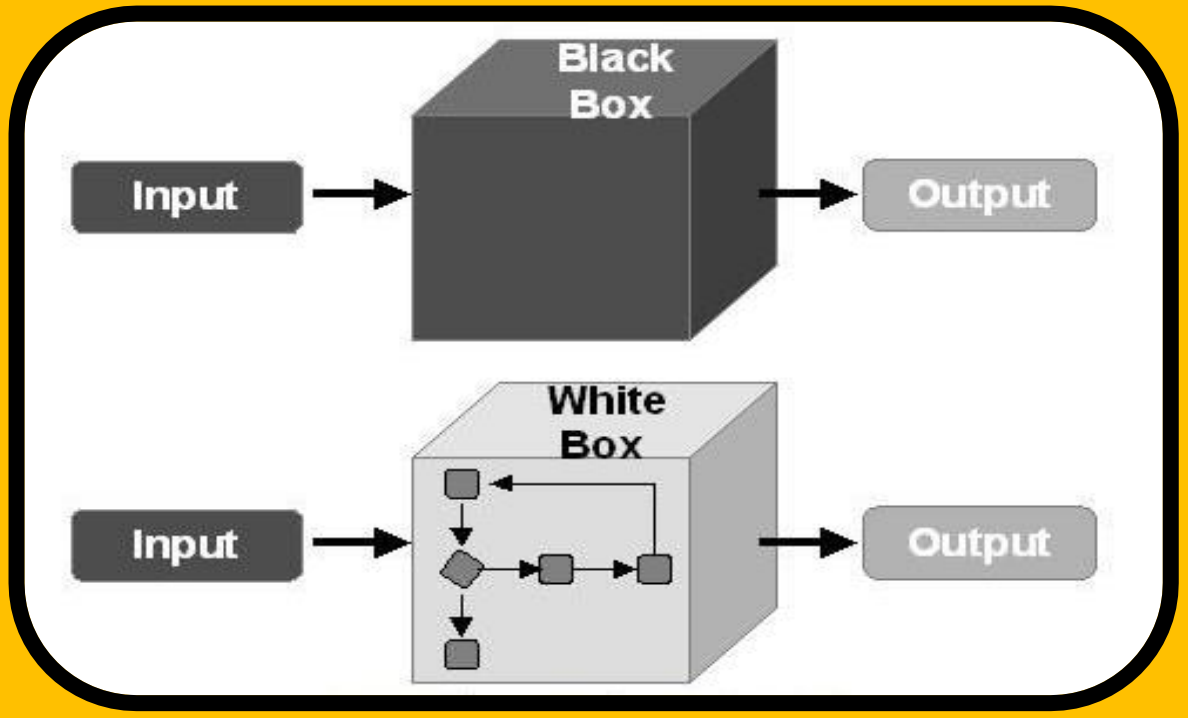
Software Testing Techniques

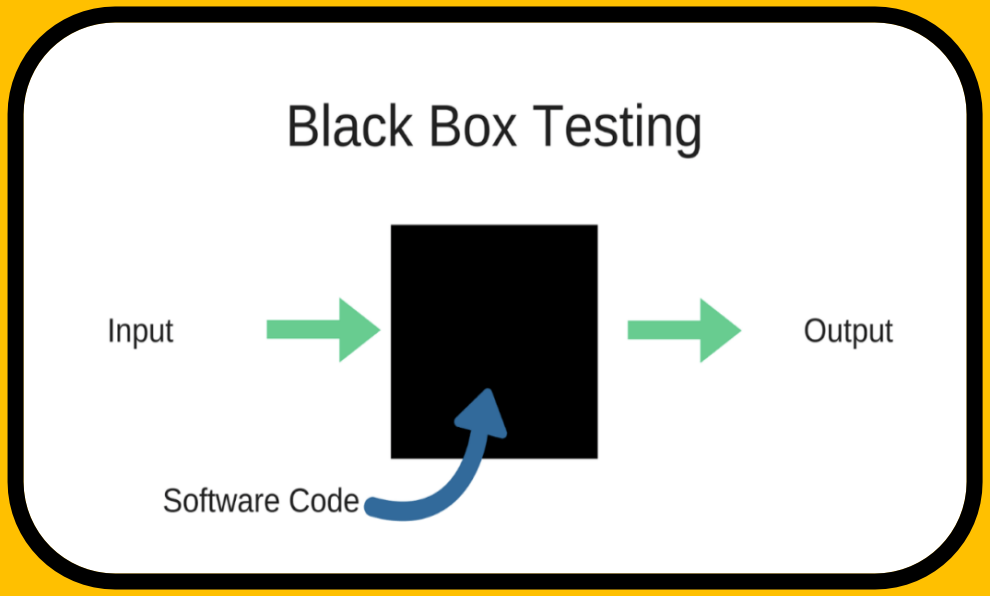
Software Testing Techniques

- Black Box Testing
- White Box Testing



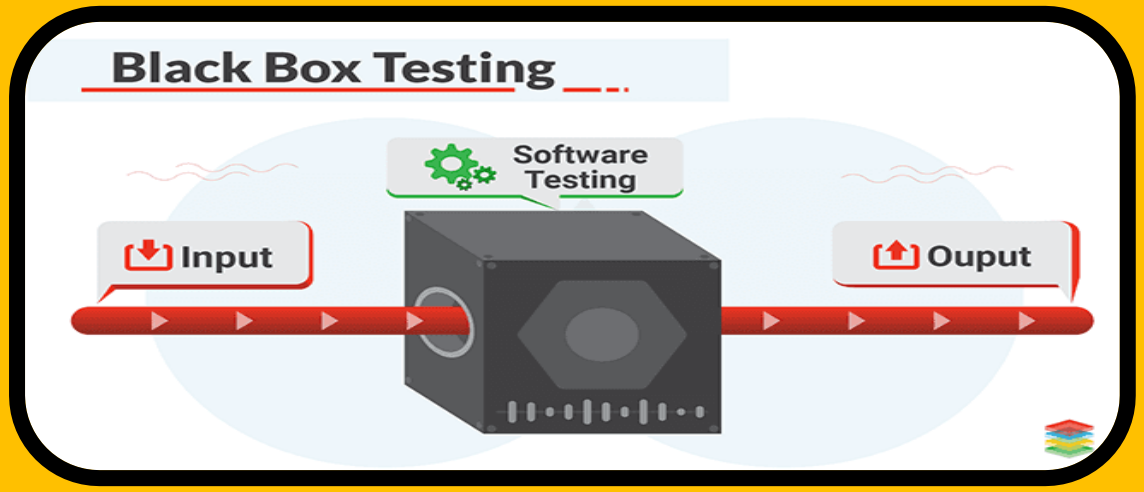
Black Box Testing

Knowing specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational while at the same time searching for errors in each function.

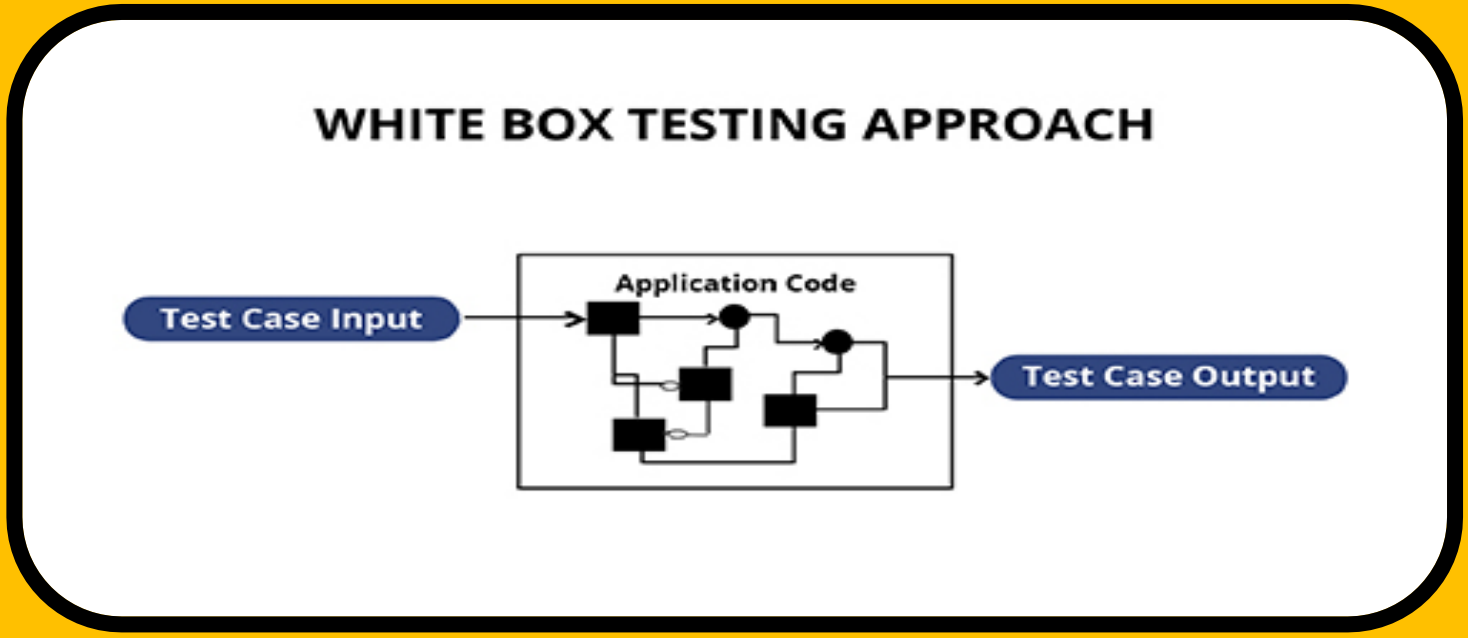


Black Box Testing

- Focus on interfaces
- Inputs and outputs
- Integrity of external information e.g., a database



White-Box Testing



... our goal is to ensure that all statements and conditions have been executed at least once ...

White-Box Testing

Why Cover?

- logic errors and incorrect assumptions are inversely proportional to a path's execution probability
- we often believe that a path is not likely to be executed; in fact, reality is often counter intuitive
- typographical errors are random; it's likely that untested paths will contain some

White-Box Testing

Cyclomatic Complexity:

It is a software metric that provides a quantitative measure of the logical complexity of a program

"The value computed for Cyclomatic complexity defines the number of independent paths in the basis set of a program and provides us with an upper bound for the number of tests that must be conducted to ensure that all statement have been executed at least once".

White-Box Testing

Cyclomatic Complexity

V(G) of a flow graph G:

Number of simple predicates (decisions) + 1

or

V(G) = E-N+2 (where E are edges and N are nodes)

or

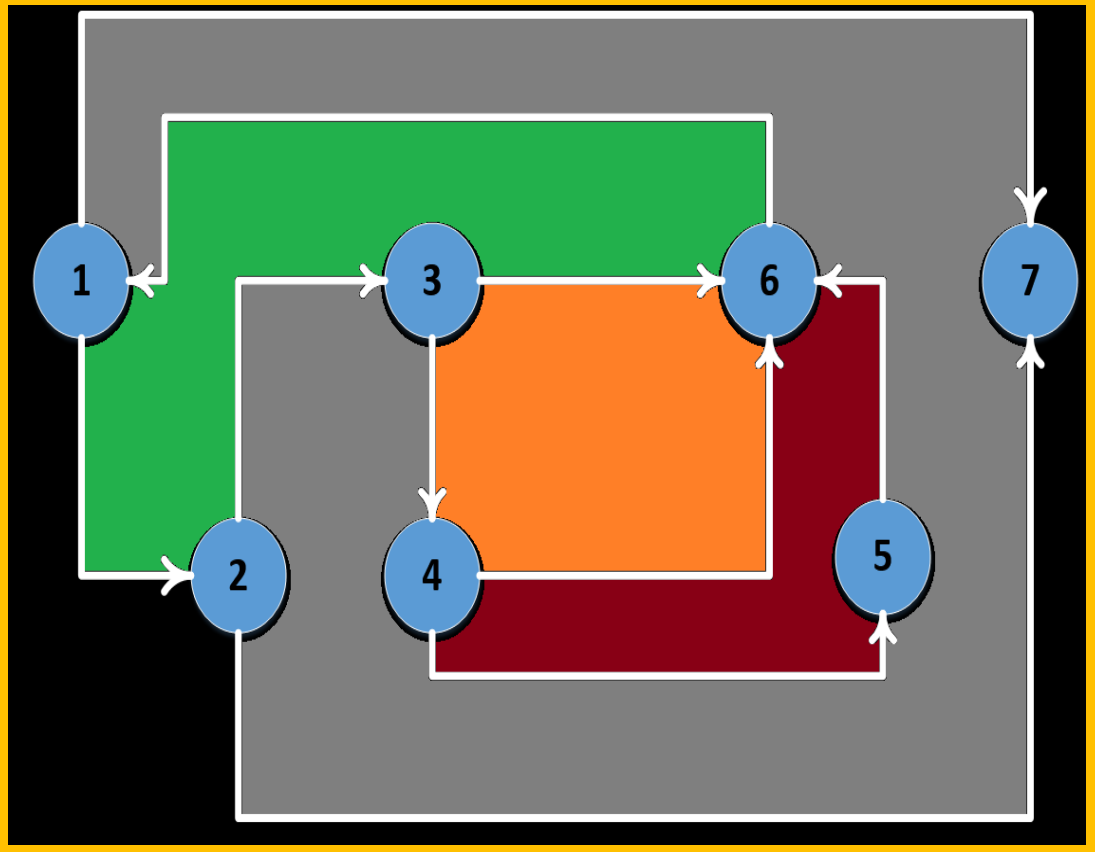
Number of enclosed areas + 1

In this case V(G) = 4

Example

```
while(value[i] != -999.0 && totinputs < 100)
{ totinputs++;
    if(value[i] >= min && value[i] <= max)
        {       totvalid++;
                sum = sum + value[i];
        }
  i++;
}
```

```
                1          2
while(value[i] != -999.0 && totinputs < 100)

{ totinputs++;
                3          4
  if(value[i] >= min && value[i] <= max)

        { totvalid++;

                sum = sum + value[i];          5

        }

  i++;          6

}
  7
```

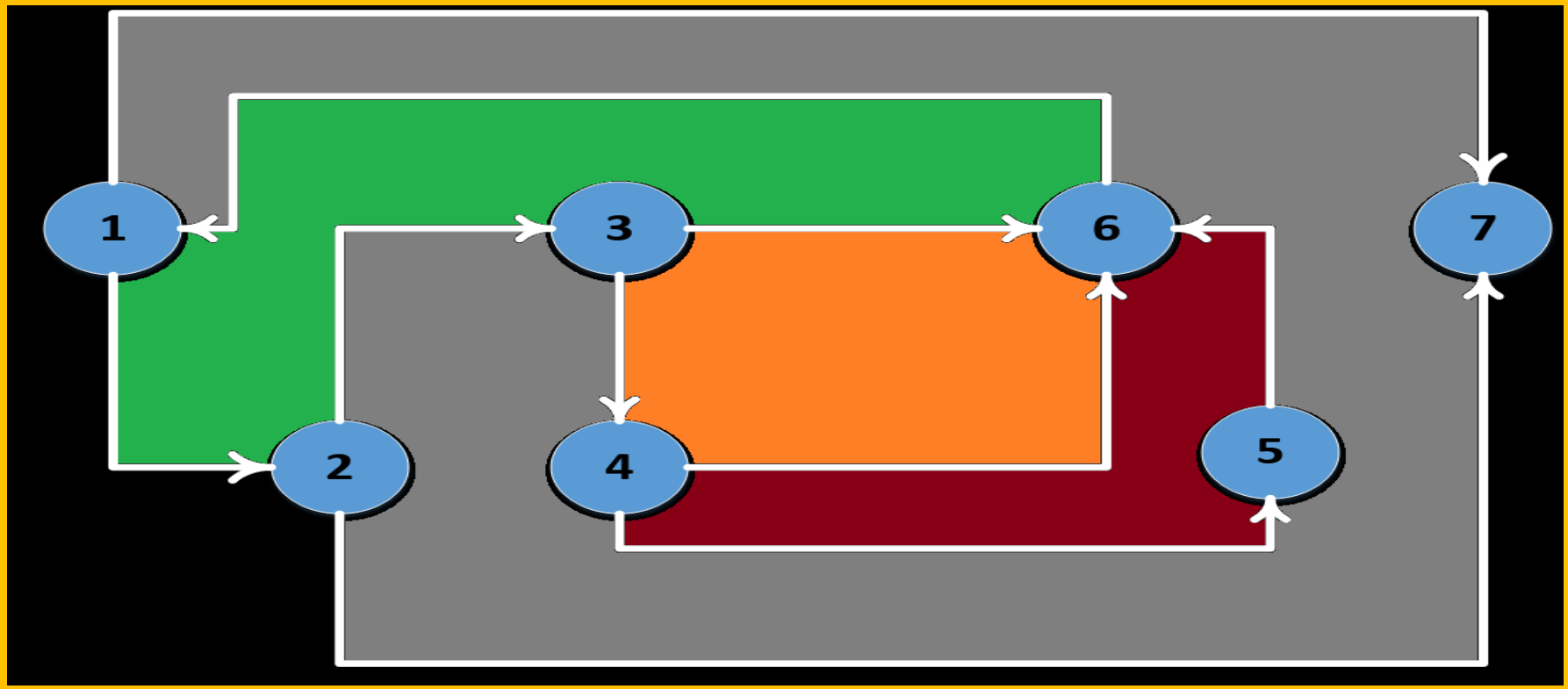


- **Cyclomatic Complexity:**

  V(G) = number of enclosed areas + 1 = 5

  V(G) = number of simple predicates + 1 = 5

  V(G) = edges - nodes + 2 = 10 - 7 + 2 = 5

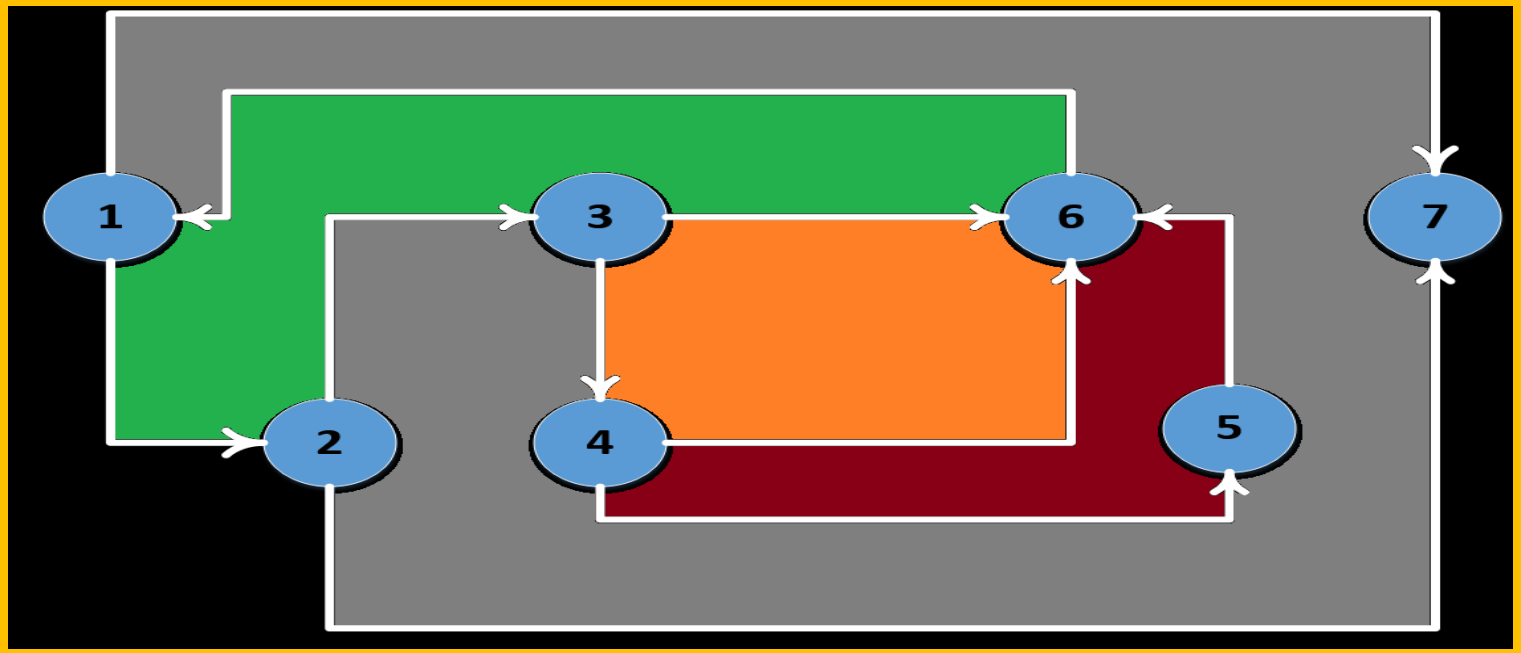


Cyclomatic Complexity by Graph Matrix

	1	2	3	4	5	6	7	Calculations	
1		X					X	= 2	= 2-1 = 1
2			X				X	= 2	= 2-1 = 1
3				X		X		= 2	= 2-1 = 1
4					X	X		= 2	= 2-1 = 1
5						X		= 1	= 1-1 = 0
6	X							= 1	= 1-1 = 0
7								---	---
Cyclomatic Complexity									= 4 + 1 = 5

Independent Paths:

1. **1-7 (**value[i] = -999.0**)**
2. **1-2-7 (**value[i] = 0, totinputs = 100**)**
3. **1-2-3-6-1-7**
4. **1-2-3-4-6-1-7**
5. **1-2-3-4-5-6-1-7**



Black-Box Testing

Also called Behavioral Testing or Functional Testing

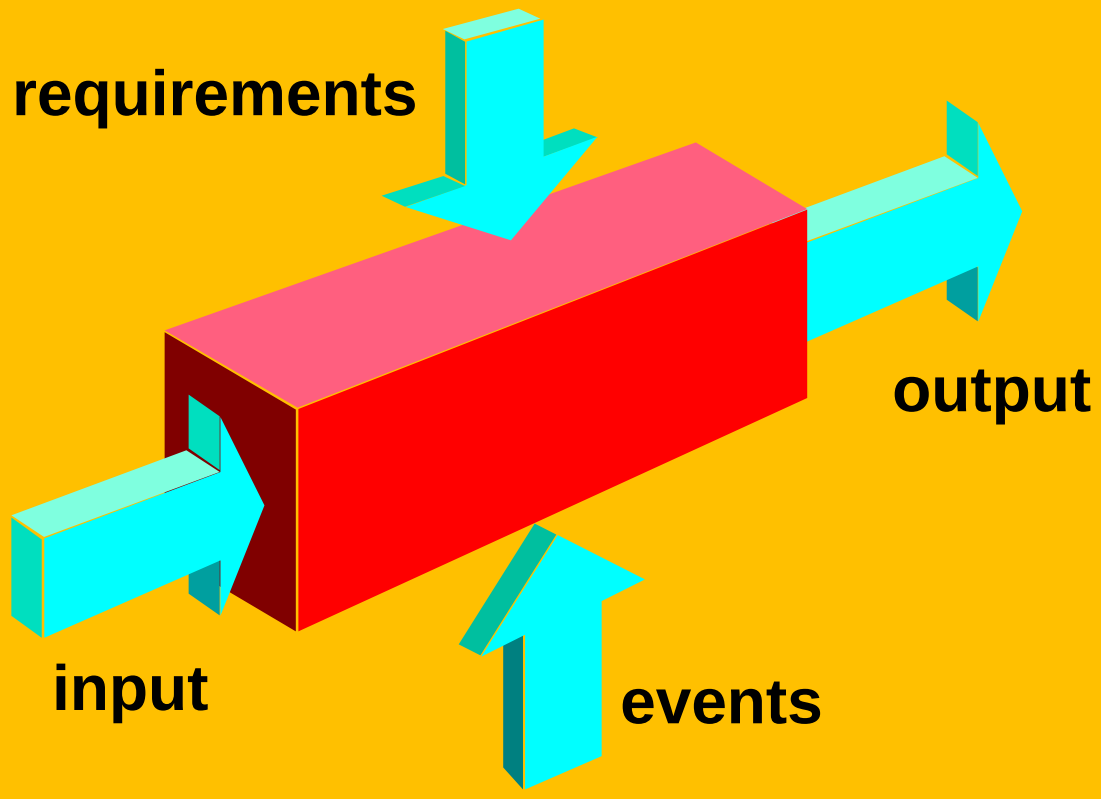
Focus on:

- **Functional Requirements of the software i.e. Inputs, Outputs**
- **Structure of program is not considered**

Black box testing is not an alternative to white box testing.

Different kind of errors are uncovered in each technique.

Black-Box Testing



Black-Box Testing

Equivalence Partitioning:

Since exhaustive testing is not possible so:

Divide the domain of all the inputs into a set of equivalence classes

Example: Determine absolute value for integer

Two behaviors: 1. For positive integers

2. For negative integers

Two classes will be formed: One for positive integers

Second for negative integers

Similarly for valid and invalid inputs we have two classes

Equivalence Partitioning

If we have input condition: 0 < count < Max

Three behaviors:

1. valid in put i.e. count within range
2. upper bound (Invalid)
3. lower bound (Invalid)

Three classes will be formed:

One for valid inputs

Second for upper bound

Third for lower bound

Black-Box Testing

Boundary Value Analysis:

It is observed that programs that work correctly for a set of values in an equivalence class, fail on some special values i.e. on boundaries.

We choose an input from equivalence classes that lies on the edge of class

Example: if range is $0.0 < x < 1$ then test cases are:

Valid inputs: 0.1, 0.9

invalid inputs: 0, 1

Black-Box Testing

Boundary Value Analysis: A guide line

- **If an input conditions specifies a range**
- **If an input condition specifies a number of values**
- **Same guideline for output conditions**
- **Boundaries of data structures are also tested.**

Practice Quiz

```
void fib (int x)
{
    int a=0,b=1,sum=0;
    for(int i=1;i<=x;i++)
    {
            sum=a+b;
            cout<<sum<<"\t";
            a=b;
            b=sum;

    }
}
```

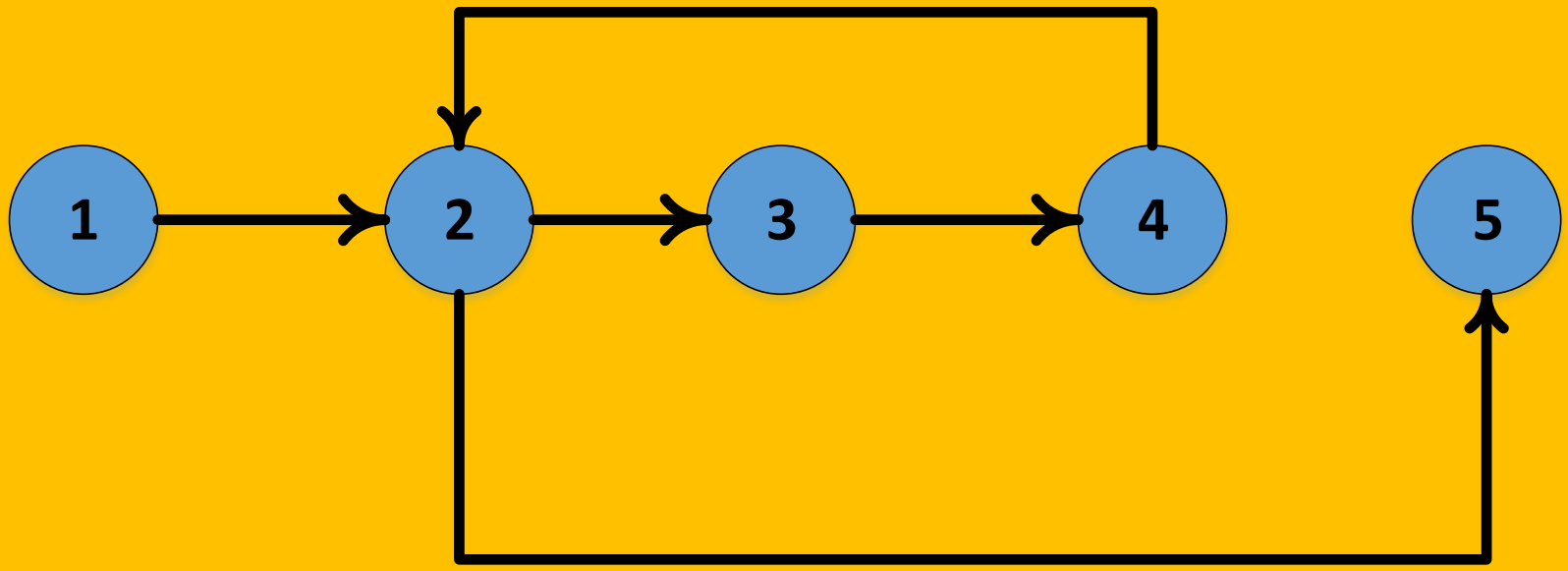
Solution

Cyclomatic Complexity:

Number of regions = 2

Number of predicates = 1+1 = 2

Edges – Nodes + 2 = 5 – 5 + 2 = 2



Practice Question

```
Public boolean find(int key)
{     int bootom = 0;              int top = elements.length-1; int last index = (bottom + top)/2; int mid;
      boolean found = key == elements [lastIndex];
                  while ((bottom <= top) && !found )
                  {
                                    mid = (bottom + top) / 2;
                                    found = key == elements[mid];
                                    if (found)
                                      {
                                                        lastIndex = mid;
                                      }
                                    else
                                      {
                                                        if (elements[mid] < key)
                                                                          {
                                                                                            bottom = mid +1;
                                                                          }
                                                        else
                                                                          {
                                                                                            top = mid – 1;
                                                                          }

                                      }

                  }
}
```