Shri Ramdeobaba College of Engineering and Management, Nagpur
Department of Computer Science and Engineering
Session: 2022-2023

Compiler Design Lab

Name : Shantanu Mane

Roll No. : 63

PRACTICAL No. 6

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

Input: Three address code statements.

Output:
1) Leader Statements
2) Basic blocks
3) Program flow graph indicating the successor & predecessor.
4) Dominators of all the basic blocks
5) Natural loop detection

Sample input: 3AC

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

Sample Output: The leader statements are:
1) count=0
3) If count > 20 GOTO 8
4) count=count + 1
8) end

The Basic blocks are:
B1: contains: 1 & 2
B2 : contains 3
B3 : contains 4 5 6 7
B4 : contains 8

The PFG is
B1->B2
B2->B3
B2->B4
B3->B2

The dominators of all basic block are:

The natural Loop is:

In [ ]:
```
TAC = {"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"}
```

In [ ]:
```
TAC
```

Out[ ]:
```
{'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'}
```

In [ ]:
```
# 1ST, 3RD, 4TH, 8TH
LEADER_STMT = []
blockList = []
for k,v in TAC.items():
    if LEADER_STMT == []:
        LEADER_STMT.append((v,1))
        blockList.append(1);
    if v.__contains__('GOTO'):
        LEADER_STMT.append((TAC[v[-1]], int(v[-1])))
        blockList.append(int(v[-1]))
    if v.__contains__('if'):
#         print(int(k)+1)
        LEADER_STMT.append((TAC[str(int(k)+1)], int(k)+1))
        blockList.append(int(k) +1)
LEADER_STMT.sort(key = lambda x: x[1])
```

In [ ]:
```
LEADER_STMT
```

Out[ ]:
```
[('count=0', 1),
 ('if count > 20 GOTO 8', 3),
 ('count=count + 1', 4),
 ('end', 8)]
```

In [ ]:
```
blockList = sorted(blockList)
blockList
```

Out[ ]:
```
[1, 3, 4, 8]
```

In [ ]:
```
blocks = {}
index = 1
for i in blockList:
    firstIndex = blockList.index(i)
    if firstIndex != len(blockList)-1:
        secondIndex = firstIndex+1
    else:
        secondIndex = firstIndex
    if firstIndex == blockList[-1] and firstIndex == secondIndex:
        blocks[f'B{index}'] = firstIndex
        index+=1
        break
    else:
        blocks[f'B{index}'] = (blockList[firstIndex], blockList[secondIndex]-1)
        index+=1
#     print(blockList[firstIndex], blockList[secondIndex]-1)
for k,v in blocks.items():
#     print(v)
    if v[0] == v[1]: # (3,3)
        blocks[k] = (v[0])
    if v[0] > v[1]: # (8,7)
        blocks[k] = (v[0])
```

In [ ]:
```
blocks
```

Out[ ]:
```
{'B1': (1, 2), 'B2': 3, 'B3': (4, 7), 'B4': 8}
```

In [ ]:
```
LEADER_STMT
```

Out[ ]:
```
[('count=0', 1),
 ('if count > 20 GOTO 8', 3),
 ('count=count + 1', 4),
 ('end', 8)]
```

In [ ]:
```
TAC
```

Out[ ]:
```
{'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'}
```

In [ ]:
```
PFG = []
for k,v in TAC.items():
    if v.__contains__("if"):
        # 1 - > 2
        for key,val in blocks.items():
            if type(val) != int:
                if int(k)-1 in val or int(k) in val:
                    first = key
            if int(k) == val or int(k)-1 == val:
                second = key
        PFG.append((first, second))
        # 2 -> 3
        for key,val in blocks.items():
            if type(val) != int:
                if int(k)+1 in val or int(k) in val:
                    first = key
            if int(k) == val or int(k)+1 == val:
                second = key
        PFG.append((second, first))
    if v.__contains__("GOTO"):
        nextstmt = v.split("GOTO ")[-1]
d        for key,val in blocks.items():
            if type(val) != int:
                if int(k) in val or int(nextstmt) in val:
                    first = key
            if int(k) == val or int(nextstmt) == val:
                second = key
        print(first, second)
PFG
```

```
B3 B4
B3 B2
```

```
Out[ ]: [('B1', 'B2'), ('B2', 'B3')]
```

```
In [ ]: # B1 -> B2
        # B2 -> B3
        # B2 -> B4
        # B3 -> B2
        PFG = []

        for k,v in TAC.items():
        #     print(k,v)
            if v.startswith("if"):
                print(int(k)-1, int(k))
                nextBlock = int(k)+1
                print(int(k), nextBlock)
                print(blocks)
                for key,val in blocks.items():
                    if type(val) != int:
                        if int(k)-1 in val or int(k) in val:
                            first = key
                    if int(k) == val or int(k)-1 == val:
                        second = key
                PFG.append((first, second))
```

```
2 3
3 4
{'B1': (1, 2), 'B2': 3, 'B3': (4, 7), 'B4': 8}
```

```
In [ ]: PFG
```

```
Out[ ]: [('B1', 'B2')]
```

```
In [ ]:
```