Lecture 17 - Nov 12

Graphs

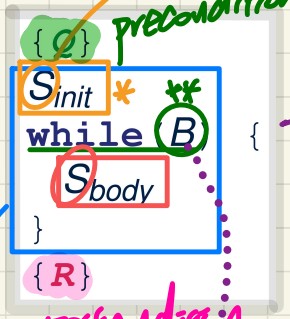
Loop Invariant (LI): Execution Flow
Relating Exit Condition, LI, Postcondition
Dijkstra's Algorithm: LI, Assumption

Announcements/Reminders

- Today's class: notes template posted
- Assignment 2 released → Edge List
  - ① working version of BFS given.
  - ② answer graph questions
- Change of Dates:
  - + Assignment 2 to be due on Wed, Nov 19
  - + Test 2 to be take place on Mon, Nov 24

Correctness of Loops: Syntax

programming statements.




```
void myAlgorithm() {
  assert Q; /* Precondition */
  Sinit
  assert I; /* Is LI established? */
  while ( B ) {
    Sbody
    assert I; /* Is LI preserved? */
  }
  assert R; /* Postcondition */
}
```

establishing LI

Iterative Algorithm

B: stay condition

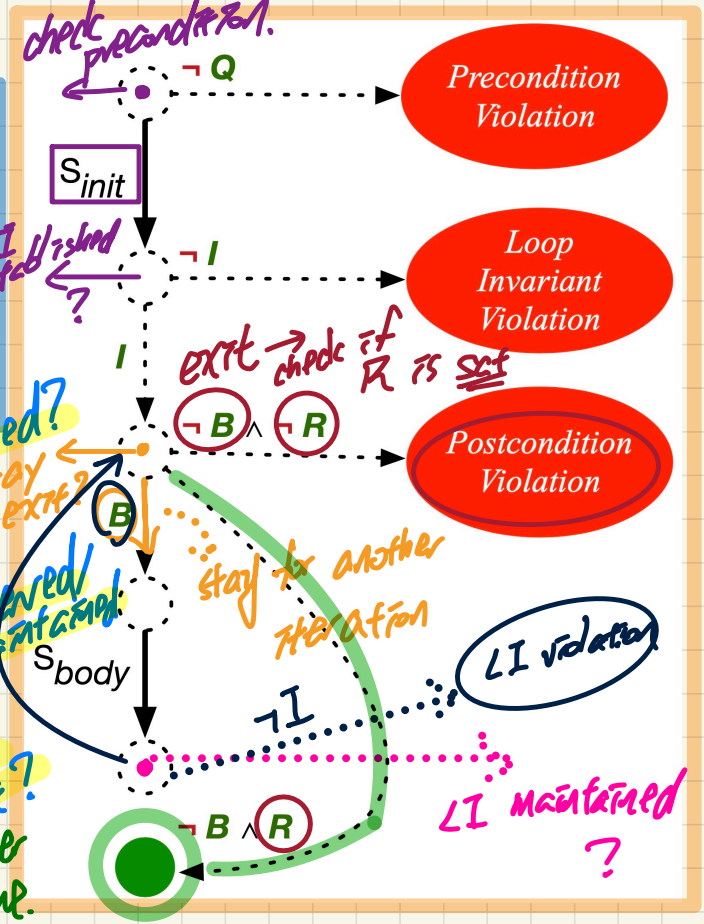
¬B: exit condition.

Is LI established?

Is LI preserved/maintained at the end of current it?

* Initialization/Preparation for iterations.

** As long as B is true, execute Sbody another time.
As soon as B is false, exit from the loop.



alt

```
Sinit
while (B) {
  assert(I);
*
}
```

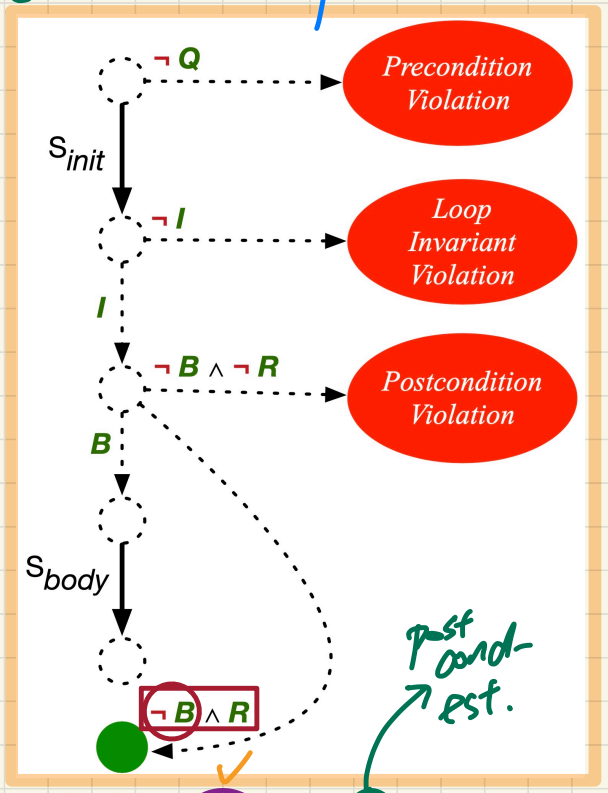
if B is false right after Sinit, we will not even get a chance to check * for establishment

Correctness of Loops: Example

```
1  void testLI() { /* Assume: integer attribute i */
2    assert i == 1; /* Precondition */
3    assert (1 <= i) && (i <= 6); /* Is LI established? */
4    while (i <= 5) {
5      i = i + 1;
6      assert (1 <= i) && (i <= 6); /* Is LI maintained? */
7    }
8    assert i == 6; /* Postcondition */
9  }
```

LI: $1 \leq i \leq 6$

It #	i	LI	B	R
init	1	✓ (est.)	✓	
1	changed to: 2	✓ (maintained)	✓	
2	3	✓	✓	
3	4	✓	✓	
4	5	✓	✓	
5	6	✓	✗ exit	✓



General: $\neg B \wedge LI \Rightarrow R$

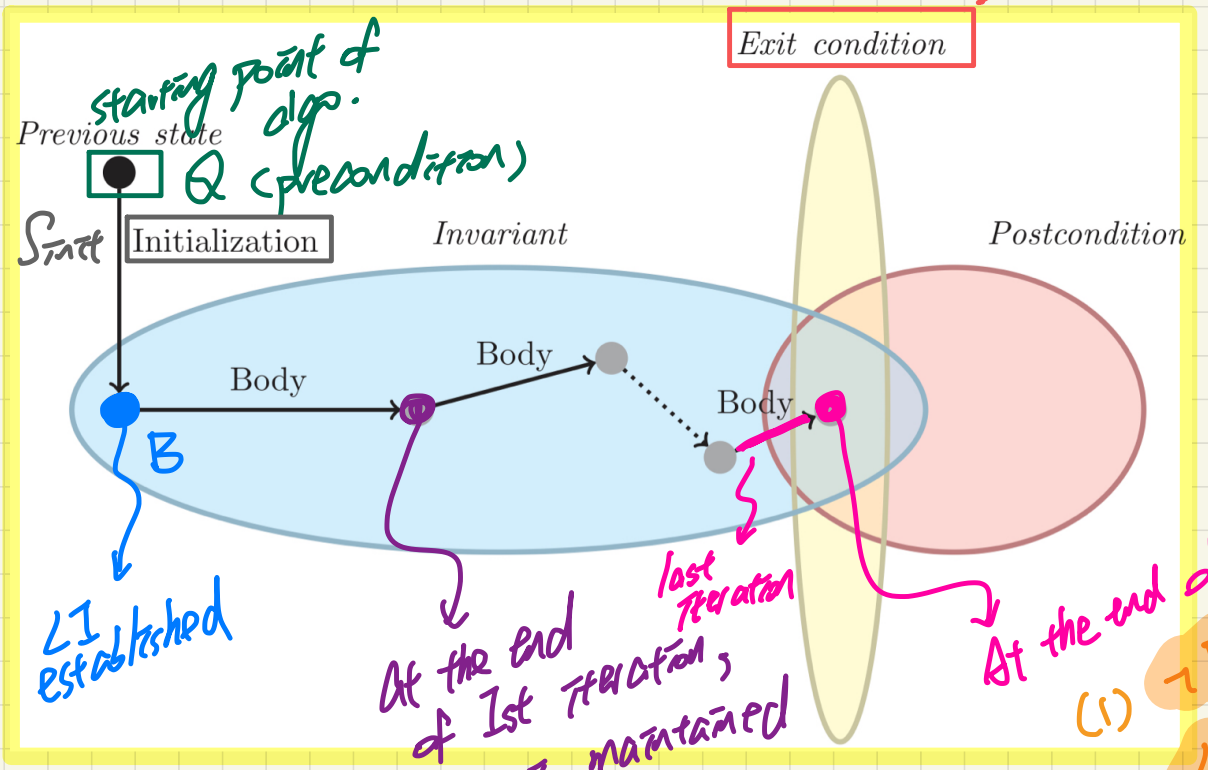
exit — maintained in last iteration

Exercises

(1) $\angle I'$: $1 \leq t \leq 5$

(2) $\angle I''$: $1 < t \leq 6$

Contracts of Loops: Visualization



starting point of algo.

Q (precondition)

¬B.

LI established

At the end of 1st iteration, LI maintained

last iteration

At the end of the last iteration.

(1) ¬B (exit)

(2) LI (maintained)

⇒ R (postcondition)

$* \forall u \cdot u \in V \wedge u \in S \Rightarrow D(u) = d(s, u)$ (w)

Correctness of Loops: Dijkstra's Shortest-Path Algorithm

Recall: A ***loop invariant*** (***LI***) is a Boolean condition.
- ***LI*** is establisehd before the 1st iteration.
- ***LI*** is preserved **at the end of** each subsequent iteration.

The (iterative) Dijkstra's algorithm has ***LI***:

> For every vertext u that has already been removed from the priority queue Q (i.e., u is considered visited), $D(u)$ equals the *true* shortest-path distance from source s to u.

$d(s, u)$

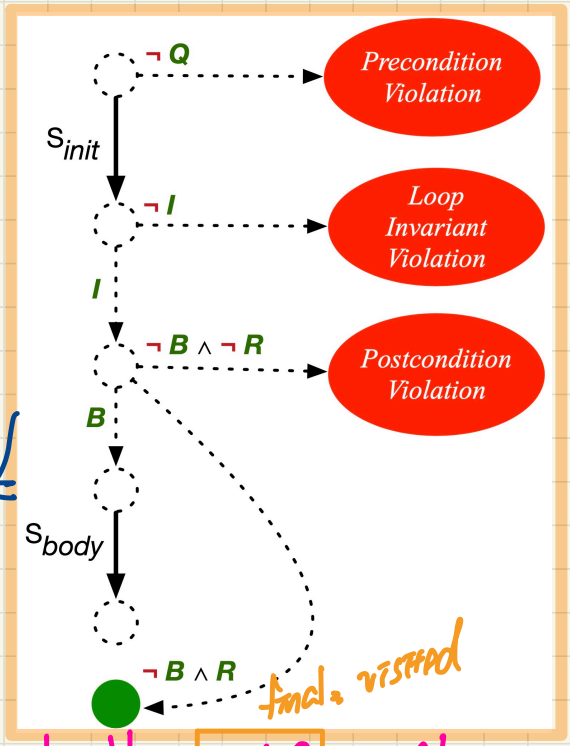
min length so far

$G = (V, E)$ Apply Dijkstra from $s \in V$

S: set of vertices removed so far.

It.	Remove	LI	S
1	w	LI_1 $D(w) = d(w,w)$	$\{w\}$
2	z	LI_2 $LI_1 \wedge D(z) = d(w,z)$	$\{w, z\}$
3	y	LI_3 $LI_2 \wedge D(y) = d(w,y)$	$\{w, z, y\}$
4	x	LI_4 $LI_3 \wedge D(x) = d(w,x)$	$\{w, z, y, x\}$

at the end of It.



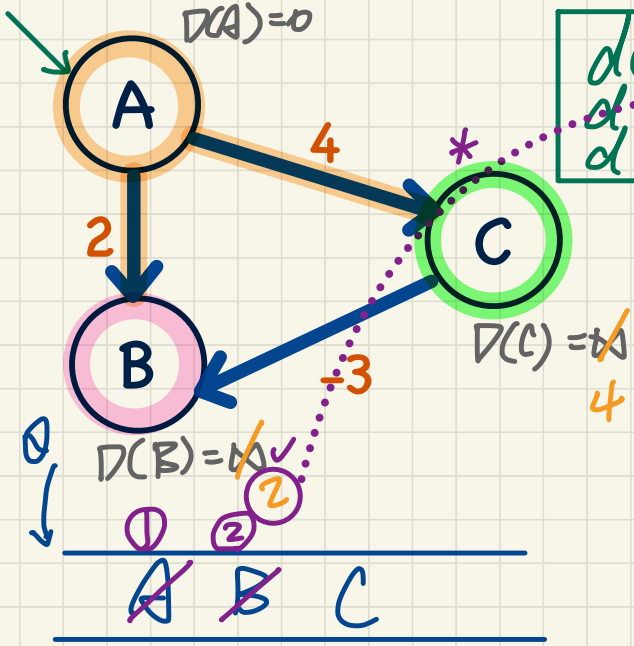
LI #1: $\forall u \cdot u \notin Q \wedge u \in V \Rightarrow D(u) = d(s, u)$ (w)

LI #2: *

Dijkstra's Shortest Path Algorithm: Negative Weights

The (iterative) Dijkstra's algorithm has ***LI***:

> For every vertext *u* that has already been removed from the priority queue *Q* (i.e., *u* is considered visited), *D*(*u*) equals the ***true*** shortest-path distance from source *s* to *u*.



$d(A,A) = 0$
$d(A,B) = 1$
$d(A,C) = 4$

* At the end of 2nd iteration, the removed vertex B's final value D=2, not equal to the true $d(A,B)=1$ ↳ violation of LI

It.	min D	changes
1	A D(A)=0	B∈Q C∈Q D(B)=2 D(C)=4 remove A from Q → S={A}
2	B D(B)=2	n.c. remove (B) from Q → S={A,B} ✓
3	C D(C)=4	B∉Q no change.