Shri Ramdeobaba College of Engineering and Management, Nagpur
Department of Computer Science and Engineering
Session: 2021-2022 [EVEN SEM]

Compiler Design Lab

Name : Janak Mandavgade

Sec : A

Roll no. : 43

Batch : A3

Subject : Compiler Design

PRACTICAL No. 6

Aim: Write a program to perform loop detection by finding leader, basic blocks and program flow graph & natural loop.

Code :

```
def find_leaders(statements):
    leaders = set()
    leaders.add(1)

    for i, statement in enumerate(statements):
        if "GOTO" in statement:
            target = int(statement.split()[-1])
            leaders.add(target)
            if i + 2 <= len(statements):
                leaders.add(i + 2)

    return leaders
```

```
def create_basic_blocks(statements, leaders):
    basic_blocks = {}
    current_block = None

    for i, statement in enumerate(statements, start=1):
        if i in leaders:
            current_block = i
            basic_blocks[current_block] = []

        basic_blocks[current_block].append(statement)

    return basic_blocks



def program_flow_graph(statements, basic_blocks):
    edges = set()

    for i, statement in enumerate(statements):
        if "GOTO" in statement:
            source = [k for k, v in basic_blocks.items() if statement in v][0]
            target = int(statement.split()[-1])
            edges.add((source, target))

            if i + 2 <= len(statements):
                edges.add((source, i + 2))
```

```
    return edges



def dominators(basic_blocks, pfg):
    dominators = {}

    for block in basic_blocks:
        if block == 1:
            dominators[block] = set()
        else:
            dominators[block] = set(basic_blocks.keys())

    while True:
        updated_dominators = dominators.copy()
        for block in basic_blocks:
            if block != 1:
                preds = {pred for pred, succ in pfg if succ == block}
                if preds:
                    updated_dominators[block] = {block} | set.intersection(*[dominators[pred] for pred
in preds])

        if dominators == updated_dominators:
            break
        else:
```

```
            dominators = updated_dominators

    return dominators



def natural_loop(pfg):
    loops = set()

    for source, target in pfg:
        if target < source:
            loops.add((target, source))

    return loops



statements = [
    "count = 0",
    "Result = 0",
    "If count > 20 GOTO 8",
    "count = count + 1",
    "increment = 2 * count",
    "result = result + increment",
    "GOTO 3",
    "end"
]
```

```
leaders = find_leaders(statements)
basic_blocks = create_basic_blocks(statements, leaders)
pfg = program_flow_graph(statements, basic_blocks)
dominators_data = dominators(basic_blocks, pfg)
loops = natural_loop(pfg)

print("Leader statements:", leaders)
print("Basic blocks:", basic_blocks)
print("Program Flow Graph:", pfg)
print("Dominators of all basic blocks:", dominators_data)
print("Natural loop:", loops)
```

Input :

```
1. count = 0
2. Result = 0
3. If count > 20 GOTO 8
4. count=count + 1
5. increment = 2 * count
6. result = result +increment
7. GOTO 3
8. end
```

Output:

```
~/Prac-678-CD$ python Prac6.py
Leader statements: {8, 1, 3, 4}
Basic blocks: {1: ['count = 0', 'Result = 0'], 3: ['If count > 20 GOTO 8'], 4: ['count = count + 1', 'increment = 2
* count', 'result = result + increment', 'GOTO 3'], 8: ['end']}
Program Flow Graph: {(3, 8), (3, 4), (4, 8), (4, 3)}
Dominators of all basic blocks: {1: set(), 3: {8, 1, 3, 4}, 4: {8, 1, 3, 4}, 8: {8, 1, 3, 4}}
Natural loop: {(3, 4)}
```