Discrete Mathematics

- Minimum Spanning Tree(MST)
- Krushal Algorithm
- Dijkstra's algorithm
- Euler path and Euler circuit
- Hamilton path and Hamilton Circuit
- SD Flow, ST Cut Flow
- Bipartite Graph
- Graph Coloring theorem,application

- Functions with its types
- Sets with types
- Computer Representation for set
- Mathematical Induction
- Recurrence Relation
- Equivalence Relation
- Propositions, Premises, Quantifiers
- Contradiction method
- Fallacy
- LCM and GCD using a euclidean algorithm
- LCM and GCD using an extended euclidean algorithm
- Rules of inference
- Pigeonhole principle
- Chinese Remainder Theorem
- Ceiling and Floor Function
- Inclusive and Exclusive Principle
- Possets
- Combination and Permutation
- Matrix and Relations
- Recursive algorithm(factorial number, b^n)
- Binomial Coefficient of expansion
- Pascal's Triangle

Microprocessor

80286 architecture/functional unit and sub-units with diagrams
80386 architecture/functional unit with registers organization
Timing Diagrams
8085 and 8086 programs

DTE-DTE, DTE DTC connection RS-232
Demux of Address line in 8085 microprocessor

PPI 8255A
PIC 8259
DMA
Instruction Group of 8085
Addressing modes of 8086
Instructions set of 8086
System Bus and its types
MultiTasking or Pipelining of 80286 Microprocessor
Pipelining of Microprocessor 8086
Push and pop operation with example
Call and Jump operation

Maths

System of linear equations, consistent and inconsistent and echelon form and reduced echelon form
Linear transformation
Linearly independent and dependent
Inverse of Matrix
LU Factorization
Consumption Matrix
Determinants and determinants without expanding
Determinants with expansion cofactors

Subspace of the vectors
Null Space of vectors
Col A, Null-A and the dimension of Col A and Null-A
Basis of vectors
Cooridante of the matrix
Change of Basis

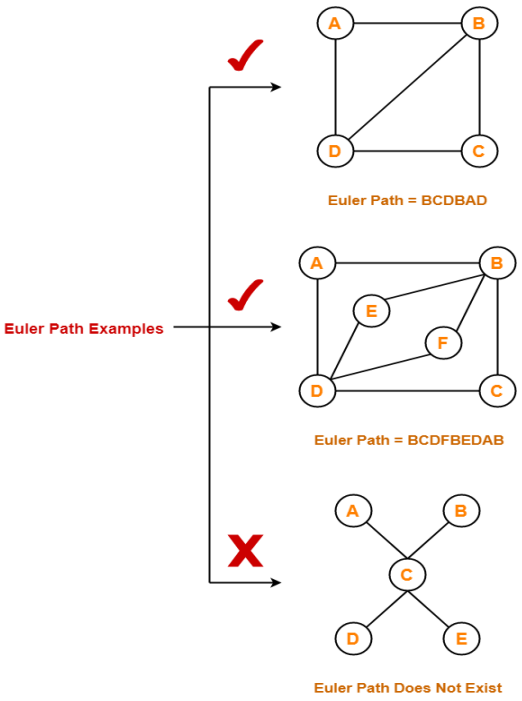
EigenValues and Eigenvectors
Unit Vector and angle between vectors
Orthogonal and orthonomal
QR Factorization of the matrix
Least square Matrix Solution
Equations of least squares line that best fits
Groups, Field, Ring
Binary operation and Integral Domain

Statistics

Dispersion and measure of dispersion
Coefficient of variance, mean and standard deviation
Karl Pearson's Correlation Coefficient and regression equation
Binomial,Normal and Poisson Distribution
Conditional Probability
Spearman's Rank
Probablity Density function and Probability Marginal Function

Eular Path:

- Cover every edges of the Graph only once
- Starts and ends with different vertices
- At most have 2 vertices with odd degrees

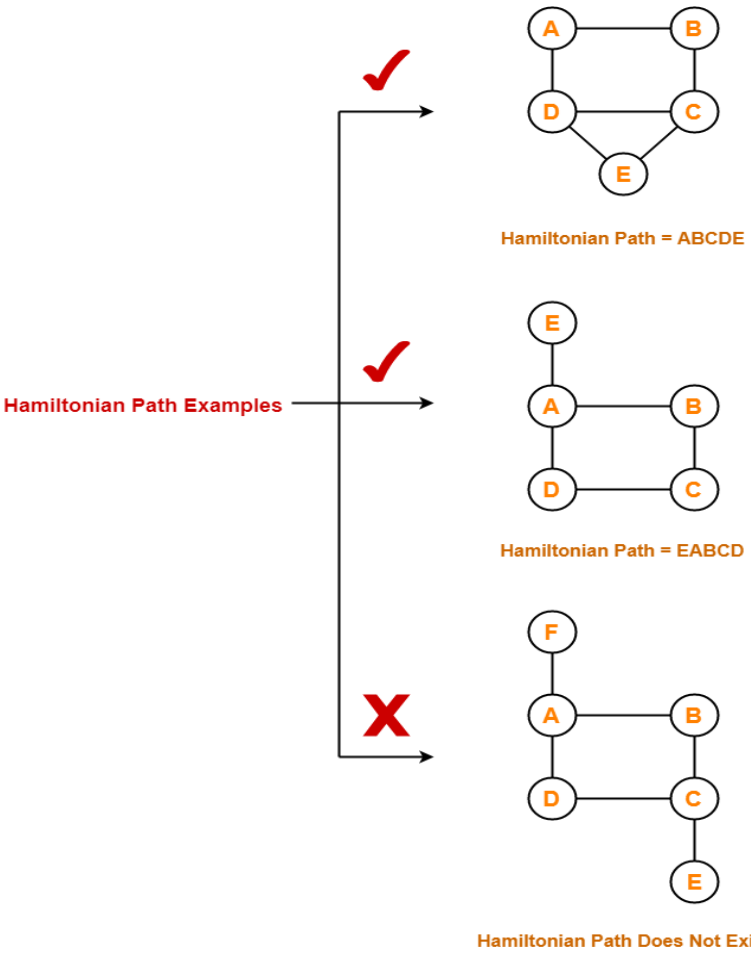


Eular Circuit

- Starts and end at the same vertex
- Closed path that visits every edge exactly once
- Every vertex must have even degree

Hamilton Path:

- Cover every vertices exactly once
- Start and end with different vertex



-

Hamilton Circuit.

-Start and end with same vertex, covering every vertices once

SD Flow

1. In graph theory, SD flow stands for "source to destination flow."
2. It is a concept used to determine the maximum flow that can pass through a network from a source node to a destination node.

3. The source node is the node where the flow originates, and the destination node is the node where the flow terminates.
4. It refers to the amount of flow that can be sent through graph
5. This flow should not violate any capacity constraints
6. It should not exceed the capacity of any edge
7. SD Flow is also known as the "maximum flow" or "maximum capacity" of a graph.

ST Cut Flow

In discrete mathematics, a cut is a partition of the vertices of a graph into two disjoint sets. A cut in a graph is said to be an "ST cut" if the two sets of vertices created by the cut contain the source vertex "S" and the destination vertex "T" of a specified edge (S, T). In other words, an ST cut is a cut that separates the source vertex S from the destination vertex T in a graph.

The capacity of an ST cut is the sum of the capacities of all edges crossing the cut, i.e., the sum of the weights of all edges that connect a vertex in the source set to a vertex in the destination set.

The concept of ST cuts is used in the Max-Flow Min-Cut theorem, which states that the maximum flow in a network is equal to the minimum capacity of an ST cut in the network. This theorem is a fundamental result in network flow theory and has important applications in various fields, including transportation, communication, and computer science.

- A cut is a partition of the vertices of a graph into two disjoint sets.
- **An ST cut is cut in a graph that separates the source vertex S from the destination vertex T.**
- The capacity of an ST cut is the s***um of the weights of all edges that cross the cut and connect a vertex in the source set to a vertex in the destination set.***

- **The Max-Flow Min-Cut theorem states that the maximum flow in a network is equal to the minimum capacity of an ST cut in the network.**
- The Max-Flow Min-Cut theorem has important applications in various fields, including transportation, communication, and computer science.

Premise

A premise, on the other hand, is a proposition that is used as the basis for a logical argument or deduction. A premise is a statement or assumption that is accepted as true in order to reach a conclusion. In a logical argument, the premises are the statements or assumptions that are used to support the conclusion.

For example, consider the following logical argument:

Premise 1: All men are mortal.

Premise 2: Socrates is a man.

Conclusion: Therefore, Socrates is mortal.

In this argument, "All men are mortal" and "Socrates is a man" are the premises, and "Socrates is mortal" is the conclusion. The premises are accepted as true in order to support the conclusion.

Existential Quantifier and Universal Quantifier

In discrete mathematics, quantifiers are used to describe the quantity of elements in a set that satisfy a certain property or condition. The two most common quantifiers are the existential quantifier and the universal quantifier.

The existential quantifier, denoted by $\exists$ (read as "there exists"), is used to indicate that at least one element in a set satisfies a certain property. For example, the statement "There exists an even integer" can be written as $\exists x$(x is even).

The universal quantifier, denoted by $\forall$ (read as "for all"), is used to indicate that every element in a set satisfies a certain property. For example, the statement "All even integers are divisible by 2" can be written as $\forall x$(x is even $\rightarrow$ x is divisible by 2).

In both cases, the variable x is used to represent an arbitrary element in the set. The quantifiers allow us to make general statements about the elements in the set without having to list them all out individually.

The Law of Detachment and Modus Ponens are related, but they are not exactly the same.

Modus Ponens is a specific form of the Law of Detachment, which states that if a conditional statement $p \rightarrow q$ is true, and p is true, then q must also be true. In other words, Modus Ponens is the rule of inference that allows us to infer the consequent q from the antecedent p and the conditional statement $p \rightarrow q$.

For example, if we know that "If it is raining, then the ground is wet" ($p \rightarrow q$) and we observe that it is indeed raining (p), then we can use Modus Ponens to conclude that "The ground is wet" (q).

On the other hand, the Law of Detachment is a more general rule of inference that applies to any valid argument in the form of "If $p \rightarrow q$ and p, then q". This means that if we have a valid argument of this form, we can use the Law of Detachment to conclude that q follows logically from the premises $p \rightarrow q$ and p.

In summary, Modus Ponens is a specific application of the Law of Detachment, where we use a conditional statement and its antecedent to infer its consequent, while the Law of Detachment is a more general principle that can be applied to any valid argument in this form.

Rules of inference are used to derive new logical statements or propositions from existing ones. Here are some common rules of inference:

Modus Ponens (MP): If p implies q and p are true, then q is true.

1. Example: If it is raining, then the ground is wet. It is raining. Therefore, the ground is wet.

Modus Tollens (MT): If p implies q and q is false, then p is false.

2. Example: If it is raining, then the ground is wet. The ground is not wet. Therefore, it is not raining.

Dijkstra's algorithm

is a popular algorithm in computer science used for finding the shortest path between two nodes in a graph with non-negative edge weights.

Dijkstra's algorithm is a way to find the shortest path between two points in a network. It starts at one point and looks at all the neighboring points, picking the one closest to the starting point. Then it looks at the neighbors of that point, and so on. It keeps track of the distances to each point and updates them if it finds a shorter path. Finally, it reaches the destination point and returns the shortest path. It's used in many applications, like finding the fastest route between two cities, but only works for networks where distances are never negative.

How does Dijkstra's algorithm works? Steps in the Simplest language.

Dijkstra's algorithm is used to find the shortest path between two points in a graph. Here are the steps in simple language:

1. Start at the initial node and set its distance to 0. Set the distances of all other nodes to infinity.
2. Mark the initial node as visited.
3. For each neighbor of the initial node, calculate the distance from the initial node to that neighbor and update its distance if it's shorter than the previous distance.
4. Select the unvisited node with the smallest distance and mark it as visited.
5. For each neighbor of the current node, calculate the distance from the current node to that neighbor and update its distance if it's shorter than the previous distance.
6. Repeat steps 4 and 5 until the destination node is marked as visited.
7. The shortest path is the sum of the distances from the initial node to the destination node, which is the value of the destination node's distance.

To keep track of the path, you can store the predecessor of each node (the previous node on the shortest path). Then, once the destination node is reached, you can follow the predecessors back to the initial node to obtain the shortest path.

Ceiling and Floor Function

The Ceiling and Floor Functions, on the other hand, are functions that are used in calculus and analysis to **round real numbers to the nearest integer.** The floor function, **denoted by ⌊x⌋, gives the largest integer that is less than or equal to x,** while the ceiling function, **denoted by ⌈x⌉, gives the smallest integer that is**

greater than or equal to x. For example, ⌊3.7⌋ = 3 and ⌈3.7⌉ = 4. These functions are useful in many applications, such as computing limits, derivatives, and integrals of functions involving real numbers.