Theory of Computing:

1. Introduction

Professor Imed Bouchrika
National Higher School of Artificial Intelligence
imed.bouchrika@ensia.edu.dz

Outline :

- Problems
- Computation and Complexity
- Sets, Functions and Relations
- Alphabets, Strings and Languages
- Proof Techniques

Problems

- Given some binary string of 0s and 1s (Example : 01010111100111), How to ensure that the binary string contains only a **single 00 substring ?**
    - Pseudo-code ?
    - Programming Language ?

Problems

- Given some binary string of 0s and 1s (Example : 01010111100111), How to ensure that the binary string contains only a **single 00 substring ?**
    - Pseudo-code ?
    - Programming Language ?
    - **Running/Execution Environment ? PC ? Vending Machine ? Automated Door ? Washing Machine ? Hardware with limited memory ?**

Problems

- Given a large text, find all valid emails within the text. Provided the email address:
    - Must contain @
    - Should contain any printable characters
    - Should have a valid domain extension (TLD)
    - ....
- Shall we write a C++/Python/Java/,,, code to do it ?

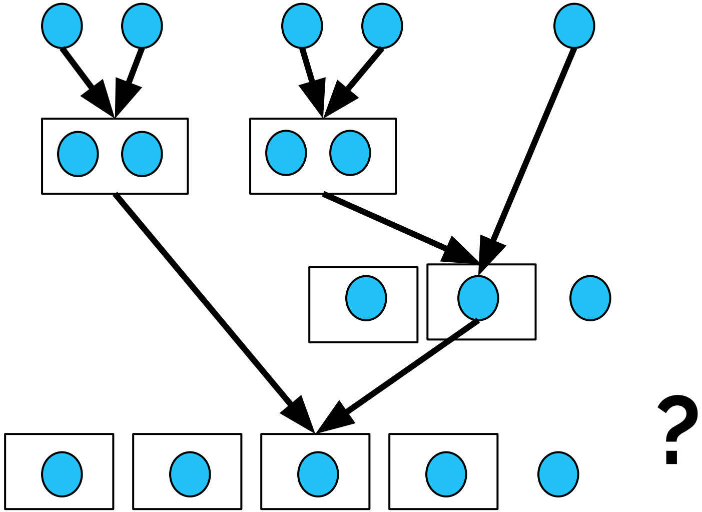
Problems

- Given five items, can we sort them ascendingly using a two-pan scale ?

- How many operations we need to perform ? Any guess ?

Problems

- Applying merge sort :



Problems

- Given five items, can we sort them ascendingly using a two-pan scale ?
    - The condition is that we need to conduct only **7 operations MAX.**
- Which Algorithm to apply ? Merge ? Shell ? Bubble ? Insertion ?

Problems

- Given five items, can we sort them ascendingly using a two-pan scale ?
  - The condition is that we need to conduct only **7 operations MAX.**
- Which Algorithm to apply ? Merge ? Shell ? Bubble ? Insertion ?

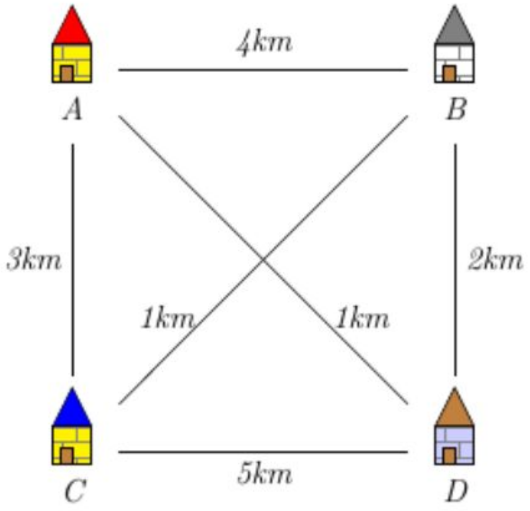
Optional Exercise
Given as a challenge in 2022/2023

Problems

- **Travelling Salesman :**

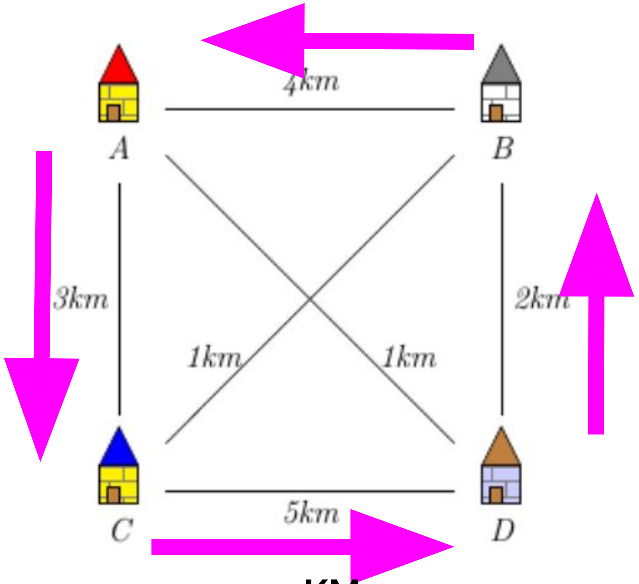
  Given a list of cities and the distances between each pair of cities,what is the shortest possible route that visits each city and returns to the origin city?

  Assume we start from **A**



Problems

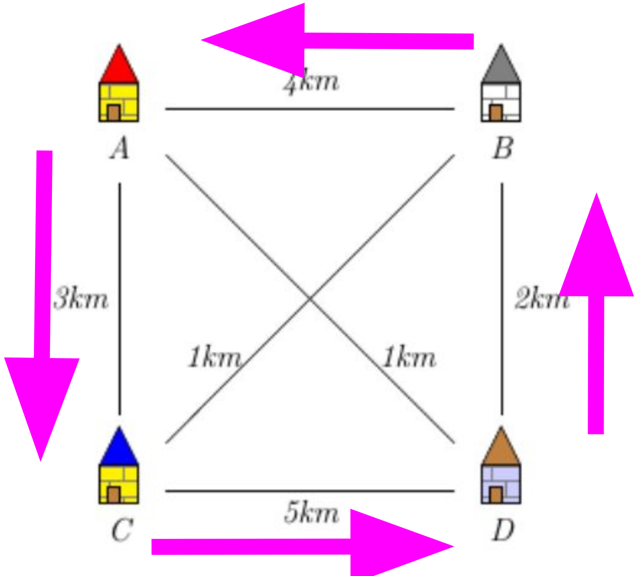
- **Travelling Salesman :** Let's start from A :



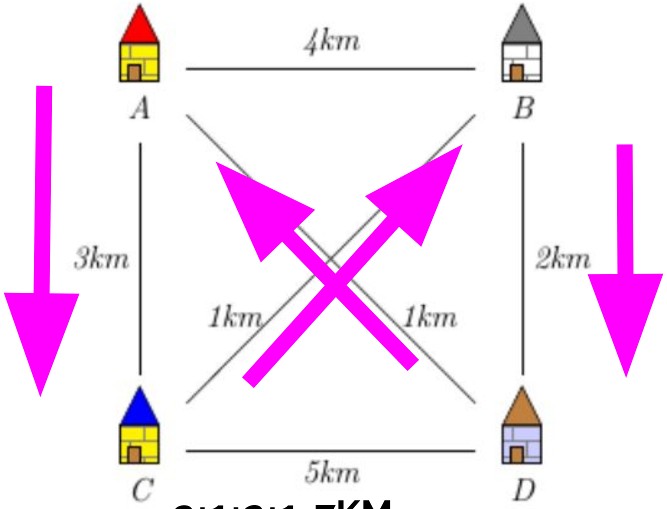
3+5+2+4=14KM

Problems

- **Travelling Salesman :** Let's start from A :



3+5+2+4=14KM



3+1+2+1=7KM

Problems

- **Travelling Salesman :**
  - How to solve the problem ? 4 cities ? Data structure ? Programming Language ?
  - Problem instance involving 4 cities → Generalize it to N cities ?

Problems

- **Travelling Salesman :**

Number of cities n	Number of paths $(n-1)!/2$
3	1
4	3
5	12
6	60
7	360
8	2,520
9	20,160
10	181,440
15	43,589,145,600
20	6.082×10^{16}
71	5.989×10^{99}

Problems

- **Travelling Salesman :**

Problem to be Solved in Logistics

Number of cities n	Number of paths $(n-1)!/2$
3	1
4	3
5	12
6	60
7	360
8	2,520
9	20,160
10	181,440
15	43,589,145,600
20	6.082×10^{16}
71	5.989×10^{99}

Problems

- **Other problems :**
  - **Partition Problem :** Given S a set of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2

Example :

{1 , 3 , 4 , 2 , 5, 7, 8 , 10} → { 1 ,2,7, 10} and {3,4,5,8 } (sum of each subset is 20)

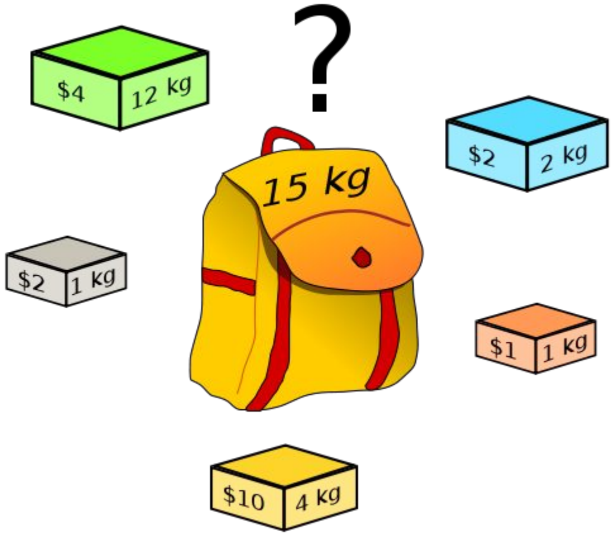
Problems

- **Other problems :**
  - **3-Partition Problem :** Given S as a set of integers can be partitioned into triplets that all have the same sum.
  - Example :

The set $S = \{20, 23, 25, 30, 49, 45, 27, 30, 30, 40, 22, 19\}$ can be partitioned into the four sets $\{20, 25, 45\}, \{23, 27, 40\}, \{49, 22, 19\}, \{30, 30, 30\}$, each of which sums to $T = 90$.

Problems

- **Other problems :**
  - **Knapsack problem :** Given a set of items, each with a **weight (kg)** and a **value ($)**, → Determine the number of each item to include in a collection so that the **total weight is less than or equal to a given limit** (For example : 15kg) and the total value is **as large as possible.**



Problems

- **Other problems :**
    - **Halting Problem :** does the following program terminate/halt:

```
input n;
assume n>1;
while (n !=1) {
   if (n is even)
    n := n/2;
  else
    n := 3*n+1;
}
```

17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Problems

- **Other problems :**
  - **A Diophantine quadratic equation :** Given three positive integers a, b, c, decide if the equation $ax^2 + by = c$ has a solution in positive integers (Unknown variables are x and y) .
    - **Example :** $x^2 + 2xy - y^3 = 13$, the solution is : $x=3$ and $y = 2$
    - What about : $x^2 - y^2 = 2$?
    - Can we write a program to brute force all integers or even real numbers ?
    - Or even prove that such equation has no solution at all ?

Computation and Complexity

- The previous problems :
  - Can we solve them on a computer ? are they computable ?
  - If they are computable : how efficient ? few seconds ? few years ?

Computation and Complexity

- Theory of Computation :
  - Is a branch of Computer Science/Mathematics dealing with how **efficiently** problems can be **computed**/solved on a **model of computation** using an algorithm

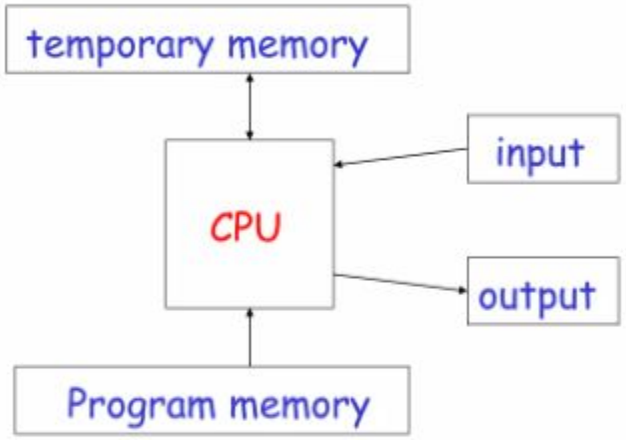
  - The field covers three major axes :
    - Automata theory
    - Computability Theory
    - Complexity Theory

Computation and Complexity

- Computational Model :
  - Is a set of allowed rules for information processing :
    - Taking some input
    - Processing a set of instruction complying with certain rules
    - Giving an output after completing the processing
      - For decision problems, the output can be :
        - Accept/Yes/True…
        - Reject/No/False…

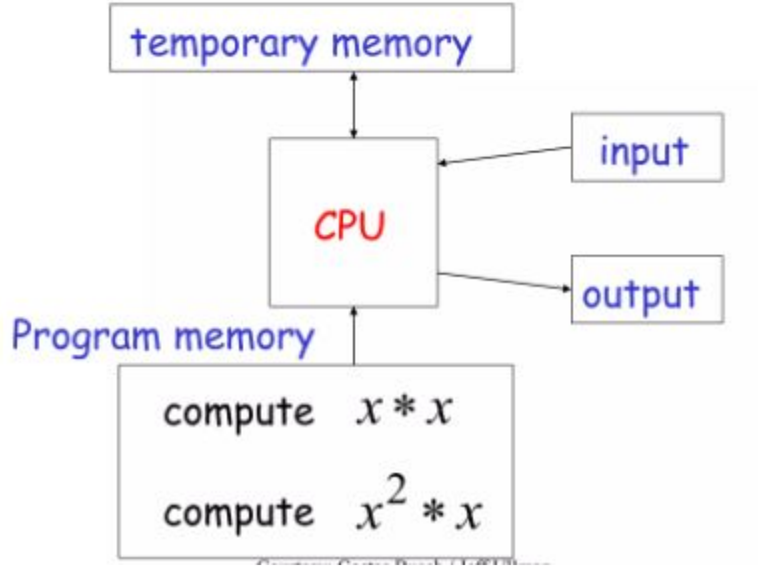
Computation and Complexity

- The computer is considered as a **physical** instance for a computational model.



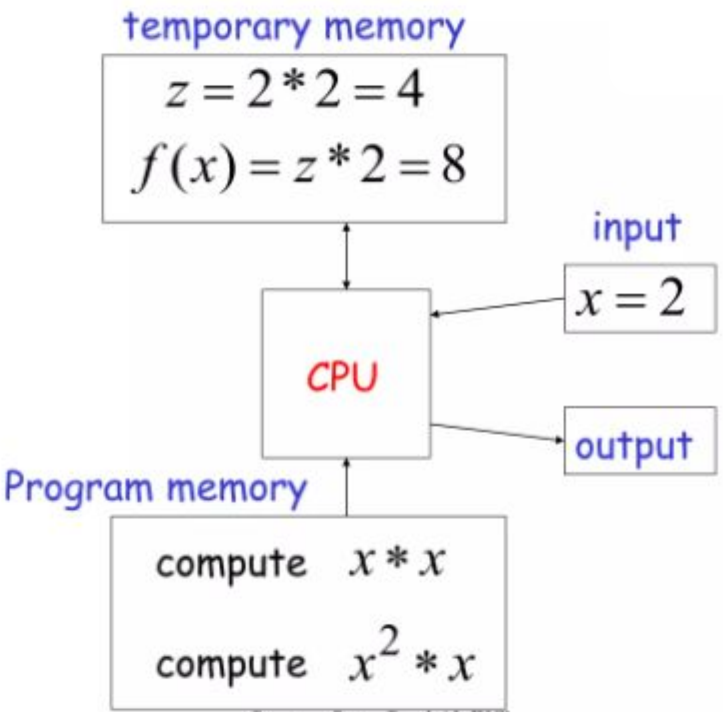
Computation and Complexity

- Given the following problem:
    - Compute the x^3 for a given number.
- The sequence of instruction is stored into the program memory section



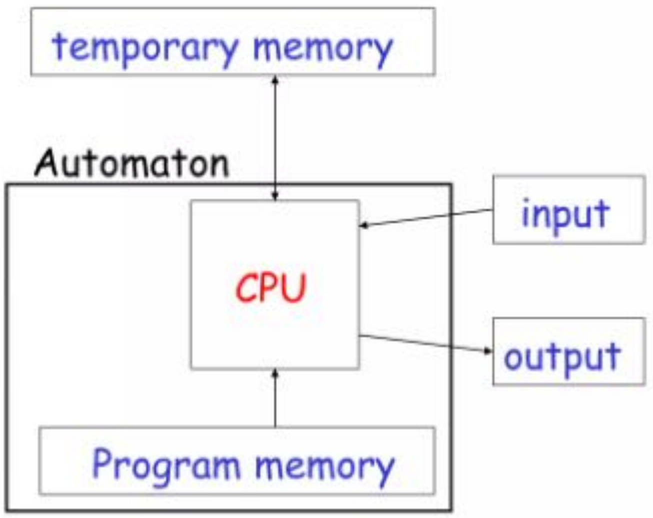
Computation and Complexity

- Given the following problem:
  - Compute the x^3 for a given number.
- The sequence of instruction is stored into the program memory section
- 2 is given as input
- Temporary memory is utilized for **processing information**



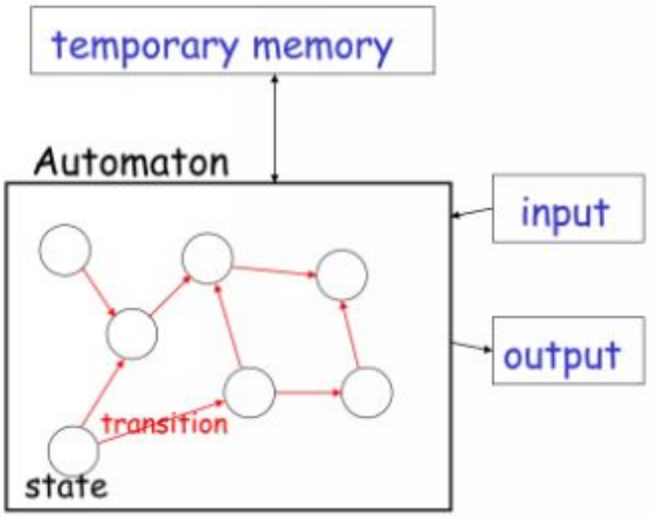
Computation and Complexity

- The part for processing the information or solving a given problem is the **Automaton**



Computation and Complexity

- The part for processing the information or solving a given problem is the **Automaton**
- In this case : CPU + Program memory for a physical device is replaced by:
  - States
  - Transitions



Computation and Complexity

- Automaton :
  - Literal Meaning : a device that can do “things” on its own or a self-operating machine
    - Plural (Automata)
  - In this course :
    - **An abstract device** for processing information to solve a given problem. (no labs, only pen and paper)

Computation and Complexity

- Automaton :
  - *are abstract models of machines that perform computations on an input by moving through a series of states or configurations.*
  - *At each state of the computation, a transition function determines the next configuration on the basis of a finite portion of the present configuration.*
  - *As a result, once the computation reaches an accepting configuration, it accepts that input. The most general and powerful automata is the Turing machine*

Definitions taken from : https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/automata-theory/basics.html

Computation and Complexity

- There are different types of automata including :
  - *Finite Automata (no temporary memory)*
  - *Pushdown Automata (has a stack)*
  - *Turing Machines (Random access memory)*

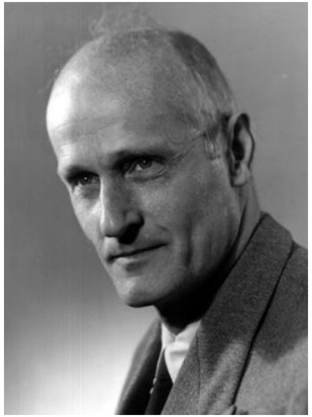
Computation and Complexity

- That's enough as an introduction, for general knowledge , read the biographies of the following scientists: (There will be some quiz questions about them)

Alan

Kurt Gödel

Alonzo Church



Stephen Kleene

Sets and Languages

- A set is a collection of elements :
  - A = { 1,2,3 }
  - B = { car,train,bus,plane}
- Conditions:
  - There should be no **duplicate elements** in a set.
  - The order of elements is not important for sets:
    - A = { 1, 2, 3 } = { 2, 1 ,3 }
- Operations of elements on a set:
  - $1 \in A$ (1 in A *or* 1 is a member of A)
  - $1 \notin B$ (1 is not in B)

Sets and Languages

- Finite set :
  - Limited or countable number of elements
  - Example :
    - C= {1,2,3,4,5,6,7,8,9,10}
    - C= {1,2,3, **…** ,10} (the use of … for a finite set.)
- Infinite Set:
  - Example :
    - S = {1,2,3,4, **…** }
    - Z = { … -2,-1,0,1,2,…}

Sets and Languages

- Empty set :
  - Contains zero element : nothing
  - We use the symbol : ø
  - Examples :
    - C = { }
    - C = ø
    - C = {ø} ?
    - C = { {} } ?

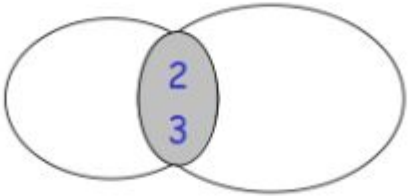
Sets and Languages

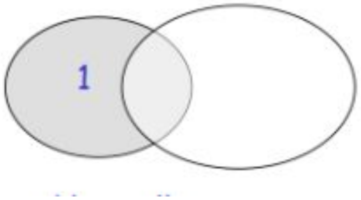
- Formal Description of a set:
  - C = { j | j >0 and j <10 }
  - E = { x | x is positive and even }
- Explicit Listing of a Set :
  - C = {1,2,3,4,5,6,7,8,9,10}
  - E = {2, 4, 6, 8, **...**}
- Informal Description of a Set :
  - Given as in plain language to describe the elements of a set.

    “The set of even numbers”

Sets and Languages

- Venn Diagram :
  - Is a graphical representation of sets





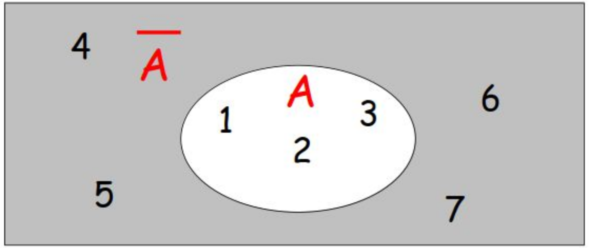
Sets and Languages

- **Operations on Sets**

  Given the following sets : A = { 1, 2 , 3 } and B = { 2 , 3 , 4 , 5 }
  - **Union :**
    - A U B = { 1, **2**,3,4,5 }
  - **Intersection**
    - A ∩ B = { 2 , 3}
  - **Difference**
    - A-B = { 1 }
      - vs
    - B-A= { 4 , 5}

Sets and Languages

- Universal Set :
  - The big set containing particular elements
    - Example : U = { 1, 2 , 3 , 4 ,5 ,6 ,7 }
  - Smaller sets can be created from the Universal set
- Complement of Set :
  - Is written as : $\overline{E}$
  - Contains the elements from the Universal set and **not contained** in E



Sets and Languages

- DeMorgan's Law

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

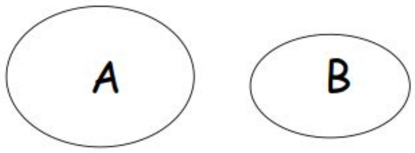
Sets and Languages

- **Subset :**
  - A set A is **a subset of** a set B if and only if everything in A is also in B. In other words : all elements of A, are included also in the set B.
    $A \subseteq B$

- **Proper Subset :**
  - If A is a subset B and A ≠ B, then A is a proper subset of B denoted as :
    $A \subset B$

Sets and Languages

- **Disjoint Sets :**
  - A and B are disjoint sets if they have no common elements.
  - $A \cap B = \emptyset$



Sets and Languages

- Set Cardinality :
  - Refers to the number of the elements within a set
    - $A = \{ 2, 6, 8 \}$
    - $|A| = 3$

Sets and Languages

- **Power set** :
  - The set of **all subsets** of a set A is called the power set of A and denoted either by
    - 2^A
    - $\mathcal{P}(A)$
  - Example : A = {a, b ,c }
  - 2^A= { ø , {a}, {b}, {c} , {a, b} , {a,c} , {b,c}, { a,b,c} }
  - Cardinality of the power set is : $2^{|A|} = 2^3 = 8$

Sets and Languages

- **Sequence :**
  - Is a list of elements with a **specific order**
  - Denoted as :
    - A = (1, 2 ,3, 4)
    - B = (2, 1, 3, 4)

    A is not equal to be B

  - Can be infinite or finite.
- **Tuple :**
  - A finite sequence can be called Tuple.
  - Tuple with K elements can be called : ***K-tuple***
  - ***2-Tuple*** can be called : **ordered pair.**

Sets and Languages

- Cartesian Products:
  - The set of all ordered pairs (a, b), where a is an element of A and b is an element of B,
  - Denoted as : A x B
  - Example:
    - For two sets : A = { 2, 4 } and B = { 2 , 3 , 5 }
    - A x B = { (2, 2), (2, 3), (2, 5), (4, 2), (4, 3), (4, 5)}
  - Cardinality :
    - |A x B | = |A| x |B|
  - Same for more than two sets (Example A x B x C x D ⇒ will produce 4-tuples of type (a, b, c, d)

Sets and Languages

- **Functions :**
    - It is a relationship which takes an input and produces an output.
    - It is sometimes called a ***mapping L***
        - *f(a)=b : f maps a to b : f*
        - $a \mapsto b$
    - The possible values that the function can take as input is called : **domain**
    - The output values come from the set called : **range**
    - The notation of the function with respect to its domain and range :
        - $f : D \rightarrow R$ (*for the case of a function that takes a single input/argument at a time).*

Sets and Languages

- **Predicate or Property :**
  - It is a special function whose range (outputs' set) is { True, False }
  - Example : function even(x)
- **Relation :**
  - A **property** whose domain is a a ***k-tuple*** is called a **relation**
  - Examples:
    - "Less_than" : N x N → {True, False}

Sets and Languages

- **Relation :**
    - Relation can be written in the form of a set considering only the ***true*** tuples
    - Example : $R = \{ (x_1, y_1), (x_2, y_2), (x_3, y_3), \dots \}$
        - If : $x_i \in A$ and $y_i \in B$
        - Then : $R \subseteq A \times B$
    - The following notation can be also used :
        - $x_i \, R \, y_i$ to denote $(x,y) \in R$

Sets and Languages

- **Equivalence Relations:**
  - Reflexive :
    - x R x
  - Symmetric :
    - x R y → y R x
    - Example :
  - Transitive:
    - x R y **and** y R z → x R z

Languages

- **Symbol :**
  - Atomic unit such as : a , 1, #, True, False (We cannot split into other units)
- **Alphabet :**
  - **Finite** set of symbols
  - Should not be empty
  - Usually denoted by Sigma : Σ
  - Examples :
    - Binary Alphabet : Σ = { 0, 1 }
    - English Alphabet : Σ = { a…z,A …Z}
    - Binary Alphabet : Σ = { a, b }
    - Unary Alphabet : Σ = { z }

Languages

- **String** :
  - is a word with a **finite** set of symbols taken from the defined set of Alphabet.
  - The empty string is denoted by ε
  - $|x|$ = the length of the string x
  - Examples:
    - x= abababab from : Σ = { a, b }
    - x=1111011 from Σ = { 0 , 1 }
    - x=ENSIA Σ = { a,...,z,A,...Z,SPACE}

Languages

- **Powers of An Alphabet : Sets of Strings** :
  - Σ is defined as the alphabet
  - Σ^k = is the set of all strings composed from the alphabet Σ and **have a length of k**
  - $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \cup \ldots$ = the set of all strings from the alphabet Σ
    - Σ^* Called the universal set of all strings
  - $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4 \cup$: All strings excluding the empty string
  - Examples for Σ = { 1, 0 }
    - Σ^2= { 00, 01 , 11 , 10 }
    - Σ^4= { 0011, 1011 ,1111 , 1110 ,... }
    - Σ^*= { ε, 0, 1 , 00, 01 , 11 , 000, 001 ... }

Languages

- **Powers of An Alphabet : Sets of Strings** :
  - If the alphabet Σ={0,1}
  - Σ^* can be also written as : $\{0,1\}^*$

Languages

- **Language :**
  - Not just words : but rules also
  - Is a subset of strings of Σ^* complying with the language rules over the alphabet Σ
  - Examples :
    - Set of words which begin with the letter b
    - Language is Algebraic expressions : (1+3)x2   (Must comply with the rules!)
    - $\{x \in \{0, 1\}^* \mid |x| > 2$ and x begins and ends with $1\}$
    - The language of Balanced Parentheses : { ε, (), (()), ((())),....} over Σ={(,) }

Proving Techniques

- Mathematical theories are constructed starting with some fundamental assumptions, called **axioms**
- A **proof** is a **convincing** logical argument that a statement is true. It must involve a proof technique with clear steps showing how to arrive logically into the final statement.
- A **theorem** is a mathematical statement proved true.

Proving Techniques

- By Contradiction:
  - For a given statement,
    - We assume that the theorem is false
    - Then show that this assumption leads to an obviously false or contradiction
    - This leads to infer that the original statement is true.

Proving Techniques

- By Contrapositive:
  - For a given implication : A → B
    - Example : if x is even , it implies that x^2 is even too.
  - It is equivalent to prove its contrapositive to : ~B → ~A
    - *If x2 is not even, then it implies that x is not even*
    - *We assume that x2 is not even, therefore, x2=2n+1*
    - *Assume that x is even,,, (2m)2=2n+1 ,, 2*2(m)2=2n+1 ? Contradiction as an even can never be odd at the same time, so x is odd.*
    - *By Contraposition,...*

Proving Techniques

- By Induction
- By Construction
- By Counter-Example