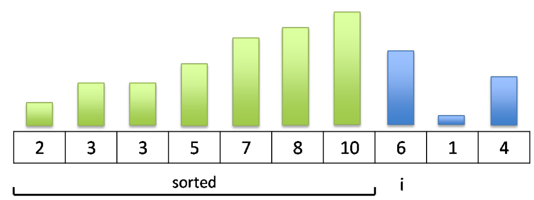
Recitation 4: A Strange Sort of Proof — Thursday February 6th

In class, we covered one quadratic sort, *selection sort*. Today, we'll look at the full correctness proof, from beginning to end, for another sorting algorithm, *insertion sort*. This is somewhat more complex of a proof, so be sure to follow along carefully!

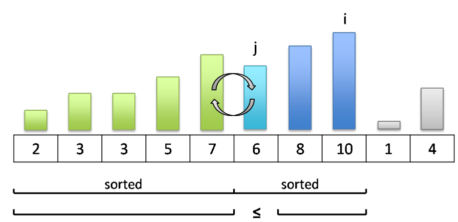
Insertion sort

```
void sort(int[] A, int n)
//@requires 0 <= n && n <= \length(A);
//@ensures is_sorted(A, 0, n);
{
  for (int i = 0; i < n; i++)
  //@loop_invariant 0 <= i && i <= n;
  //@loop_invariant is_sorted(A, 0, i);
  {
    int j = i;

    while (j > 0 && A[j-1] > A[j])
    //@loop_invariant 0 <= j && j <= i;
    //@loop_invariant is_sorted(A, 0, j);
    //@loop_invariant is_sorted(A, j, i+1);
    //@loop_invariant le_segs(A,0,j, A,j+1,i+1);
    {
      swap(A, j-1, j);
      j--;
    }
  }
}
```



Outer loop invariant, schematically



Inner loop invariant, schematically

To proceed, we need to follow the same four steps we have used all semester to show correctness for a function with a loop. But when we get to the preservation step, the loop body is itself a loop, so we must repeat the processes. Below is the structure we will follow.

1. Prove outer loop invariants hold INITially
2. Show that outer loop invariants are PREServed
   (a) Prove inner loop invariants hold INITially
   (b) Show that inner loop invariants are PREServed
   (c) Prove that the inner loop TERMinates
   (d) Show that the inner loop invariants and negation of the inner loop guard imply the outer loop invariants
3. Show that the outer loop TERMinates
4. Prove that the postcondition holds on EXIT

1. Prove that, given the preconditions, the outer loop invariants INITially hold.

2. In order to prove that the outer loop invariants are correct, we will need the negation of the *inner* loop guard along with the preservation and correctness of the inner loop invariants.

   (a) Assume that the outer loop invariants hold for some iteration of the loop. Prove that the **inner** loop invariants initially hold on this iteration. You can imagine you are trying to show the loop invariants are true initially for the following block of code:

```
6  //@assert 0 <= i && i <= n;
7  //@assert is_sorted(A, 0, i);
8
9  int j=i;
10
11 while (j > 0 && A[j-1] > A[j])
12 //@loop_invariant 0 <= j && j <= i;
13 //@loop_invariant is_sorted(A, 0, j);
14 //@loop_invariant is_sorted(A, j, i+1);
15 //@loop_invariant le_segs(A, 0, j, A, j+1, i+1);
16 {
17   swap(A, j-1, j);
18   j--;
19 }
```

   Line 12:

   Line 13:

   Line 14:

   Line 15:

   (b) Given that the **inner** loop invariants hold INITially, prove that they are PREServed.

   Line 12:

   Line 13:

   Line 14:

   Line 15:

   (c) Prove the inner loop TERMinates:

   We successfully proved the inner loop invariants correct! Now all that's left is to prove the outer ones.

(d) Prove the outer loop invariants are PREServed, given that the inner loop invariants hold.

____

____

____

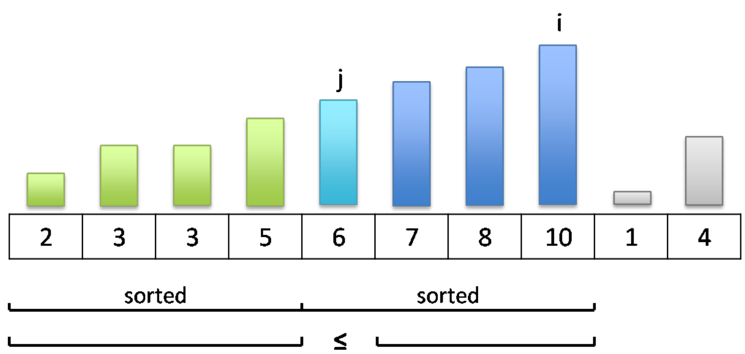
3. Prove that the outer loop TERMinates: ______________________________

4. Finally, prove that the termination of the outer loop and the negation of the outer loop guard prove the postcondition on EXIT. The figure below may be a helpful visualization as you do so.

____

____

____



Whew! We're done!