This week, the lecture was only 50 minutes long.

These are *rough notes* based on my own notes I prepared for the Week 7 lecture. They go with the annotated lecture slides you can see on the "More" page of the course website. Although they are rough, I'm providing them here in case they help you remember what we did in lecture.

1 Legend

Text that was already on the slides is in a box, like this.
Descriptions of things that were already on the slides are formatted like this.

> Text I planned to write during lecture is formatted like this, with a line to the left.
>
> *Descriptions of other things I planned to draw or write or do are formatted like this.*

Anything else is text I planned to say, or things I planned to ask.

2 Big Oh notation

2.1 Definition of Big Oh

Big O Notation
Let $\mathcal{F}$ be the set of all functions from $\mathbb{N}$ to $\mathbb{R}^+$.
For any $f \in \mathcal{F}$, let

$$O(f) = \{g \in \mathcal{F} \mid \exists c \in \mathbb{R}^+.\ \exists b \in \mathbb{N}.\ \forall n \in \mathbb{N}.\ (n \geq b \text{ IMPLIES } g(n) \leq cf(n))\}$$

In other words:

> $f \in O(g)$ means for all sufficiently large n, $g(n)$ is at most a constant factor times $f(n)$.
>
> *(Draw example: graph with $3n$ vs. n^2. Draw a line where we can put b. We can take $c = 1$.)*

2.2 Properties of Big Oh

Properties of big O
Summary on "Further reading" page of course website.

- Constant factors don't matter:
  If $d > 0$ is a constant, then $df(n) \in O(f(n))$ and $f(n) \in O(df(n))$.

$5n^2 \in O(n^2)$

- Low order terms don't matter:
  If $\lim_{n\to\infty} \frac{h(n)}{g(n)} = 0$ then $g(n) + h(n) \in O(g(n))$.

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

- Transitivity:
  If $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$ then $f(n) \in O(h(n))$
- Summation rules:
  If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$ then $f_1(n)+f_2(n) \in O(g_1(n)+g_2(n))$.
  $\max\{f(n), g(n)\} \in O(f(n) + g(n))$
  $f(n) + g(n) \in O(\max\{f(n), g(n)\})$
- Product rule:
  If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$ then $f_1(n)f_2(n) \in O(g_1(n)g_2(n))$

$3 \cdot (2^n + 1) \cdot (n^3 + n + 2) \in O(2^n \cdot n^3)$

Exercise: prove some of these.

Exponents and logarithms

- Exponents matter:
  If $a \leq b$ then $n^a \in O(n^b)$.
  If $a > b$ then $n^a \notin O(n^b)$.

$n^2 \in O(n^{2.5})$

$n^3 \notin O(n)$

- Bases of exponents matter:
  If $1 < a \leq b$ then $a^n \in O(b^n)$.
  If $1 < b < a$ then $a^n \notin O(b^n)$.

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

$3^n \notin O(2^n)$

- Bases of logarithms don't matter:
  For all $a, b > 1$, $\log_a(n) \in O(\log_b(n))$

$\log_2(n) \in O(\log_{10}(n)$

- Exponential functions grow faster than polynomial functions:
  $\forall b > 1$ and $a \in \mathbb{R}$, $n^a \in O(b^n)$ but $b^n \notin O(n^a)$.

$n^{100} \in O(2^n)$

$2^n \notin O(n^{100})$

- Polynomial functions grow faster than polylogarithmic functions:
  For all $a, b > 0$, $(\log n)^a \in O(n^b)$ but $n^b \notin O((\log n)^a)$.

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

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

2.3 Definition of big Ω and big Θ

For any $f \in \mathcal{F}$,

$$O(f) = \{g \in \mathcal{F} \mid \exists c \in \mathbb{R}^+. \exists b \in \mathbb{N}. \forall n \in \mathbb{N}. (n \geq b \text{ IMPLIES } g(n) \leq cf(n))\}$$

$$\Omega(f) = \{g \in \mathcal{F} \mid \exists c \in \mathbb{R}^+. \exists b \in \mathbb{N}. \forall n \in \mathbb{N}. (n \geq b \text{ IMPLIES } g(n) \geq cf(n))\}$$

$$\begin{aligned}\Theta(f) =& O(f) \cap \Omega(f) \\ =& \{g \in \mathcal{F} \mid \exists c_1 \in \mathbb{R}^+. \exists c_2 \in \mathbb{R}^+. \exists b \in \mathbb{N}. \forall n \in \mathbb{N}. \\ & (n \geq b \text{ IMPLIES } c_1 f(n) \leq g(n) \leq c_2 f(n))\}\end{aligned}$$

3 Solving recurrences

Solving recurrences
Define $T : \mathbb{N} \to \mathbb{N}$ by:
- $T(0) = 3$
- For $n \in \mathbb{N}$, $T(n+1) = 2T(n) + 1$

Find a **closed form** for $T(n)$.

(Arrow to the equations.) Recurrence relation

(Arrow to "closed form".) Simple expression with no recursion.

n	$T(n)$
0	3
1	7
2	15
3	31
4	63

Guess: $T(n) = 4 \cdot 2^n - 1$

Verify: prove it (usually by induction)

Solving a recurrence means finding a closed form: writing the same function as a simple expression with no recursion.
Methods for solving recurrences:

1. Guess and verify
2. Repeated substitution and verify
3. Special techniques for "divide and conquer" recurrences / Master Theorem
4. Transformations
5. Solving linear recurrences using characteristic polynomials

We just did guess and verify.

Repeated substitution and verify
Define $M : \mathbb{Z}^+ \to \mathbb{N}$ by

- $M(1) = c$
- For $n > 1$, $M(n) = M(\lceil n/2 \rceil) + M(\lfloor n/2 \rfloor) + dn$

where $c, d \in \mathbb{N}$ are constants.

Assume n is a power of 2.

$M(n) = 2M(n/2) + dn$

$$\begin{aligned} M(n) =& 2M(n/2) + dn \\ =& 2[2M(n/2^2) + dn/2] + dn \\ =& 2[2[2M(n/2^3) + dn/2^2] + dn/2] + dn \\ \vdots& \text{ find the pattern} \\ =& 2^k M(n/2^k) + kdn \end{aligned}$$

When $k = \log_2 n$, get $M(n) = nM(1) + dn \log_2 n = cn + dn \log_2 n$.

Now, prove this is correct using induction. For $k \in \mathbb{N}$, let $P(k) =$ "$M(2^k) = c2^k + dk2^k$".

What if n not a power of 2?

(Repeat definition of M.)
Theorem 1. $M(n) \in O(n \log n)$

We ended the lecture here. (I explained that if we ignore n that are not powers of 2, the theorem is not hard: we have already proved $M(n) = nc + dn \log n$ in that case, and we can use properties of big O.)