



# Flow of presentation

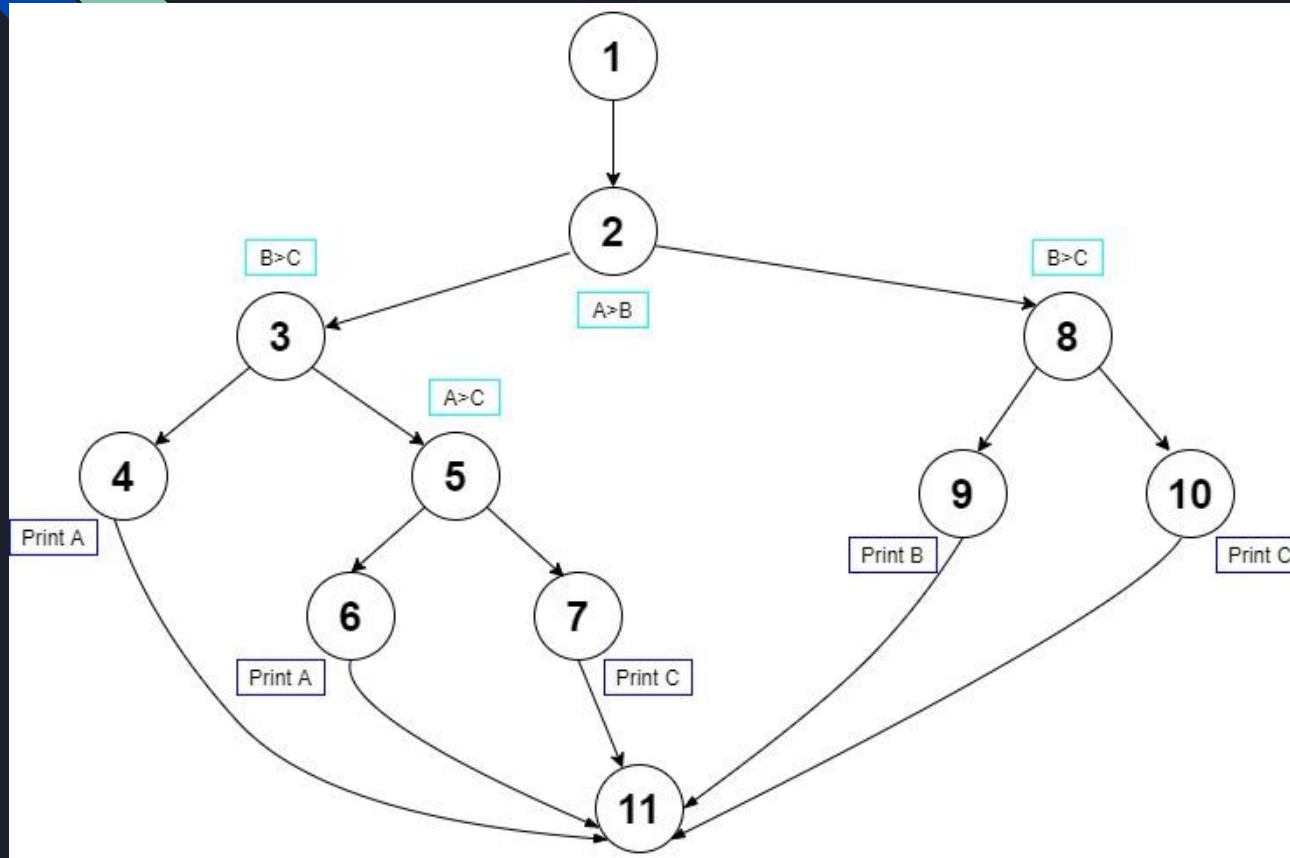
1. What is Test Case Optimization in Software Testing?
2. Need for Test Case Optimization
3. Solution for Test Case Optimization? Genetic Algorithm
4. Genetic Algorithm and its Phases
5. Demonstrating Genetic Algorithm on our example.
6. Conclusion and References



# What is Test Case Optimization ?

- We will explain test case *Optimization*, by demonstrating Software Testing on an example
- We will consider a C program for finding 'Greatest among 3 numbers'
- Then we will dry run a few random test cases on DFD (Data Flow Diagram) and explain the problem in this method
- Our objective is to *optimise* these initial test cases for complete path coverage

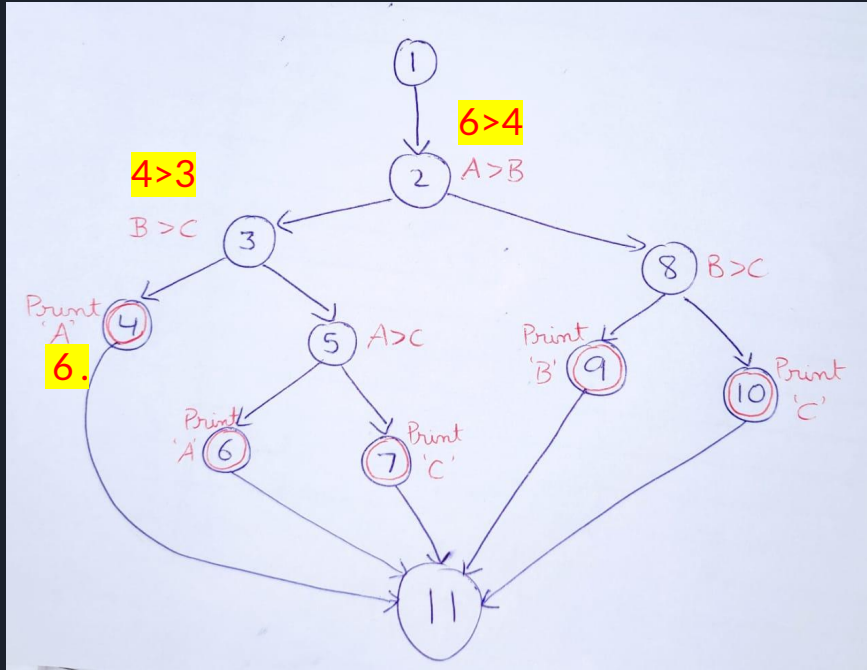
# DFD for 'Greatest Among 3 numbers'



## Five Paths

1. 1-2-3-4-11
2. 1-2-3-5-6-11
3. 1-2-3-5-7-11
4. 1-2-8-9-11
5. 1-2-3-10-11

# Dry Run Test Cases



## Paths

- 1-2-3-4-11
- 1-2-3-5-6-11
- 1-2-3-5-7-11
- 1-2-8-9-11
- 1-2-3-10-11

	A	B	C	Path
1.	0	6	1	4
2.	6	4	3	1
3.	5	1	4	2
4.	10	5	6	2
5.	6	2	1	1



# Problem

- As our program had 5 paths, we took 5 automatically generated test cases
- But these test cases cover only 3 out of 5 paths i.e

$3/5 = 60\%$  path coverage

- A perfect set of test cases would be the one which has 100% path coverage
- We need to find a solution to *Optimize these test cases* to have complete path coverage
- Solution: *GENETIC ALGORITHM !*



# Genetic Algorithm

- Given by John Holland and David Goldberg
- Inspired by Charles Darwin's Theory of natural evolution
- This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.



# Phases of Genetic Algorithm

1. Initial population
2. Fitness function
3. Selection
4. Crossover
5. Mutation

We will explain these phases by continuing with our example



# Initial Population

	A	B	C	Path
1.	0	6	1	4
2.	6	4	3	1
3.	5	1	4	2
4.	10	5	6	2
5.	6	2	1	1

This set of test cases is our initial population. Each input variable is converted in 8 bits binary number and applied crossover operation and mutation operation according the proposed method.





# Fitness Function

In this example, we take path coverage as fitness function. If the path coverage is more than 95% then stop the process. If the path coverage is less than 95% but more than 75%, then apply mutation operation else apply crossover operation.

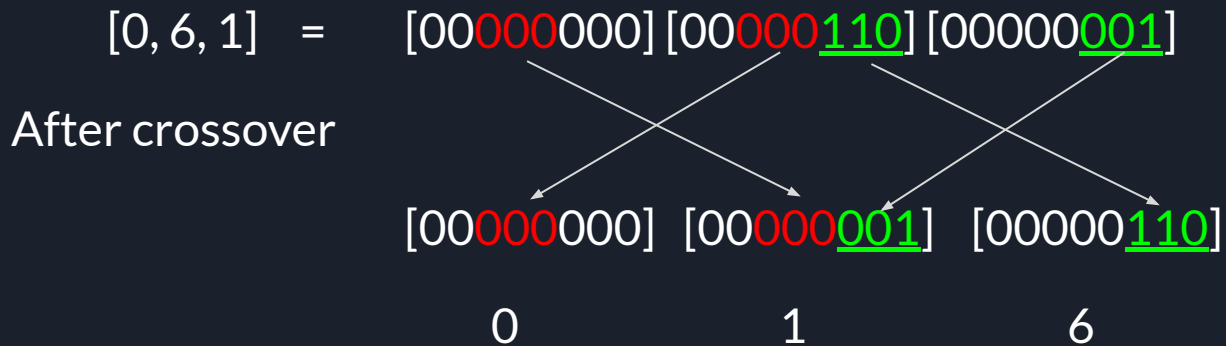
	A	B	C	Path
1.	0	6	1	4
2.	6	4	3	1
3.	5	1	4	2
4.	10	5	6	2
5.	6	2	1	1

$$\text{Path Coverage} = 3/5 = 60\%$$

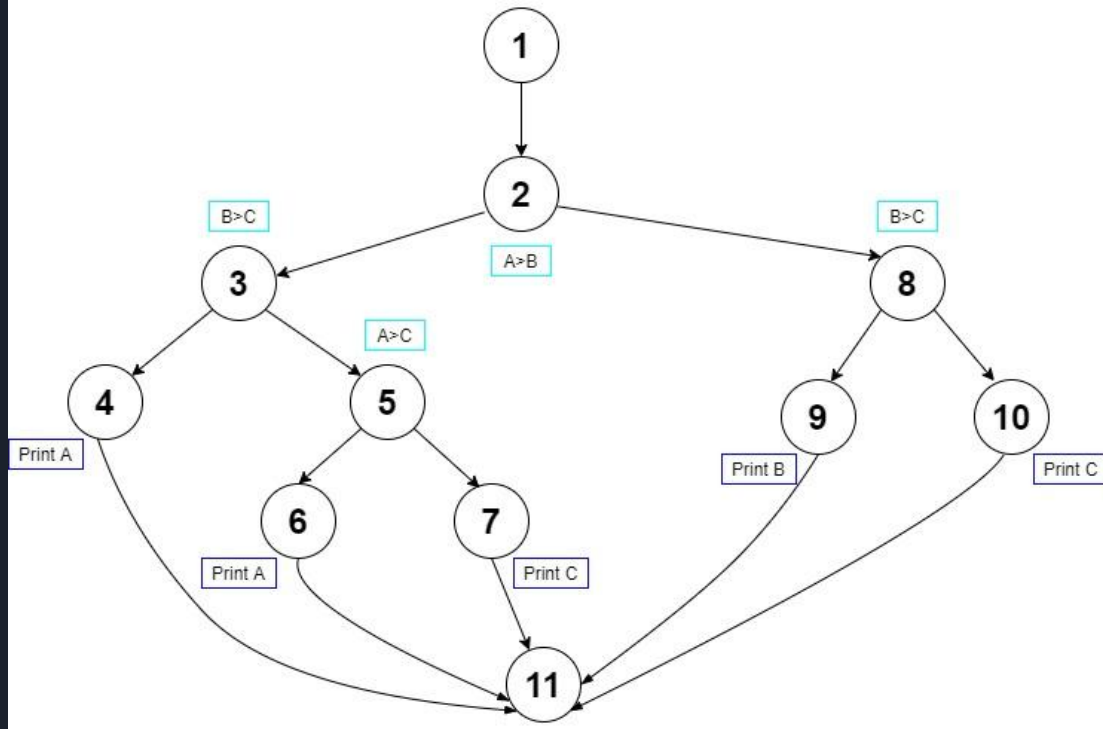
# Crossover

In Crossover operation three bits of variable 'a' are exchanged with 3 bits of variable 'b' and three bits of variable 'b' exchanged with three bits of variable 'c'.

Consider first test case [0,6,1]



# After Crossover



Path

[0,6,1] 1-2-8-9-11

[0,1,6] 1-2-8-10-11

## After applying Crossover to all test cases

A	B	C	Path
0	6	1	4
6	4	3	1
5	1	4	2
10	5	6	2
6	2	1	1

Initially



A	B	C	Path
0	1	6	5
6	3	4	2
5	4	1	1
2	12	5	4
6	1	2	2

Now

## After applying Crossover

A	B	C	Path
0	1	6	5
6	3	4	2
5	4	1	1
2	12	5	4
6	1	2	2

### Path Coverage

5,2,1,4 out of 1,2,3,4,5  
 $4/5 = 80\%$

In one iteration we went from 60% path coverage to 80% path coverage.

Now as Path Coverage > 75% , we will apply mutation operation.

# Mutation

A	B	C	Path
0	1	6	5
6	3	4	2
5	4	1	1
2	12	5	4
6	1	2	2

In mutation operation, any one random bit of any one variable is flipped.

Consider the test case [6,3,4]

[6, 3, 4] = [00000110] [00000011] [00000100]

After mutation


[00000110] [00000011] [00001100]

6

3

12

## After Mutation



A	B	C	Path
0	1	6	5
6	3	12	3
5	4	1	1
2	12	5	4
6	1	2	2

Path Coverage  
5,3,1,4,2 out of 1,2,3,4,5  
 $5/5 = 100\%$

Hence, we have achieved 100% path coverage by *Optimizing* our initial test cases.

Therefore we can use *Genetic Algorithm* for optimising test cases.



## Conclusion of '*Test Case Optimization using Genetic Algorithm*'

In this project, we analyzed how GAs helped in software testing. The results show how software testing using Genetic Algorithms can be used to achieve 100% path coverage. A fitness function has a key role for GA, in the presented method the fitness function takes path coverage as fitness function. Using crossover and mutation operations on initial population we went from 60% path coverage to 100% path coverage. Thus, to increase the efficiency of Software testing, Genetic Algorithms are being used to optimise test cases.