Logic and Discrete Structures

Freshmen's Course

Computers and Information Technology (EN)

Dr. Antonius N. STANCIU, PhD

Informal Proofs

- Definition
- Proof Primers
  - An *informal proof* is a demonstration that some statement is true
- Elementary Logic Operators
  - Negation (NOT)
  - Conjunction (AND)
  - Disjunction (OR)

Negation

- Changes the statements into its opposite

S	Non *S*
F (0)	***T (1)***
T (1)	***F (0)***

- S = **Earth is a star**
- Non S = **Earth is not a star** (rather than non-Earth is a star)

Conjunction and disjunction

A	B	$A \cdot B$	$A+B$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Exclusive OR

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Conditional Statements

- If A, then B

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

- There is no connection between the hypothesis and the conclusion
  - If the Moon is made out of cheese, then $1 = 2$.
  - If $1 = 1$, then the Moon is made out of cheese.

Conditional Statements

- When the hypothesis of a conditional is false, we say that the conditional is vacuously true
  - If $1=2$, then $39=12$
- If the conclusion is true, we say that the conditional is trivially true
  - If $1 = 2$, then $2 + 2 = 4$
- The converse does not always have the same truth value

Equivalent Statements

S	*T*
¬ ***(A or B)***	***(***¬ ***A) and (***¬ ***B)***
¬ (A ***and*** B)	(¬ A) ***or*** (¬ B)
A and (B ***or*** C)	***(A and*** B) ***or*** (A ***and*** C)
A or (B ***and*** C)	***(A or*** B) ***or*** (A ***or*** C)
if A, then B	***if*** ¬ ***B, then*** ¬ ***A***
if A, then B	***(***¬ A) ***or*** B
¬ ***(if A, then B)***	***A and*** (¬ B)

About (Prime) Numbers and Divisibility

- Integers:
  - Odd
    - …, -5, -3, -1, 1, 3, 5, …
  - Even:
    - …, -4, -2, 0, 2, 4, …
- The number d divides the number n (or $d \mid n$), if $d \neq 0$ and there is a number k, so $n = dk$.

Divisibility Properties

- If $d|a$ and $a|b$, then $d|b$
- If $d|a$ and $d|b$, then $d|(ax + by)$ for any integers x and y
- The number $p > 1$ is a *prime* number if its only positive prime dividers are 1 and p
- The prime numbers are extremely important for the computer science and engineering

Proof Techniques

Exhaustive Checking

- If n is an integer and $2 \leq n \leq 7$, then $n^2 + 2$ is not divisible by 4
  - Demonstration: $6, 11, 18, 27, 38, 51$
- Exhaustive checking cannot be used to prove a statement that involves infinitely many things to check (at least not yet and not on actual computers 😢)
- Sometimes, a counterexample helps

Proof Techniques

Conditional Proof

- If A, then B
  - It is based on the assumption that the hypothesis is true
  - It can be performed in several steps, until reaching the conclusion B
    - If $\boldsymbol{A}$, then M
    - If M, then N
    - If N, then P
    - If P, then $\boldsymbol{B}$

- Example 1)
  - If x and y are odd numbers, then their sum will be even
    - $x + y = (2k + 1) + (2m + 1) = 2k + 2m + 2 = 2(k + m + 1)$

- Example 2)
  - If $d|a$ and $a|b$, then $d|b$
    - $b = an = (dm)n = d(mn)$

Proof Techniques

Proof by Contradiction (Refutation)

- A contradiction is a false statement
- We start out by assuming that the statement to be proved is false. Then we argue until we reach a contradiction
- (A ⇒ B is the same as A · ¬B = False)
- Example
  - If n is an integer, then $n^2 + 2$ is not divisible by 4
    - We assume that this statement is false. Then $4 \mid (n^2 + 2)$ for an integer n
    - It gives $n^2 + 2 = 4k,$ for an integer k. If n is even, then $n = 2m$
    - Substituting n we have $4k = n^2 + 2 = (2m)^2 + 2 = 4m^2 + 2,$ or $2k = 2m^2 + 1$
    - So n cannot be even. If it is not even, it can only be odd. If it is odd then
    - $4k = n^2 + 2 = (2m + 1)^2 + 2 = 4m^2 + 4m + 3$, same as $4k - 4m^2 - 4m = 3$
    - But 3 is not divisible by 4!

Proof Techniques

If and Only If (Iff) Proofs

- A iff B is the same as A then B **AND** if B then A
- Example
  - x is odd if and only if (iff) $x^2 + 2x + 1$ is even
    - x is odd iff $x = 2k + 1$ for any integer k (by definition)
    - iff $x + 1 = 2k + 2$ for any integer k (algebra)
    - iff $x + 1 = 2m$ for any integer m (algebra)
    - iff $x + 1$ is even (by definition)
    - iff $(x + 1)^2$ is even (Exercise 8a in the book)
    - iff $x^2 + 2x + 1$ is even (algebra) QED
- A *constructive approach* is usually preferred

Sets

- Definition:
  - A collection of things (often called *elements*, *objects* or *members*)
- Synonyms:
  - Set, collection, bunch, group, class…
- The belonging (membership) of an element to a set:
  - $x \in S$
  - $x, y \in S$
- Notations
  - Symbolic: S
  - Explicit: $\{ a, b, c, d \}$
  - Analytic: $\{ x \mid P \}$

Set Notations

- $S = \{x, y, z\}$
- $A = \{x, \{x, y\}\}$
  - $x \in A$
  - $\{x, y\} \in A$
- $\{H, E, L, L, O\}$
- $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$
  - $\{1, 2, \ldots, 12\}$
  - $\{1, 2, 3, \ldots, 11, 12\}$

Set Notation

- Singleton
  - We call singleton a set with only one member
  - $\{ a \}, \{ 123 \}$ etc.
- Void set (or null set, or empty set)
  - $\{ \}$
  - $\varnothing$
- Examples:
  - $\{ \{ \} \}, \{ \varnothing \}$

About sets

- Sets' equality:
  - $A = B$
  - $\{u, g, h\} = \{h, u, g\}$
  - $A \neq B$
  - $\{a, b, c\} \neq \{a, b\}$
  - $\{a\} \neq \varnothing$
- Sets' characteristics:
  - No order
  - No redundancy
- Sets:
  - Finite
  - Infinite

About sets

- Numbers:
  - Natural numbers:
    - $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$
    - $\mathbb{N}^* = \{1, 2, 3, \ldots\}$
  - Integers:
    - $\mathbb{Z} = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$

- Analytical representation (through its properties)
  - Odd integer numbers:
    - $A = \{ x \mid x = 2k + 1, k \in \mathbb{Z} \}$
  - Rational numbers:
    - $\mathbb{Q}$
  - Real numbers:
    - $\mathbb{R}$

Subsets

- If all the elements of A are to be found in B, then A is called a subset of B
  - $A \subset B$

- Examples:
  - $\{a, b\} \subset \{a, b, c\}$
  - $\{0, 1, 2\} \subset \mathbb{N}$
  - $\mathbb{N} \subset \mathbb{Z}$
  - $A \subset A$
  - $\varnothing \subset A$
  - $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$

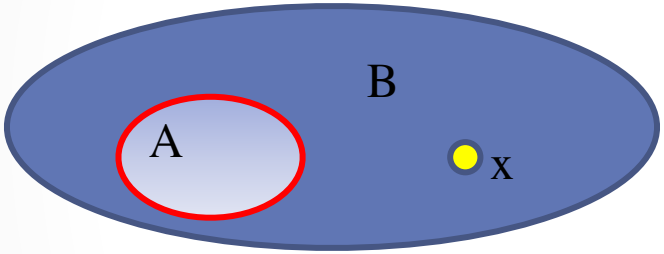
Subsets

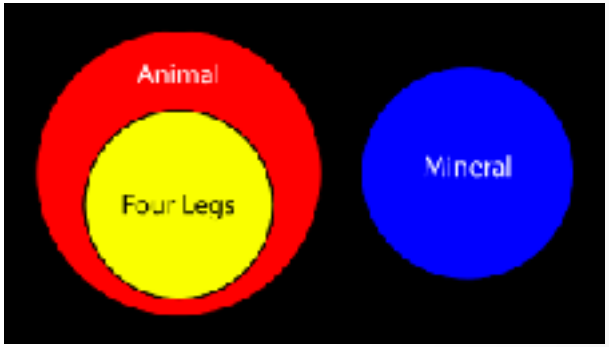
- If A is not a subset of B we write $A \not\subset B$
  - $\{a, b\} \not\subset \{a, c\}$
  - $\{0, -1, -2\} \not\subset \mathbb{N}$

- The conceptof subset is different from the concept of membership
  - If $A = \{a, b, c\}$, then:
    - $\{a\} \subset A$
    - $a \in A$
    - $\{a\} \notin A$
    - $a \not\subset A$
  - If $A = \{a, \{b\}\}$, then:
    - $a \in A,\ \{b\} \in A,\ \{a\} \subset A$ și $\{\{b\}\} \subset A$. In turn, neither
    - $b \notin A$ nor $\{b\} \not\subset A$

Subsets

- The Venn diagram (actually is Euler's) of the subset A included in the set B





- John VENN (1834–1923)
- Leonhard EULER (1707–1783)

The Power of a Set

- This functions returns all the subsets of the set given as an argument
- Example:
  - Given the set $S = \{a, b, c\}$ we have:
  - $\text{power}(S) = \{\varnothing\ , \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, S\}$

Sets' Equality

- $A = B$ means $A \subset B$ **and** $B \subset A$

Statement to be proved	Demonstration Strategy
$\boldsymbol{A \subset B}$	For any and each x ∈ A, we have to show that x ∈ B
$\boldsymbol{A \not\subset B}$	We find an element x ∈ A, that also obeys x ∉ B
$\boldsymbol{A = B}$	We show that A ⊂ B **and** B ⊂ A

Subset Proofs

- Given:
  - $A = \{x \mid x \text{ prime and } 42 \leq x \leq 51\}$
  - $B = \{x \mid x = 4k + 3 \text{ and } k \in \mathbb{N}\}$
- Prove that:
  - $A \subset B$
- Proof:
  - Take x from A. Then either $x = 43$ or $x = 47$
  - One can see that $43 = 4(10) + 3$ and $47 = 4(11) + 3$.
  - In both cases $x \in B$
  - Therefore $A \subset B$ (Q.E.D.)

Non-Subset Proofs

- Given:
  - $A = \{x \mid 3k + 1 \text{ and } k \in \mathbb{N}\}$
  - $B = \{x \mid 4k + 1 \text{ and } k \in \mathbb{N}\}$
- Prove that:
  - $A \not\subset B$
  - $B \not\subset A$
- Proof:
  - We have $A = \{1, 4, 7, \ldots\}$ and $B = \{1, 5, 9, \ldots\}$
  - Since $\{4\} \in A$ **and** $\{4\} \notin B$, it results that $A \not\subset B$
  - Since $\{5\} \in B$ **and** $\{5\} \notin A$, it results that $B \not\subset A$ (Q.E.D.)

Equality Proofs

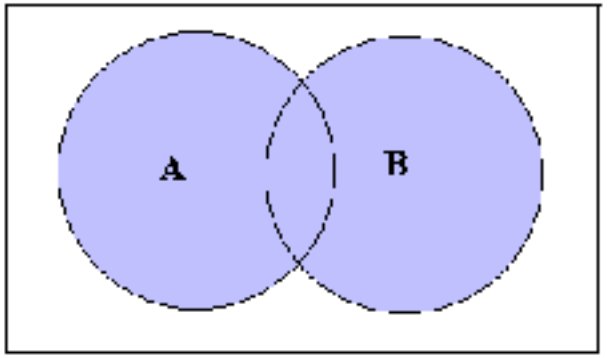
- Given:
  - $A = \{x \mid x \text{ prime and } 12 \leq x \leq 18\}$,
  - $B = \{x \mid x = 4k + 1 \text{ and } k \in \{3, 4\}\}$
- Prove that:
  - $A = B$
- Proof:
  - If $x \in A$, then either $x = 13$ or $x = 17$
  - We check that $13 = 4(3) + 1$ and $17 = 4(4) + 1$, so $x \in B$ and, consequently, $A \subset B$ (†)
  - If $x \in B$, then either $x = 4(3) + 1$ or $x = 4(4) + 1$, so $x \in A$ and, consequently $B \subset A$ (‡)
  - From (†) **and** (‡) it results that $A = B$ (Q.E.D.)

Operations on Sets

- Union
- Intersection
- Universal Complemen

Union

- $A \cup B = \{x \mid x \in A \textbf{ or } x \in B\}$
- If $A = \{a, b, c\}$ and $B = \{c, d\}$, then $A \cup B = \{a, b, c, d\}$



Union's Properties

- $A \cup \varnothing = A$
- $A \cup B = B \cup A$ ($\cup$ is commutative)
- $A \cup (B \cup C) = (A \cup B) \cup C$ ($\cup$ is associative)
- $A \cup A = A$
- $A \subset B$ iff $A \cup B = B$

Subset's Condition

- Example
  - Prove that $A \subset B$ iff $A \cup B = B$

- Proof
  - First, we prove that $A \subset B$ implies $A \cup B = B$
    - If it is true and we have $x \in A \cup B$, then $x \in A$ or $x \in B$
    - Since we assumed $A \subset B$, we have $x \in B$, meaning $A \cup B \subset B$
    - Since we always have $B \subset A \cup B$, it means that $A \cup B = B$ (Q.E.D. 1)
  - Then, we prove that $A \cup B = B$ implies $A \subset B$
    - If it is true and we have $x \in A$, then $x \in A \cup B$
    - Since we have assumed that $A \cup B = B$, it means that $x \in B$, giving $A \subset B$ (Q.E.D. 2)
  - Out of (Q.E.D. 1) **and** (Q.E.D. 2) it results that $A \subset B$ iff $A \cup B = B$ (Q.E.D.)

Notations for Unions

- In the case of union of n sets $A_1, \ldots, A_n$, we may write:

$$\bigcup_{i=1}^{n} A_i = A_1 \cup \cdots \cup A_n$$

- If we have the union of an infinite number of sets $A_1, A_2, \ldots, A_n, \ldots,$ we may write it as:

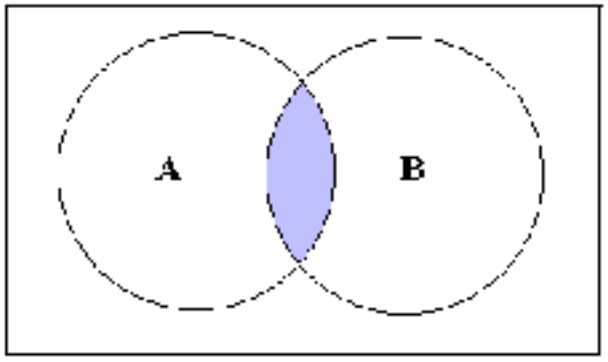
$$\bigcup_{i=1}^{\infty} A_i = A_1 \cup \cdots \cup A_n \cup \cdots$$

- If I is a set of indices and A_i is a set set for each $i \in I$, then the union of all these sets may be written as:

$$\bigcup_{i \in I} A_i.$$

Intersection

- $A \cap B = \{x \mid x \in A$ **and** $x \in B\}$
- If $A = \{a, b, c\}$ and $B = \{c, d\}$, then $A \cap B = \{c\}$
- if $A \cap B = \varnothing$, then A and B are called ***disjoints***



Intersection's Properties

- $A \cap \varnothing = \varnothing$
- $A \cap B = B \cap A$ ($\cap$ is commutative)
- $A \cap (B \cap C) = (A \cap B) \cap C$ ($\cap$ is associative)
- $A \cap A = A$
- $A \subset B$ iff $A \cap B = A$

Notations of Intersections

- If we have the intersection of n sets $A_1, \ldots, A_n$, we may write it as:

$$\bigcap_{i=1}^{\infty} A_i = A_1 \cap \cdots \cap A_n$$

- If we have the intersection of an infinite number of sets $A_1, A_2, \ldots, A_n, \ldots$, we may write it as:

$$\bigcap_{i=1}^{\infty} A_i = A_1 \cap \cdots \cap A_n \cap \cdots$$

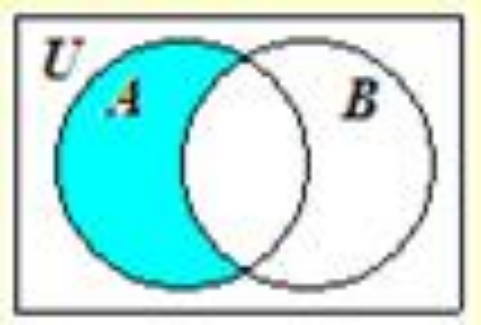
- If I is a set of indices and A_i is a set for every $i \in I$, then the intersection of all these sets may be written as:

$$\bigcap_{i \in I} A_i.$$

The Difference of Sets

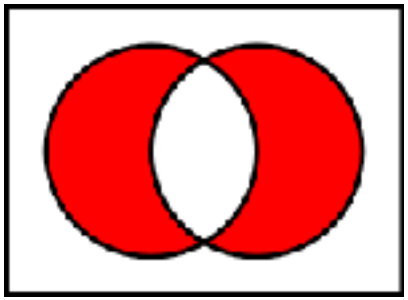
- If A and B are sets, then the difference $A - B$ is the set of all elements of A that are not in B
  - $A - B$ is also called the relative complement of B over A

- $A - B = \{x \mid x \in A \textbf{ and } x \notin B\}$



The Symmetric Difference

- $A \oplus B = \{x \mid$ either $x \in A$, or $x \in B$ (but not both!)$\}$

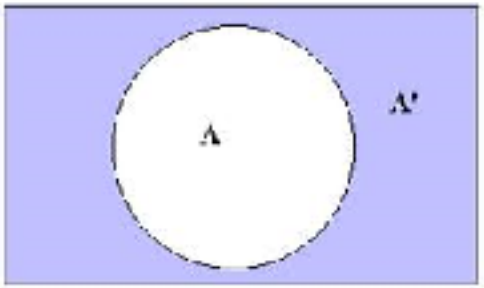


- $A \oplus B = (A \cup B) - (A \cap B)$

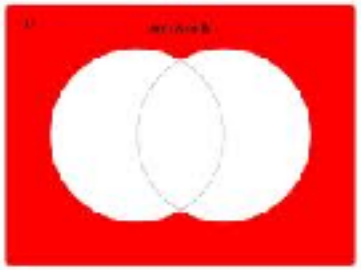
- Can you prove that $(A \oplus B) \oplus C = A \oplus (B \oplus C)$?
  - One solution (among many others) is to draw a Venn diagram for each member of the afore mentioned equation

The Universal Complement

- If we are talking about the situation when a certain set is always a subset of a set U (called Universe), then the difference $U - A$ will be called universal complement of A and it will be marked as A'



- Example: $(A \cup B)'$



Combining Set Operations

- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
  - $\cap$ is distributive over $\cup$
- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
  - $\cup$ is distributive over $\cap$
- $A \cap (A \cup B) = A$
  - absorption law
- $A \cup (A \cap B) = A$
  - absorption law

Combining Set Operations

- Exemple:
  - prove that $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

- Proof: $x \in A \cap (B \cup C)$
  - iff $x \in A$ and $x \in (B \cup C)$
  - iff $x \in A$ and either $x \in B$ or $x \in C$
  - iff either $(x \in A$ and $x \in B)$ or $(x \in A$ și $x \in C)$
  - iff $x \in (A \cap B) \cup (A \cap C)$ (Q.E.D.)

Complement's Properties

- $(A')' = A$
- $\varnothing' = U$ and $U' = \varnothing$
- $A \cap A' = \varnothing$ and $A \cup A' = U$
- $A \subset B$ iff $B' \subset A'$
- $(A \cup B)' = A' \cap B'$
  - de Morgan's Law
- $(A \cap B)' = A' \cup B'$
  - de Morgan's Law
- $A \cap (A' \cup B) = A \cap B$
  - Absorption Law
- $A \cup (A' \cap B) = A \cup B$
  - Absorption Law

Counting the Elements of Finite Sets

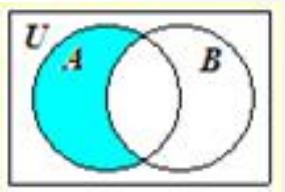
- The size of a set is given by its ***cardinality***, written as $|S|$
- For instance, if $S = \{a, b, c\}$,
  - then $|S| = |\{a, b, c\}| = 3$
- Union Rule:
  - $|A \cup B| = |A| + |B| - |A \cap B|$

Counting the Elements of Finite Sets

- The Union Rule can be extended over more than two sets:
  - $|A \cup B \cup C| = |A \cup (B \cup C)|$
  - $= |A| + |B \cup C| - |A \cap (B \cup C)|$
  - $= |A| + |B| + |C| - |B \cap C| - |A \cap (B \cup C)|$
  - $= |A| + |B| + |C| - |B \cap C| - |(A \cap B) \cup (A \cap C)|$
  - $= |A| + |B| + |C| - |B \cap C| - |A \cap B| - |A \cap C| + |A \cap B \cap C|$

Counting the Elements of Finite Sets

- Difference Rule: $|A - B| = |A| - |A \cap B|$
- The Difference rule is easily observable (and understandable) with the help of Venn diagrams
- Other two special cases are also intuitive and can be expressed as:
  - If $B \subset A$, then $|A - B| = |A| - |B|$
  - If $A \cap B = \varnothing$, then $|A - B| = |A|$



Bags *(Multisets)*

- Bags are collections of objects (just like the sets are) which allow the reputations of their elements

- The main characteristics are:
  - The elements of a bag may occur more than once
  - There is no order among the elements of a bag

- One can define:
  - Equality ($[h, u, g, h] = [h, h, g, u]$, but $[h, u, g, h] \neq [h, u, g]$)
  - Inclusion ($[a, b] \subset [a, b, a]$, but $[a, b, a] \not\subset [a, b]$)

Bags *(Multisets)*

- One can define:
  - The bags' sum, written as $A + B$ (if x appears m times in A and n times in B, then x appears $m + n$ times in $A + B$)
    - For instance: $[2, 2, 3] + [2, 3, 3, 4] = [2, 2, 2, 3, 3, 3, 4]$
  - Union and Intersection. (Let A and B be bags and let m and n the number of occurrences of x in A and B, respectively). Put the larger value of m și n occurrences of x in $A \cup B$ and the smaller number of m and n occurrences of x in $A \cap B$.
    - For instance: $[2, 2, 3] \cup [2, 3, 3, 4] = [2, 2, 3, 3, 4]$ and
    - $[2, 2, 3] \cap [2, 3, 3, 4] = [2, 3]$
- Homework:
  - Cantor's Set Theory
  - Russell's Paradox

Ordered structures

- Tuples
  - They are collections of **ordered** objects, called elements
  - If a tuple has n elements, then it will be called an *n-tuple*
  - The particular cases of 2-tuple and 3-tuple are called *double* and triple, respectively
  - Two n-tuples, $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$, are equal if $x_i = y_i$ for i between 1 and n

- Important tuple characteristics
  - The elements can be repeated
  - The elements are ordered

Cartesian Product of Sets

- If A and B are sets, then their cartesian product denoted by $A \times B$, is the set of all doubles (a, b), knowing that $a \in A$ and $b \in B$
  - $A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$
- If one of the sets is void set, then the cartesian product of that set with another set is also a void set

Cartesian Product of Sets

- The cartesian product can be extended on as many sets it is needed
- The cartesian product of a set with itself n times can be written as A^n
- Please remember that $A^0 = \{( )\}$ și $A^1 = \{(a)\}$
  - Note that $A^0 \neq \{\ \}$ și $A^1 \neq A$
  - Please write, for example, A^0, A^1, A^2 și A^3, if $A = \{a, b, c\}$

Cartesian Product of Sets

- Working with tuples requires random access to any of its elements

- For instance, if $t \in A \times B \times C$, then t can be written in several ways:

$$(t_1, t_2, t_3),$$
$$(\iota(1), \iota(2), \iota(3)),$$
$$(t[1], t[2], t[3]),$$
$$(t(A), t(B), t(C)),$$
$$(A(\iota), B(\iota), C(\iota)).$$

Vectors, Arrays, Articles

- În computer science, an unidimensional array of n elements is an n-tuple in the cartesian product A^n
  - We can look at the cartesian product A^n aș being the set of all unidimensional arrays of *n* elements of the set *A*
  - If $x = (x_1, \ldots, x_n)$, then the element x_i is usually denoted, in computers, as $x[i]$

Vectors, Arrays, Articles

- A two-dimensional array, often called a *matrix*, may be perceived as a table which is indexed by rows and columns

- If we have a matrix x of m rows and n columns, we say that x is an array of m by n elements
  - For instance, if x is a 3 by 4 array, then it can be represented as:

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{12} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix}$$

Vectors, Arrays, Articles

- The same x array can be represented also as a triple with quadruple elements:

$$x = ((x_{11}, x_{12}, x_{13}, x_{14}), (x_{21}, x_{22}, x_{23}, x_{24}), (x_{31}, x_{32}, x_{33}, x_{34}))$$

- In computer programming, x_{ij} is usually written as `x[i,j]`
- One can see that the cartesian product $(A^4)^3$ is the set of all two-dimensional arrays over A having 3 rows and 4 columns
- This can be extended: for instance, $((A^5)^7)^4$ is the set of all cubic arrays over A made of the quadruples having septuple elements which are quintuple elements from A
  - Tuples are:
    - Easily understandable
    - The basic building blocks for representing the information in computers
    - Easy implementable in computers (please see the example in the handbook, as homework)

Lists

- They are finite successions of zero or more elements that can repeat themselves
- The main difference compared to tuples lies in the access pattern:
  - Random access for tuples
  - Sequential access for lists
- Specifically, în lists we have straight access to only two elements:
  - The list's head, meaning its first element
  - The list's tail, meaning the list of all other elements

Lists

- **Usual notations and elementary operations:**

- The void list (or the empty list): $< >$
- $L = <w, x, y, z>$ is a list of length 4
- For list L we have the functions `head(`L`)` and `tail(`L`)` to extract the head and, respectively, the tail of L
  - For example, `head(`L`)` $= w$ and `tail(`L`)` $= <x, y, z>$
- The void list has no head nor tail
- One can add an element to a list by making it the head of the new list
  - `cons(`h`, `L`)` $= <h, w, x, y, z>$
  - `cons(head(`L`), tail(`L`))` $= L$

Lists

- There are no restrictions regarding the type of elements contained within a list
- That's why it is a recommended method for the representation of information in computers
- For instance:

L	head(L)	tail(L)
$\langle a, \langle b \rangle \rangle$	a	$\langle \langle b \rangle \rangle$
$\langle \langle a \rangle, \langle b, a \rangle \rangle$	$\langle a \rangle$	$\langle \langle b, a \rangle \rangle$
$\langle \langle \langle\ \rangle, a, \langle\ \rangle \rangle, b, \langle\ \rangle \rangle$	$\langle \langle\ \rangle, a, \langle\ \rangle \rangle$	$\langle b, \langle\ \rangle \rangle$

Lists

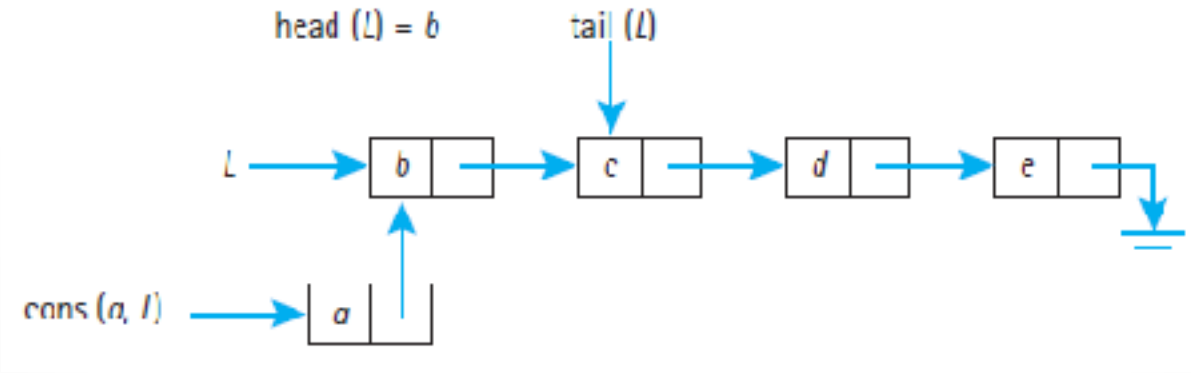
- If all the elements of a list L belong to a certain set A, then we say that list L is a list over set A
- For example, if $L = \{a, b, c\}$, then the next lists are lists over set A:

$$\langle\,\rangle,\ \langle a\rangle,\ \langle a, b\rangle,\ \langle b, a\rangle,\ \langle b, c, a, b, c\rangle$$

- The collection of all lists over set A is usually denoted by `lists(A)`

Lists' Representation in Computers

- Let $L = <b, c, d, e>$



Strings and Languages

- A string is an ordered and finite succession of elements that are placed next to each other (by juxtaposition)
- The elements of a string are taken from a set called alphabet
- If A îs an alphabet, the the strings made with elements from A are called strings over A
  - Examples of strings over $A = \{a, b, c\}$: a, ba, bba, $aacabb$
- The string with zero elements is called the void string (or the empty string) and is usually denoted using the Greek letter Λ
- The length of a string is given by the number of its elements
  - For instance, $|\Lambda| = 0$ and $|\text{aacabb}| = 6$

Strings' Concatenation

- It is made by merely juxtaposition
  - For instance, if $s = aab$ and $t = ba$ are two strings over the alphabet $\{a, b\}$, then the concatenation $st = aabba$
- The empty string is a neutral element for concatenation
  - $s\Lambda = \Lambda s = s$
- There is a powerful association between strings and lists in their representation in computers, coming from the necessity to decompose a string into its elements, which can be then represented as a list
  - For instance, the string $aacabb$ can be represented as the list $<a, a, c, a, b, b>$

Languages

- Are sets of strings
- If A is an alphabet, then a language is a set of strings over A
- The set of all strings over A is written as A^*
- Therefore, any language over A is a subset of A^*

Languages

- If A is an alphabet, then the sets $\varnothing$, $\{\Lambda\}$, A, and A^* are four examples of languages over A
- For example, if $A = \{a\}$:
  - $\varnothing$, $\{\Lambda\}$, $\{a\}$, and $\{\Lambda , a, aa, aaa, \ldots\}$ are four languages over A
- For any natural number n, the concatenation of s with itself n times is written s^n
- For instance:
  - $s^0 = \Lambda$, $s^1 = s$, $s^2 = ss$, and $s^3 = sss$

$$\{a^n \mid n \in \mathbb{N}\} = \{\Lambda , a, aa, aaa, \ldots\}.$$
$$\{ab^n \mid n \in \mathbb{N}\} = \{a, ab, abb, abbb, \ldots\}$$
$$\{a^n b^n \mid n \in \mathbb{N}\} = \{\Lambda , ab, aabb, aaabbb, \ldots\}$$
$$\{(ab)^n \mid n \in \mathbb{N}\} \quad \{\Lambda , ab, abab, ababab, \ldots\}.$$

Numerals

- Are written numbers
- In terms of strings, numerals are nonempty strings that represent a number
- The roman numerals are strings that represent positive integers using the alphabet $\{I, V, X, L, C, D, M\}$
- The decimal numbers represent natural numbers by using the alphabet $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- The binary numbers are based on the binary alphabet $\{0, 1\}$
  - For instance, MDCLXVII = 1667 = 11010000011

Languages' Product

- Being sets, the languages can be combined using the usual set operations (union, intersection, difference, complement)
- Two languages, L and M, can be combined to obtain the set of all concatenations of strings in L with strings in M
- This new language is called the product of L and M

$$LM = \{st \mid s \subset L \text{ and } t \subset M\}$$

- For example, if $L = \{ab, ac\}$ and $M = \{a, bc, abc\}$, then $LM = \{aba, abbc, ababc, aca, acbc, acabc\}$

Languages' Product Properties

- Neutral elements: $L\{\Lambda\} = \{\Lambda\}L = L$
  $L\varnothing \quad \varnothing L \quad \varnothing.$
- Associativity: $L(MN) = (LM)N$
- Noncommutativity: $LM \neq ML$
- For any natural number n, the product of a language L with itself n times is defined as:

$$L^n = \{s_1 s_2 \dots s_n \mid s_k \in L \text{ for each } k\}$$

$$L^0 = \{\Lambda\}$$

The Languages' Closure

- If L is a language, then its closure is usually denoted by L^* and represents the set of all possible string concatenations from L

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots \cup L^n \cup \cdots$$

- $x \in L^*$ iff $x \in L^n$. In other words, $x \in L^*$ iff either $x = \Lambda$, or $x = l_1 l_2 \ldots l_n$

- The positive closure of language L is usually denoted as L^+ and is defined as:

$$L^+ = L^1 \cup L^2 \cup L^3 \cup \cdots$$

The Languages' Closure

- From the definition, it results that $L^* = L^+ \cup \{\Lambda\}$, but it is not necessarily true that $L^+ = L^* - \{\Lambda\}$
- For instance, if $L = \{\Lambda , a\}$, then $L^+ = L^*$
- The languages' closure properties:

$\{\Lambda\}^* - \varnothing^* - \{\Lambda\}$.

$\Lambda \in L$ if and only if $L^+ - L^*$.

$L^* = L^* L^* = (L^*)^*$.

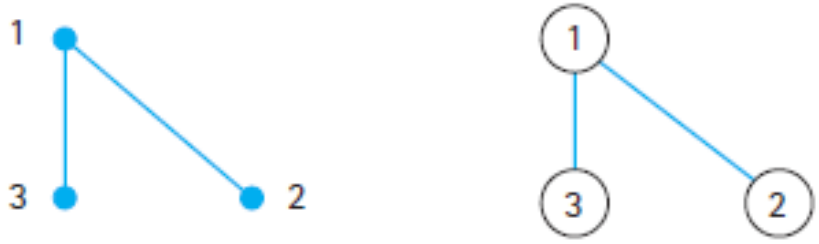
$(L^* M^*)^* = (L^* \cup M^*)^* = (L \cup M)^*$.

$L\,(M\ L)^* - (LM)^* L$.

Graphs

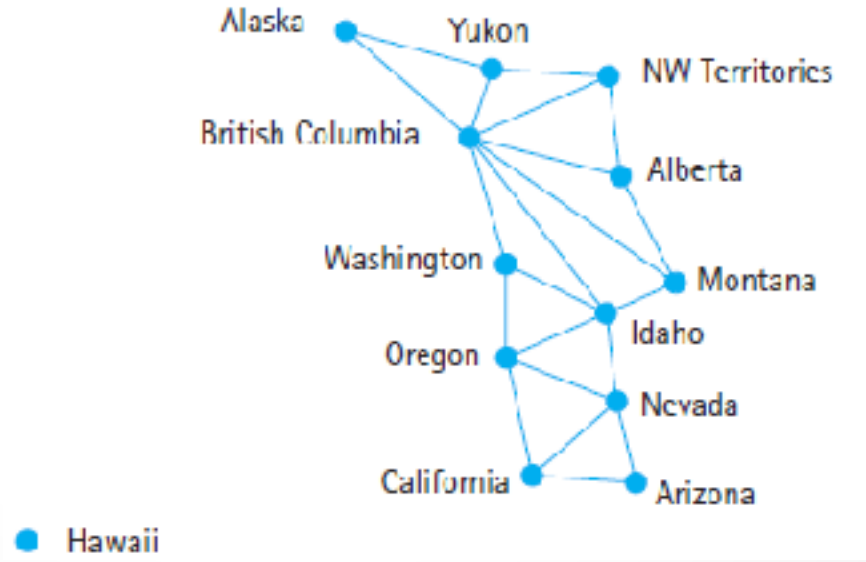
- Are sets of objects in which some objects are connected to each other
- The objects are named *vertices* or *nodes*
- The connections are called *edges*
- They can be graphically represented, which somehow explains the given name to this structures

Representing graphs



Graphs' Coloring

- A graph is n-colorable if there is an assignment of n colors to its nodes such that any two distinct adjacent nodes have distinct colors
- The smallest number of colors is called the chromatic number of that graph

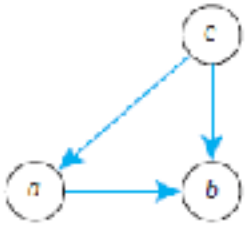


Graphs' Coloring

- A graph is said to be *planar* if it can be represented or drawn on a plane such that no edges intersect
- A complete graph (all nodes are connected with all others) with four nodes is planar, while a five node one isn't
- Any planar graph is 4-colorable (Keneth APPEL and Wolfgang HAKEN, 1976)

Terminology

- A *directed* graph has each edge pointing in one direction



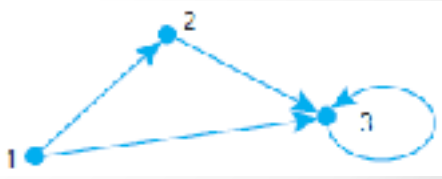
Source

Drain

- The *degree* of a node is given by the number of edges that it touches; we add 2 to that number for each loop (a loop is an edge that starts and ends at the same node

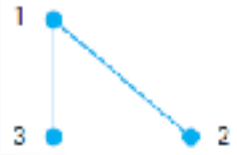
- For oriented graphs *in-degrees* și *ex-degrees*

- In digraphs we have *sources* and *drains*



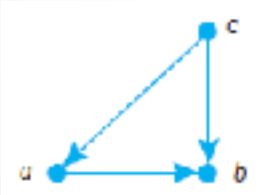
Computational Representation of Graphs

- Computational representation of undirected graphs



$\{1, 2, 3\}$ $\{\{1, 2\}, \{1, 3\}\}$

- Computational representation of directed graphs



$\{a, b, c\}$ $\{(a, b), (c, b), (c, a)\}$

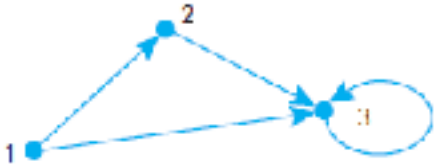
Weighted Graphs

- The graphs can have information attached to each edge
- The attached information is usually called a weight
- Analytical (or computational) representations of a weighted edge can be done with 3-tuples (triples)

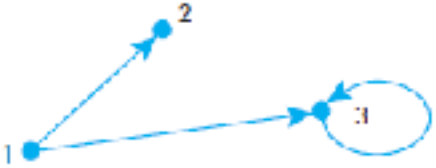
$$(a,\ b,\ w)$$

Particularities

- Graphs and binary relations
  - One can observe that any binary relation R on a set A can be represented as a directed graph of the form $G = (A, R)$, with nodes A and edges R
  - For instance, let $A = \{1, 2, 3\}$ and $R = \{(1,2), (1, 3), (2, 3), (3, 3)\}$



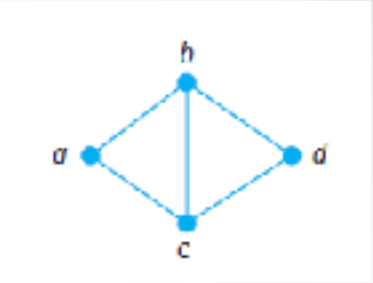
- Subgraphs are portions of some graphs that require separate treatments and discussions
  - A graph (V', E') is a subgraph of graph (V, E) if $V' \subset V$ and $E' \subset E$



Paths in Graphs

- Going from one vertex to another along connected edges
- The path from x_0 to x_n is $x_0, x_1, \ldots, x_n$, as long there is an edge x_{i-1}, x_i for each $1 \leq i \leq n$
- A cycle is a path which starts and end în the same vertex, without repeating edges
- The graphs with no cycles are called non-cyclical graphs
- The length of a path is given by the number of edges involved
- The graphs are connected if there is a path between any pair of vertices

Example



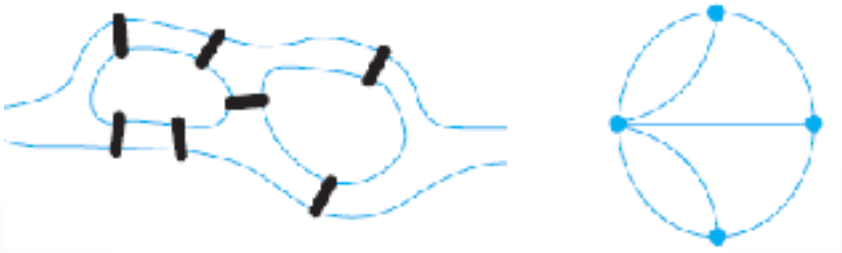
- The b, c, d, b, a path visits b twice and its length is 4
- The a, b, c, b, d path visits b twice, uses bc edge twice and its length is 4
- The a, b, c, a path is cycle of length 3
- The a, b, a path has the length 2, but it is not a cycle

One problem

- Please draw the diagram on the left without lifting the pen from the pencil and without drawing and edge twice
- The second drawing demonstrates the graph nature of the given problem

Another problem

- It is the famous problem of the seven bridges of Königsberg which connected, around XVIII century, two islands of the Pregel river with its banks
- The city has to be visited by crossing a bridge only once
- Leonhard EULER (1707–1783) demonstrated that there is no such a solution and found the general conditions for it to exists



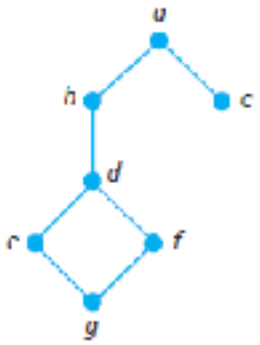
Graph Traversal

- Traversing a graph starts at a vertex v and visits all vertices x reachable by a path
- An already visited node will not be visited again
- There are two largely used algorithms:
  - Breadth First
  - Depth First

Breadth-First Algorithm

- The procedure `visit(v, k)` is called, so:

for $k := 0$ to $n - 1$ do visit(v, k) od.



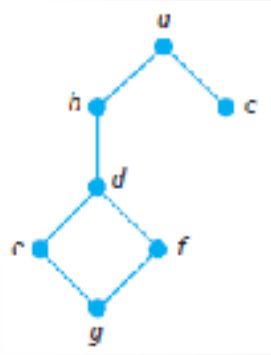
- Starting at a, We have several solutions:

$a\ b\ c\ d\ e\ f\ g$ $a\ b\ c\ d\ f\ e\ g$ $a\ c\ b\ d\ e\ f\ g$ $a\ c\ b\ d\ f\ e\ g$

Depth-First Algorithm

- We call `DF(v)` recursively, so:

```
DF(v):  if v has not been visited then
          visit v;
          for each edge from v to x do DF(x) od
        fi
```

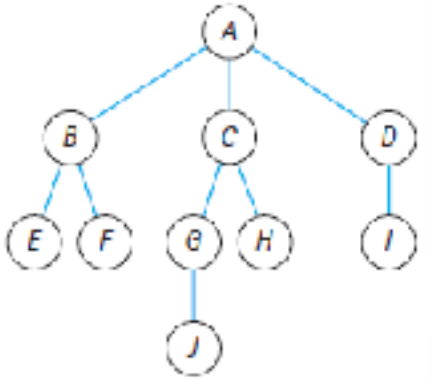


- Starting at a, we have four solutions:

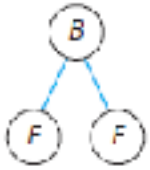
$a\ b\ d\ e\ g\ f\ c$ $\quad$ $a\ b\ d\ f\ g\ e\ c$ $\quad$ $a\ c\ b\ d\ e\ g\ f$ $\quad$ $a\ c\ b\ d\ f\ g\ e$

Trees

- Are similar representations to those used in biology
- The trees have:
  - A root
  - Vertices and leaves (parents and children)
  - Branches



- The trees can be ordered
- Subtrees can be extracted



Representing algebraic expressions

- The algebraic expression $x - y$ can be represented as an ordered tree, having the minus sign in the root, x in the lefthand subtree, and y in the righthand subtree

$3 - (4 + 8)$

$(4 + 8) - 3$

Representing Trees

- Any representation has to allow the reconstruction
- One of the methods is using lists

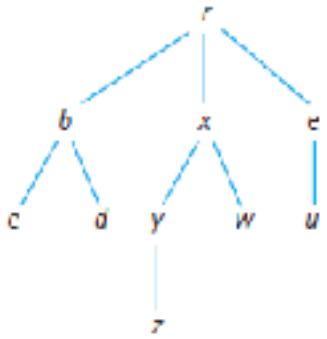
$3 - (4 + 8)$

$\langle -, \langle 3 \rangle, \langle +, \langle 4 \rangle, \langle 8 \rangle \rangle \rangle$

$(4 + 8) - 3$

Representing trees

- Another example:



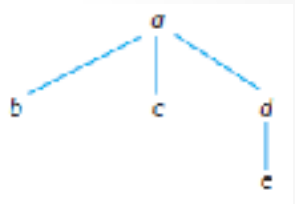
$$T = \langle r, \{\langle b, \{\langle c\rangle, \langle d\rangle\}\rangle, \langle x, \{\langle y, \{\langle z\rangle\}\rangle, \langle w\rangle\}\rangle, \{\langle e, \{\langle u\rangle\}\rangle\}\rangle$$

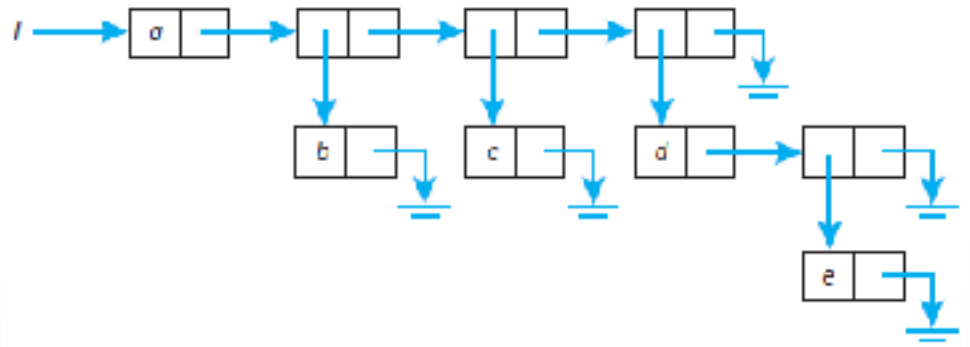
- Please note that T has the following subtrees:

$$\langle b, \{\langle c\rangle, \langle d\rangle\}\rangle$$
$$\langle x, \{\langle y, \{\langle z\rangle\}\rangle, \langle w\rangle\}\rangle$$
$$\langle e, \{\langle u\rangle\}\rangle$$

Representing Trees in Computers

- Representing the tree expressed by $T = \langle a, \langle b \rangle, \langle c \rangle, \langle d, \langle e \rangle \rangle \rangle$

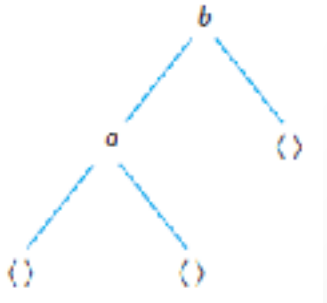




Binary Trees

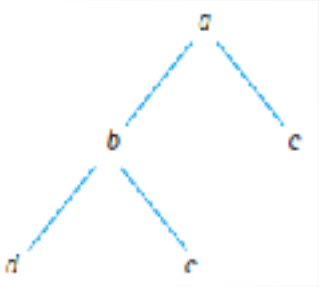
- Are those ordered trees which are either null, or or each node contains two subtrees (called lefthand and righthand trees) which are also binary trees

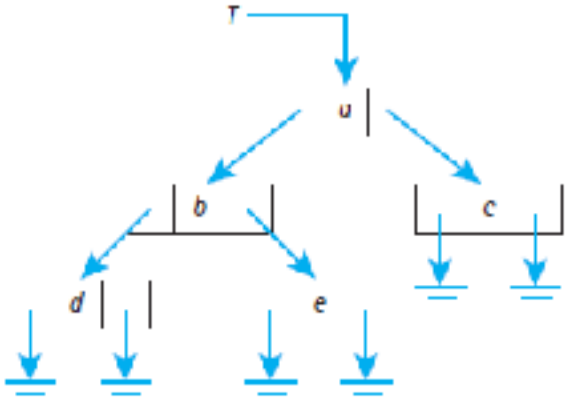
$\langle L, x, R \rangle$.



Computer Representations of Binary Trees

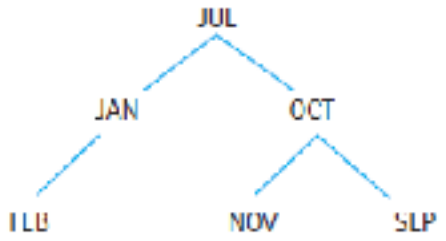
- Let the tree $T = \langle a, \langle b \rangle, \langle c \rangle, \langle d, \langle e \rangle \rangle \rangle$





Binary Search Trees

- It is important to minimize the depth
- Consider the year's months FEB, JAN, JUL, NOV, OCT, SEP, alphabetically ordered



- The depth of this binary search tree is 2

Spanning Trees

- A spanning tree for a connected graph is a subgraph that is a tree and contains all the vertices of the graph.

- A minimal spanning tree for a connected weighted graph is a spanning tree such that the sum of the edge weights is minimum among all spanning trees

Prim's Algorithm (1957)

- Constructs a minimal spanning tree for any undirected connected weighted graph
- Starting with any vertex, the algorithm searches for an edge of minimum weight connected to the vertex
- It adds the edge to the tree and then continues by trying to find new edges of minimum weight such that one vertex is in the tree and the other vertex is not

Prim's Algorithm (1957)

- Variables:
  - The set of graph's vertices, V
  - The set of spanning tree's vertices, W
  - The set of spanning tree's vertices, S

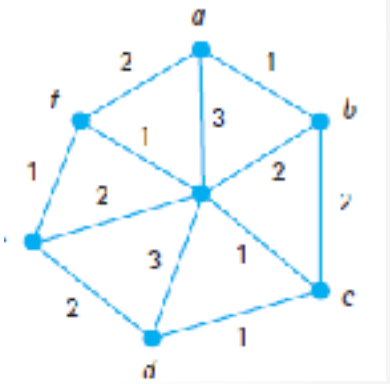
1. Initialize $S := \varnothing$.
2. Pick any vertex $v \in V$ and set $W := \{v\}$.
3. **while** $W \neq V$ **do**
   Find a minimum weight edge $\{x, y\}$ where $x \in W$ and $y \in V - W$;
   $S := S \cup \{\{x, y\}\}$;
   $W := W \cup \{y\}$

od

Prim's Algorithm (1957)

- A minimal spanning tree construction

S	S
$\{\}$	$\{a\}$
$\{\{a,b\}\}$	$\{a,b\}$
$\{\{a,\},\{b,c\}\}$	$\{a,b,c\}$
$\{\{a,b\},\{b,c\},\{c,d\}\}$	$\{a,b,c,d\}$
$\{\{a,b\},\{b,c\},\{c,d\},\{c,g\},\{g,f\}\}$	$\{a,b,c,d,g\}$
$\{\{a,b\},\{b,c\},\{c,d\},\{c,g\},\{g,f\}\}$	$\{a,b,c,d,g,f\}$
$\{\{a,b\},\{b,c\},\{c,d\},\{c,g\},\{g,f\},\{f,e\}\}$	$\{a,b,c,d,g,f,e\}$

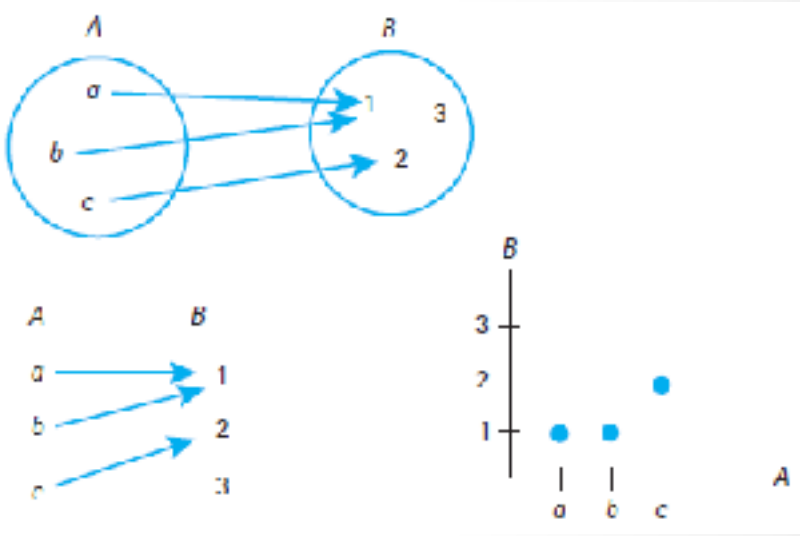


Functions

- Gottfried-Wilhelm von LEIBNITZ (1692)
- If A and B are sets and we associate to **each** element from A **exactly one** element from B, then we have $f : A \rightarrow B$
- If $x \in A$ and $y \in B$, then we can write $f(x) = y$
- Alternative terms for functions: *association (mapping), transformation, operator*

Representing functions

- Analytical representation
  - $f(x) = x^2$
- Exhaustive representation
  - Let $A = \{a, b, c\}$, $B = \{1, 2, 3\}$ and $g : A \to B$
  - We have: $g(a) = 1$, $g(b) = 1$ și $g(c) = 2$
- Graphical representation
- Truth tables



Terminology

- Type of function $f : A \to B \qquad f(x) = y$,
  - A is the function's f domain and B is its codomain
  - x is the function's f argument and y is its returned value

- Function's range $\text{range}(f) = \{f(a) \mid a \in A\}$

- Function's image $f(S) = \{f(x) \mid x \subset S\}$ where $S \subset A$

- Function's pre-image $f^{-1}(T) = \{a \in A \mid f(a) \in T\}$ where $T \subset B$

Terminology

- Functions' equality
  - If $f, g : A \to B$ and $f(x) = g(x)$ for every $x \in A$, then $f = g$
- Defining a function by cases
  - For instance, the absolute value function: $\text{abs}(x) \begin{cases} x & \text{if } x \geq 0 \\ x & \text{if } x < 0 \end{cases}$
  - Such functions can also be defined by using *if-then-else* rules:

$$\text{abs}(x) = \text{if } x \geq 0 \text{ then } x \text{ else } -x$$

  - Classifying the roots of the quadratic equation: $ax^2 + bx + c = 0$

```
classifyRoots(a, b, c)   if b^2 − 4ac > 0 then
                             "The roots are real and distinct."
                         else if b^2 − 4ac < 0 then
                             "The roots are complex conjugates."
                         else
                             "The roots are real and repeated."
```

Some Useful Functions

- The *floor* functions extracts the nearest integer less or equal to a given real number, commonly denoted as $\lfloor x \rfloor$
- The *ceiling* function extracts the nearest integer greater or equal to a given real number, commonly denoted as $\lceil x \rceil$
- Properties:

a. $\lfloor x+1 \rfloor = \lfloor x \rfloor + 1$.
b. $\lceil x-1 \rceil = \lceil x \rceil - 1$.
c. $\lfloor x \rfloor = \lceil x \rceil$ if and only if $x \in Z$.
d. $\lfloor x \rfloor = \lceil x-1 \rceil$ if and only if $x \notin Z$.
e. $\lfloor x \rfloor = \lceil x-1 \rceil$ if and only if $x \notin Z$.

- Some values:

x	2.0	1.7	1.3	1.0	0.7	0.3	0.0	0.3	0.7	1.0	1.3	1.7	2.0
$\lfloor x \rfloor$	-2	-2	-2	-1	-1	-1	0	0	0	1	1	1	2
$\lceil x \rceil$	-2	-1	-1	-1	0	0	0	1	1	1	2	2	2

Greatest Common Divisor…

- …for two integers, not both zero, is the greatest integer which divides them both and is denoted as:

$$\gcd(a, b)$$

- Properties:

a. $\gcd(a, b) = \gcd(b, a) = \gcd(a, -b)$.

b. $\gcd(a, b) = \gcd(b, a - bq)$ for any integer q.

c. If $y = \gcd(a, b)$, then there are integers x and y such that $y = ax + by$.

d. If $d \mid ab$ and $\gcd(d, a) = 1$, then $d \mid b$.

The Division Algorithm

- If we have the integers a and b with b non-zero, the we have the integers q and r so that:

$$a = bq + r$$

- To be noted that $0 \leq r < |b|$.

- Euclid's algorithm receives two natural numbers a and b, both not zero, and returns the greatest common divisor:

while $b > 0$ **do**
  Construct a $bq + r$, where $0 < r < b$;
  a : b;
  $b := r$
od;
Output a.

The Mod Function

- If a and b are integers, with b strictly positive, the the remainder of the division of a to b is written $a \ mod \ b$

$$a \bmod b = a - b\lfloor a/b \rfloor$$

- Having a constant divisor n, one can create the set

$$N_n = \{0, 1, 2, \ldots, n-1\}$$

- Examples of certain values:

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$x \bmod 1$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$x \bmod 2$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$x \bmod 3$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0
$x \bmod 4$	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
$x \bmod 5$	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0

Properties of the Mod Function

- **a.** $x \bmod n = y \bmod n$ iff n divides $x - y$ iff $(x - y) \bmod n = 0$.
  b. $(x + y) \bmod n = ((x \bmod n) + (y \bmod n)) \bmod n$.
  c. $(xy) \bmod n = ((x \bmod n)(y \bmod n)) \bmod n$.
  d. If $ax \bmod n = ay \bmod n$ and $\gcd(a, n) = 1$, then $x \bmod n = y \bmod n$.
  e. If $\gcd(a, n) = 1$, then $1 \bmod n = ax \bmod n$ for some integer x.

- An example: converting decimal integers into binary numbers

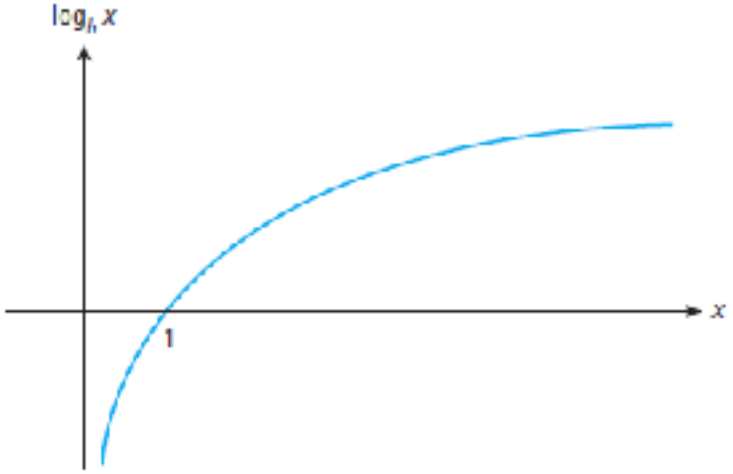
The Logarithmic Function

- Measures the size of the exponents
- If b is a positive real non-unitary number, then it means that $\log_b x = y$ and $b^y = x$,
- The base 2 logarithm is often used in computers

x	1	2	4	8	16	32	64	128	256	512	1024
$\log_2 x$	0	1	2	3	4	5	6	7	8	9	10

The Logarithmic Function

- The logarithmic function's graph:



- Properties:

$\log_b (b^x) = x.$

$\log_b (x\ y) = \log_b x + \log_b y.$

$\log_b (x^y) = y \log_b x.$

$\log_b (x/y) = \log_b x - \log_b y.$

$\log_a x = (\log_a b)(\log_b x).$

Partial Functions

- They are like ordinary functions, except that they may not be defined for some elements of the domain
- One obvious example is the division function which cannot be completely defined on $\mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ because of the 2-tuples like $(x, 0)$
- The values on which the function cannot be defined may be redirected to a reserved value associated to an error message

Constructing Functions

- Complex functions can be created by combining simple functions
- The combining method under discussion is called function composition
- The map function is a useful tool for displaying a list of values of a function
- Many programming languages rely on these ideas and concepts

Composing Functions

- It is a natural process, often performed in an intuitive manner
  - For instance, `floor(log2(5))`
- The composition of two functions, f and g, is usually denoted by $f \circ g$ and defined as: $(f \circ g)(x) = f(g(x))$.
- Function composition makes sense only if $g{:}A \to B$, $f{:}C \to D$ and $B \subset C$
- So, $f \circ g : A \to D$

Composing Functions

- Is associative
  - For instance, if f, g and h are functions of the right type for composition, then

$$(f \circ g) \circ h = f \circ (g \circ h)$$

- The equality is easy to observe by evaluating the expressions:

$$((f \circ g) \circ h)(x) = (f \circ g)(h(x)) = f(g(h(x)))$$
$$(f \circ (g \circ h))(x) = f((g \circ h)(x)) = f(g(h(x)))$$

Composing Functions

- Usually, it is not commutative
- It is enough to find a single value for which $f \circ g$ is different from $g \circ f$
  - For instance, if $f(x) = x+1$ and $g(x) = x^2$, we have, for $x=5$
    - $(f \circ g)(5) = f(g(5)) = f(5^2) = 5^2 + 1 = 26$
    - $(g \circ f)(5) = g(f(5)) = g(5+1) = (5 + 1)^2 = 36$
- The function that always returns the argument is called the identity function

$$f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$$

The Sequence, Distribute, and Pairs Functions

- The sequence function `seq` has the type $\mathbb{N} \rightarrow lists(\mathbb{N})$ and is defined as:

$$\text{seq}(n) = \langle 0, 1, \ldots, n \rangle$$

- The distribute function `dist` has the type $A \times lists(B) \rightarrow lists(A \times B)$ and is defined as

$$\text{dist}(x, \langle r, s, t \rangle) = \langle (x, r), (x, s), (x, t) \rangle$$

- The `pairs` function takes two lists of equal lengths and returns the list of pairs of corresponding elements:

$$\text{pairs}(\langle a, b, c \rangle, \langle d, e, f \rangle) = \langle (a, d), (b, e), (c, f) \rangle$$

Composition of Functions with Different Arities

- Take, for instance, the function:

$$f(x, y) = \text{dist}(x, \text{seq}(y)).$$

- In this case, `dist` has two arguments and `seq` has one argument

- Se evaluează expresia $f(5,3)$:

$$\begin{aligned} f(5,3) &= \text{dist}(5, \text{seq}(3)) \\ &= \text{dist}(5, \langle 0, 1, 2, 3 \rangle) \\ &= \langle (5,0), (5,1), (5,2), (5,3) \rangle \end{aligned}$$

The Map Function

- It is an useful tool for computing a function's list of values
- It takes a function $f{:}A \rightarrow B$ and a list of values from A and returns a list of values from B obtained by applying function f on the list of values from A

$$\text{map}(f, \langle x_1, \ldots, x_n \rangle) = \langle f(x_1), \ldots, f(x_n) \rangle$$

- In other words:

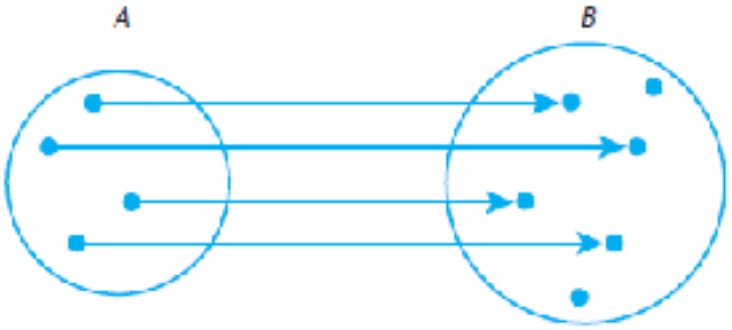
$$\text{map: } (A \rightarrow B) \times \text{lists}(A) \rightarrow \text{lists}(B)$$

The Map Function

- It is a higher order function, meaning it is a function that has another function as its argument or as its output value
- This is an important property found in most good programming languages
- The composition or tupling operations are examples of higher order functions

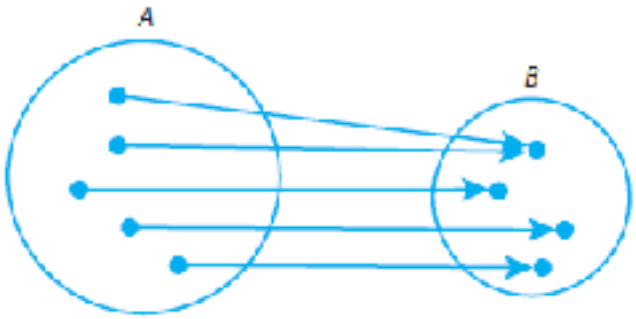
Functions' Properties

- Injective functions:
  - A function $f : A \rightarrow B$ is injective if $x \neq y$ implies that $f(x) \neq f(y)$ or if $x = y$ implies that $f(x) = f(y)$



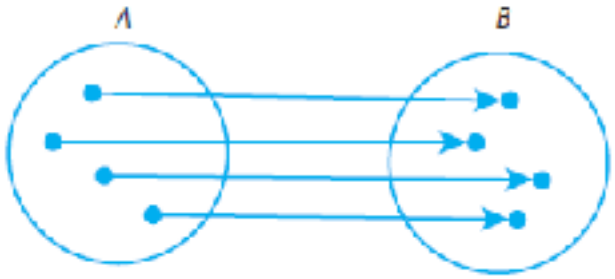
Functions' Properties

- Surjective functions:
  - A function $f : A \rightarrow B$ is surjective if $\texttt{range}(f) = B$
  - In other words, any $b \in B$ can be written as $b = f(x)$ for some $x \in A$



Functions' Properties

- Bijective functions:
  - A function is bijective if it is both injective **and** surjective



A Bijection Example

- Let $f : (0, 1) \rightarrow R^+$ be defined as:

$$f(x) = \frac{x}{1 \quad x}$$

- To prove that f is an injection, let $f(x) = f(y)$:

$$\frac{x}{1-x} = \frac{y}{1-y}$$

- We cross multiply, then subtract $-xy$ from both sides, and we obtain $x = y$
- To prove that f is a surjection, let $y > 0$ and then try to find $x \in (0, 1)$ so that $f(x) = y$, by solving

$$\frac{x}{1-x} = y, \qquad x = \frac{y}{y+1}$$

- It results that $f(y/(y+1)) = y$, and since $y > 0$, it results that $0 < y/(y+1) < 1$, so f is surjective. Q.E.D.

Inverse Functions

- The bijections always come in pairs
- If $f : A \to B$ is a bijection, then there is a $g : B \to A$, defined by $g(b) = a$ if $f(a) = b$
- The inverse of f is also a bijection and we have $g(f(a)) = a$ for all $a \in A$ and $f(g(b)) = b$ for all $b \in B$
- To be noted that there is only one inverse function for any given bijective function
- If g and h would be two inverse functions of the bijective function f, then:

$$\begin{aligned} g(x) &= g(\mathrm{id}_B(x)) \\ &= g(f(h(x))) && (\text{since } f \circ h = \mathrm{id}_B) \\ &= \mathrm{id}_A\ (h(x)) && (\text{since } g \circ f = \mathrm{id}_A) \\ &= h(x). \end{aligned}$$

The Mod Function and Inverse Functions

- Let $n > 1$ and let $f : N_n \rightarrow N_n$, where a and b are integers, defined as:

$$f(x) = (ax + b) \bmod n.$$

- The function f is bijective if and only if $\gcd(a, n) = 1$, in which case the inverse function is defined as:

$$f^{-1}(x) = (kx + c) \bmod n.$$

- where c is an integer satisfying $f(c) = 0$, and k is an integer so that $1 = ak + nm$, for some integer m

Relationships between Injection and Surjection

- If f and g are injective, then $f \circ g$ is also injective
- If f and g are surjective, then $f \circ g$ is also surjective
- If f and g are bijective, then $f \circ g$ is also bijective
- There is an injection from A to B if and only if there is a surjection from B to A

The Pigeonhole Principle

- If m pigeons fly into n pigeonholes, where m is strictly greater than n, then at least one pigeonhole will have two or more pigeons

- In other words, if A and B are finite sets so that $|A| > |B|$, then any function from A to B will have at least two elements from A associated to the same element from B

- Said differently, no such function will be injective

Simple Ciphers

- Bijections and inverse functions are very important for applications that **encipher** and **decipher** information, these applications are usually called **ciphers**

- We will index the 26 letters of the lowercase alphabet by the $\mathbb{N}_{26}$ set

- We describe a circular cipher that translates the letters in the original message by 5 positions ("abc" becomes "fgh")

Simple Ciphers

- In other words, the cipher is represented by the translation function:

$$f(x) = (x + 5) \bmod 26$$

- The function $f : \mathbb{N}_{26} \rightarrow \mathbb{N}_{26}$ is bijective

- For decoding we use the inverse function:

$$f^{-1}(x) = (x - 5) \bmod 26$$

- The cipher used in this case is **additive** and **monoalphabetic**

Simple Ciphers

- A **multiplicative** cipher translates every character by multiplying with a constant, like in this example:

$$g(x) = 3x \bmod 26$$

- The function g is bijective and has the inverse:

$$g^{-1}(x) = 9x \bmod 26$$

- The **affine** cipher is monoalphabetic, but uses two translation methods, like in this example:

$$f(x) = 3((x + 5) \bmod 26) \bmod 26 = (3x + 5) \bmod 26$$

Simple Ciphers

- The affine function presented in the previous example is bijective, since $\gcd(3, 26) = 1$, having the inverse:

$$f^{-1}(x) = (9x + 7) \bmod 26$$

- Some ciphers leave one or more characters in the same position (fixed):
  - An additive cipher with $x=26k$ is fixed for all characters
  - A multiplicative cipher always leaves the first character in the same position

- We can be certain that no character remains in the same position only if $\gcd(a\text{-}1, 26)$ does not divide b

Hash Functions

- A *hash* function maps arbitrary length data to a fixed to a fixed length data (see *Hash Function* in Wikipedia)
- The there the data is accessible through hash indexes is called *hash* table
- Let S be a three letter abbreviation list of year's months

Hash Functions

- One can define the *hash* function $f: S \to \{0, 1, \ldots, 11\}$:

$$f(XYZ) = (\text{ord}(X) + \text{ord}(Y) + \text{ord}(Z)) \bmod 12$$

- For A to Z capitals and a to z letters, the ASCII (American Standard Code for Information Interchange) codes are between 65 and 90 and from 97 to 122, respectively

- In January's case we have:

$$\begin{aligned} f(\text{JAN}) &= (\text{ord}(X) + \text{ord}(Y) + \text{ord}(n)) \mod 12 \\ &= (74 + 97 + 110) \mod 12 \\ &= 5. \end{aligned}$$

Hash Functions

- Accordingly, the list of all values of f is:

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
5	5	0	3	7	1	11	9	8	6	7	4

- Please note that f is not injective

- The indexes for January and for February are of the same value and, while we cannot put two different data at the same address, it means that we have a **collision**

- When a collision occurs, one of the data mapped at that address will be placed there, while the rest of them to some of the other available addresses

Hash Functions

- There are several methods to find a key in collision with another key
- One of those methods searches successively in alternate locations, so it is named *linear probing*
- For instance, if a collision occurs at the K location, then it will look at the next alternate locations:

$$(k+1) \bmod n,\ (k+2) \bmod n,\ \ldots,\ (k+n) \bmod n$$

Numerability

- This is a short discussion about counting the sets which may not be finite
- Some techniques which can be used in automated calculus will be derived
- For instance, we will observe that there are limits about what we can count

Comparing the Size of Sets

- If we have two sets, A and B, with a bijection between them, then those sets have the same number of elements

$$|A| = |B|$$

- If there an injection between A and B, then:

$$|A| \leq |B|$$

- If there is an injection, but no bijection, between them, then:

$$|A| < |B|$$

Countable Sets

- Informally, a set is countable if we can count its elements in distinct steps, no matter how many elements it contains

- If we have the set A with n elements, then we can represent those elements as:

$$x_0, x_1, x_2, \ldots, x_{n-1}$$

- If we associate each element x_k with its index k, then we have a bijection between A and $\{0, 1, \ldots, n\text{-}1\}$

- If A is infinite, the bijection is between A and N

The Definition of Numerability

- We can conclude that a set is countable if there is a bijection between it and $\mathbb{N}$ (countable infinite set)
- Otherwise, it is not countable
- The properties of numerability:
  - Any subset of $\mathbb{N}$ is countable
  - S is countable iff $|S| \leq |N|$
  - Any subset of a countable set is also countable
  - Any image of a countable set is also countable

The Limits of Numerability

- The set of computer programs is infinitely countable
- Since there is “only” a countable number of computer programs, it gives that what we can compute has limits
  - For instance, there are a lot of functions defined on $\mathbb{N}$ with values on $\mathbb{N}$
  - Consequently, there are programs for computing a countable subset of those functions
- The rațional numbers can be computed
- Also, there are a lot of irrational numbers that can be computed (like π or e)

Superior Cardinalities

- Cantor demonstrated that there are more subsets of a countable set than elements of that set

$$|A| < |\text{power}(A)|$$

- There is an infinite countable languages set over a finite countable alphabet
  - Using the above delicte relation, we can use a infinite sequence of sets with growing cardinality
  - One can demonstrate that $|R| = |\text{power}(N)|$, resulting $|N| < |R| < |\text{power}(\text{power}(N))| < \cdots$
  - In this case, there is a known set *S*, so $|S| = |\text{power}(\text{power}(N))|$?

The Continuum Hypothesis

- Îs there a set S with a cardinality between that of $\mathbb{N}$ and that of $\mathbb{R}$?

- The answer is that nobody knows!
  - Funny thing is that those assuming there is such a set will not encounter any contradiction in their inferences
  - Those assuming the opposite also will not experience any contradiction în their demonstration
  - The latest hypothesis is called *the continuum hypothesis*
  - For instance, if we can demonstrate the existence of a set S, so $|\mathbb{N}| \leq |S| < |\mathbb{R}|$, then, according to this hypothesis, we have $|\mathbb{N}| = |S|$

Thank you!

I wish you Merry Christmas and a Happy New Year!