ECE 322

SOFTWARE TESTING AND MAINTENANCE

Fall 2025

Assignment #5

Due date: Monday, November 17, 2025 by 3:00 PM

Total: 40 points

---

10 points

1. Using the modified condition/branch coverage criterion, propose test cases for the following expression

$$(a||!b)\&\&(c||(!d\&\&a))$$

Note that the test set could not be unique; show selected possibilities.

Solution

The table below displays all combinations of variables along with the values of the logic expression (z).

	a	*b*	*c*	*d*	*z*
1	F	F	F	F	F
2	F	F	F	T	F
3	F	F	T	F	T
4	F	F	T	T	T
5	F	T	F	F	F
6	F	T	F	T	F
7	F	T	T	F	F
8	F	T	T	T	F
9	T	F	F	F	T
10	T	F	F	T	F
11	T	F	T	F	T
12	T	F	T	T	T
13	T	T	F	F	T
14	T	T	F	T	F
15	T	T	T	F	T
16	T	T	T	T	T

All candidate test cases can be listed as follows:
For a:
- [#1: input(F F F F), output(F), #9: input(T F F F), output(T)]
- [#5: input(F T F F), output(F), #13: input(T T F F), output(T)]
- [#7: input(F T T F), output(F), #15: input(T T T F), output(T)]
- [#8: input(F T T T), output(F), #16: input(T T T T), output(T)]
For b:
- [#3: input(F F T F), output(T), #7: input(F T T F), output(F)]
- [#4: input(F F T T), output(T), #8: input(F T T T), output(F)]
For c:
- [#1: input(F F F F), output(F), #3: input(F F T F), output(T)]
- [#2: input(F F F T), output(F), #4: input(F F T T), output(T)]
- [#10: input(T F F T), output(F), #12: input(T F T T), output(T)]
- [#14: input(T T F T), output(F), #16: input(T T T T), output(T)]
For d:
- [#9: input(T F F F), output(T), #10: input(T F F T), output(F)]
- [#13: input(T T F F), output(T), #14: input(T T F T), output(F)]

5 points

2. Through symbolic execution, obtained are the following path constraints

PC1: $X>Y$ && $X^2<Z$ && $Z>0$, $Z<W$

PC2: $W<2*Z+1$ && $Z>3$

PC3: $X>Y$ && $X<Z$ && $Z-X>2$

PC4: $X<Z$ && $Z<-2$ && $X>0$

Propose test cases.

Solution

PC1: $X=2$, $Y=1$, $Z=5$, $W=6$

PC2: $W=7$, $Z=4$

PC3: $X=2$, $Y=1$, $Z=5$

PC4: not feasible

5 points

3.(i) As black box testing tests all functionalities of a system (identified in specifications), why do we need white box testing? Offer some compelling arguments.

(ii)can you achieve 100% statement coverage criterion. If not, explain why. Show an illustrative example (python code).

(iii)is symbolic execution dynamic or static testing process.

Solution

(i) White box testing allows to quantify the quality of software; for instance cyclomatic complexity categorizes the complexity of code and impacts software maintenance.
(ii) Sometimes the 100% coverage cannot be achieved; for instance, some parts of code are not reachable.
(iii) It is static.

10 points

4. The vector representations of the following paths formed over edges a-b-c-d-e-f-g-h-i-j-k-l-m are shown below

1 1 1 0 1 1 0 0 0 0 0 0 0
1 1 2 1 2 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 1 1 0
0 0 0 0 0 1 1 1 0 0 0 0 1
0 0 0 0 0 0 1 1 0 1 0 1 0

Are these paths linearly independent?

Solution

Checking linear independence requires computing the rank of the matrix whose rows are the vector representations of the paths. We have

```
T=np.array([[1,1,1,0,1,1,0,0,0,0,0,0,0],
            [1,1,2,1,2,1,0,0,0,0,0,0,0],
            [0,0,0,0,0,0,1,0,1,0,1,1,0],
            [0,0,0,0,0,1,1,1,0,0,0,0,1],
            [0,0,0,0,0,0,1,1,0,1,0,1,0]])
```

The function matrix_rank(T) (`from numpy.linalg import matrix_rank`) returns rank(T)=5, so the paths are linearly independent.

10 points

5. Construct a control flow graph of the following code and determine its cyclomatic complexity.

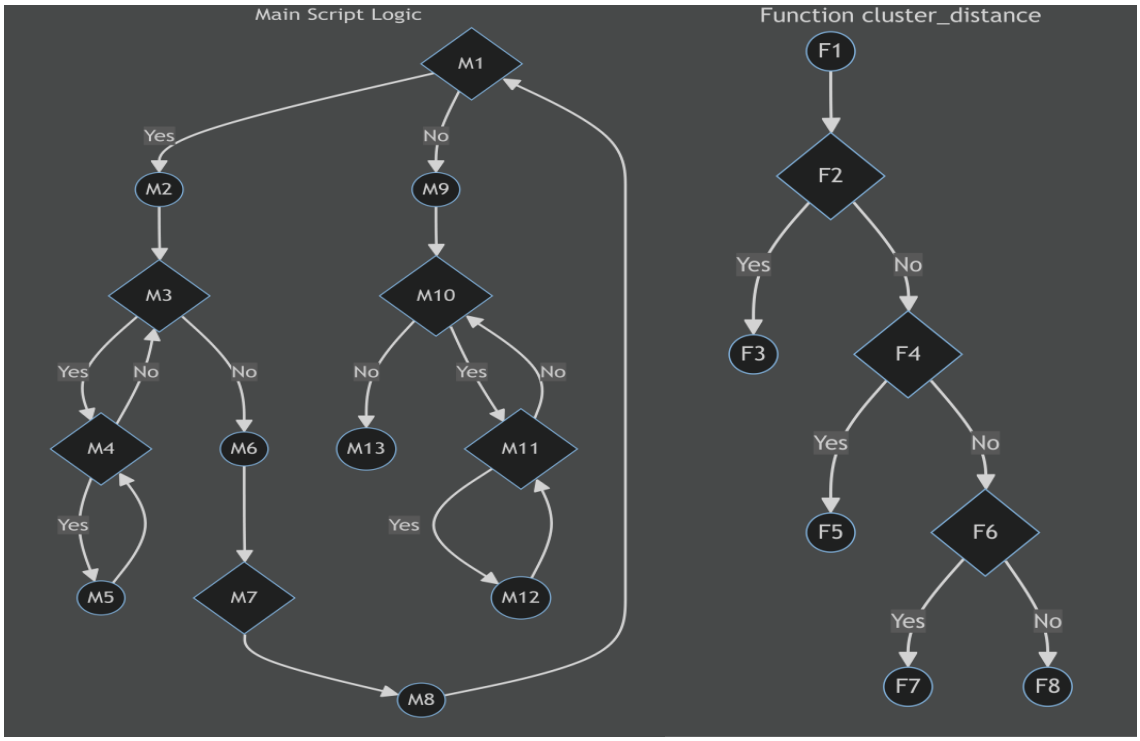
```
def cluster_distance(c1, c2):
    """Compute inter-cluster distance depending on linkage type."""
    dists = cdist(X[c1], X[c2])
    if linkage == 'single':
        return np.min(dists)
    elif linkage == 'complete':
        return np.max(dists)
    elif linkage == 'average':
        return np.mean(dists)
    else:
        raise ValueError("linkage must be 'single', 'complete', or 'average'")

while len(clusters) > n_clusters:
    # Compute all pairwise distances
    distances = np.full((len(clusters), len(clusters)), np.inf)
    for i in range(len(clusters)):
        for j in range(i + 1, len(clusters)):
            distances[i, j] = cluster_distance(clusters[i], clusters[j])

    # Find two clusters with minimum distance
    i, j = np.unravel_index(np.argmin(distances), distances.shape)

    # Merge them
    new_cluster = clusters[i] + clusters[j]
    clusters = [c for k, c in enumerate(clusters) if k not in (i, j)]
    clusters.append(new_cluster)

# Assign labels
labels = np.zeros(len(X), dtype=int)
for idx, cluster in enumerate(clusters):
    for sample_index in cluster:
        labels[sample_index] = idx

return labels
```

Solution:

Block No.	Code Block	D
F1	def cluster_distance(c1, c2): dists = cdist(X[c1], X[c2])	
F2	if linkage == 'single':	1
F3	return np.min(dists)	
F4	elif linkage == 'complete':	1
F5	return np.max(dists)	
F6	elif linkage == 'average':	1
F7	return np.mean(dists)	

F8	else: raise ValueError(...)	
M1	while len(clusters) > n_clusters:	1
M2	distances = np.full((len(clusters), len(clusters)), np.inf)	
M3	for i in range(len(clusters)):	1
M4	for j in range(i + 1, len(clusters)):	1
M5	distances[i, j] = cluster_distance(clusters[i], clusters[j])	
M6	i, j = np.unravel_index(np.argmin(distances), distances.shape) new_cluster = clusters[i] + clusters[j]	
M7	clusters = [c **for** k, c in enumerate(clusters) **if** k not in (i, j)]	2
M8	clusters.append(new_cluster)	
M9	labels = np.zeros(len(X), dtype=int)	
M10	for idx, cluster in enumerate(clusters):	1
M11	for sample_index in cluster:	1
M12	labels[sample_index] = idx	
M13	return labels	

The control diagram can be depicted as:



In the diagram,

- For function “cluster_distance” part, there are three logic branches, as the result, the cyclomatic complexity is V(G1) = 3 + 1 = 4
- For the “main_logic” part, there are six loops and an “if” logic expression, the cyclomatic complexity is V(G2) = 7 + 1 = 8