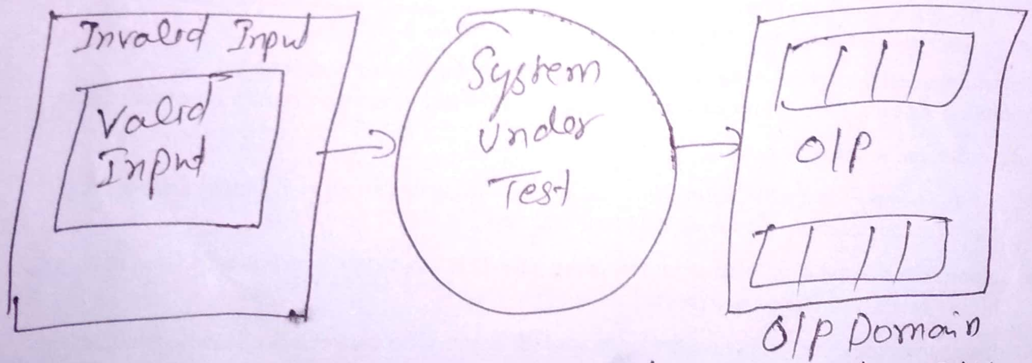
Equivalence Class Testing :->

(1) The equivalence class are identified by taking each Input condition and partitioning it into Valid and Invalid classes. For eg: If an input condition speifies a range of values from 1 — 999. We identify one valid equivalence Class {1 < item < 999] and two Invalid equivalence classes {item < 1} and [item > 999]

(2) Generate the test Cases using the equivalence classes identified in the previous step. This is performed by writing test cases covering all the valid equivalence classes. Then a test case is written for each ~~individual~~ invalid equivalence class, so that no test contains more than one invalid class.



for eg: -> A program to accept any no. 1—99. four equivalence classes

(I) Any no. between 1 and 99 is valid Input

(II) Any number less than 1, this include 0 and all negative numbers.

(III) Any number greater than 99.

(IV) if it is not a number, It should not be accepted

ecision Table Based Testing :→

Decision tables are useful for describing situations in which a number of combinations of action are taken under varying sets of conditions. Decision tables, have been used to represent and analyse complex logical relationship since early 1960's.

Decision Table Terminology

Condition stub	Entry							
C1	True				False			
C2	True		False		True		False	
C3	True	False	True	False	True	False	True	False
Action stub a1	X	X			X			
a2	X		X			X		
a3		X						
a4				X		X		X

Cause effect Graphing Techniques :→

no. of test cases – 2^n.

for different Input combinations.

→ Used to pointing out incompleteness and ambiguities in the specification.

→ Cause – Input
effect – O/P.

→

~~one Renew 10 Iteration Perplexity 2000 sum~~

~~10 1957.475 2222~~

~~15 10 1957 → 1969~~

~~20 10 1962~~

~~25 10~~

~~30 10 32+11~~

~~35 10 1933 43~~

① Cause and effect are identified ③
by reading spesification

② Boolean graph linking the cause and effects.

③ Graph is annotated with combination of causes and effects

④ The graph is converted into a limited entry decision table. each column in the table represents a test case.

⑤ The column in the decision table are converted into test cases

→ Identity function

If c_1 is 1
The e_1 is 1 else 0.

→ Not function

If c_1 is 1
Then e1 is 0 else e1 is 1.

→ OR funtion.

if c_1 ~~or~~ c_2 or c_3 is 1
then e_1 is 1 else e1 is 0.

→ AND funtion

If both c_1 and c_2 are 1
Then e_1 is 1; else e1 is zero

E constraints states that it must always be true that at most one of C1 or C2 can be 1

→ I - atleast one of C1, C2 and C3 must always be 1, C1, C2 and C3 cannot be 0 simultaneously.

→ O - constraints states that one and only one of C1 and C2 must be 1

→ R - constraints states that for C1 to be 1 C2 must be 1.

→ Example :→

Causes are C1 → side x is less than sum of sides x and Z.

C2 → side y is less than sum of sides x and z

C3 → side z is less than sum of x & y

C4 → side x is equal to side y.

C5 → side x is equal to side z

C6 → side y is equal to side z

effects :→ e1 - not a triangle

e2 - Scalene triangle

e3 - Isosceles triangle

e4 → Equilateral "

e5 - Impopssible stage

ructural Testing :→ Code Testing.

) Path Testing → It requires complete knowledge of the program's structure and used by developers to unit test their own code. not good for system testing.

→ generating a set of paths that will cover every branch in the program

→ finding a set of test-cases that will exeute every path in this set of program paths.

Flow Graph :→ The control flow of a program can be analyzed using a graphical representation known as flow graph. The flow graph is a directed graph in which nodes are either entire statements or fragments of a statement. and edges represents flow of control.

DD – Path :→ DD- Path Graph :→ The next ~~pat~~ step is to draw a DD path graph from the flow graph.. The DD path is known as Decision to Decision path Graph. Here we concentrate only on decision nodes. The nodes of flow graph, which are in a sequence are combined into a single node.

Hence DD- Graph is a directed graph in which nodes are sequence of statements and edges represents control flow between node.

Indepemt Paths :→ find Independent paths that ID path is any path through the DD path graph that introduces at-least one new set of processing statemens or new conditions is quite Interesting to use Independent Path to Ensure that

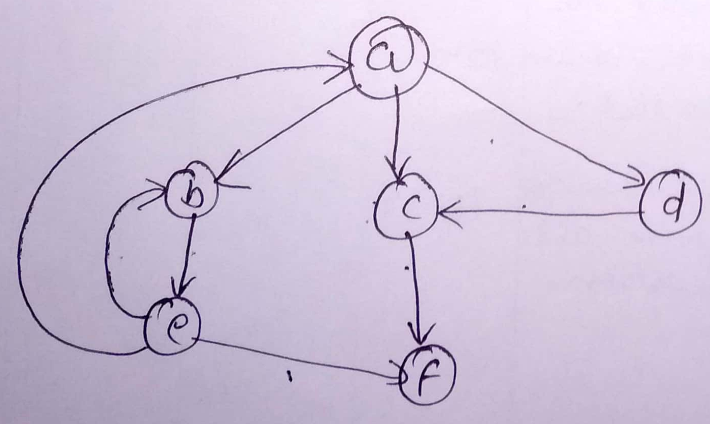
(I) Every statement i in the program has been executed at least once.

(II) Every branch has been exercised for True and false conditions

Cyclomatic Complexity :→ This approach is used to find the number of Independent path through a program. This provides us the Upper bound for the number of tests that must be ~~executed~~ conducted to ensure that all statements have been executed at least once and every condition has been executed on its true and false side.

Cyclomatic Complexity of a graph G with n vertices, e edges, and P connected components is

$$V(G) = e - n + 2P$$



$e = 9$
$n = 6$
$P = 1$

Fig:

$V(G) = 9 - 6 + 2 * 1$

$= 9 - 6 + 2 = 5$

= acf
= abef
= adcf
= abeacf
= abebef

ata flow Testing :→

In this we concentrate on the usage of variables and the focus points are

① Statements where variables receive values

② statements where those values are used or referenced.

③ Define and Reference variables anomolies

① A variable is defined but not used

② A variable is used but never defined

③ A variable is defined twice before it is used.

Definitions :→ Defining node:→