DISCRETE MATHEMATICS
GENERIC ELLECTIVE(GI1T2)

Miss Neha
Assistant Professor
DEPARMENT OF MCA

PATNA WOMENS'S COLLEGE

Meaning of Discrete

Discrete means something i.e.; **countable**, **independent, separated**

For example

Total number of students in class-(Particular Value)

Height of students in class-(Continuous Value)

Definition

Discrete mathematics is a branch of mathematics which deals with discrete objects.

Discrete objects are those which are separated from (not connected to/distinct from) each other. automobiles, houses, people etc. are all discrete objects.

Why Discrete Mathematics?

Design efficient computer systems.

↓

To design efficient one must have better understanding of computer system

↓

Working of CPU

↓

CPU works on logic gates

↓

Logic gates based on discrete data i.e. 0,1

Contents

1. Logic and Proofs How do computers think

Logic: Boolean Algebra

Proof: induction, contradiction

2. Counting How many steps are needed to sort n numbers?

- Sets, Combinations, Permutations, Binomial theorem,
- Pigeonhole principle, Recursions, Generating functions

3. Graph Theory Computer networks, circuit design, data structures

- Relations, Graphs, Degree sequence, Eulerian graphs, Trees

4. Number Theory

- Number sequence, Euclidean algorithm

Kind of problems solved by discrete mathematics

- How did Google manage to build a fast search engine?
- What is the foundation of internet security?
- How many ways are there to choose a computer password?
- What is the probability of winning a lottery?
- Is there a link between two users in a social network?
- What is the shortest path between two cities using a transportation system?
- How can a list of integers sorted in increasing order? How many steps are required to do such a sorting?

Problem Solving requires mathematical rigor

- Your boss is not going to ask you to solve
  - an MST (Minimal Spanning Tree) or
  - a TSP (Travelling Salesperson Problem)

- Rarely will you encounter a problem in an abstract setting

- However, he/she may ask you to build a model for the company's salesman to minimize the cost of travelling

- It is up to you to determine
  - a proper model for representing the problem and
  - a correct or efficient algorithm for solving it

Examples of Graph Models:

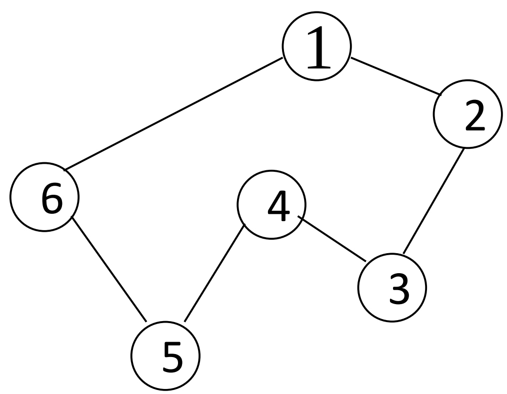
Traveling Salesman Problem

There are n cities. The salesman

- starts his tour from City 1,
- visits each of the cities exactly once,
- and returns to City 1.

For each pair of cities i, j there is a cost c_{ij} associated with traveling from City i to City j .

- Goal: Find a minimum-cost tour.



Situations where counting techniques are used

- There are 4 jobs that should be processed on the same machine.

  (*Can't be processed simultaneously*).

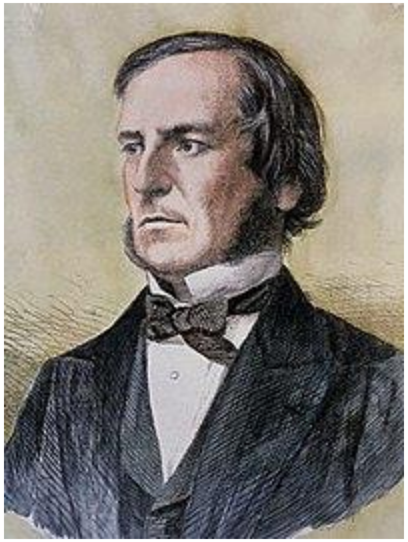
  Here is an example of a possible schedule:

- Counting question: What is the number of all possible schedules?
- Optimization question: Find a schedule that minimizes the average completion time of the four jobs.

What's your job?

- Build a mathematical model for each scenario
- Develop an algorithm for solving each task
- Justify that your solutions work
  - Prove that your algorithms terminate. Termination
  - Prove that your algorithms find a solution when there is one. Completeness
  - Prove that the solution of your algorithms is correct Soundness
  - Prove that your algorithms find the best solution (i.e., maximize profit). Optimality (of the solution)
  - Prove that your algorithms finish before the end of life on earth. Efficiency, time & space complexity

BOOLEAN ALGEBRA



Invented By: George Boole

Definition

- Boolean algebra is a mathematical system for the manipulation of variables that can have one of two values.
  - In formal logic, these values are “true” and “false.”
  - In digital systems, these values are “on” and “off,” 1 and 0, or “high” and “low.”

- Boolean expressions are created by performing operations on Boolean variables.
  - Common Boolean operators include AND, OR, and NOT.

Operators

- Binary Operators
  - AND

    $z = x \bullet y = x\,y$ $\qquad$ $z=1$ *if* $x=1$ *AND* $y=1$
  - OR

    $z = x + y$ $\qquad$ $z=1$ *if* $x=1$ *OR* $y=1$
  - NOT

    $z = \overline{x} = x'$ $\qquad$ $z=1$ *if* $x=0$
- Boolean Algebra
  - Binary Variables: only ‘0’ and ‘1’ values
  - Algebraic Manipulation

Operator Precedence

- Parentheses

  $(\ldots) \bullet (\ldots)$

- NOT

  $x' + y$

- AND

  $x + x \bullet y$

- OR

High

to

low

$$x\,[\,y + z\,\overline{(w + x)}\,]$$

$$(w + x)$$

$$\overline{(w + x)}$$

$$z\,\overline{(w + x)}$$

$$y + z\,\overline{(w + x)}$$

$$x\,[\,y + z\,\overline{(w + x)}\,]$$

Boolean Algebra Postulates

1. **Commutative Law**

   $x \bullet y = y \bullet x$ $\qquad$ $x + y = y + x$

2. **Identity Law**

   $x \bullet 1 = x$ $\qquad$ $x + 0 = x$

3. **Complement Law**

   $x \bullet x' = 0$ $\qquad$ $x + x' = 1$

4. **Associative Law**

   $x(y \bullet z) = (x \bullet y) \bullet z$ $\qquad$ $x+(y + z)=(x + y)+z$

5. **Distributive Law**

   $x \bullet(y + z) =x \bullet y + x \bullet z$ $\qquad$ $x + y \bullet z=(x + y)\bullet(x + z)$

BASIC THEOREMS

AND

PROPERTIES OF BOOLEAN ALGEBRA

Boolean Algebra Theorems

- Duality Principal
  - *The duality principal states that every algebraic expression is deducible if operator and identity elements are interchanged.*

<u>Example</u>

$$x \bullet (y + z) = (x \bullet y) + (x \bullet z)$$

$$x + (y \bullet z) = (x + y) \bullet (x + z)$$

Boolean Algebra Theorems

- Theorem 1 Idempotent Law
  - $x \bullet x = x$ $\qquad x + x = x$
- Theorem 2 Dominance Law
  - $x \bullet 0 = 0$ $\qquad x + 1 = 1$
- Theorem 3: *Involution Law*
  - $(x')' = x$ $\qquad \overline{(\overline{x})} = x$
- Theorem 4: *De Morgan's Law*
  - $(x \bullet y)' = x' + y'$ $\qquad (x + y)' = x' \bullet y'$
  - $\overline{(x \bullet y)} = \overline{x} + \overline{y}$ $\qquad \overline{(x + y)} = \overline{x} \bullet \overline{y}$
- Theorem 5: *Absorption Law*
  - $x \bullet (x + y) = x$ $\qquad x + (x \bullet y) = x$

Basic Theorems

- **Basic Theorems**: proven by the postulates of table

Theorem 1(a): $x + x = x$

$x = x + 0$	by Identity Law
$= x + x \,.\, x'$	Complement's Law
$= (x + x).(x + x')$	Distributive Law
$= (x + x) \cdot 1$	Identity Law
$= x + x$	

Theorem 1(b): $x \cdot x = x$

$x = x \cdot 1$	by Identity Law
$= x \cdot (x + x')$	Complement's Law
$= x \cdot x + x \cdot x'$	Distributive Law
$= x \cdot x + 0$	Complement's Law
$= x.\, x$	Identity Law

Basic Theorems

Absorption Law: $x + xy = x$

$$
\begin{aligned}
x + xy &= x \cdot 1 + x \cdot y && \text{by Identity Law} \\
&= x \cdot (1 + y) && \text{Distributive Law} \\
&= x\,(y+1) && \text{Commutative Law} \\
&= x \cdot 1 && \text{Dominance Law} \\
&= x && \text{Identity law}
\end{aligned}
$$

- The theorems of Boolean algebra can be shown to hold true by means of truth tables.

x	y	$x \cdot y$	$x + x \cdot y$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

First absorption theorem

Boolean Functions

- Boolean Expression also acts as a function which means its takes some input and mapped them to an output

  Example: $F = x + y' z$

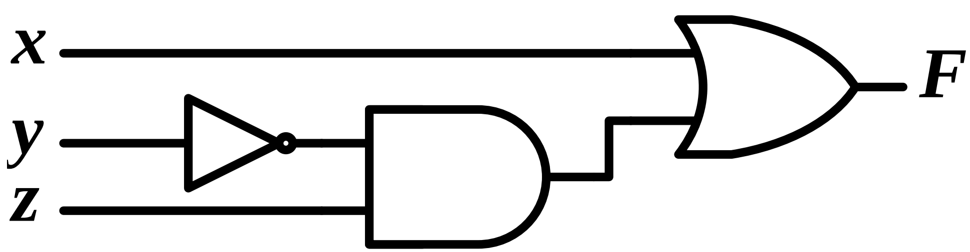
- Truth Table

  A Boolean function can

  be represented in a truth table.

  the binary combinations for the truth table obtained by counting from 0 through 2^n-1

- Logic Circuit



x	y	y'	z	y'· z	F
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	1	0	0
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	0	1

Simplification of the algebraic

- A one way to represent Boolean function in a truth table.
- In algebraic form, it can be expressed in a variety of ways.
- By simplifying Boolean algebra, we can reduce the number of gates in the circuit and the number of inputs to the gate.

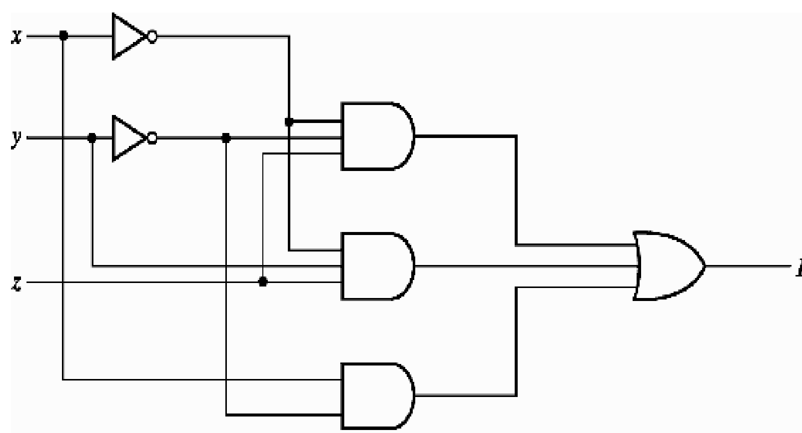
Before simplification of Boolean function

- Consider the following Boolean function:
  $F_2 = x'y'z + x'yz + xy'$

- This function with logic gates is shown in Fig. 2-2(a)

- The function is equal to 1 when xyz = 001 or 011 or when xyz = 10x .

x	y	z	F_2
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



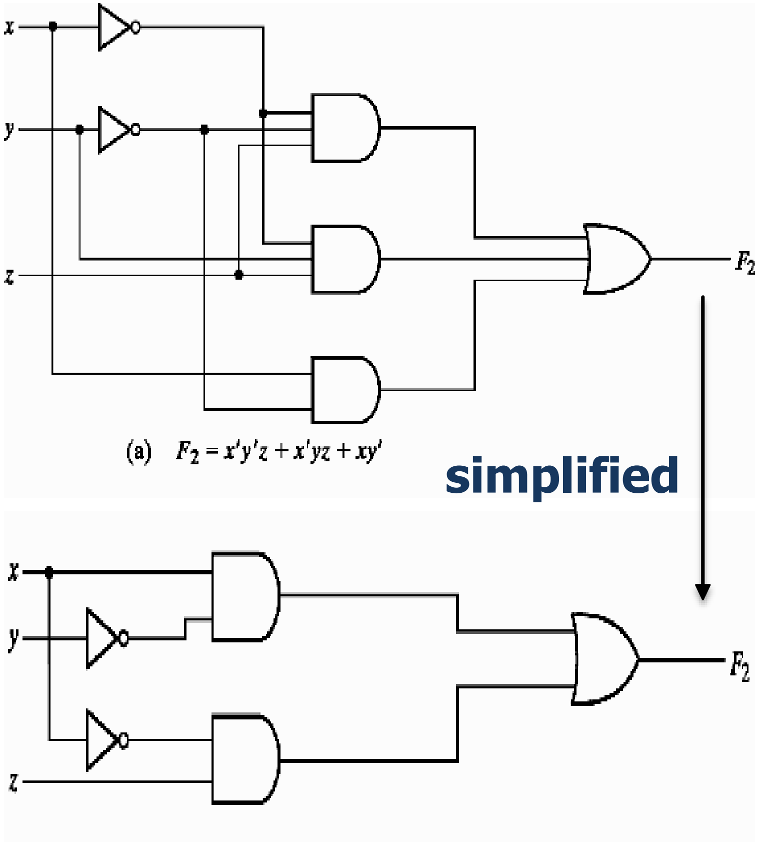
(a) $F_2 = x'y'z + x'yz + xy'$

After simplification of Boolean function

- Simplify the following Boolean function:

  $F_2 = x'y'z + x'yz + xy'$
  $= x'z(y' + y) + xy'$
  $= x'z + xy'$

- In 2-2 (b), would be preferable because it requires less wires and components.



(a) $F_2 = x'y'z + x'yz + xy'$

(b) $F_2 = xy' + x'z$

Fig. 2-2 Implementation of Boolean function F_2 with gates

Algebraic Manipulation

- *Literal*:

  A single variable within a term that may be complemented or not.

- Use Boolean Algebra to simplify Boolean functions to produce simpler circuits

  Example: Simplify to a minimum number of literals

  $F = x + x'y$ (3 *Literals*)

  $= x + (x'y)$

  $= (x + x')(x + y)$ ***Distributive law* (+ over •)**

  $= (1)(x + y) = x + y$ (2 *Literals*)

Canonical Form

- Boolean function expressed in **SOP** or **POS** from are said to be in canonical form.
- **SOP (Sum of Product)**

When Boolean function is expressed as a sum of minterms, its is called its sum of product expansion.

- **POS(Product of Sum)**

When Boolean function is expressed as a product of maxterms, its is called its product of sum expansion.

Canonical Forms

- **Minterm**
  - Product (*AND* function)
  - Contains all variables in normal or complemented form.
  - Evaluates to '1' for a specific combination

Example

$A = 1$
$B = 1$
$C = 1$

$$\left.\begin{matrix} A = 1 \\ B = 1 \\ C = 1 \end{matrix}\right\} \begin{matrix} A & & B & & C \\ (1) & \bullet & (1) & \bullet & (1) \\ \downarrow & & \downarrow & & \downarrow \\ 1 & \bullet & 1 & \bullet & 1 = 1 \end{matrix}$$

	A B C	Minterm	
0	0 0 0	m_0	$\bar{A}\bar{B}\bar{C}$
1	0 0 1	m_1	$\bar{A}\bar{B}C$
2	0 1 0	m_2	$\bar{A}B\bar{C}$
3	0 1 1	m_3	$\bar{A}BC$
4	1 0 0	m_4	$A\bar{B}\bar{C}$
5	1 0 1	m_5	$A\bar{B}C$
6	1 1 0	m_6	$AB\bar{C}$
7	1 1 1	m_7	ABC

Canonical Forms

- Maxterm
  - Sum (*OR* function)
  - Contains all variables
  - Evaluates to '0' for a specific combination

Example

$$\left.\begin{array}{l} A = 0 \\ B = 0 \\ C = 0 \end{array}\right\} \quad \underset{0}{A} + \underset{0}{B} + \underset{0}{C} = 0$$

	A B C	Maxterm	
0	0 0 0	M_0	$A+B+C$
1	0 0 1	M_1	$A+B+\overline{C}$
2	0 1 0	M_2	$A+\overline{B}+C$
3	0 1 1	M_3	$A+\overline{B}+\overline{C}$
4	1 0 0	M_4	$\overline{A}+B+C$
5	1 0 1	M_5	$\overline{A}+B+\overline{C}$
6	1 1 0	M_6	$\overline{A}+\overline{B}+C$
7	1 1 1	M_7	$\overline{A}+\overline{B}+\overline{C}$

Canonical Forms

- Truth Table *to* Boolean Function

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

$$F = \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + A\overline{B}C + ABC$$

Canonical Forms

- Sum of *Minterms*

$$F = \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + ABC$$

$$F = m_1 + m_4 + m_5 + m_7$$

$$F = \sum(1,4,5,7)$$

- Product of *Maxterms*

$$\bar{F} = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}$$

$$\bar{\bar{F}} = \overline{\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}}$$

$$F = \overline{\bar{A}\bar{B}\bar{C}} \cdot \overline{\bar{A}B\bar{C}} \cdot \overline{\bar{A}BC} \cdot \overline{AB\bar{C}}$$

$$F = (A+B+C)(A+\bar{B}+C)(A+\bar{B}+\bar{C})(\bar{A}+\bar{B}+C)$$

$$F = M_0 \qquad M_2 \qquad M_3 \qquad M_6$$

$$F = \prod(0,2,3,6)$$

	A B C	F	$\bar{F}$
0	0 0 0	0	1
1	0 0 1	1	0
2	0 1 0	0	1
3	0 1 1	0	1
4	1 0 0	1	0
5	1 0 1	1	0
6	1 1 0	0	1
7	1 1 1	1	0

Standard Forms

- Sum of Products (SOP)

$$F = \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + ABC$$

$$A\bar{B}(\bar{C} + C)$$

$$= A\bar{B}(1)$$

$$= A\bar{B}$$

$$AC(\bar{B} + B)$$

$$= AC$$

$$\bar{B}C(\bar{A} + A)$$

$$= \bar{B}C$$

$$F = \bar{B}C(\cancel{\bar{A} + A}) + A\bar{B}(\cancel{\bar{C} + C}) + AC(\cancel{\bar{B} + B})$$

$$F = \bar{B}C + A\bar{B} + AC$$

Standard Forms

- Product of Sums (POS)

$$\bar{A}B(\bar{C}+C)$$

$$\bar{F} = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C}$$

$$B\bar{C}(\bar{A}+A)$$

$$\bar{A}\bar{C}(\bar{B}+B)$$

$$\bar{F} = \bar{A}\bar{C}\cancel{(\bar{B}+B)} + \bar{A}B\cancel{(\bar{C}+C)} + B\bar{C}\cancel{(\bar{A}+A)}$$

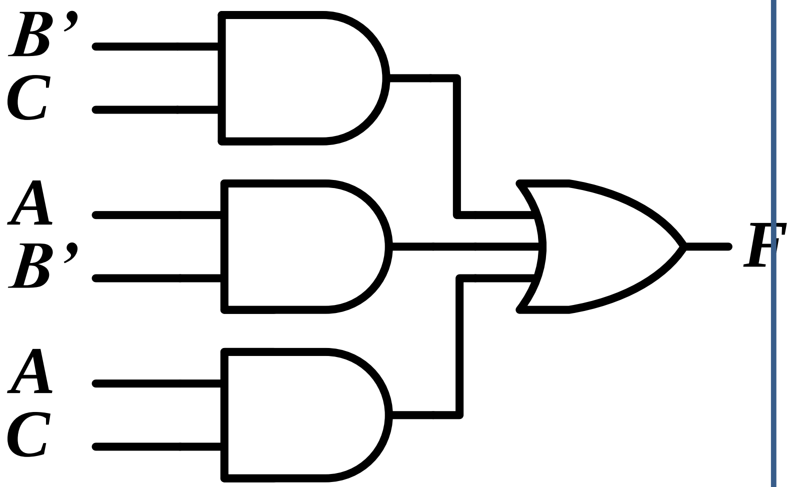
$$\overline{\overline{F}} = \overline{\bar{A}\bar{C} + \bar{A}B + B\bar{C}}$$

$$F = (A+C)(A+\bar{B})(\bar{B}+C)$$

Two-Level Implementations

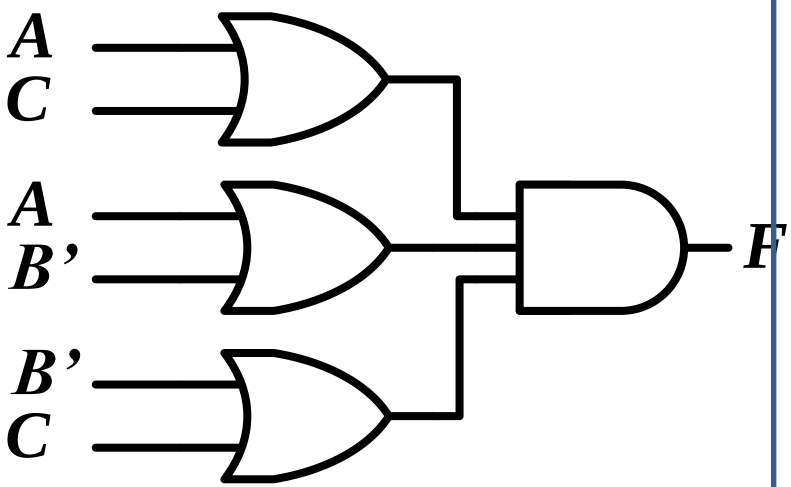
- Sum of Products (SOP)

$$F = \overline{B}C + A\overline{B} + AC$$



- Product of Sums (POS)

$$F = (A + C)(A + \overline{B})(\overline{B} + C)$$



The Karnaugh Map

- Feel a little difficult using Boolean algebra laws, rules, and theorems to simplify logic?
- A K-map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP or POS expression possible, known as the <u>minimum expression</u>.

What Is K-map

- It’s similar to truth table; instead of being organized (i/p and o/p) into columns and rows, the K-map is an array of cells in which each cell represents a binary value of the input variables.
- The cells are arranged in a way so that simplification of a given expression is simply a matter of properly grouping the cells.
- K-maps can be used for expressions with 2, 3, 4, and 5 variables.

THE 3 VARIABLE K-MAP

- There are 8 cells as shown:

AB \ C	0	1
00	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$
01	$\overline{A}B\overline{C}$	$\overline{A}BC$
11	$AB\overline{C}$	ABC
10	$A\overline{B}\,\overline{C}$	$A\overline{B}C$

THE 4-VARIABLE K-MAP

AB \ CD	00	01	11	10
00	$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$	$\overline{A}\,\overline{B}\,\overline{C}D$	$\overline{A}\,\overline{B}CD$	$\overline{A}\,\overline{B}C\overline{D}$
01	$\overline{A}B\overline{C}\,\overline{D}$	$\overline{A}B\overline{C}D$	$\overline{A}BCD$	$\overline{A}BC\overline{D}$
11	$AB\overline{C}\,\overline{D}$	$AB\overline{C}D$	$ABCD$	$ABC\overline{D}$
10	$A\overline{B}\,\overline{C}\,\overline{D}$	$A\overline{B}\,\overline{C}D$	$A\overline{B}CD$	$A\overline{B}C\overline{D}$

K-map SOP Minimization

- The K-Map is used for simplifying Boolean expressions to their minimal form.
- A minimized SOP expression contains the fewest possible terms with fewest possible variables per term.
- Generally, a minimum SOP expression can be implemented with fewer logic gates than a standard expression.

MAPPING A STANDARD SOP EXPRESSION

For an SOP expression in standard form:

- A 1 is placed on the K- map for each product term in the expression.
- Each 1 is placed in a cell corresponding to the value of a product term.

AB \ C	0	1
00	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$
01	$\overline{A}B\overline{C}$	$\overline{A}BC$
11	$AB\overline{C}$	ABC
10	$A\overline{B}\,\overline{C}$	$A\overline{B}C$

MAPPING A STANDARD SOP EXPRESSION (FULL EXAMPLE)

The expression:

$$\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + AB\overline{C} + A\overline{B}\,\overline{C}$$

000 001 110 100

AB \ C	0	1
00	1	1
01		
11	1	
10	1	

Practice:

$A\overline{B}\,\overline{C} + AB\overline{C} + ABC + ABC$

$\overline{A}BC + A\overline{B}C + A\overline{B}\,\overline{C}$

$\overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}BCD$

MAPPING A NONSTANDARD SOP EXPRESSION

- A Boolean expression must be in standard form before you use a K-map.
  - If one is not in standard form, it must be converted.
- You may use the procedure mentioned earlier or use numerical expansion.

K-MAP SIMPLIFICATION OF SOP EXPRESSIONS

- After an SOP expression has been mapped, we can do the process of *minimization:*
  - Grouping the 1s
  - Determining the minimum SOP expression from the map

Grouping Of 1s

- You can group 1s on the K-map according to the following rules by enclosing those adjacent cells containing 1s.
- **The goal** is to maximize the size of the groups and to minimize the number of groups.

Grouping Of 1s (Rules)

1. A group must contain either 1,2,4,8,or 16 cells (depending on number of variables in the expression)
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group do not have to be adjacent to each other.
3. Always include the largest possible number of 1s in a group in accordance with rule 1.
4. Each 1 on the map must be included in at least one group. The 1s already in a group can be included in another group as long as the overlapping groups include noncommon 1s.

Grouping The 1s (Example)

AB \ C	0	1
00	1	
01		1
11	1	1
10		

AB \ C	0	1
00	1	1
01	1	
11		1
10	1	1

Grouping Of 1s (Example)

AB \ CD	00	01	11	10
00	1	1		
01	1	1	1	1
11				
10		1	1	

AB \ CD	00	01	11	10
00	1			1
01	1	1		1
11	1	1		1
10	1		1	1