Introduction

Foundations of Computing Science

Pallab Dasgupta
Professor,
Dept. of Computer Sc & Engg

Comments on Alan Turing's Paper

"On Computable Numbers, with an Application to the Entscheidungs Problem."

This is a bizarre paper. It begins by defining a computing device absolutely unlike anything I have seen, then proceeds to show—I haven't quite followed the needlessly complicated formalism—that there are numbers that it can't compute.

As I see it, there are two alternatives that apply to any machine that will ever be built: Either these numbers are too big to be represented in the machine, in which case the conclusion is obvious, or they are not; in that case, a machine that can't compute them is simply broken!

Any tabulating machine worth its rent can compute all the values in the range it represents, and any number computable by a function—that is, by applying the four operations a number of times—can be computed by any modern tabulating machine since these machines—unlike the one proposed here with its bizarre mechanism——have the four operations hardwired. It seems that the "improvement" proposed by Turing is not an improvement over current technology at all, and I strongly suspect the machine is too simple to be of any use.

If the article is accepted, Turing should remember that the language of this journal is English and change the title accordingly.

Source: http://www.fang.ece.ufl.edu/reject.html

Curriculum

Discrete Structures -- Sets, Relations and Functions; Proof Techniques, Algebraic Structures, Morphisms, Posets, Lattices and Boolean Algebras.

Logic -- Propositional Calculus and Predicate Calculus, Satisfiability and Validity, Notions of soundness and completeness

Languages AND Automata Theory -- Chomsky Hierarchy of Grammars and the corresponding acceptors, Turing Machines, Recursive and Recursively Enumerable Languages; Operations on Languages, closures with respect to the operations.

Computability -- Church-Turing Thesis, Decision Problems, Decidability and Undecidability, Halting Problem of Turing Machines; Problem reduction (Turing and Mapping reduction).

Computational Complexity:

Time Complexity -- Measuring Complexity, The class P, The class NP, NP-Completeness, Reduction, co-NP, Polynomial Hierarchy.

Space Complexity – Savitch's Theorem, The class PSPACE.

Books

1. Michael Sipser -- Theory of Computation, Cengage Learning India.
2. J.P. Trembley and R. Manohar -- Discrete Mathematical Structures with Applications to Computer Science, McGraw Hill Book Co.
3. John E. Hopcroft and J.D.Ullman -- Introduction to Automata Theory, Languages and Computation, Narosa Pub. House, N. Delhi.
4. H.R. Lewis and C.H.Papadimitrou -- Elements of the Theory of Computation, Prentice Hall, International, Inc.

+

Additional materials made available during the course.

Set Theory

A *set* is a group of objects represented as a unit

- Example: Set of odd positive integers less than 50 and divisible by 5

  { 5, 15, 25, 35, 45 }

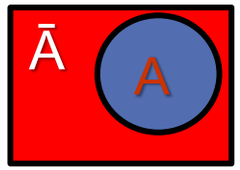
Let A and B be two sets. A is a *subset* of B ($A \subseteq B$) if every element of A is also an element of B, i.e., $x \in A \Rightarrow x \in B$

- A is a proper subset of B ($A \subset B$) if A is a subset of B and $A \neq B$
- Example: $A \subset B$, where

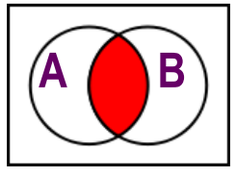
  B = set of odd positive integers less than 50 & divisible by 5 ≡ {5, 15, 25, 35, 45}

  A = set of odd positive integers less than 50 & divisible by 15 ≡ {15, 45}

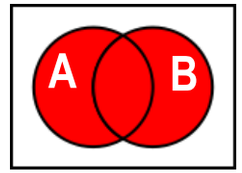
Set Operations



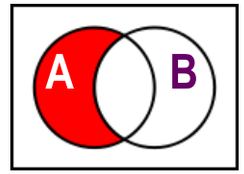
Complement
(Ā)



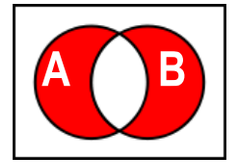
Intersection
(A ∩ B)



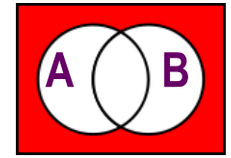
Union
(A U B)



Difference
(A – B) = (A ∩ B')



Symmetric
Difference
(A Δ B)



DeMorgan's Law
(A U B)' = A' ∩ B'

Set Theory (contd…)

Notations

- N = Set of natural numbers
- Z = Set of integers [Z^+ = Set of positive integers]
- R = Set of real numbers [R^+ = Set of positive real numbers]
- Q = Set of rational numbers
- C = Set of complex numbers

Power Set of A, P(A) is the set of all subsets of A

- A = { 1, 2 } then P(A) = { Φ, {1}, {2}, {1, 2} }, Here Φ is *Null Set*

Cartesian Product of A and B (written as A × B) is the set of all pairs where the first element is a member of A and the second element is a member of B

- Example: A = {1, 2} and B = {x, y, z},

  Then, A × B = {(1, x), (1, y), (1, z), (2, x), (2, y), (2, z)}

Function

A *function (or mapping)* is an object that sets up an input-output relationship

- If *f* is a function whose output value is *b* when the input value is *a*, we write *f(a)* = *b*
- Let $f(x_1) = y_1$ and $f(x_2) = y_2$. If $y_1 \neq y_2$, then $x_1 \neq x_2$.

The set of possible inputs to a function is its *domain*

The set of outputs of a function is its *range*

- *f* is a function with domain D and range R is represented as, *f* : D → R

Example:

Consider the function, *f* : {1, 2, 3, 4, 5, 6} → {0, 1, 2}

- The function *f* takes positive integers less than 7 and outputs the result modulo 3; i.e., f(n) = n%3
- Domain of *f* is, D = {1, 2, 3, 4, 5, 6}
- Range of *f* is, R = {0, 1, 2}

Relation

A property whose domain is a set of *k*-tuples ($A \times A \times \ldots \times A$) is called a *relation*

- If k = 2 the relation is called a *binary relation*
  - Example: *less than (<)* is a binary relation

k number of *A*s

A binary relation *R* is an *equivalence relation* if R satisfies the following conditions:

- R is reflexive i.e., $\forall x, xRx$
- R is symmetric i.e., $\forall x \forall y, (xRy \Rightarrow yRx)$
- R is transitive i.e., $\forall x \forall y \forall z, (xRy \text{ and } yRz \Rightarrow xRz)$

A binary relation *R* is a *partial-order relation* if R satisfies the following conditions:

- R is reflexive i.e., $\forall x, xRx$
- R is anti-symmetric i.e., $\forall x \forall y, (xRy \text{ and } yRx \Rightarrow x \equiv y)$
- R is transitive i.e., $\forall x \forall y \forall z, (xRy \text{ and } yRz \Rightarrow xRz)$

Graph

An *undirected graph* is a set of points with lines connecting some of the points

- G = (V, E) where V is the set of vertices and E is the set of edges

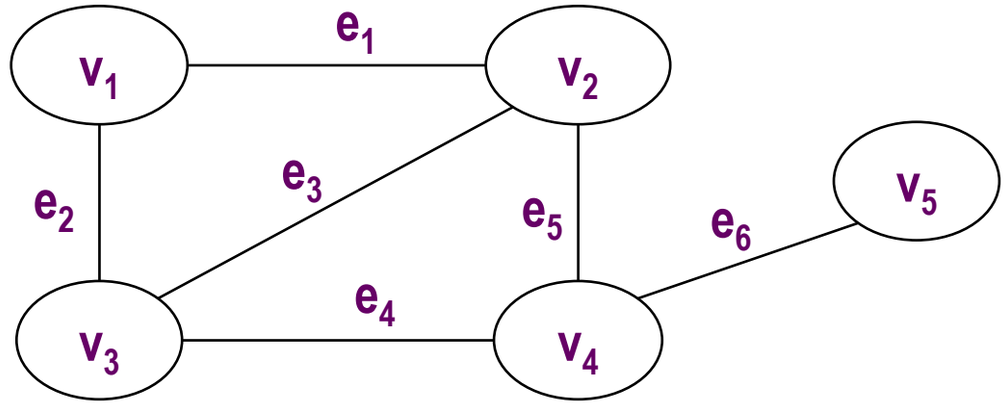
Number of edges incident at a particular node (v) is the *degree [d(v)]* of the node

Example: G = (V, E), where

Set of Vertices: V ={ v_1, v_2, v_3, v_4, v_5 }

Set of Edges: E ={ $e_1, e_2, e_3, e_4, e_5, e_6$ }

Degrees: $d(v_1) = 2$, $d(v_2) = 3$, $d(v_3) = 3$, $d(v_4) = 3$ and $d(v_5) = 1$



Graph (contd…)

G is a *subgraph* of H if the nodes of G are a subset of the nodes H, and the edges of G are the edges of H on the corresponding nodes

- Example: Subgraph H = (V_H,E_H) where;

$V_H = \{v_2, v_3, v_4, v_5\}$ and $E_H = \{e_3, e_4, e_5, e_6\}$

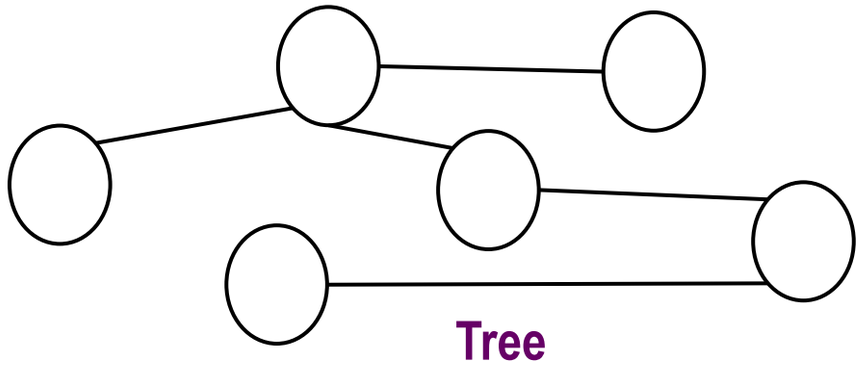
A *path* in a graph is a sequence of nodes connected by edges

- v_1, v_2, v_3, v_4, v_5 is a path

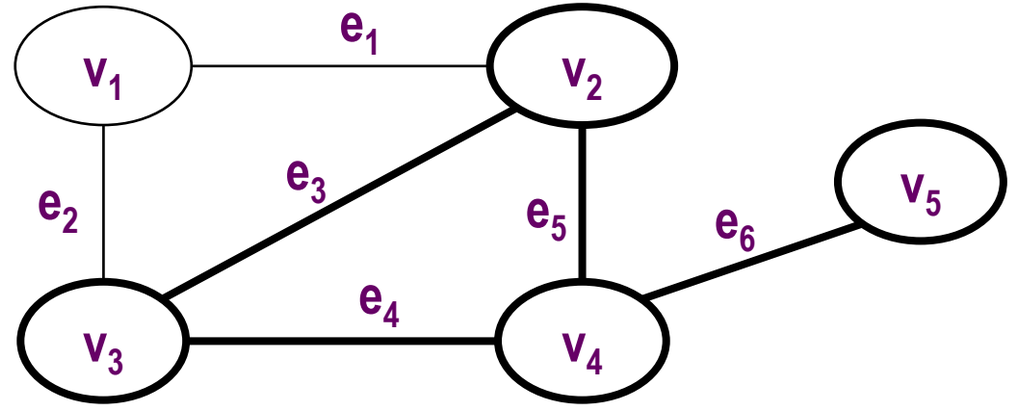
A path is a *cycle* if it starts and ends in the same node

- v_1, v_2, v_4, v_3, v_1 is a cycle

A graph is a *tree* if it is connected and has no cycle



Tree



Graph (contd…)

If a graph has arrows instead of lines, the graph is called a *directed graph*

- Edges from vertex *i* to vertex *j* are represented as pairs *(i, j)*
- Out-degree [$d^+(v)$]: number of arrows pointing from a particular node (v)
- In-degree [$d^-(v)$]: number of arrows pointing to a particular node (v)

Example: G = (V, E) where,

- Set of vertices, V = {1, 2, 3, 4, 5}
- Set of directed edges, E = {(1,2), (1,5), (2,1), (2,3), (2,4), (5,3), (5,4)}
- In-degrees and Out-degrees,

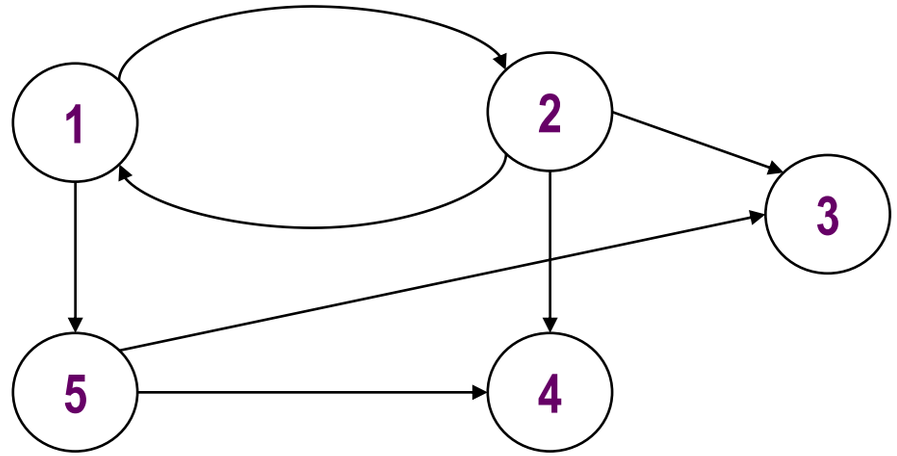
  $d^+(1) = 2$; $d^-(1) = 1$

  $d^+(2) = 3$; $d^-(2) = 1$

  $d^+(3) = 0$; $d^-(3) = 2$

  $d^+(4) = 0$; $d^-(4) = 2$

  $d^+(5) = 2$; $d^-(5) = 1$



Boolean Logic

It is a mathematical system built around two values TRUE and FALSE

- **The value TRUE and FALSE are called Boolean values and are often represented by the values 1 and 0**

Basic operations are as follows:

- **Negation (~) ,Conjunction (Λ) ,Disjunction (V)**

Truth Table of Basic operations:

Negation

a	~a
0	1
1	0

Conjunction

a	b	a Λ b
0	0	0
0	1	0
1	0	0
1	1	1

Disjunction

a	b	a V b
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Logic (contd…)

Several other Boolean operations occasionally appear

- **Exclusive-Or (XOR): $P \oplus Q \equiv (\sim P \wedge Q) \vee (P \wedge \sim Q) \equiv \sim(P \Leftrightarrow Q)$**
- **Implication: $P \Rightarrow Q \equiv \sim P \vee Q$**
- **Equality: $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$**

Distributive law:

- **$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$**
- **$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$ [Dual]**

Commutative law:

- **$P \vee Q \equiv Q \vee P$ and $Q \wedge R \equiv R \wedge Q$**