Verification of mutable Arrays

19CSE205 : PROGRAM REASONING

Dr. Swaminathan J

Assistant Professor

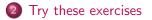
Department of Computer Science and Engineering



Jul - Dec 2020



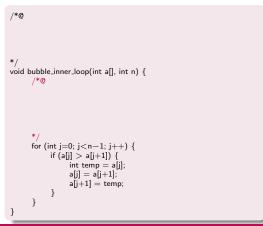






Implementing the inner loop of bubble sort.

- Starting from index 0, every adjacent pair of elements are compared.
- If left one is bigger, it is swapped with right.
- As a result, the largest element moves to the rightmost/last position (n-1).



 What are the constraints on n?
 What are the constraints on array range?

< ∃ ►



Implementing the inner loop of bubble sort.

- Starting from index 0, every adjacent pair of elements are compared.
- If left one is bigger, it is swapped with right.
- As a result, the largest element moves to the rightmost/last position (n-1).

```
/*@ requires n > 0; requires \valid(a + (0..n-1));
```

```
*/
void bubble_inner_loop(int a[], int n) {
    /*@
```

 What is the expected at the end of the loop?
 How does the last element relate to the rest?

< ∃ ►



Implementing the inner loop of bubble sort.

- Starting from index 0, every adjacent pair of elements are compared.
- If left one is bigger, it is swapped with right.
- As a result, the largest element moves to the rightmost/last position (n-1).

```
/*@
      requires n > 0;
      requires \forall alid(a + (0..n-1));
      ensures \ forall integer i;
           0 \le i \le n-1 = a[n-1] > a[i];
*/
void bubble_inner_loop(int a[], int n) {
      /*@
      for (int j=0; j<n-1; j++) {
           if (a[j] > a[j+1]) {
                 int temp = a[i];
                 a[i] = a[i+1];
                 a[i+1] = temp:
```

5. What is the entry span of the loop?6. What is the exit point of the loop?

< ∃ ►



Implementing the inner loop of bubble sort.

- Starting from index 0, every adjacent pair of elements are compared.
- If left one is bigger, it is swapped with right.
- As a result, the largest element moves to the rightmost/last position (n-1).

loop invariant $0 \le j \le n-1$;

```
  \label{eq:states} \begin{array}{l} */ \\ \text{for (int } j{=}0; \, j{<}n{-}1; \, j{+}+) \; \{ \\ & \text{if } (a[j] > a[j{+}1]) \; \{ \\ & \text{int } temp = a[j]; \\ & a[j] = a[j{+}1]; \\ & a[j{+}1] = temp; \\ \; \} \\ \} \end{array}
```

7. How do we capture the progress made as iterations proceed? $\rightarrow \forall i \in [0,j), arr[j] \ge arr[i]$

→ Ξ →



Implementing the inner loop of bubble sort.

- Starting from index 0, every adjacent pair of elements are compared.
- If left one is bigger, it is swapped with right.
- As a result, the largest element moves to the rightmost/last position (n-1).

```
/*@
      requires n > 0;
      requires \forall alid(a + (0..n-1));
      ensures \ forall integer i;
           0 \le i \le n-1 = a[n-1] > a[i];
*/
void bubble_inner_loop(int a[], int n) {
      /*@
        loop invariant \forall integer i:
           0 \le i \le i = 3 afil \ge 3 afil:
        loop invariant 0 \le i \le n-1;
      for (int i=0; i< n-1; i++) {
           if (a[j] > a[j+1]) {
                 int temp = a[i];
                 a[i] = a[i+1];
                 a[i+1] = temp:
```

8. Which variables can be legally assigned within the loop?

- 3 → 1



Implementing the inner loop of bubble sort.

- Starting from index 0, every adjacent pair of elements are compared.
- If left one is bigger, it is swapped with right.
- As a result, the largest element moves to the rightmost/last position (n-1).

```
/*@
      requires n > 0;
      requires \forall alid(a + (0..n-1));
      ensures \ forall integer i;
           0 \le i \le n-1 = a[n-1] > a[i];
*/
void bubble_inner_loop(int a[], int n) {
      /*@
        loop invariant \forall integer i;
           0 \le i \le j = a[j] \ge a[i];
        loop invariant 0 \le i \le n-1;
        loop assigns j, a[0..j+1];
      */
      for (int i=0; i< n-1; i++) {
           if (a[j] > a[j+1]) {
                 int temp = a[i];
                 a[i] = a[i+1];
                 a[i+1] = temp:
```

9. What must be the termination condition for the loop?

→ Ξ →



Implementing the inner loop of bubble sort.

- Starting from index 0, every adjacent pair of elements are compared.
- If left one is bigger, it is swapped with right.
- As a result, the largest element moves to the rightmost/last position (n-1).

```
/*@
      requires n > 0;
      requires \forall alid(a + (0..n-1));
      ensures \ forall integer i;
           0 \le i \le n-1 = a[n-1] \ge a[i];
*/
void bubble_inner_loop(int a[], int n) {
      /*@
        loop invariant \forall integer i;
           0 \le i \le j = a[j] \ge a[i];
        loop invariant 0 \le i \le n-1;
        loop assigns i, a[0,.i+1];
        loop variant n - 1 - j;
      *
      for (int i=0; i< n-1; i++) {
           if (a[j] > a[j+1]) {
                 int temp = a[i];
                 a[i] = a[i+1];
                 a[i+1] = temp:
```

- 4 ∃ ▶



 In the above implementation of bubble sort inner loop, pairwise comparison is done from 0 to n-1 and pairs are swapped when necessary, resulting in moving of max element to (n-1)th position. Now, change the implementation to do this pairwise swapping upto a certain given index k.

```
void bubble_inner_range(int a[], int n, int k) { ... }
```

- 2. Make appropriate changes to the post condition and loop invariant to verify the correctness of this changed implementation.
- 3. In a similar manner, implement the inner loop of selection sort. Essentially it finds the position where the max element resides and swaps it with the last position. i.e. if max element occurs at index 3, a[3] is swapped with a[n-1] so that the last element becomes the max.
- 4. Given an array a[] write a program to compute the cumulative array. For example, if a[] = $\{1, 3, 0, 5, 4\}$, then cumulative array = $\{1, 1+3, 1+3+0, 1+3+0+5, 1+3+0+5+4\}$ = $\{1, 4, 4, 9, 13\}$. Provide loop invariant.

(4) (日本)