Worksheet 3: Asymptotic Analysis

Rin Meng 51940633
Kevin Zhang 10811057
Mika Panagsagan 29679552
Priyansh Mathur 84491356

January 23, 2025

3 Progress Measures for While Loops

Assume that `FindNeighboringInversion(A)` consumes an array `A` and returns an index i such that `A[i] > A[i+1]` or returns -1 if no such inversion exists. Let's work out a bound on the number of iterations of the loop below in terms of n, the length of the array `A`.

```
Let i = FindNeighboringInversion(A)
While i >= 0:
    Swap A[i] and A[i+1]
    Set i to FindNeighboringInversion(A)
```

1. Give and work through two small inputs

   Let us define two arrays A where

$$A = [3, 1, 4, 2, 5]$$

   (a) since $A[i] > A[i+1] = A[0] > A[1]$, $i = 0$

   (b) Since $i \geq 0$, we swap $A[0]$ and $A[1]$, so $A = [1, 3, 4, 2, 5]$

   (c) Set $i =$ `FindNeighboringInversion(A)`

   (d) Now we choose that the next inversion occurs at $i = 2$ since $A[2] > A[3]$

(e) Since $i \geq 0$, we swap $A[2]$ and $A[3]$, so $A = [1, 3, 2, 4, 5]$.

(f) Set $i =$ `FindNeighboringInversion(A)`

(g) Now we choose that the next inversion occurs at $i = 1$ since $A[1] > A[2]$

(h) Since $i \geq 0$, we swap $A[1]$ and $A[2]$, so $A = [1, 2, 3, 4, 5]$.

(i) Set $i =$ `FindNeighboringInversion(A)`

(j) There are no inversions, therefore $i = -1$

(k) Since $i < 0$, we terminate the loop.

$$A = [5, 4, 3, 2, 1]$$

(a) since $A[i] > A[i + 1] = A[0] > A[1]$, $i = 0$

(b) Since $i \geq 0$, we swap $A[0]$ and $A[1]$, so $A = [4, 5, 3, 2, 1]$

(c) Set $i =$ `FindNeighboringInversion(A)`

(d) Now we choose that the next inversion occurs at $i = 1$ since $A[1] > A[2]$

(e) Since $i \geq 0$, we swap $A[1]$ and $A[2]$, so $A = [4, 3, 5, 2, 1]$.

(f) Set $i =$ `FindNeighboringInversion(A)`

(g) Now we choose that the next inversion occurs at $i = 2$ since $A[2] > A[3]$

(h) Since $i \geq 0$, we swap $A[2]$ and $A[3]$, so $A = [4, 3, 2, 5, 1]$.

(i) Set $i =$ `FindNeighboringInversion(A)`

(j) Now we choose that the next inversion occurs at $i = 3$ since $A[3] > A[4]$

(k) Since $i \geq 0$, we swap $A[3]$ and $A[4]$, so $A = [4, 3, 2, 1, 5]$.

From this, we can see that the number of iterations is very long.

2. Define an inversion (not just a neighboring one), and prove that if an inversion exists at all, a neighboring inversion exists.

   An inversion is defined as a pair of indices $(i, i + 1)$ such that $A[i] > A[i + 1]$.

Proof by contradiction: Assume that an inversion exists at index i such that

$$A[i] > A[i+1]$$

and there is no neighboring inversion. This implies that $A[i+1] \leq A[i+2]$. However, this contradicts the definition of an inversion. Therefore, if an inversion exists, a neighboring inversion must exist.

3. Give upper-bounds and lower-bounds on the number of inversions in A.

   Upperbound:

   $$O(\frac{n(n-1)}{2})$$

   Justification: If the array is arranged in decreasing order, then there are $n(n-1)/2$ inversions.

   Lowerbound:

   $$\Omega(0)$$

   Justification: If the array is arranged in increasing order, then there are no inversions.

4. Give a "measure of progress" for each iteration of the loop in terms of inversions. (I.e., how can we measure that we're making progress toward terminating the loop?)

   The measure of progress is the number of inversions in the array A. If the number of inversions decreases, then we are making progress towards terminating the loop.

5. Give an upper-bound on the number of iterations the loop could take.

   The upper-bound on the number of iterations the loop could take is $O(n^2)$.

6. Prove that this algorithm sorts the array A (i.e., removes all inversions from the array).

   Proof by contradiction: Assume that the algorithm does not sort the array A. This implies that there exists an inversion in the array A. However, the algorithm terminates when there are no inversions in the array A. Therefore, the algorithm must sort the array A.