one of the major techniques in dynamic testing

streamstech

What is black-box testing?

→ a s/w testing method in which the functionalities of s/w applications are tested without having knowledge of internal code structure, implementation details and internal paths.

→ mainly focuses on input and output of s/w applications

→ is entirely based on s/w requirements and specification

→ aka functional testing

Aims to:

1) test modules independently.
2) Detect incorrect or missing functions.
3) Identify interface errors.
4) Assess system behavior and performance
5) Test the system under maximum load and stress
6) Ensure the s/w meets user/customer acceptance within defined acceptable limits

What are the types of errors detected by black-box testing?

1) incorrect or missing function
2) interface errors.
3) errors in data structures or external database access
4) Behavior or performance errors
5) Initialization and termination errors.

Boundary value Analysis (BVA)

→ a technique that uncovers the bugs at the boundary of input values.

→ input values near boundary have higher chances of error.

* Boundary value checking
* Robustness testing method
* Worst case testing method

Boundary value checking: BVC

→ In this method, the test cases are designed by holding one variable at its extreme value and other variables at the nominal values in the input domain

between range

Extreme value can be selected at

1) Min
2) value just above the min (min^+)
3) max
4) value just below the max (max^-)

* If we take 2 variables, the combination will be: (comb for test case)

1) Anom, Bnom
2) Amax, Bnom
3) Anom Bmax
4) $Amax^-$, Bnom
5) Anom, $Bmax^-$
6) Amin, Bnom
7) Anom, Bmin
8) $Amin^+$, Bnom
9) Anom, $Bmin^+$

* For n variable → $4n + 1$ test cases

Robustness testing method:

→ extended BVC

Here are 2 more extreme values:

1) value just above max (max^+)
2) value just below min (min^-)

4 more test cases added:

prev 9

10) Amax+, Bnom
11) Amin−, Bnom
11) Bmax+, Anom
12) Bmin−, Anom

* n variable → 6n+1 test cases

Wonst case testing method:

* In BVC, just 1 variable in the boundary, we assumed
* Here, more than 1 variable can be on the boundary

but not take max+, min−

total test cases:

* 1 variable in boundary
* 2 " " "

Test cases:

1) Anom, Bnom 2) . . . . 9) Anom, Bmin+

same as BVC

10) Amax, Bmax+ −
11) Amax, Bmax
12) Amax, Bmin
13) Amax, Bmin+
14) Amin, Bmin
15) Amin+, Bmin+
16) Amax−, Bmax−
17) Amin, Bmax−
18) Bmax, Amax−
19) Bmax, Amin
10) Bmax, Amin+
21) Bmin, Amax−
22) Amin, Bmin+
23) Bmin, Amin+
24 Amin+, Bmax−
25) Amax−, Bmin+

n variable → 5^n test case

Ex: 4.1) prognam detenmines if an intoyen is a prime numben
nange : [1, 100]

Now, design test cases fon BVC, nobust and wonst case testing method.

a) Test cases using BVC:

n = 1 [one voniable]
test case : 4.1 + 1 = 5

Here, min = 1, min+ = 2

max = 100, max− = 99

nom = 50 to 55

Design of test cases:

1 → not prime

2 → prime

100 → not prime

99 → not prime

53 → prime

b) Test cases using robust:

n = 1, test case : 6.1 + 1 = 7

min = 1 → not prime

max = 100 → np

min+ = 2 → p

max− = 99 → np

min− = 0 → invalid

max+ = 101 → "

nom = 53 → p

c) n=1 ; test = $5^1 = 5$ (same as BVC)

worst case

4.2) a^b → function ; a [1, 10]
b [1, 5]

test cases using BVC, robust, worst case.

Soln:

a)

BVC

	a	b
max	10	5
max −	9	4
min	1	1
min+	2	2
nom	5	3

n = 2
test case
= 4.2 + 1
= 9

output using test cases:

a	b	output
10	3	1000
~~10~~	~~4~~	.
9	3	729
1	3	1
2	3	8
5	3	125

a	b	output
5	5	3125
5	4	625
5	1	5
5	2	25

b) Robust.:

$n = 2$

$6 \cdot 2 + 1 = 13$

test case:

	a	b
max	10	5
min	1	1
max-	9	4
min+	2	2
nom	5	3
max+	11	6
min-	0	0

test case:

a	b	o/p
10	3	1000
1	3	1
9	3	729
2	3	8
5	3	125
11	3	invalid
0	3	invalid

a	b	o/p
5	5	3125
5	1	5
5	4	625
5	2	25
5	6	invalid
5	0	invalid

e) worst-case: $n = 2$

test case: $5^2 = 25$

test case:

a	b	o/p
10	5	100000
10	4	10000
10	1	10
10	2	100
10	3	1000
9	5	59049
9	4	6561
9	1	9
9	2	81
9	3	729
1	5	1

a	b	o/p
1	4	1
1	1	1
1	2	1
2	5	32
2	4	16
2	1	2
2	2	4
5	5	3125
5	4	625
5	1	5
5	2	25
5	3	125