm is not an issue in asymptotic notation.
- Exponentials with different bases differ by a exponential factor (not a constant factor).
  - **Example:** $2^n = (2/3)^n * 3^n$.

Polylogarithms

- **For $a \geq 0$, $b > 0$,** $\lim_{n\to\infty} (\lg^a n \,/\, n^b) = 0$, so $\lg^a n = o(n^b)$, and $n^b = \omega(\lg^a n)$
  - Prove using L'Hospital's rule repeatedly

- $\lg(n!) = \Theta(n \lg n)$
  - Prove using Stirling's approximation (in the text) for $\lg(n!)$.

Exercise

Express functions in A in asymptotic notation using functions in B.

A	B	
$5n^2 + 100n$	$3n^2 + 2$	$A \in \Theta(B)$

$A \in \Theta(n^2)$, $n^2 \in \Theta(B) \Rightarrow A \in \Theta(B)$

$\log_3(n^2)$	$\log_2(n^3)$	$A \in \Theta(B)$

$\log_b a = \log_c a / \log_c b$; $A = 2\lg n / \lg 3$, $B = 3\lg n$, $A/B = 2/(3\lg 3)$

$n^{\lg 4}$	$3^{\lg n}$	$A \in \omega(B)$

$a^{\log b} = b^{\log a}$; $B = 3^{\lg n} = n^{\lg 3}$; $A/B = n^{\lg(4/3)} \to \infty$ as $n \to \infty$

$\lg^2 n$	$n^{1/2}$	$A \in o(B)$

$\lim_{n\to\infty} (\lg^a n / n^b) = 0$ (here $a = 2$ and $b = 1/2$) $\Rightarrow A \in o(B)$

Summations

Sequences

- Sequence: an ordered list of elements
  - Like a set, but:
    - Elements can be duplicated.
    - Elements are ordered.

Sequences

- A sequence is a function from a subset of Z to a set S.
  - Usually from the positive or non-negative integers.
  - a_n is the image of n.

- a_n is a term in the sequence.
- $\{a_n\}$ means the entire sequence.
  - The same notation as sets!

Example sequences

- $a_n = 3n$
  - The terms in the sequence are $a_1, a_2, a_3, \ldots$
  - The sequence $\{a_n\}$ is $\{ 3, 6, 9, 12, \ldots \}$

- $b_n = 2^n$
  - The terms in the sequence are $b_1, b_2, b_3, \ldots$
  - The sequence $\{b_n\}$ is $\{ 2, 4, 8, 16, 32, \ldots \}$

- Note that these sequences are indexed from 1
  - Not always, though! You need to pay attention to the start of a sequence.

Summations

- Why do we need summation formulae?

  For computing the running times of iterative constructs (is a simple explanation).

  Example: Maximum Subvector

  Given an array $A[1\ldots n]$ of numeric values (can be positive, zero, and negative) determine the subvector $A[i\ldots j]$ ($1 \leq i \leq j \leq n$) whose sum of elements is maximum over all subvectors.

1	-2	2	2

Maximum Subvector

```
MaxSubvector(A, n)
        maxsum ← 0;
        for i ← 1 to n
          do for j = i to n
                  sum ← 0
                  for k ← i to j
                       do sum += A[k]
                  maxsum ← max(sum, maxsum)
        return maxsum
```

- T(n) = $\sum_{i=1}^{n} \sum_{j=i}^{n} \sum_{k=i}^{j} 1$
- NOTE: This is not a simplified solution. What *is* the final answer?

Summations

How do you know this is true?

$$\sum_{i=1}^{k}\left(ca_i + b_i\right) = c\sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i$$

Use associativity to separate the *b*s from the *a*s.

Use distributivity to factor the *c*s.

Summations you should know

What is S = 1 + 2 + 3 + … + n?

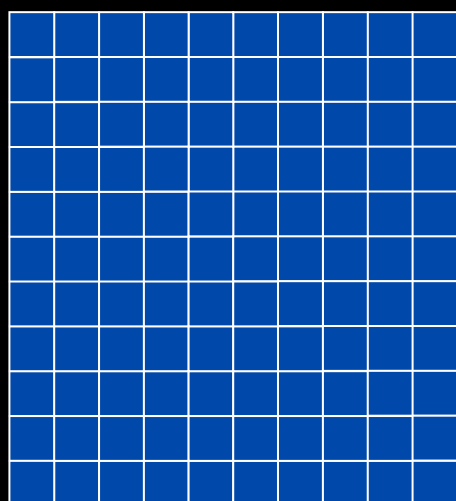
S	=	1	+	2	+	…	+	n	Write the sum.
S	=	n	+	n-1	+	…	+	1	Write it again.
2S	=	n+1	+	n+1	+	…	+	n+1	Add together.

You get n copies of (n+1). But we've over added by a factor of 2. So just divide by 2.

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

Summations example/picture

$$\sum_{i=1}^{10} i = 1 + 2 + 3 + \ldots + 10$$



$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{10} i = \frac{10 \times 11}{2} = 55$$

$$\sum_{i=1}^{100} i = \frac{100 \times 101}{2} = 5050$$

We now have a square 10 (n) by 11 (n+1) with area 110 units

We need half of that (10x11)/2

Summations you should know

What is S = 1 + 3 + 5 + … + (2n - 1)?

Sum of first n odds.

$$\sum_{k=1}^{n}(2k-1) = 2\sum_{k=1}^{n}k - \sum_{k=1}^{n}1$$

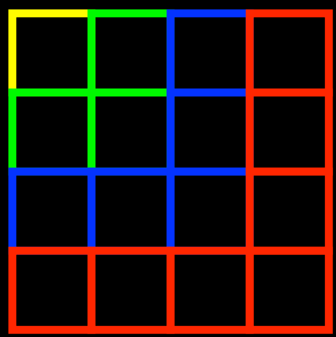
$$= 2\left(\frac{n(n+1)}{2}\right) - n$$

$$= n^2$$

Summations you should know

What is S = 1 + 3 + 5 + … + (2n - 1)?

Sum of first n odds.



Summations you should know

What is $S = 1 + r + r^2 + \ldots + r^n$

Geometric Series

$$\sum_{k=0}^{n} r^k = 1 + r + \ldots + r^n$$

Multiply by r

$$r\sum_{k=0}^{n} r^k = r + r^2 + \ldots + r^{n+1}$$

Subtract 2nd from 1st

$$\sum_{k=0}^{n} r^k - r\sum_{k=0}^{n} r^k = 1 - r^{n+1}$$

factor

$$(1-r)\sum_{k=0}^{n} r^k = 1 - r^{n+1}$$

divide

$$\sum_{k=0}^{n} r^k = \frac{1 - r^{n+1}}{(1-r)}$$

DONE!

Summations you should know

What about:

$$\sum_{k=0}^{\infty} r^k = 1 + r + \ldots + r^n + \ldots$$

If r ≥ 1 this blows up.

If r < 1 we can say something.

$$\sum_{k=0}^{\infty} r^k = \lim_{n \to \infty} \sum_{k=0}^{n} r^k$$

$$= \lim_{n \to \infty} \frac{1 - r^{n+1}}{(1 - r)} \qquad = \frac{1}{(1 - r)}$$

In-class exercise

- Find an expression for the following summation.
  - $S = (1x2) + (2x3) + (3x4) + \ldots + n(n+1) = ?$
  - Hint: Consider $(n+1)^3 - n^3$.

In-class exercise

$$S = 1 \cdot \binom{n}{0} + 2 \cdot \binom{n}{1} + 3 \cdot \binom{n}{2} + ... + (n+1)\binom{n}{n} = ?$$

Consider the binomial series expansion, and ponder what happens when you differentiate both sides…

$$(1+x)^n = \sum_{i=1}^{n} \binom{n}{i} x^i$$

Important summations and techniques

- **Constant Series:** For integers *a* and *b*, $a \le b$,

$$\sum_{i=a}^{b} 1 = b - a + 1$$

- **Linear Series (Arithmetic Series):** For $n \ge 0$,

$$\sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

- **Quadratic Series:** For $n \ge 0$,

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

Important summations and techniques

- **Cubic Series:** **For $n \geq 0$,**

$$\sum_{i=1}^{n} i^3 = 1^3 + 2^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$$

- **Geometric Series:** **For real $x \neq 1$,**

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n = \frac{x^{n+1} - 1}{x - 1}$$

For |x| < 1,

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}$$

Important summations and techniques

- **Linear-Geometric Series:** For $n \geq 0$, real $c \neq 1$,

$$\sum_{i=1}^{n} ic^{i} = c + 2c^{2} + \cdots + nc^{n} = \frac{-(n+1)c^{n+1} + nc^{n+2} + c}{(c-1)^{2}}$$

- **Harmonic Series:** n^{th} harmonic number, $n \in I^{+}$,

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

$$= \sum_{k=1}^{n} \frac{1}{k} = \ln(n) + O(1)$$

Important summations and techniques

- **Telescoping Series:**

$$\sum_{k=1}^{n} a_k - a_{k-1} = a_n - a_0$$

- **Differentiating Series:** For |x| < 1,

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

Important summations and techniques

- **Approximation by integrals:**
  - For monotonically increasing f(n)

$$\int_{m-1}^{n} f(x)dx \le \sum_{k=m}^{n} f(k) \le \int_{m}^{n+1} f(x)dx$$

  - For monotonically decreasing f(n)

$$\int_{m}^{n+1} f(x)dx \le \sum_{k=m}^{n} f(k) \le \int_{m-1}^{n} f(x)dx$$

- **How?**

Important summations and techniques

- ***n*th harmonic number**

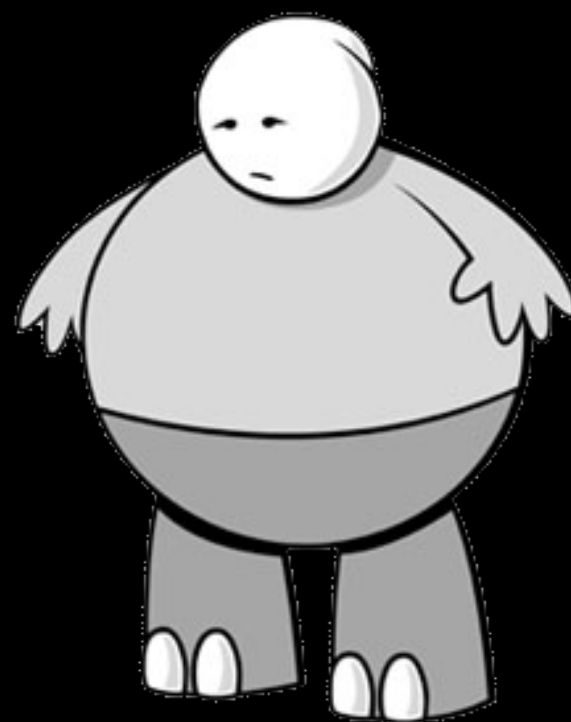
$$\sum_{k=1}^{n} \frac{1}{k} \geq \int_{1}^{n+1} \frac{dx}{x} = \ln(n+1)$$

$$\sum_{k=2}^{n} \frac{1}{k} \leq \int_{1}^{n} \frac{dx}{x} = \ln n$$

$$\Rightarrow \sum_{k=1}^{n} \frac{1}{k} \leq \ln n + 1$$

Wrap-up

- What are the different asymptotic bounds on functions?
- How are the asymptotic bounds related?
- Asymptotic bounds and algorithmic efficiency
- Summations
  - Basic summations (formulae)
  - Tricks for certain series
    - Telescoping
    - Differentiation
    - …



You should never forget the definitions for the Θ, O, Ω, o, ω notations. They help us analyze algorithms.