Software Engineering : Lab-9

Name – Patel Aryan

Id – 202201511

Que-1 Convert the code comprising the beginning of the doGraham method into a control flow graph (CFG). You are free to write the code in any programming language.

- ➔ Python code for CFG :

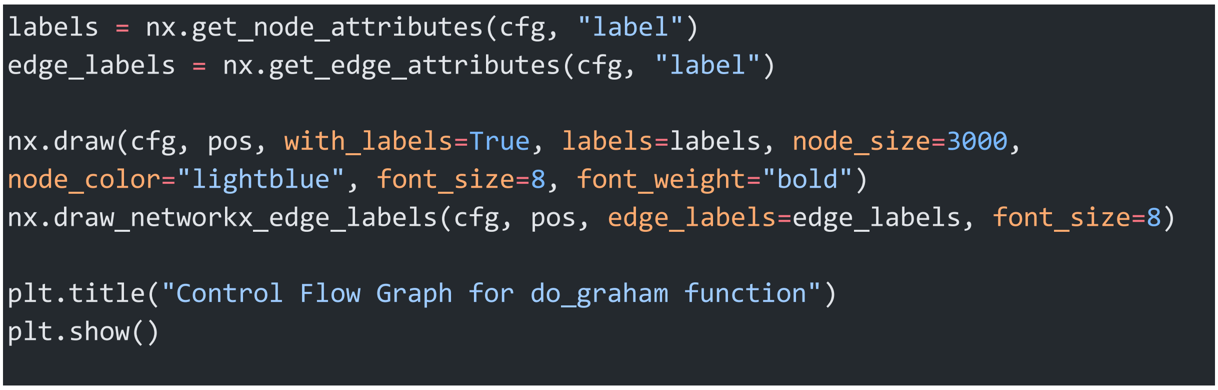
```python
import networkx as nx
import matplotlib.pyplot as plt

# Create a directed graph
cfg = nx.DiGraph()

# Define nodes for each code segment
cfg.add_node("Start", label="Start of do_graham function")
cfg.add_node("Initialize min_index", label="min_index = 0")
cfg.add_node("Loop start", label="for i in range(1, n):")
cfg.add_node("Check condition", label="if points[i].y < points[min_index].y or (points[i].y == points[min_index].y and points[i].x > points[min_index].x):")
cfg.add_node("Update min_index", label="min_index = i")
cfg.add_node("Loop end", label="End of for loop")
cfg.add_node("Swap points", label="points[0], points[min_index] = points[min_index], points[0]")
cfg.add_node("Return", label="return points")

# Define edges between nodes to represent control flow
cfg.add_edge("Start", "Initialize min_index")
cfg.add_edge("Initialize min_index", "Loop start")
cfg.add_edge("Loop start", "Check condition")
cfg.add_edge("Check condition", "Update min_index", label="True")
cfg.add_edge("Check condition", "Loop end", label="False")
cfg.add_edge("Update min_index", "Loop start")
cfg.add_edge("Loop end", "Swap points")
cfg.add_edge("Swap points", "Return")

# Draw the control flow graph
pos = nx.spring_layout(cfg)
```

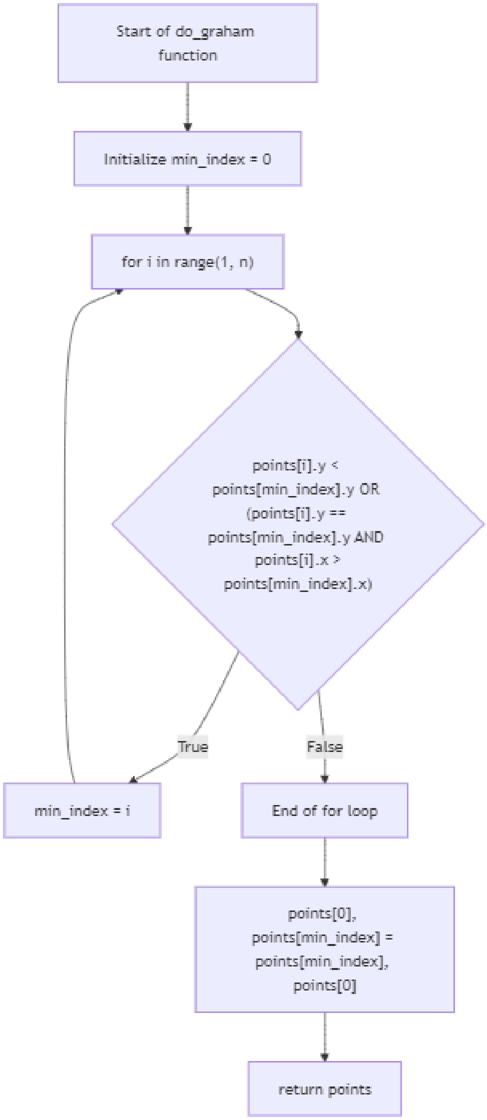


```
labels = nx.get_node_attributes(cfg, "label")
edge_labels = nx.get_edge_attributes(cfg, "label")

nx.draw(cfg, pos, with_labels=True, labels=labels, node_size=3000,
node_color="lightblue", font_size=8, font_weight="bold")
nx.draw_networkx_edge_labels(cfg, pos, edge_labels=edge_labels, font_size=8)

plt.title("Control Flow Graph for do_graham function")
plt.show()
```

Generated Control Flow Graph :



Start of do_graham function
Initialize min_index = 0
for i in range(1, n)
points[i].y < points[min_index].y OR (points[i].y == points[min_index].y AND points[i].x > points[min_index].x)
True
False
min_index = i
End of for loop
points[0], points[min_index] = points[min_index], points[0]
return points

Que-2 Construct test sets for your flow graph that are adequate for the following criteria: Statement Coverage, Branch Coverage, Basic Condition Coverage.

➔

Test Cases	**Input**	**Coverage**
TC1	[(0, 0), (1, 1), (2, 2)]	Statement
TC2	[(1, 1), (2, 0), (3, 3)]	Branch
TC3	[(0, 0), (1, 0), (2, 2)]	Branch
TC4	[(0, 0), (1, 2), (2, 0)]	Basic condition
TC5	[(0, 0), (1, 0), (2, 1)]	Basic condition
TC6	[(1, 1), (0, 0), (-1, -1), (2, 2)]	Tests negative coordinates
TC7	[(0, 0), (0, 0), (1, 1), (2, 2)]	Tests handling of duplicate points
TC8	[(0, 0), (2, 2), (1, 1), (3, 0)]	Tests y comparison with varied x values
TC9	[(2, 2), (3, 2), (1, 3), (4, 3)]	Branch coverage for same y, different x
TC10	[(5, 5), (3, 5), (7, 5), (6, 5)]	Tests condition with same y values