In flow graphs, loops represent the repetitive execution of a sequence of statements. They are essential for handling iterative tasks and controlling the flow of execution in programs. Loops allow certain code blocks to be repeated multiple times until a specific condition is met.

Here’s an explanation of loops in flow graphs:

1. Flow Graphs: Flow graphs are graphical representations of the control flow of a program. They depict the sequence of instructions and the paths that control can take during program execution. Flow graphs consist of nodes and edges, where nodes represent basic blocks or statements, and edges represent the flow of control between these blocks.
2. Loop Structures: Loops create repetitive control flow structures in flow graphs. They allow a set of instructions to be executed repeatedly until a certain condition is satisfied. Common loop structures include the “while” loop, “for” loop, and “do-while” loop, each with its own syntax and control flow behavior.
3. Loop Header: In flow graphs, loops are typically represented by a loop header node. The loop header node contains the condition or control variable that determines whether the loop will continue or terminate. It serves as the entry point to the loop structure.
4. Loop Body: The loop body contains the statements or basic blocks that are executed repeatedly as long as the loop condition evaluates to true. The loop body is connected to the loop header, and once executed, control flows back to the loop header for evaluation of the loop condition.
5. Loop Exit: The loop exit represents the point where control exits the loop structure when the loop condition becomes false. This exit point is usually connected to the subsequent statements or basic blocks that follow the loop.

6. Back-Edges: In flow graphs, back-edges represent the flow of control from within the loop body back to the loop header. They create a loop in the graph, allowing the loop to repeat until the loop condition is false.
7. Loop Invariants: Loop invariants are statements or conditions that remain true throughout the execution of a loop. They are typically located before or after the loop body and are not affected by the iterations of the loop. Loop invariants play an important role in loop optimization and can help improve the efficiency of loop execution.
8. Loop Control Flow: The control flow within a loop is determined by the loop condition. If the condition evaluates to true, the loop body is executed, and control flows back to the loop header for evaluation again. If the condition evaluates to false, control exits the loop, and execution continues with the statements following the loop.

Example, take source code as shown below:

```
a = 0;
b = b+c;
c = 0;
If (a>d) {
c = b;
b++;
}
else {
c = d;
d++;
}
a= b+d;
```

Basic blocks in above source code are

(1)	(2)	(3)	(4)

a = 0;
b = b+c;
c = 0;
If (a>d)
	c = b;
	b++;
		c = d;
		d++;
			a= b+d;

Flow graph for above Basic blocks