CS1101S
AY25/26 sem 1
github.com/lostmusician

01 recursion

examples

```
function factorial(n) {
    if (n === 0) return 1; // base case
    return n * factorial(n - 1); // recursive case
}

function factorial(n) {
    return iter(1, 1, n);
}

function iter(product, counter, n) {
    return counter > n
        ? product // base case
        : iter(product * counter, counter + 1, n); // recursive case
}
```

```
function fib(n) {
    function f(n, k, x, y) {
        return (k > n)
            ? y
            : f(n, k + 1, y, x + y);
    }
    return (n < 2) ? n : f(n, 2, 0, 1);
}
```

```
function gcd(a, b) {
    return b === 0
        ? a
        : gcd(b, a % b);
}
```

wishful thinking: assuming that the back is already solved

CPS

```
function append_iter(xs, ys){
    // iterative process
    function app(xs, ys, c) {
        return is_null(xs)
        ? c(ys)
        : app(tail(xs), ys,
            x => c(pair(head(xs), x))
            );
    }
    return app(xs, ys, x => x);
}
```

```
function fast_expt(b, n) {
    return n === 0
        ? 1
        : is_even(n)
        ? square(fast_expt(b, n / 2))
        : b * fast_expt(b, n - 1);
}
```

```
function fast_expt_cps(b, n, c) {
    return n === 0
        ? c(1)
        : is_even(n)
        ? fast_expt_cps(b, n / 2, x => c(square(x)))
        : fast_expt_cps(b, n - 1, x => c(b * x));
}
```

02 lists and trees

A tree of certain data items is a list whose elements are such data items, or trees of such data items.
make tree takes 3 args: entry, left branch, right branch

```
function BST_to_list(bst) {
    if (is_null(bst)) {
        return null;
    } else {
        const ltree = head(tail(bst));
        const num = head(bst);
        const rtree = head(tail(tail(bst)));
        return append(BST_to_list(ltree),
                      pair(num,
                           BST_to_list(rtree)));
    }
}
```

```
function map_tree(f, tree) {
    return map(sub_tree => !is_list(sub_tree)
                ? f(sub_tree)
                : map_tree(f, sub_tree)
               , tree);
}
```

```
function flatten_tree(xs) {
    function h(xs, prev) {
        return is_null(xs)
            ? prev
            : is_list(xs)
                ? append(flatten_tree(xs), prev)
                : pair(xs, prev);
    }
    return accumulate(h, null, xs);
}
```

```
function insert(bst, item) {
    if (is_empty_tree(bst)) {
        return make_tree(item,
                         make_empty_tree(),
                         make_empty_tree());
    } else {
        if (item < entry(bst)) {
            // smaller than entry(left branch)
            return make_tree(entry(bst),
                             insert(left_branch(bst),
                                    item),
                             right_branch(bst));
        } else if (item > entry(bst)) {
            // bigger than entry (right branch)
            return make_tree(entry(bst),
                             left_branch(bst),
                             insert(right_branch(bst),
                                    item));
        } else {
            // equal to entry.
            return bst;
        }
    }
}
```

```
function find(bst, name) {
    return is_empty_tree(bst)
        ? false
        : name === entry(bst)
         ? true
         : name < entry(bst)
          ? find(left_branch(bst), name)
          : find(right_branch(bst), name);
}
```

matrix

remember to use listref

```
function transpose(M) {
    const nR = length(M); // number of rows
    const nC = length(head(M)); // columns
    return map(c => map(row => list_ref(row, c), M), enum_list(0, nC - 1));
}

function row_sums(M) {
    return map(row => accumulate((x, sum) => x + sum, 0, row), M);
}
```

```
function map_using_accumulate(f, xs) {
    return accumulate((x, result) =>
            pair(f(x), result), null, xs);
}

function filter_using_accumulate(pred, xs) {
    return accumulate(
        (x, result) => pred(x)
        ? pair(x, result) : result,
        null,
        xs
    );
}
```

03 permutations and combinations

```
function permutations(s) {
    return is_null(s)
        ? list(null)
        : accumulate(append, null,
                     map(x => map(p => pair(x, p)),
                     permutations(remove(x, s))),
                     s));
}

function subsets(s) {
    return accumulate(
        (x, s1) => append(s1,
                          map(ss => pair(x, ss), s1)),
        list(null),
        s);
}
function choose(n, r) {
    if (n < 0 || r < 0) {
        return 0;
    } else if (r === 0) {
        return 1;
    } else {
        // Consider the 1st item, there are 2 choices:
        // To use, or not to use
        // Get remaining items with wishful thinking
        const to_use = choose(n - 1, r - 1);
        const not_to_use = choose(n - 1, r);

        return to_use + not_to_use;
    }
}

function combinations(xs, r) {
    if ( (r !== 0 && xs === null) || r < 0) {
        return null;
    } else if (r === 0) {
        return list(null);
    } else {
        const no_choose =
            combinations(tail(xs), r);
        const yes_choose =
            combinations(tail(xs),
                                      r - 1);
        const yes_item = map(x =>
            pair(head(xs), x),
                         yes_choose);
        return append(no_choose, yes_item);
    }
}
```

02. COMPOUND STATEMENTS

operations

1 $\sim$: negation (not)
2 $\wedge$: conjunction (and)
2 $\vee$: disjunction (or) - coequal to $\wedge$
3 $\rightarrow$: if-then

logical equivalence

- identical truth values in truth table
- definitions
- to show non-equivalence:
    - truth table method (only needs 1 row)
    - counter-example method

conditional statements

hypothesis → conclusion

antecedent → consequent

- **vacuously true** : hypothesis is false
- **implication law** : $p \to q \equiv \sim p \vee q$
- common if/then statements:
  - if p then q: $p \to q$
  - p if q: $q \to p$
  - p only if q: $p \to q$
  - p iff q: $p \leftrightarrow q$
- **contrapositive** : $\sim q \to \sim p$
- **inverse** : $\sim p \to \sim q$
- **converse** : $q \to p$

converse ≡ inverse
statement ≡ contrapositive

- r is a **necessary** condition for s: $\sim r \to \sim s$ and $s \to r$
- r is a **sufficient** condition for s: $r \to s$
- **necessary** & **sufficient** : $\leftrightarrow$

valid arguments

- determining validity: construct truth table
  - valid $\leftrightarrow$ conclusion is true when premises are true
- **syllogism** : (argument form) 2 premises, 1 conclusion
- **modus ponens** : $p \to q;\ p;\ \therefore q$
- **modus tollens** : $p \to q;\ \sim q;\ \therefore \sim p$
- **sound argument** : is valid & all premises are true

fallacies

converse error	inverse error
$p \to q$	$p \to q$
q	$\sim p$
$\therefore p$	$\therefore \sim q$

03. QUANTIFIED STATEMENTS

- **truth set** of $P(x) = \{x \in D \mid P(x)\}$
- $P(x) \Rightarrow Q(x) : \forall x(P(x) \to Q(x))$
- $P(x) \Leftrightarrow Q(x) : \forall x(P(x) \leftrightarrow Q(x))$

relation between $\forall, \exists, \wedge, \vee$

- $\forall x \in D, Q(x) \equiv Q(x_1) \wedge Q(x_2) \wedge \cdots \wedge Q(x_n)$
- $\exists x \in D \mid Q(x) \equiv Q(x_1) \vee Q(x_2) \vee \cdots \vee Q(x_n)$

05. SETS

notation

- set roster notation [1]: $\{x_1, x_2, \ldots, x_n\}$
- set roster notation [2]: $\{x_1, x_2, x_3, \ldots\}$
- set-builder notation: $\{x \in \mathbb{U} : P(x)\}$

definitions

- **equal sets** : $A = B \leftrightarrow \forall x(x \in A \leftrightarrow x \in B)$
  - $A = B \leftrightarrow (A \subseteq B) \wedge (A \supseteq B)$
- **empty set**, $\emptyset$: $\emptyset \subseteq$ all sets
- **subset** : $A \subseteq B \leftrightarrow \forall x(x \in A \to x \in B)$
- **proper subset** : $A \subsetneq B \leftrightarrow (A \subseteq B) \wedge (A \neq B)$
- **power set** of A : $\mathcal{P}(A) = \{X \mid X \subseteq A\}$
  - $|\mathcal{P}(A)| = 2^{|A|}$, given that A is a finite set
- **cardinality** of a set, $\mid A \mid$: number of distinct elements
- **singleton** : sets of size 1
- **disjoint** : $A \cap B = \emptyset$

methods of proof for sets

- direct proof
- element method
- truth table

boolean operations

- **union:** $A \cup B = \{x : x \in A \vee x \in B\}$
- **intersection:** $A \cap B = \{x : x \in A \wedge x \in B\}$
- **complement** (of B in A): $A \backslash B = \{x : x \in A \wedge x \notin B\}$
- **complement** (of B): $\bar{B}$ or $B^c = U \backslash B$
  - set difference law: $A \backslash B = A \cap \bar{B}$

ordered pairs and cartesian products

- **ordered pair** : (x, y)
  - $(x, y) = (x', y') \leftrightarrow x = x'$ and $y = y'$
- **Cartesian product** :
  $A \times B = \{(x, y) : x \in A$ and $y \in B\}$
  - $|A \times B| = |A| \times |B|$
- **ordered tuples** : expression of the form $(x_1, x_2, \ldots, x_n)$

06. FUNCTIONS

definitions

- **function/map** from A to B : assignment of each element of A to exactly one element of B.
  - $f : A \to B$: "f is a function from A to B"
  - $f : x \to y$: "f maps x to y"
  - **domain** of f = A
  - **codomain** of f = B
  - **range/image** of f = $\{f(x) : x \in A\}$
    $= \{y \in B \mid y = f(x)$ for some $x \in A\}$
- **identity function** on A, $\mathrm{id}_A : A \to A$
  - $\mathrm{id}_A : x \to x$
  - range = domain = codomain = A
  - (E6.1.24) $f \circ \mathrm{id}_A = f$ and $\mathrm{id}_A \circ f = f$
- **well-defined function** : every element in the domain is assigned to exactly one element in the codomain

equality of functions

- same codomain and domain
- for all $x \in$ codomain, same output

function composition

- $(g \circ f)(x) = g(f(x))$
- for $(g \circ f)$ to be well defined, codomain of f must be equal to the domain of g
- × commutative
- ✓ **associative** - (T6.1.26) $f \circ (g \circ h) = (f \circ g) \circ h$

image & pre-image

for $f : A \to B$

- if $X \subseteq A$, **image** of X,
  $f(X) = \{y \in B : y = f(x)$ for some $x \in X\}$
- if $Y \subseteq B$, **pre-image** of Y,
  $f^{-1}(Y) = \{x \in A : y = f(x)$ for some $y \in Y\}$

injection & surjection

- **surjective** (onto) : codomain = range
  - $\forall y \in B, \exists x \in A\ (y = f(x))$
  - surjective test: $\forall Y \subseteq B, Y \subseteq f(f^{-1}(Y))$
- **injective** : one-to-one
  - $\forall x, x' \in A(f(x) = f(x') \Rightarrow x = x')$
  - injective test: $\forall X \subseteq A, X \subseteq f^{-1}(f(X))$
- **bijective** : both surjective & injective
  - bijective $\Leftrightarrow$ has an inverse (T6.2.28)

inverse

- $\forall x \in A, \forall y \in B(f(x) = y \Leftrightarrow g(y) = x)$
- **uniqueness** of inverses (P2.6.16)
  - if g, g' are inverses of $f : A \to B$, then $g = g'$

07. INDUCTION

mathematical induction

to prove that $\forall n \in \mathbb{Z}_{\geq m}(P(n))$ is true,

- base step: show that $P(m)$ is true
- induction step: show that $\forall k \in \mathbb{Z}_{\geq m}(P(k) \Rightarrow P(k+1))$ is true.
  - induction hypothesis: assumption that $P(k)$ is true

strong MI

to prove that $\forall n \in \mathbb{Z}_{\geq 0}(P(n))$ is true,

- base step: show that $P(0), P(1)$ are true
- induction step: show that
  $\forall k \in \mathbb{Z}_{\geq 0}(P(0) \cdots \wedge P(k+1) \Rightarrow P(k+2))$ is true.

justification:

- $P(0) \wedge P(1)$ by base case
- $P(0) \wedge P(1) \to P(2)$ by induction with $k = 0$
- $P(0) \wedge P(1) \wedge P(2) \to P(3)$ by induction with $k = 1$
- $\ldots$
- we deduce that $P(0), P(1), \ldots$ are all true by a series of **modus ponens**

well-ordering principle

- every nonempty subset of $\mathbb{Z}_{\geq 0}$ has a smallest element.
- application: recursion has a base case

RECURSION

a sequence is **recursively defined** if the definition of a_n involves $a_0, a_1, \ldots, a_{n-1}$ for all but finitely many $n \in \mathbb{Z}_{\geq 0}$.

recursive definitions

e.g. recursive definition for $\mathbb{Z}$

1. (**base clause**) $0 \in \mathbb{Z}_{\geq 0}$
2. (**recursion clause**) If $x \in \mathbb{Z}_{\geq 0}$, then $x + 1 \in \mathbb{Z}_{\geq 0}$
3. (**minimality clause**) Membership for $\mathbb{Z}_{\geq 0}$ can be demonstrated by (finitely many) successive applications of the clauses above

recursion vs induction

- **recursion** - to define the set
- **induction** - to show things about the set

well-formed formulas (WFF)

in propositional logic

define the set of WFF(Σ) as follows

1. (base clause) every element ρ of Σ is in WFF(Σ)
2. (recursion clause) if x, y are in WFF(Σ), then $\sim x$ and $(x \wedge y)$ and $(x \vee y)$ are in WFF(Σ)
3. (minimality clause) Membership for WFF(Σ) can be demonstrated by (finitely many) successive applications of the clauses above

08. NUMBER THEORY

divisibility

transitivity of divisibility
If $a \mid b$ and $b \mid c$, then $a \mid c$.
closure lemma (non-standard name)
Let $a, b, d, m, n \in \mathbb{Z}$. If $d \mid m$ and $d \mid n$, then $d \mid am + bn$.
division theorem
$\forall n \in \mathbb{Z}$ and $d \in \mathbb{Z}^+$, $\exists! q, r \in \mathbb{Z}$ s.t.
$n = dq + r$ and $0 \leq r < d$
$q = n\ div\ d = \lfloor n/d \rfloor$
$r = n \bmod d = n - dq$

base-b representation

of positive integer n is $(a_\ell a_{\ell-1} \ldots a_0)_b$
where $\ell \in \mathbb{Z}_{\geq 0}$ and $a_0, a_1, \ldots, a_\ell \in \{0, 1, \ldots, b-1\}$
s.t. $n = a_\ell b^\ell + a_{\ell-1} b^{\ell-1} + \cdots + a_0 b^0$ and $a_\ell \neq 0$

greatest common divisor

- if $m \neq 0$ and $n \neq 0$, then $\gcd(m, n)$ exists and is positive.
  - gcd: *Euclidean Algorithm*
  - integer linear combination: *Extended Euclidean Algorithm*

Bezout's Lemma:
For all $m, n \in \mathbb{Z}$ with $n \neq 0$, there exist $s, t \in \mathbb{Z}$ such that $\gcd(m, n) = ms + nt$.
Euclid's Lemma:
Let $m, n \in Z^+$. If p is prime and $p \mid mn$, then $p \mid m$ or $p \mid n$.

- (E8.4.3) $m \bmod n = 0 \Leftrightarrow \gcd(m, n) = n$
- (L8.4.11) $\forall x, y, r \in \mathbb{Z}$,
  $x \bmod y = r \Rightarrow \gcd(x, y) = \gcd(y, r)$

prime factorization thoerem

- **(aka Fundamental Theorem of Arithmetic)**: Every integer $n \geq 2$ has a unique prime factorization in which the prime factors are arranged in nondecreasing order.

modular arithmetic

$n \bmod d$ is always non-negative.

Let $a, b, c \in \mathbb{Z}$ and $n \in \mathbb{Z}^+$.
congruence
$a \equiv b \pmod n \Leftrightarrow a \bmod n = b \bmod n$
Then $\exists k \in \mathbb{Z}(a = nk + b$ and $n \mid (a - b))$
reflexivity
$a \equiv a \pmod n$
symmetry
$a \equiv b \pmod n \to b \equiv a \pmod n$
transitivity
$a \equiv b \pmod n \wedge b \equiv c \pmod n \to a \equiv c \pmod n$

addition & multiplication

If $a \equiv b (\bmod n)$ and $c \equiv d (\bmod n)$,

- (P8.6.6) $a + c \equiv (b + d)(\bmod n)$
- (P8.6.13) $ac \equiv bd(\bmod n)$

additive inverse

b is an *additive inverse* of $a \bmod n \Leftrightarrow a + b \equiv 0 \pmod n$.
b is an *additive inverse* of $a \bmod n \Leftrightarrow b \equiv -a \pmod n$.

multiplicative inverse

b is a multiplicative inverse of $a \bmod n \Leftrightarrow ab \equiv 1 \pmod{n}$.

- If b, b' are multiplicative inverses of a, then $b \equiv b' \pmod{n}$.
- exists $\Leftrightarrow \gcd(a, n) = 1$.
  - a, n are coprime
- to find multiplicative inverse: **Euclidean Algorithm**

09. EQUIVALENCE RELATIONS

relations

Let R be a relation from A to B and $(x, y) \in A \times B$. Then:
xRy for $(x, y) \in R$ and $x \not R y$ for $(x, y) \notin R$

- a relation from A to B is a subset of $A \times B$.
- a **(binary) relation** on set A is a relation from A to A.
  - subset of A^2
- **inverse relation**: $xR^{-1}y \Leftrightarrow yRx$

reflexivity, symmetry, transitivity

Let A be a set and R be a relation on A.

reflexive
$\forall x \in A\ (xRx)$
symmetric
$\forall x, y \in A\ (xRy \Rightarrow yRx)$
transitive
$\forall x, y, z \in A\ (xRy \wedge yRz \Rightarrow xRz)$

- **equivalence relation**: a relation that is reflexive, symmetric and transitive
- **equivalence class**: the set of all things equivalent to x

equivalence classes

Let A be a set and R be an equivalence relation on A.

- $[x]_R$: **equivalence class** of x with respect to R
  $\forall x \in A, [x]_R = \{y \in A : xRy\}$
- A/R : The set of all equivalent classes
  $A/R = \{[x]_R : x \in A\}$

$xRy \Rightarrow [x] = [y] \Rightarrow [x] \cap [y] \neq \emptyset$

partitions

- a **partition** of a set A is a set $\mathscr{C}$ of *non-empty subsets* of A such that
  $(\geq 1)\ \ \forall x \in A,\ \exists S \in \mathscr{C}(x \in S)$
  $(\leq 1)\ \ \forall x \in A,\ \forall S, S' \in \mathscr{C}(x \in S \wedge x \in S' \Rightarrow S = S')$
- **components** : elements of a partition
- every partition comes from an equivalence relation

partial orders

Let A be a set and R be a relation on A.

- R is **antisymmetric** if $\forall x, y \in A\ (xRy \wedge yRx \rightarrow x = y)$
  - includes vacuously true cases (e.g. $xRy \Leftrightarrow x < y$)
- x and y are **comparable** if $\forall x, y \in A\ (xRy \vee yRx)$
- R is a **(non-strict) partial order** if R is reflexive, antisymmetric and transitive.
  - $\preccurlyeq$ - partial order
  - $x \prec y \Leftrightarrow x \preccurlyeq y \wedge x \neq y$ (NOT a partial order)
  - Hasse diagram
- R is a **(non-strict) total order** if R is a partial order and x and y are comparable

min and max

Let $\preccurlyeq$ be a partial order on a set A, and $c \in A$.

- c is a **minimal element** if $\forall x \in A\ (x \preccurlyeq c \Rightarrow c = x)$
  - nothing is strictly below it
- c is a **maximal element** if $\forall x \in A\ (c \preccurlyeq x \Rightarrow c = x)$
  - nothing is strictly above it
- c is the **smallest element** or **minimum element** if $\forall x \in a\ (c \preccurlyeq x)$.
- c is the **largest element** or **maximum element** if $\forall x \in a\ (x \preccurlyeq c)$.

linearization

Let A be a set and $\preccurlyeq$ be a partial order on A.
Then there exists a total order $\preccurlyeq^*$ on A such that
$\forall x, y \in A\ (x \preccurlyeq y \Rightarrow x \preccurlyeq^* y)$

10A. COUNTING

permutations

$P(n, r) = \frac{n!}{(n-r)!}$ (also ${}_nP_r, P^n_r$)

- **multiplication/product rule:** An operation of k steps can be performed in $n_1 \times n_2 \times \cdots \times n_k$ ways.
- **addition/sum rule:** Suppose a finite set A equals the union of k distinct mutually disjoint subsets $A_1, A_2, \ldots, A_k$. Then
  $|A| = |A_1| + |A_2| + \cdots + |A_k|$
- **difference rule:** if A is a finite set and $B \subseteq A$, then
  $|A \backslash B| = |A| = |B|$
- **complement:** $P(\bar{A}) = 1 - P(A)$
- **inclusion/exclusion rule:** $|A \cup B \cup C| =$
  $|A|+|B|+|C|-|A\cap B|-|B\cap C|-|C\cap A|+|A\cap B\cap C|$

permutations with indistinguishable objects

For n objects with n_k of type k indistinguishable from each other, the total number of distinguishable permutations
$= \frac{n!}{n_1! n_2! \ldots n_k!}$

pigeonhole principle

For any function f from a finite set X with n elements to a finite set Y with m elements and for any positive integer k, if $k < \frac{n}{m}$, then there is some $y \in Y$ such that y is the image of at least $k + 1$ distinct elements of X.

- A function from a finite set to a smaller finite set cannot be injective.
- **presentation:**
  - There are m ¡object M¿ (pigeons) and n ¡object N¿
  - Thus, by Pigeonhole Principle, ...

combinations

$\binom{n}{r} = \frac{n!}{r!(n-r)!}$ (also $C(n, r), {}_nC_r, C_{n,r}, {}^nC_r$)

r-combinations from n elements with **repetition**
$= \binom{r+n-1}{r}$

pascal's formula

Suppose $n, r \in \mathbb{Z}^+$ with $r \leq n$. Then
$\binom{n+1}{r} = \binom{n}{r-1} + \binom{n}{r}$

binomial theorem

$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$
binomial coefficient: $\binom{n}{k}$

10B. PROBABILITY

probability

Let S be a sample space. For all events A and B in S, a *probability function* P satisfies the following axioms:

1. $0 \leq P(A) \leq 1$
2. $P(\emptyset) = 0$ and $P(S) = 1$
3. $(A \cap B = \emptyset) \Rightarrow [P(A \cup B) = P(A) + P(B)]$
4. $P(\bar{A}) = 1 - P(A)$
5. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

expected value

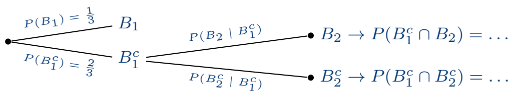
For possible outcomes $a_1, a_2, \ldots, a_n$ which occur with probabilities $p_1, p_2, \ldots, p_n$, the **expected value** is
$\sum_{k=1}^{n} = a_k p_k$

- **linearity** of expectation
  - $E[X + Y] = e[X] + E[Y]$
  - $E\left[\sum_{i=1}^{n} c_i \cdot X_i\right] = \sum_{i=1}^{n} (c_i \cdot E[X_i])$

conditional probability

The conditional probability of A given B,
$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$

probability tree:



Bayes' theorem

Suppose a sample space S is a union of mutually disjoint events $B_1, B_2, \ldots, B_n$ and A is an event in S.
For $k \in \mathbb{Z}$ and $1 \leq k \leq n$,

$$P(B_k \mid A) = \frac{P(A|B_k) \cdot P(B_k)}{\sum_{i=1}^{n} \left(P(A|B_i) \cdot P(B_i)\right)}$$

application of Bayes' theorem

$$P(B_1 \mid A) = \frac{P(A|B_1) \cdot P(B_1)}{P(A|B_1) \cdot P(B_1) + P(A|B_2) \cdot P(B_2)}$$

Let A be the event that the person test positive for a disease.
B_1: the person actually has the disease.
B_2: the person does not have the disease.

true positives: $P(B_1 \mid A)$	false negatives: $P(\bar{A} \mid B_1)$
false positives: $P(A \mid B_2)$	true negatives: $P(\bar{A} \mid B_2)$

independent events

A and B are **independent** *iff*
$P(A \cap B) = P(A) \cdot P(B)$
A, B and C are **pairwise independent** *iff*

1. $P(A \cap B) = P(A) \cdot P(B)$
2. $P(B \cap C) = P(B) \cdot P(C)$
3. $P(A \cap C) = P(A) \cdot P(C)$

A, B and C are **mutually independent** *iff*

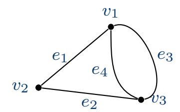
1. A, B and C are pairwise independent
2. $P(A \cap B \cap C) = P(A) \cdot P(B) \cdot P(C)$

11. GRAPHS

- mathematical structures used to model pairwise relations between objects

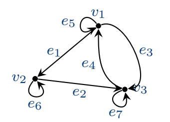
types of graphs

undirected graph



$e_1 = \{v_1, v_2\}$

directed graph



$e_1 = (v_1, v_2)$

undirected graph

- denoted by $G = (V, E)$, comprising
  - nonempty set of *vertices/nodes*, $V = \{v_1, v_2, \ldots, v_n\}$
  - a set of *edges*, $E = \{e_1, e_2, \cdots, e_k\}$
- $e = \{v, w\}$ for an undirected edge E incident on vertices v and w

directed graph

- denoted by $G = (V, E)$, comprising
  - nonempty set V of *vertices*
  - a set E of *directed edges* (ordered pair of vertices)
- $e = (v, w)$ for an directed edge E from vertex v to vertex w

simple graph

- **undirected graph** with no loops or parallel edges

complete graph

- a complete graph on n vertices, $n > 0$, denoted K_n, is a simple graph with n vertices and exactly one edge connecting each pair of distinct vertices

bipartite graph

- a simple graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V
- **complete bipartite graph:** $K_{m,n}$
  - bipartite graph on two disjoint sets U and V such that every vertex in U connects to every vertex in V
  - denoted $K_{m,n}$ where $|U| = m, |V| = n$

subgraph of a graph

H is a subgraph of $G \Leftrightarrow$

- every vertex in H is also a vertex in G
- every edge in H is also an edge in G
- every edge in H has the same endpoints as it has in G

degree

- **degree** of $v, deg(v) =$ number of edges incident on v
- **total degree** of G = sum of the degrees of all vertices of G
  total degree of $G = 2\times$ (number of edges of G)
- (C10.1.2) the total degree of a graph is even
- (P10.1.3) in any graph there are an even number of vertices of odd degree

trails, paths and circuits

Let G be a graph; let v and w be vertices of G.

- **walk** (from v to w): a finite alternating sequence of adjacent vertices and edges of G.
  - e.g. $v_0 e_1 v_1 e_2 \dots v_{n-1} e_n v_n$
  - **length** of walk: the number of edges, n
- a **trivial walk** from v to v consists of the single vertex v
- **trail** (from v to w): a walk from v to w that does not contain a repeated edge
- **path** (from v to w): a trail that does not contain a repeated vertex
- **closed walk**: walk that starts and ends at the same vertex
- **circuit/cycle**: an undirected graph $G(V, E)$ where
  - $V = \{x_1, x_2, \dots, x_n\}$
  - $E = \{\{x_1,x_2\},\{x_2,x_3\},\dots,\{x_{n-1},x_n\},\{x_n,x_1\}\}$
  - $n \in \mathbb{Z}_{\geq 3}$
  - aka a closed walk that does not contain a repeated edge
- **simple circuit/cycle**: does not have any other repeated vertex except the first and last
- (an undirected graph is) **cyclic** if it contains a loop/cycle

connectedness

- vertices v and w are connected $\Leftrightarrow$ $\exists$ a walk from v to w
- graph G is connected $\Leftrightarrow$ $\forall$ vertices $v, w \in V, \exists$ a walk from v to w

connected component

- a connected subgraph of the largest possible size
- graph H is a connected component of graph $G \Leftrightarrow$
  1. H is a subgraph of G
  2. H is connected
  3. no connected subgraph of G has H as a subgraph and contains vertices or edges that are not in H

Euler circuit

- **Euler circuit**: a circuit that contains every vertex and traverses every edge of G exactly once
- **Eulerian graph**: graph that contains an Euler circuit

T10.2.3

Euler circuit $\Leftrightarrow$ connected and every vertex has positive even degree

T10.2.4

Eulerian graph $\Leftrightarrow$ every vertex has positive even degree

- **Euler trail** (from v to w): a sequence of adjacent edges and vertices that starts at v, ends at w, and passes through every vertex of G at least once, and traverses every edge of G exactly once.

C10.2.5

$\exists$ Euler trail $\Leftrightarrow$ G is connected; v, w have odd degree; all other vertices of G have positive even degree

Hamiltonian circuit

- **Hamiltonian circuit** (for G): a *simple circuit* that includes every vertex of G.
  - does not need to include all the edges of G (unlike Euler circuit)
- **Hamilton(ian) graph**: contains a Hamiltonian circuit
- If G is a Hamiltonian circuit, then G has subgraph H where:
  1. H contains every vertex of G
  2. H is connected
  3. H has the same number of edges as vertices
  4. every vertex of H has degree 2

matrix representations of graphs

- **equal matrices** $\Leftrightarrow$ A and B are the same size and $a_{ij} = b_{ij}$ for all $i = 1, 2, \dots, m$ and $i = 1, 2, \dots, n$
- **square matrix**: equal number of rows and columns
- **main diagonal**: all entries $a_{11}, a_{22}, \dots, a_{nn}$
- **symmetric matrix** $\Leftrightarrow \forall i, j \in \mathbb{Z}^+_{\leq n} (a_{ij} = a_{ji})$

adjacency matrix

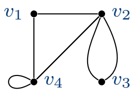
The adjacency matrix of a **directed graph** G is the $n \times n$ matrix $A = (a_{ij})$ over the set of non-negative integers such that

a_{ij} = number of **arrows** from v_i to v_j $\forall i, j = 1, 2, \dots, n$

$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 2 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

The adjacency matrix of an **undirected graph** G is the $n \times n$ matrix $A = (a_{ij})$ over the set of non-negative integers such that

a_{ij} = number of **edges** from v_i to v_j $\forall i, j = 1, 2, \dots, n$



$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 2 & 1 \\ 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

identity matrix

The $n \times n$ identity matrix,

$$I_n = (\delta_{ij}) = \begin{cases} 1, \text{if } i = j \\ 0, \text{if } i \neq j \end{cases} \quad \text{for all } i, j = 1, 2, \dots, n$$

matrix multiplication

scalar product

$$\begin{bmatrix} a_{i1} & a_{i2} & \dots & a_{in} \end{bmatrix} \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{nj} \end{bmatrix} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}$$

matrix product

Let $A = (a_{ij})$ be an $m \times k$ matrix and $B = (b_{ij})$ be a $k \times n$ matrix with real entries.

$AB = (c_{ij}) = \sum_{r=1}^{k} a_{ir} b_{rj}$

$\times$ commutative $\quad$ ✓ associative

nth power of a matrix

For any $n \times n$ matrix **A**, the powers of A are defined as follows:

$A^0 = I$ where I is the $n \times n$ identity matrix

$A^n = AA^{n-1} \quad \forall n \in \mathbb{Z}_{\geq 1}$

counting walks of length N

number of walks of length n from v_i to v_j = the ij-th entry of A^n

isomorphism

- graph isomorphism ($\cong$) is an equivalence relation.

Let $G = (V_G, E_G)$ and $G' = (V_{G'}, E_{G'})$ be two graphs.
$G \cong G' \Leftrightarrow$ there exist bijections $g : V_G \to V'_G$ and $h : E_G \to E'_G$ that preserve the edge-edgepoint functions of G and G' in the sense that $\forall v \in V_G$ and $e \in E_G$, v is an endpoint of $e \Leftrightarrow g(v)$ is an endpoint of $h(e)$.

planar graph

- a graph that can be drawn on a two-dimensional plane without edges crossing.
  - divides a plane into *regions/faces* (includes 'outside' the graph)

Euler's formula:

For a connected planar simple graph $G = (V, E)$ with $e = |E|$ and $v = |V|$ and f faces,

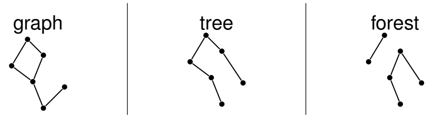
$f = e - v + 2$

Kuratowski's Theorem

A finite graph is planar $\Leftrightarrow$ does not contain a subgraph that is a subdivision of the complete graph K_5 or the complete bipartite graph K_3

trees

- **tree** $\Leftrightarrow$ graph that is circuit-free and connected
  - **(L10.5.4)** If G is a connected graph with n vertices and $n-1$ edges, then G is a tree.
- **trivial tree**: graph that comprises a single vertex
- **forest** $\Leftrightarrow$ graph is circuit-free and not connected
  - a group of trees
- **terminal vertex**: a vertex of degree 1
- **internal vertex**: a vertex of degree greater than 1



rooted trees

- **rooted tree**: a tree in which there is one vertex that is distinguished from the others and is called the root.
- **level** (of a vertex): the number of edges along the unique path between it and the root
- **height** (of a rooted tree): the maximum level of any vertex of the tree
- children, parent, siblings, ancestor, decendant

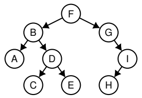
binary tree

- **binary tree**: a rooted tree in which every parent has at most 2 children
  - at most one left child and at most one right child
- **full binary tree**: a binary tree in which every parent has exactly 2 children
- (left/right) **subtree**: Given any parent v in a binary tree T, the binary tree whose root is the (left/right) child of v, whose vertices consist of the left child of v and all its descendants, and whose edges consist of all those edges of T that connect the vertices of the left subtree.

T10.6.1: Full Binary Tree Theorem

If T is a full binary tree with k internal vertices, then T has a total of $2k + 1$ vertices and has $k + 1$ terminal vertices.

binary tree traversal



F-B-G-A-D-I-C-E-H

Breadth-First Search (BFS)

- starts at the root
- visits its adjacent vertices
- visits the next level

Depth-First Search (DFS)

- **pre-order**
  - current vertex $\to$ left subtree $\to$ right subtree
- **in-order**
  - left subtree $\to$ current vertex $\to$ right subtree
- **post-order**
  - left subtree $\to$ right subtree $\to$ current vertex

pre-order DFS	*in-order DFS*	*post-order DFS*
F-B-A-D-C-E-G-I-H	A-B-C-D-E-F-G-H-I	A-C-E-D-B-H-I-G-F

spanning trees

- **spanning tree** (for a graph G): a subgraph of G that contains every vertex of G and is a tree.
  - $w(e)$ - weight of edge e
  - $w(G)$ - total weight of G
- **weighted graph**: each edge has an associated positive real number weight
  - **total weight**: sum of the weights of all edges
- **minimum spanning tree**: least possible total weight compared to all other spanning trees

Kruskal's algorithm

For a connected weighted graph G with n vertices:

1. initialise T to have all the vertices of G and no edges.
2. let E be the set of all edges in G; let $m = 0$
3. while $(m < n - 1)$
   - 3.1. find and remove the edge e in E of least weight
   - 3.2. if adding e to the edge set of T does not produce a circuit:
     - i. add e to the edge set of T
     - ii. set $m = m + 1$

Prim's algorithm

For a connected weighted graph G with n vertices:

1. pick any vertex v of G and let T be the graph with this vertex only
2. let V be the set of all vertices of G except v
3. for $(i = 0$ to $n - 1)$
   - 3.1. find the edge e in G with the least weight of all the edges connected to T. let w be the endpoint of e.
   - 3.2. add e and w to the edge and vertex sets of T
   - 3.3. delete w from v

LOGICAL EQUIVALENCES

commutative laws	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
associative laws	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
distributive laws	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
identity laws	$p \wedge true \equiv p$	$p \vee false \equiv p$
idempotent laws	$p \wedge p \equiv p$	$p \vee p \equiv p$
universal bound laws	$p \vee true \equiv true$	$p \wedge false \equiv false$
negation laws	$p \vee \sim p \equiv true$	$p \wedge \sim p \equiv false$
double negation law	$\sim(\sim p) \equiv p$	—
absorption laws	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
De Morgan's Laws	$\sim(p \vee q) \equiv \sim p \wedge \sim q$	$\sim(p \wedge q) \equiv \sim p \vee \sim q$

SET IDENTITIES

commutative laws	$A \cap B = B \cap A$	$A \cup B = B \cup A$
associative laws	$(A \cap B) \cap C = A \cap (B \cap C)$	$(A \cup B) \cup C = A \cup (B \cup C)$
distributive laws	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
identity laws	$A \cap U = A$	$A \cup \emptyset = A$
idempotent laws	$A \cap A = A$	$A \cup A = A$
universal bound laws	$A \cap \emptyset = \emptyset$	$A \cup U = U$
complement laws	$A \cap \overline{A} = \emptyset$	$A \cup \overline{A} = U$
double **complement** law	$\overline{(\overline{A})} = A$	—
absorption laws	$A \cup (A \cap B) = A$	$A \cap (A \cup B) = A$
De Morgan's Laws	$\overline{A \cup B} = \overline{A} \cap \overline{B}$	$\overline{A \cap B} = \overline{A} \cup \overline{B}$

proven:

number theory

- E1.1 - the product of 2 consecutive odd numbers is always odd.
- E1.5 - the difference between 2 consecutive squares is always odd
- E1.4 - the sum of any 2 even integers is even
- T4.6.1 - there is no greatest integer
- T8.2.8 - there are infinitely many prime numbers
- T4.3.1 - for all positive integers a and b, if $a|b$, then $a \leq b$.
- P4.6.4 - for all integers n, if n^2 is even then n is even
- T4.2.1 - all integers are rational numbers
- T4.2.2 - the sum of any 2 rational numbers is rational
- E1.7 - there exist irrational numbers p and q such that p^q is rational
- T4.7.1 - $\sqrt{2}$ is irrational.
- T4.3.2 - the only divisors of 1 are 1 and -1.

divisibility

- L8.1.5 - Let $d, n \in \mathbb{Z}$ with $d \neq 0$. Then $d \mid n \Leftrightarrow n/d \in \mathbb{Z}$
- L8.1.9 - Let $d, n \in \mathbb{Z}$. If $d \mid n$, then $-d \mid n$ and $d \mid -n$ and $-d \mid -n$
- L8.1.10 - Let $d, n \in \mathbb{Z}$. If $d \mid n$ and $d \neq 0$, then $|d| \leq |n|$
- L8.2.5 - **Prime Divisor Lemma** (non-standard name):
  - Let $n \in \mathbb{Z}_{\geq 2}$. Then n has a prime divisor.
- P8.2.6 - **sizes of prime divisors**:
  - Let n be a composite positive integer. Then n has a prime divisor $p \leq \sqrt{n}$.

base-b representation

- T8.3.13 - $\forall n \in \mathbb{Z}^+, \exists! \ell \in \mathbb{Z}_{\geq 0}$ and $a_0, a_1, \ldots, a_\ell \in \{0, 1, \ldots, b-1\}$ such that ¡the definition of base-b representation¿ holds.

logic

- T3.2.1 - negation of a universal statement:
  - $\sim(\forall x \in D, P(x)) \equiv \exists x \in D \mid \sim P(x)$
- T3.2.2 - negation of an existential statement:
  - $\sim(\exists x \in D \mid P(x)) \equiv \forall x \in D, \sim P(x)$

sets

- T5.1.14 - there exists a unique set with no element. It is denoted by $\emptyset$.
- E5.3.7 - for all A, B: $(A \cap B) \cup (A \backslash B) = A$
- T5.3.11(1) - let A, B be disjoint finite sets. Then $|A \cup B| = |A| + |B|$
- T5.3.11(2) - let $A_1, A_2, \ldots, A_n$ be pairwise disjoint finite sets. Then $|A_1 \cup A_2 \cup \cdots \cup A_n| = |A_1| + |A_2| + \cdots + |A_n|$
- T5.3.12 - **Inclusion-Exclusion Principle**:
  - for all finite sets A and B, $|A \cup B| = |A| + |B| - |A \cap B|$

induction

- L7.3.19 - If $x \in \text{WFF}^+(\Sigma)$, then assigning false to all elements of Σ makes x evaluate to false.
- T7.3.20 - $\sim(\forall x \in \text{WFF}(\Sigma), \exists y \in \text{WFF}^+(\Sigma) \;\; y \equiv x) \equiv \exists x \in \text{WFF}(\Sigma) \;\; \forall y \in \text{WFF}^+(\Sigma) \;\; y \not\equiv x$ aka $\sim$ (not) must be included in the definition of WFF.

relations

- E9.2.11 - The equality relation R on a set A has equivalence classes of the form $[x] = \{y \in A : x = y\} = \{x\}$ where $x \in A$
- T9.3.4 - Let R be an equivalence relation on a set A. Then A/R is a partition of A.
- T9.3.5 - If $\mathscr{C}$ is a partition of A, then there is an equivalence relation of R on A such that $A/R = \mathscr{C}$.
- L9.5.5 - Consider a partial order $\preccurlyeq$ on set A.
  - A smallest element is minimal.
  - There is at most one smallest element.

graphs

- L10.2.1 - Let G be a graph.
  - L10.2.1a - If G is connected, then any two distinct vertices of G can be connected by a path
  - L10.2.1b - If vertices v and w are part of a circuit in G and one edge is removed from the circuit, then there still exists a trail from v to w in G.
  - L10.2.1c - If G is connected and G contains a circuit, then an edge of the circuit can be removed without disconnecting G.
- L10.5.1 - Any non-trivial tree has at least one vertex of degree 1.
- T10.5.2 - Any tree with n vertices $(n > 0)$ has $n - 1$ edges.
- L10.5.3 - If G is any connected graph, C is any circuit in G, and one of the edges of C is removed from G, then the graph that remains is still connected.
- L10.5.4 - If G is a connected graph with n vertices and $n - 1$ edges, then G is a tree.
- T10.6.1 - If T is a full binary tree with k internal vertices, then T has a total of $2k + 1$ vertices and has $k + 1$ terminal vertices.
- T10.6.2 - For non-negative integers h, if T is any binary tree with height h and t terminal vertices, then $t \leq 2^h$.
- P10.7.1 -
  1. Every connected graph has a spanning tree.
  2. Any two spanning trees for a graph have the same number of edges

abbreviations

- L - lemma
- E - example
- P - proposition
- T - theorem