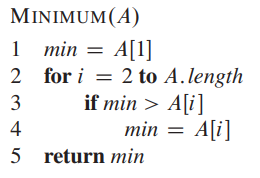
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
| C:\Users\saif\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\final design.jpg | **Course Name:** | **Design and Analysis of Algorithms** | **Course Code:** | **CS2009** |
| **Degree Program:** | **BSCS** | **Semester:** | **FALL 2022** |
| **Due Date:** | **Thursday, February 16, 2023** | **Total Marks:** | **10 + 10 + 10 = 30** |
| **Section:** | **4B** | **Page(s):** | **2** |
| **Exam Type:** | **Assignment 1** |  |  |
| **Student : Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Roll No.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section:\_\_\_\_\_\_\_** | | | | |
| **Instruction/Notes:** |  | | | |

**Loop invariants**

Use loop invariants to prove the correctness of the following algorithms

Note: Use the code given in this assignment.

1. Find Min



Loop Invariant: At the start of each iteration min holds the value of smallest element A[1…A.length].

Initialization: Before first iteration min is initialized to A[1] so loop invariant holds first iteration.

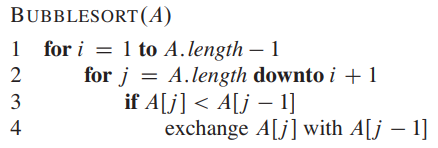
Maintenance: At the end of each iteration, if A[i] < min then min is updated with the value of A[i]. Min then holds the smallest element in A[1…i].

Termination: Loop terminates when i is equal to A.length so min holds the smallest element in A[1…A.length].

Example:

A = {3,5,6,3,9,0,3};

1. Min=3
2. Min=3
3. Min=3
4. Min=3
5. Min=3
6. Min=0
7. Min=0
8. Bubble Sort



Loop Invariant: At the start of each iteration of outer loop elements in A[1…i-1] are sorted.

Initialization: Before the first iteration i==1 so A[1…0] is empty. Therefore, loop invariant holds.

Maintenance: At the end of each outer loop iteration, largest element in A[i…A.length] bubbles up to the end and A[1…i] is still sorted.

Termination: Loop terminates when i==A.length-1 hence A[1…A.length-1] is sorted since inner loop insures the largest element is at the end of each subarray

Example:

A = {3,5,6,3,9,0,3};

1. 3 5 6 3 9 0 3
2. 0 3 5 6 3 9 3
3. 0 3 3 5 6 3 9
4. 0 3 3 3 5 6 9
5. 0 3 3 3 5 6 9
6. 0 3 3 3 5 6 9
7. 0 3 3 3 5 6 9
8. 0 3 3 3 5 6 9
9. Selection Sort

SELECTIONSORT(*A*)

1. **for**
3. **for**
4. **if**

Loop Invariant: At the start of each iteration of outer loop elements in A[1…j-1] are sorted.

Initialization: Before first iteration j==1 so A[1…0] is empty. Therefore, first iteration holds.

Maintenance: At the end if each iteration of outer loop smallest\_index in A[i…A.length]. Then the A[smallest\_index] is swapped with A[j]. Hence, array is sorted for A[1…j-1].

Termination: When outer loop terminates j==A.length-1 and A[j…A.length-1] is sorted, sorted, since smallest\_idex gets the element of the smallest index in A[1…A.length] and swap with A[i]. Hence A is sorted.

A = {3,5,6,3,9,0,3};

1. 3 5 6 3 9 0 3
2. 0 5 6 3 9 3 3
3. 0 3 6 5 9 3 3
4. 0 3 3 5 9 6 3
5. 0 3 3 3 9 6 5
6. 0 3 3 3 5 6 9
7. 0 3 3 3 5 6 9
8. 0 3 3 3 5 6 9