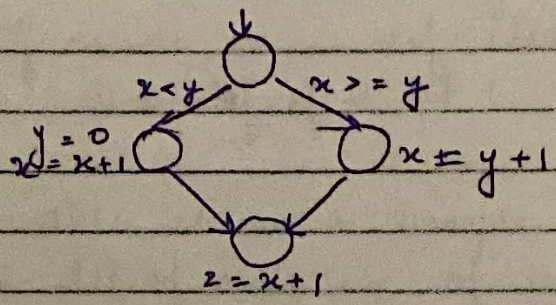
Week-3

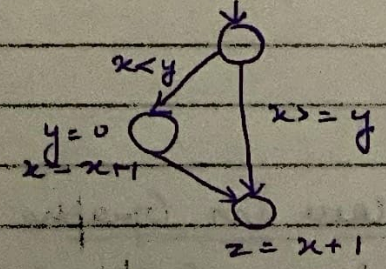
4.1 Graph Coverage Criteria

- model software artifacts as graphs → 3 types of criteria
  1. structural, data flow, call graphs.
- A CFG (control flow graph) models all execution of a method by describing control structures. Nodes → seq. of statements (basic blocks). Edges → transfer of control from 1 statement to the next.

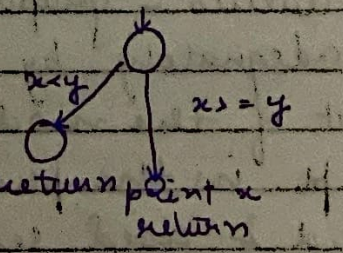
Eg.1
```
if (x<y)
{
  y=0
  x=x+1
}
else
{ x=y+1
}
z=x+1
```



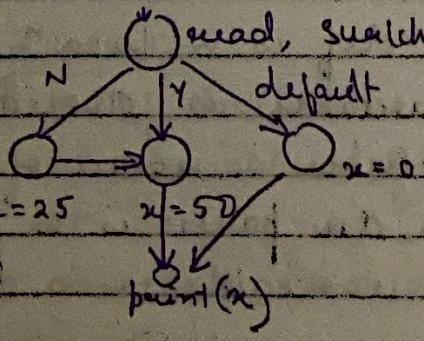
Eg.2
```
if (x<y)
{
  y=0
  x=x+1
}
z=x+1
```



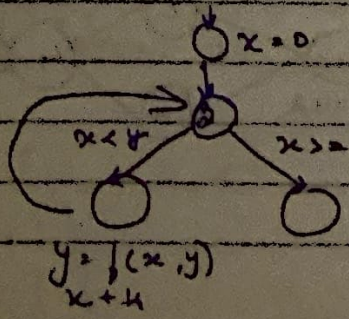
Eg.3
```
if (x<y)
{ return
}
print(x)
return
```



Eg.4
```
read(c)
switch(c)
{ case N   z=25
  case Y   x=50
           break
  default  x=0
           break
}
print(x)
```

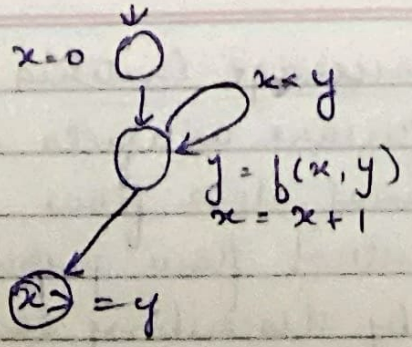


Eg.5
```
x=0
while (x<y)
{
  y=f(x,y)
  x=x+1
}
```



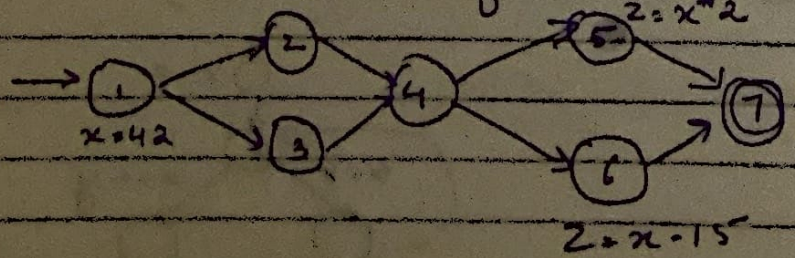
Eg

```
x=0
do
{y = f(x,y)
 x = x+1
}while (x<y)
print (y)
```



Data Flow in Graphs

- Data values – graph models of prog. can be tested adequately by including value of variables as a part of the model. Data values are created at some pt. in the prog. used later. They can be created & used several times.
- A def is a loca^n where a value of the problem variable is stored into memory. A use is a loca^n where a value of the variable is accessed
- A du-pair is a pair of loca^n (l_i, l_j) such that a variable v is defined at l_i and used at l_j.



def(1) = {x}

def(5) = def(6) = {z}

use(5) = {x}

use(6) = {x}

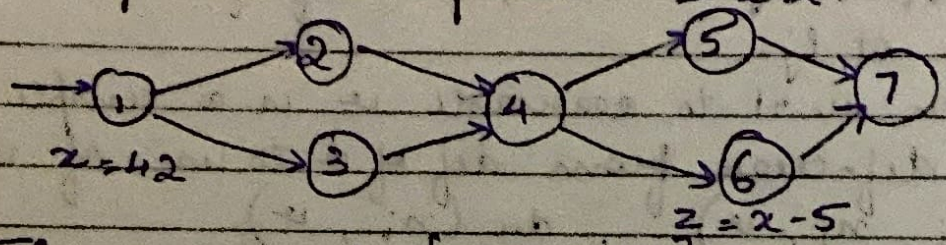
- Pattern matching eg.
- The def of a variable may or may not reach a particular use. A def of a variable v at location l_i will not reach use of v at location l_j if there is no path from l_i to l_j.
- A path from l_i to l_j is def-clear w.r.t. variable v if v is not given another value on any of the node or edges in the path. If there is a def-clear path from l_i to l_j w.r.t. v, the def of v at l_i reaches the use at l_j.
- A du-path w.r.t to variable v is a simple path that is def-clear from def of v to use of v.
  du (n_i, n_j, v) , du (n_i, v)

L3 Data Flow Graph Coverage Criteria

- This criteria is defined as sets of du-paths. Consider all du-paths w.r.t. to a given variable defined in a given node. The def-path set du (n_i, v) is the set of du paths w.r.t variable v that start at node n_i.
- A def-pair set, du (n_i, n_j, v) is the set of du-paths w.r.t to variable v that start at node n_i & end at node n_j.

  du (10, iSub) = {[10, 3, 4] ; [10, 3, 4, 5] .... }
  du (10, 4, iSub) = {[10, 3, 4]}
  du (10, 5, iSub) = {[10, 3, 4, 5]}

- A test path p is said to du tour a sub-path d w.r.t v if p tours d & the portion of p to which d corresponds is def-clear w.r.t. v. We can also allow def-clear side-trips w.r.t to v while touring a du-path.
- TR1 → each def at least reaches 1 use.
  TR2 → each def reaches all possible uses.
  TR3 → each def reaches all possible uses thru all possible du paths.



TR1 → def(x) → [1,2,4,6,7]
TR2 → use(x) → {[1,2,4,5,7], [1,2,4,6,7]}
TR3 → du-paths(x) → {[1,2,4,5,7], [1,2,4,6,7], [1,3,4,5,7], [1,3,4,6,7]}

- Subsumption

all du-paths coverage
↓
all uses coverage
↓
all defs coverage

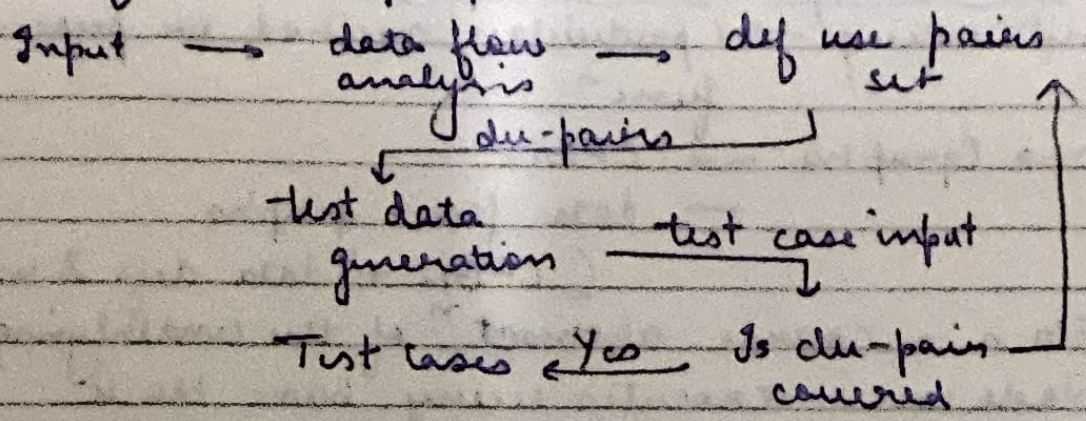
Complete Path
↓
Prime Path

all du-paths → all uses → all def

edg-pair → edges → node

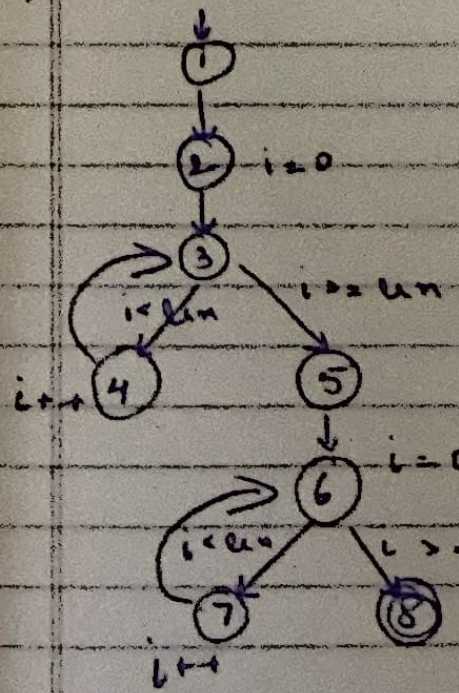
Comp. round trip → single round trip

- Data flow testing - process



L4 Data Flow Testing Example

Eg. Statistics code.



node	defs	use
1	numbers, sum, len	numbers
2	i	
3		
4	sum, i	numbers, i, sum
5	mid, mean, varsum, i	", len, sum
6		
7	varsum, i	varsum, nos, i, mean
8	var, sd	all

var → du pairs → du paths

nos. → (1,4), (1,5), (1,7) → [1,2,3,4], [1,2,3,5], [1,2,3,5,6,7]

then find the unique ones.

↓

Create test case to cover all du-paths.

V5 **Unit Testing Based on Graphs**

- unit testing → particular method or procedure or func^{n}.
  - → Graphs → CFGs.
    - → data flow graphs (CFGs + data defs & uses)
- CFG is a coarse abstract^{n} of the underlying code.
  - Node cov. → executes every basic block
  - Edge cov. → " decisions / transfer of control
  - Prime path cov. → executes loops structurally.
- Data flow graphs provides useful info. about values of variables as the prog. executes.