SETS

Property-Law	Set Theory Formula
Commutativity	A∩B = B∩A
	A∪B = B∪A
Associativity	A∩ (B∩C) = (A∩B)∩C
	A∪ (B∪C) = (A∪B)∪C
Distributivity	A ∩(B∪C) = (A ∩B) ∪ (A∩C)
Idempotent	A ∩ A = A
	A ∪ A = A
Identity	A ∪ Ø = A
	A ∩ U = A
Absorption	A ∩ (A ∪ B) = A
	A ∪ (A ∩ B) = A
Domination	A ∩ Ø = Ø
	A ∪ U = A
Set Difference	A – A = Ø
	B – A = B∩ A'
	B – A = B – (A∩B)
	(A – B) = A if A∩B = Ø
	(A – B) ∩ C = (A∩ C) – (B∩C)
	A ΔB = (A-B) U (B- A
Double complement	(A')' = A
Empty & Universal Set	Ø' = ∪ & ∪' = Ø
Complement Law	A∪A' = U
	A∩A' = Ø
	A' = U – A
De Morgan's	(A ∪B)' = A' ∩B'
	(A∩B)' = A' ∪ B'
Symmetric Difference	A ⊕ B = (A ∪ B) - (A ∩ B)
Carninality of set (| |)	Number of distinct elements in a set
P(S): powerset of a set	the set of all possible subsets, including the empty set (∅) and S itself
Cardinality of Powerset: (n = number of elements in the set)	P(S) = 2^n
Addition principle	|A∪B| = |A| + |B|
Inclusion–Exclusion Principle	|A∪B| = |A| + |B| − |A ∩ B| **or** |A ∪ B ∪ C| = |A| + |B| + |C| - |A ∩ B| - |A ∩ C| - |B ∩ C| + |A ∩ B ∩ C|

Common Sets

Z = set of integers = {...,−2,−1,0,1,2,...}

Z* = set of integers excluding 0 = {...,−2,−1,1,2,...}

N = set of natural numbers = {1,2,3,...}

W = set of whole numbers = {0,1,2,3,...}

Q = set of rational numbers = {...,-3/2, 1/4, 7/5,...} (denominator ≠0)

R = set of real numbers (ALL numbers)

BOOLEAN

Property-Law	Boolean Formula
Boundedness	A+1 = 1
	A.0 = 0
Sum of products: f(x,y,z) = x.y+x.z+y.z	From T.T. use only lines that f= true. If input=1 use uncomplemented, if input = 0 use complemented
Product of sums: f(x,y,z) = (x+y).(x+z).(y+z)	
K-Maps ≠ T.T. only for 2,3,4,5 variables	Cell = binary value, number of cells = 2^k

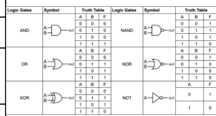
Truth Table

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z	0	1	0	1	0	1	0	1
f	0	0	0	1	1	1	1	1

Karnaugh Map

	y'z'	y'z	yz	yz'
x'	0	0	1	0
x	1	1	1	1

⇒ f(x,y,z) = x+y.z



HALF ADDER: used for 2 bit addition

A,B: sum (XOR) + carry out (AND)

FULL ADDER: for more than 2 bits (bits + carry bit)

A, B, carry-in: sum + carry out (2 half adders + OR)

INDUCTION & RECURSION

Induction	Assume P(n) = True then show that P(n+1) = True
1. Basis Step	Test+Show that for n=1, P(1) = True
2. Induction Hypothesis	Assume n=k and so P(n) = P(k)
3. Induction Step	Show that P(k+1) is Also True
4. Conclusion	P is True for all n, n+1
Strong Induction	Assume P(n) = True then show that P(n+1) = True
1. Basis Step	Test+Show that for n=1, P(1) = True
2. Induction Hypothesis	Assume P(i) = True ∀ positive integers i from 1 to k
3. Induction Step	Show that P(k+1) is also true
4. Conclusion	P is True for all n, n+1
Recursion	Function calls itself, breaking a problem into smaller steps
1. Basis Clause	Specify some initial elements
2. Recursive Clause	Establishes a systematic way to generate new elements from known elements
3. Terminal Clause	Guarantees that the only way to obtain the elements of S is clauses 1 + 2
Linear Recurrence	Each term in a sequence is a linear function of earlier terms (ex. Fibonacci Sequence)
Homogenous:	$a_n = c_1 \cdot a_{n-1} + c_2 \cdot a_{n-2} + \ldots + c_k \cdot a_{n-k}$
Non-Homogenous:	$a_n = c_1 \cdot a_{n-1} + c_2 \cdot a_{n-2} + \ldots + c_k \cdot a_{n-k} + f(n)$
Arithmetic Sequence	The difference between 2 terms is a constant c
	$a_n = a_1 + (n-1) \cdot d$
Sum of the first n terms	$S_n = \frac{n}{2} \cdot (a_1 + a_n)$
Geometric Sequence	The difference between 2 terms is a ratio r
	$a_n = a_1 \cdot r^{n-1}$
Sum of the first n terms	$S_n = \frac{a(1-r^n)}{1-r}$

PROPOSITIONAL LOGIC

Rules of Inference	
Modus Ponens	(p ∧ (p → q)) ⇒ q, if p → q is True and p is True, then q is True
Modus Tollens	((p → q) ∧ ¬q) ⇒ ¬p, if p → q is True and ¬q is True, then ¬p is True
Hypothetical Syllogism	((p → q) ∧ (q → r)) ⇒ (p → r), if p → q is True and q → r is True, then p → r is True
Disjunctive Syllogism	((p ∨ q) ∧ ¬p) ⇒ q, if (p ∨ q) is True and ¬p is True, then q is True
Addition	p ⇒ (p ∨ q), if p is True, then (p ∨ q) is True
Conjunction	((p) ∧ (q)) ⇒ (p ∧ q), if p is True and q is True, then (p ∧ q) is True
Simplification	(p ∧ q) ⇒ p, if (p ∧ q) is True, then p is True
Resolution	((p ∨ q) ∧ (¬p ∨ r)) ⇒ (q ∨ r), if (p ∨ q) is True and (¬p ∨ r) is True, then (q ∨ r) is True
Logical Equivalences	
Implication	A → B ≡ ¬A ∨ B.
Biconditional	A ↔ B ≡ (A → B) ∧ (B → A).

PREDICATE LOGIC

Quantifiers	Symbol
Universal Quantifier	∀ : "For all" - ∀x P(x) = P(x) is true for all x in the domain.
For a collection of sets	$\forall x\ P(x) = P(n_1) \vee P(n_2) \vee \ldots$
Existential Quantifier	∃: "There exists." ∃x P(x) = there is at least one x for which P(x) is true.
For a collection of sets	$\exists x\ P(x) = P(n_1) \wedge P(n_2) \wedge \ldots$
Unique Existential Quantifier	∃!: there exists a unique ...
Rules of Inference	
Universal Instantiation	If ∀x P(x) is true, then P(c) is true for any constant c.
Existential Generalization	If P(c) is true for some constant c, then ∃x P(x) is true.
Universal Modus Ponens	If ∀x (A(x) → B(x)) is true and A(c) is true, then B(c) is true.
Existential Modus Ponens	If ∃x (A(x) ∧ B(x)) is true, then B(c) is true for some constant c.
Logical Equivalences	
Negation of Quantified Statements	¬(∀x P(x)) ≡ ∃x ¬P(x)
	¬(∃x P(x)) ≡ ∀x ¬P(x)
De Morgan's Laws for Quantifiers:	¬(∀x P(x)) ≡ ∃x ¬P(x)
	¬(∃x P(x)) ≡ ∀x ¬P(x)
Binding Variables in quantifiers	ex. ∃x P(x,y) = x: bound, y:free
Order of Precedence	important only if different quantifiers = **∀,∃ ⇒** ¬ ⇒ ∨ ⇒ ∧ ⇒ → ⇒ ↔

Tautology: A compound proposition that is true for all possible truth value assignments to its constituent propositions.

Contradiction: A compound proposition that is false for all possible truth value assignments to its constituent propositions. (¬P → ¬Q)

Propositional Statements	**T.T of Implication (if p, then q)**
Converse: Q → P	
Contrapositive: ¬Q → ¬P	
Inverse: ¬P → ¬Q	**T.T of Biconditional (p if and only if q)**
Truth Tables	
Number of lines of n variables = 2^n	

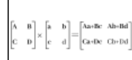
T.T of Implication:

p	q	p→q
T	T	T
T	F	F
F	T	T
F	F	T

T.T of Biconditional:

p	q	p↔q
T	T	T
T	F	F
F	T	F
F	F	T

GRAPHS

Walk	sequence of vertices & edges. **v** & **e** can be repeated
Trail	a walk, no **e** is repeated
Path	a trail, no **v** & **e** can be repeated
Cirquit	a closed trail, **v** can be repeated
Circle	a closed path, vertex reachable from self
Eulerian Path	a walk, traverses each edge only once **(traversable)**
Eulerian Cycle	Eulerian path, starts & ends in same vertex
Hamiltonian Path	a path, visits each vertex only once **(traceable)**
Hamiltonian Cycle	a Hamiltonian path, starts & ends in same vertex
Transitive closure	G →G*: Check which paths do not have a directed edge and add it
Vertex Degree	Number of incident edges
Degree Sequence	Monotonic non-increasing sequence of vertices degrees
Sum of D.S	Always EVEN - twice the number of edges
Number of Edges	Sum of D.G. / 2
Simple Graph	No loops, no parallel edges - n vertices = n-1 degree
Regular Graph	All vertices = same degrees - r regular = r degree
Complete Graph	Simple graph, every 2 vertices are connected with an edge
Isomorphic Graphs	If there exists Bijection that preserves adjacency
	Every element of G1 maps to 1 element in G2
	G1 & G2 = SAME D.S.
Bipartite Graphs	Vertices can be divided in 2 disjoint sets such that all edges have 1 endpoint in each set
	Created sets have the same D.S.
Breadth First Search	Traversal algorithm, explores a graph by visiting its vertices layer by layer.
	Starts from a specified source vertex and moves outward to its neighbors before visiting their neighbors.
Adjacency Matrix	Square matrix, leading diagonal = loops
	Undirected graph: number of edges = half the sum of all elements
	Directed graph: the squared matrix = number of paths of length 2 between 2 vertices
Matrix - Addition	Add each corresponding elements (only same size matrices)
Matrix - Multiplication	Only if number of rows in M1 = number of columns in M2
(2x2) x (2x2)	
(3x2) x (2x3)	
Weighted Graphs	Each edge is assigned a numerical value

ALGORITHMS

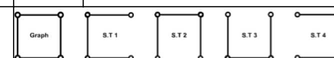
Dijkstra's (example): To find the shortest path between nodes in a weighted graph - Greedy Approach

Stage	Vertex	Shortest Distance from S	Previous Vertex	Not Visited
initialization	a	0	undefined	a,b,c,d
	b	∞	undefined	
	c	∞	undefined	
	d	∞	undefined	
1	a	0	undefined	b,c,d
	b	x	a	
	c	x	a	
	d	∞	undefined	
2	a	0	undefined	c,d
	b	x	a	
	c	x	a	
	d	x	b	
3	...	...	...	

Algorithm continues until all nodes are visited and shortest path has been found.

TREES

Trees	Undirected graphs, no cycles, all vertices are conected
Forest	Disconnected graph with no cycles
Rooted tree	1 vertex has been designated as the root & every vertex is a directed path away from the root
Spanning tree	Connected subgraph of main graph, contains all vertices of original graph but without the cycles



ALGORITHMS

Kruskal: Starts with the cheapest edge in a weighted spanning tree
Continues adding cheapest edges

Prim: Starts with any node in a weighted spanning tree
Continues adding cheapest edges

Depth of node	Number of edges from root to that node
Height of node	Longest path from that node to a leaf
Depth/Height of tree	Maximum path length across all nodes
M-ary trees	Rooted tree, every node has children ≤ m
	Regular: Every node has exactly m children
	At most, m^h vertices at level h
Isomorphic trees	Always same D.S. - Same D.S. does not mean isomorphic
Binary Search Tree	**Left Subtree:** Values in the nodes ≤ the value of the node itself. **Right Subtree:** Values in the nodes ≥ the value of the node itself.
Building a B.S.T	Short values in ascending order and find the root **(r = floor [(min + max)/2])**
Divide & Conquer	Repeatedly split ascending order lists in half until search returns the required record

FUNCTIONS

Injective (One-to-One)	Every element in Set A , maps to one element in set B
	f(a) = f(b) ↔ a = b or f(a) ≠ f(b) ↔ a ≠ b
Horizontal Line test	Line intersects graph only once
Vertical Line test	Not a function if intersects more than once
Surjective	Every element in Set B , has at least one preimage in set A
	f(x) = y, solve for x; prove for ∀y in the Co-Domain, ∃x in the Domain
	If Range = Co-Domain: Always Surjective
From Graph:	x axis = Domain, y axis = Range
Inverse Function	If function = bijective, then inverse: Set f(x) = y, Swap x and y, solve for y. Swap x and y again
Composite Functions	f(x)=2x & g(x)=x+3: (f∘g)(x)=f(g(x))=f(x+3)=2(x+3)=2x+6; (g∘f)(x)=g(f(x))=g(2x)=2x+3
Logarithms	y = a^x ↔ log base a of y = x

RELATIONS

Reflexive	aRa, ∀a∈S; **digraph:** always loop on every element; **matrix:** leading diagonal contains only 1s
Symmetric	∀a,b∈S, if aRb then bRa; **digraph:** no single connection (either no connection or parallel edges); **matrix:** symmetric in respect to leading diagonal
Antisymmetric	∀a,b∈S, if aRb & bRa then a=b; **digraph:** no parallel edges; **matrix:** if i≠j & mij≠0 then mij = 0
Transitive	∀a,b,c∈S, if aRb & bRc then aRc
Equivalance	Reflexive + Symmetric + Transitive
Partial Order	Reflexive + Antisymmetric + Transitive
Total Order	Partial Order + ∀a,b∈S, aRb or bRa

COMBINATORICS

Product Rule	**Independent events:** n x m possible ways. For Sets: |A x B| = |A| x |B|
Addition Rule	**Mutually Exclusive events: n + m** possible ways. For Sets: |A ∪ B| = |A| + |B| as long as A,B are disjoint
Subtraction Rule (Inclusion Exclusion Principle)	To count the total number of distinct elements and avoid double counting or overlapping |A ∪ B| = |A| + |B| - |A ∩ B|
Pigeonhole Principle	If k positive integer & k+1 objects are placed inside k containers, then at least 1 container has 2 or more objects. If N objects are placed in k boxes, at least one box contains ⌈N/k⌉objects (ceiling)
Permutations (Order Is Important)	Ordered arrangement of distinct elements in a set; r-permutations = arrangement of r objects
Number of Permutations	**P(n,r) = (n!) / (n-r)!** *(ex. How many ways to select 1st, 2nd & 3rd prize winner out of 50 people: P(50,3) = 50x49x48 = ...*
Combinations (Order Not Important)	r-combination of elements in a set is an unordered arrangement of r elements
Number of Combinations	**Binomial coefficient:** $C(n,r) = \frac{n!}{(n-r)!r!} = \frac{P(n,r)}{r!}$
Binomials	2 terms connected with + or - ; $(x+y)^2 = x^2 + 2xy + y^2$; $(x+y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$
Binomial Theorem	$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$
Pascal's Identity	$\binom{n}{k} + \binom{n}{k-1} = \binom{n+1}{k}$
Pascal's Triangle	Number Triangle, numbers arranged in staggered rows such that every element is the sum of the 2 elements directly above it.