Software Engineering

Assignment 4: Testing

3 Ba INF 2018-2019

Evelien Daems
Lars Van Roy

November 18, 2018

Control Flow Graph

Een control flow graph is een visuele weergave van een programma waarmee alle mogelijke paden doorheen een programma kunnen gegenereerd worden. Elke binaire node (een node waar twee pijlen uitkomen) komt overheen met een conditie in het programma. Elke mogelijke sequentie van nodes dat eindigt in een eindstaat is een mogelijk pad door het programma, het is echter niet gegarandeerd dat dit pad ook bestaat. Sommige condities kunnen enkel slagen indien andere condities ook slagen en of slagen sowieso x aantal keren. Het is dus niet omdat de sequentie bestaat in de control flow graph dat het ook een mogelijk pad is. Voor de flow graph van de calculate() functie zie appendix A.

Cyclomatic Complexity

De cyclomatic complexity kan berekend worden door het aantal edges - het aantal nodes te beschouwen + 2 of het aantal binaire conditie nodes + 1. Beide geven ons in dit geval 10. Deze waarde symboliseert een bovengrens voor het aantal mogelijke onafhankelijke paden door de flowgraph.
aantal edges - aantal nodes + 2 = 29 - 21 + 2 = 10
aantal binarie condities + 1 = 9 + 1 = 10

independent paths

Wanneer we beginnend van het kortst mogelijke pad verdergaan door steeds een nieuw onafhankelijke pad te nemen krijgen we een mogelijkheid om de verschillende paden voor te stellen. Een nieuw pad wordt onafhankelijk genoemd van de voorgaande paden als er in het nieuwe pad een node voorkomt die nog niet voorkwam in de voorgaande paden. Wanneer we dit doen bekomen we de volgende paden.

index	pad
0	1, 2, 6, 7, 21
1	1, 2, 6, 7, 8, 9, 10, 7, 21
2	1, 2, 3, 2, 6, 7, 8, 9, 10, 7, 21
3	1, 2, 3, 4, 5, 2, 6, 7, 21
4	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 10, 7, 21
5	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 12, 10, 7, 21
6	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 12, 10, 7, 21
7	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 12, 10, 7, 21
8	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 12, 10, 7, 21
9	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 12, 10, 12, 10, 7, 10, 12, 10, 7, 21

Table 1: alle mogelijke paden

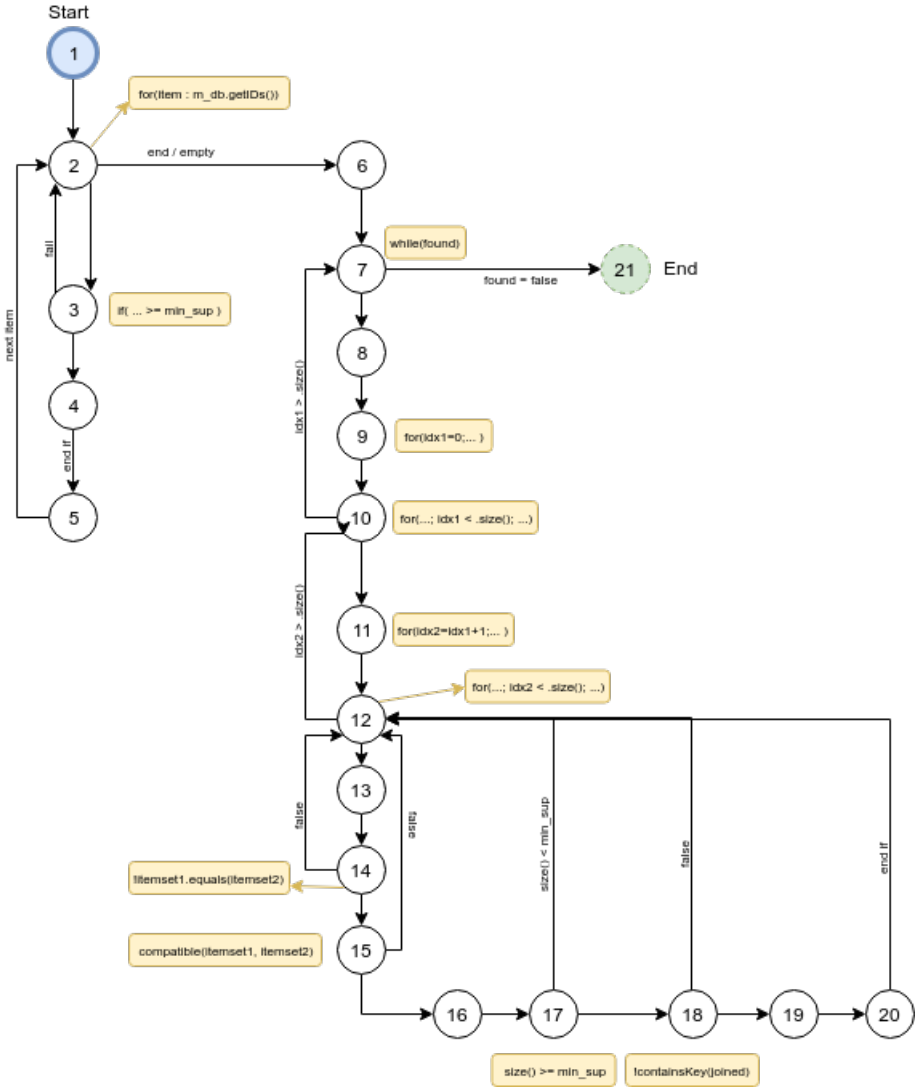
test cases

Bij nadere observatie van deze paden zien we dat het merendeel hiervan niet bereikbaar is. sommige condities kunnen niet falen zonder dat andere in het programma ook falen en omgekeerd. Uiteindelijk blijven volgende 4 paden over met bijhorende input en output. Hierbij zijn triviale inputvelden, of m.a.w. inputvelden waarvoor de waarde niet relevant is om het gewenste pad te bekomen, weggelaten.

index	input	output
1	m_dbṁ_tidlist = {}	{{}, {}}
2	m_dbṁ_tidlist = {0:{0}}; min_sup = 2;	{{}, {}}
4	m_dbṁ_tidlist = {0:{0}}; min_sup = 1;	{{[0]}, {}}
9	m_dbṁ_tidlist = {0:{1,2}, 1:{1}}; min_sup = 1;	{{[0], [1]}, {[0, 1]}, {}}

Table 2: test cases

appendix A


Start
1
for(Item : m_db.getIDs())
2
end / empty
6
fail
3
if(... >= min_sup)
next item
4
end if
5
while(found)
7
found = false
21
End
8
idx1 > .size()
9
for(idx1=0;...)
10
for(...; idx1 < .size(); ...)
idx2 > .size()
11
for(idx2=idx1+1;...)
for(...; idx2 < .size(); ...)
12
13
false
14
itemset1.equals(itemset2)
false
15
compatible(itemset1, itemset2)
size() < min_sup
false
end if
16
17
18
19
20
size() >= min_sup
!containsKey(joined)

appendix B

index	pad	input	output
0	1, 2, 6, 7, 21	bestaat niet	bestaat niet
1	1, 2, 6, 7, 8, 9, 10, 7, 21	m_dbm_tidlist = {}	{{}, {}}
2	1, 2, 3, 2, 6, 7, 8, 9, 10, 7, 21	m_dbm_tidlist = {0:{0}}; min_sup = 2;	{{[0]}, {}}
3	1, 2, 3, 4, 5, 2, 6, 7, 21	bestaat niet	bestaat niet
4	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 10, 7, 21	m_dbm_tidlist = {0:{0}}; min_sup = 1;	{{[0]}, {}}
5	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 12, 10, 7, 21	bestaat niet	bestaat niet
6	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 12, 10, 7, 21	bestaat niet	bestaat niet
7	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 12, 10, 7, 21	bestaat niet	bestaat niet
8	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 12, 10, 7, 21	bestaat niet	bestaat niet
9	1, 2, 3, 4, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 12, 10, 12, 10, 7, 10, 12, 10, 7, 21	bestaat niet	bestaat niet

Table 3: paden + test cases