# ECS20 takeaways

## I. Propositions, implications, propositional logic

- Proposition is a declarative statement w/ T, F value.
- Predicates $P(x)$ ← bind the predicate, $\exists_x, \forall_x, \ldots$ to become a proposition $\rightarrow \{T, F\}$.
- Implications $p \rightarrow q$
- Contrapositive $\neg q \rightarrow \neg p$
- Leads to – modus ponens
  – modus tollens.
- De Morgan's Laws (1) $\neg(p \wedge q) \equiv \neg p \vee \neg q$
  (2) $\neg(p \vee q) \equiv \neg p \wedge \neg q$
- Logical operators, $\wedge$, $\vee$, $\neg$, $\oplus$, etc.
- Existential quantifiers, $\forall_x$, $\exists_x$, $\neg\exists_x$, etc.

## II. Basic methods of proof

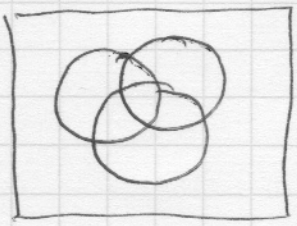
- Direct $p \rightarrow q$
- Contrapositive $\neg q \rightarrow \neg p$
- Contradiction
  – propositions $\neg p \rightarrow (r \wedge \neg r)$
  – implications $p \wedge \neg Q \rightarrow$ contradiction
- Constructive existence
- Nonconstructive existence

## III. Sets: Collections of objects, order does not matter, no repeats of elements

Set builder notation

$$A = \{ x \mid P(x) \}$$

Venn diagrams



$\mathcal{U}$

$\emptyset = \{\}$ distinct from $\{\emptyset\} = \{\{\}\}$

$\emptyset \subset A$ (empty set is a subset of any set)

Power set A, the set of all subsets

$$|P(A)| = 2^{|A|}$$

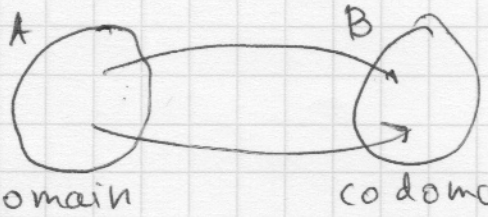
De Morgan's Law

$$\overline{A \cup B} \equiv \overline{A} \cap \overline{B}$$

$$\overline{A \cap B} \equiv \overline{A} \cup \overline{B}$$

## IV. Functions $f: A \to B$

- Must map every element in A to an element in B. (not unique necessarily)




- One-to-one (injective); Each a maps to a unique b.
- Onto (surjective); Every element in b is mapped to.
- Bijection (inverse fun'c exist); onto + 1-to-1.

# V. Sequences & summations.

- A sequence is an ordered list

$$\{a_0, a_1, a_2, \ldots a_m\}$$

- arithmetic series $a_j = a_0 + jd$ where $a_0, d$ are given constants.
- geometric sequence $a_j = a_0 r^j$ where $a_0, r$ are given constants

- Summations.

$$\sum_{j=0}^{n} a_j = \sum_{k=0}^{n} a_k = a_0 + \sum_{k=1}^{n} a_k$$

$$= a_0 + \sum_{k=1}^{n-1} a_k + a_n \quad , \text{ etc.}$$

Also can renormalize indices, e.g.

Let $k = j+1$ then $\sum_{j=0}^{n} a_j = \sum_{k=1}^{n-1} a_k$

Specific sums:

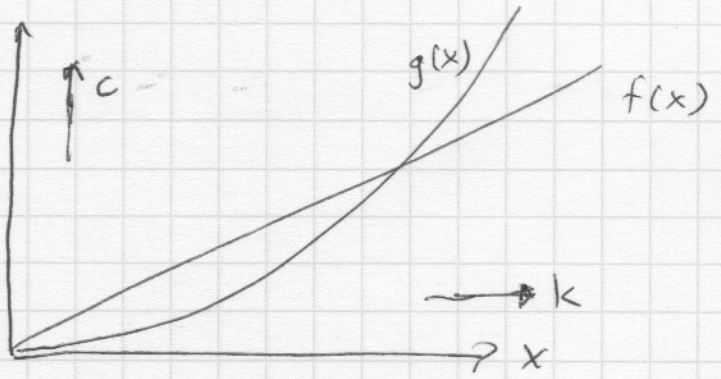
$$\sum_{j=0}^{n} j = n(n+1)/2$$

$$\sum_{j=0}^{n} a_0 r^j = \begin{cases} (n+1)a_0 & \text{if } r = 1 \\ \dfrac{a_0 r^{n+1} - a_0}{r-1} & \text{if } r \neq 1 \end{cases}$$

## VI. Complexity of algorithms and fun'cs.

Given two funcs $f(x)$, $g(x)$

- $f(x) \in O(g(x))$ means $|f(x)| \leq c|g(x)|$ for $x > k$.
  (c and k are "witnesses")
- $f(x) \in \Omega(g(x))$ means $|f(x)| \geq c|g(x)|$ for $x > k$.
- $f(x) \in \Theta(g(x))$ means $f(x) \in O(g(x))$ for some $c_1, k_1$,
  and $f(x) \in \Omega(g(x))$ for some $c_2, k_2$.




Very simple for polynomials:

$$a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = \sum_{j=0}^{n} a_j x^j \leq \sum_{j=0}^{n} |a_j| x^n$$

$$\boxed{c = \sum_{j=0}^{n} |a_j|} \quad k = 1, \quad n \text{ is order of polynomial}$$

## VII. Algorithms. A proceedure for processing information.

a) Basic structures: if-then, while, for, if-else-then, etc.

b) Complexity of algorithms (Big-O notation)

c) Recursive and inductive algorithms.

a) Basic algorithms

- search (linear, binary).
- sorting (bubblesort)
- Psuedocode.

```
" ex func( )
      if P(x)
          do this
      else if
  return 0.... "
```

b) Complexity of algorithms:

- # of operations (very tedious; not so relevant)
- How does run time scale with input size?

Complexity classes

| | |
|---|---|
| $O(1)$ | constant |
| $O(\log n)$ | |
| $O(n)$ | |
| $O(n \log n)$ | |
| $O(x^n)$ | Polynomial |
| $O(n!)$ | Factorial |
| $O(2^n)$ | Exponential |

Polynomial & below "tractable"

(Don't forget the functions, $\lfloor x \rfloor$, $\lceil x \rceil$)

## VIII. Proof by mathematical induction.

Prove $P(n)$ for $n \in \mathbb{N}, \mathbb{Z}^+$

- Inductive hypothesis $P(n)$
- Prove base case $P(n_0) = T$ where $n_0 = \{0, 1, 2, \ldots\}$
- Direct proof $P(n) \rightarrow P(n+1)$

Typically used for summations, inequalities, Fibonacci #'s.

Simple for summations:

$$P(n) := \text{"} \sum_{i=0}^{n} f(i) = \text{simple func}(n) \text{"}$$

Raw expression for $n+1$ in terms of $n$

$$\sum_{i=0}^{n+1} f(i) = \underbrace{\sum_{i=0}^{n} f(i)}_{\text{simplefunc}(n)} + f(n+1) = \underbrace{\text{simplefunc}(n) + f(n+1)}_{\text{algebra to make it look like } P(n+1)}$$

write of $P(n) := \text{Sum}(n) = \text{func}(n)$

Goal $\Rightarrow$ $P(n+1) := \text{Sum}(n+1) = \text{func}(n+1)$

# IX. Recursion & iteration

Recursion: define an object in terms of its predecessors.

arith. series: $a_n = a_{n-1} + d$

Fibo #'s: $f_0 = 0;\ f_1 = 1;\quad f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$.

Recursive algorithms.
(if and while loops to build from top down)

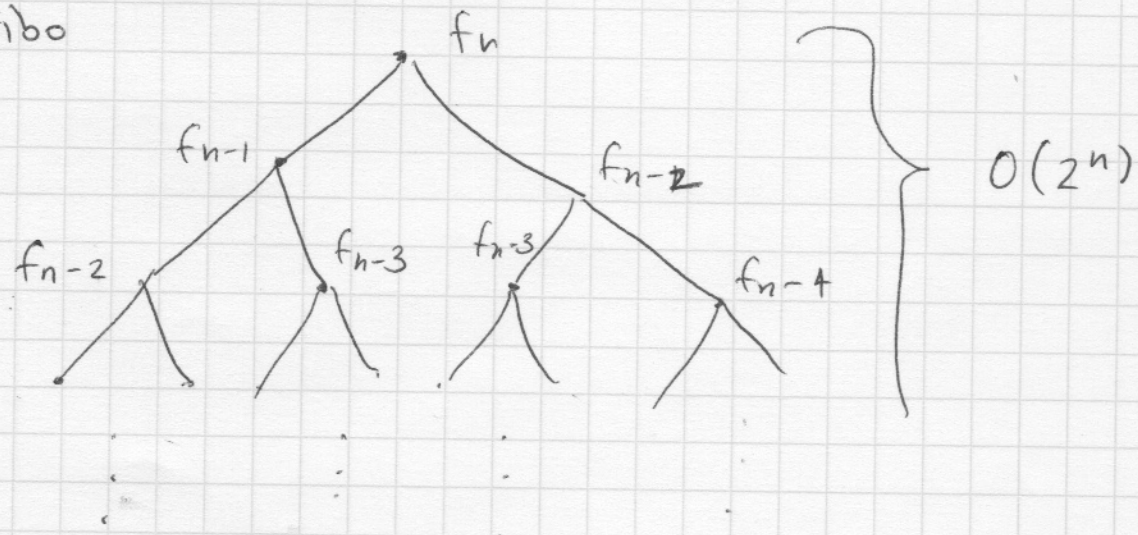
- specify base case
- else call the func for one value smaller

e.g. Recurse factorial func.

```
recfac (n)
    if n = 1
        return 1
    else
        return  n · recfac (n-1)
```

O (n)

Recursive Fibo



# X. Counting & probability

1) Product rule (outcomes multiply "and")

2) Sum rule (outcomes add "or")

usually a combination of sum & product.

3) Inclusion-exclusion $|A \cup B| = |A| + |B| - |A \cap B|$

4) Permutation $P(n,r)$ ordered sequence.

5) Combinations $C(n,r)$ subsets.

$$P(n,r) = C(n,r)\, r!$$

6) Probability $p(E) = \dfrac{|E|}{|S|}$