CSCI 4041, Spring 2019, Written Assignment 2

Due Tuesday, 2/5/19, 1:00 PM (submission link on Canvas)

Group17 member:
Thomas Tang - tang0376
Jackie Ji - ji000011
Song Liu - liux4169

This is a collaborative assignment; you may work in a group of 1-3 students. However, you may not consult or discuss the solutions with anyone other than the course instructor, the TAs, or the other members of your group, nor may you use material found from outside sources as part of your solutions. In addition, if you do choose to work in a group, each group member must participate in coming up with the solution to each problem, and must be able to explain the group's answer if asked: dividing the problems amongst the group members is not acceptable.

Complete the following problems and submit your solutions in a single pdf file to the Written Assignment 2 submission link on Canvas. If you're working in a group, only one person should submit your answers, but make sure that you include the name and x500 of each group member at the top of the file, and that you are all in one of the Written Assignment Groups in Canvas. Typed solutions are preferred, but pictures or scans of a handwritten solutions in pdf form are acceptable so long as your solutions are clearly legible.

This assignment contains 4 problems, and each is worth 10 points, for a total of 40 points.

Your solutions to these problems must be clearly explained in a step-by-step manner; for most problems, the explanation will be worth far more points than the actual answer.

Problems 1-3 concern Selection Sort, which works by finding the minimum value in the array, and placing it in A[1], then finding the second smallest value, and placing it in A[2], and so on. Pseudocode for Selection Sort can be found below:

```
SELECTION-SORT(A)
1     for i = 1 to A.length-1
2           for j = i+1 to A.length
3                 if A[j] < A[i]
4                       exchange A[i] with A[j]
```

1. (Adapted from Exercise 2-2.2)
    a. Show every step of Selection Sort on the array [5, 2, 3, 4, 1] (that is, write out the array after every exchange operation).
    When i = 1, j = 2, [2,5,3,4,1];

When i = 1, j = 5, [1,5,3,4,2];
When i = 2, j = 3, [1,3,5,4,2];
When i = 2, j = 5, [1,2,5,4,3];
When i = 3, j = 4, [1,2,4,5,3];
When i = 3, j = 5, [1,2,3,5,4];
When i = 4, j = 5, [1,2,3,4,5].

b. When using Selection Sort on an any array of size 5, what are the minimum and maximum number of comparisons (that is, the number of times line 3 executes)?
Min: 4+3+2+1=10
Max: 4+3+2+1=10

c. When using Selection Sort on an any array of size 5, what are the minimum and maximum number of exchanges (that is, the number of times line 4 executes)?
Max: 4+3+2+1=10
Min: 0

2. (Adapted from Exercise 2.2-2) Give a best-case and worst-case running time for Selection Sort using Θ-notation, and argue informally that your answers are correct.

Θ(n^2) is the running time for both best-case and worst-case. It will go through both inner loop and outer loop which gives the Θ(n^2) no matter what input.

3. (Adapted from Problem 2-2) You may assume for this problem that all elements in the array being passed to Selection Sort are distinct.

a. Here is a possible loop invariant for the inner for loop of Selection Sort:

> At the start of each iteration of the for loop on lines 2-4, A[i] is the smallest element in the subarray A[i...j-1].

Prove that this loop invariant holds. Your proof should resemble the structure of the loop invariant proofs in Sections 2.1 and 2.3 of the textbook, including arguments for Initialization, Maintenance, and Termination.

Initialization: At the start of iteration where j = i+1 of the inner loop, A[i] is the smallest element in the subarray A[i..i]. Since A[i...i] only have one element, the only element A[i] must be the smallest.

Maintenance:

Assume: at the start of iteration j = i+k of the inner loop, A[i] is the smallest element in the subarray A[i...i+k-1].

Prove: at the start of iteration j = i+k+1 of the inner loop, A[i] is the smallest element in the subarray A[i...i+k]. If the A[i+k] is smaller than A[i], then after the i+kth iteration it will exchange with A[i]. The old A[i+k] becomes the new A[i], which now makes A[i] the smallest in the subarray A[i..i+k]. If the A[i+k] is

bigger than A[i], then A[i] is the smallest in the subarray A[i...i+k] without any operation.

Termination: at the start of iteration j = A.length+1 of the inner loop, A[i] is the smallest element in the subarray A[i...A.length].

b. Using your answer from part a, state precisely a loop invariant for the outer for loop of Selection Sort in lines 1-4, and prove that this loop invariant holds.

Loop invariant: at the start of the ith iteration of the outer loop, the smallest i-1 elements of A are in sorted order in first i-1 indices.

Initialization:at the start of the 1st iteration of the outer loop, the smallest 0 elements of A are in sorted order in first 0 indix. Since it is no elements, it is vacuously true.

Maintenance:

Assume: at the start of the i= kth iteration of the outer loop, the smallest k-1 elements of A are in sorted order in first k-1 indices.

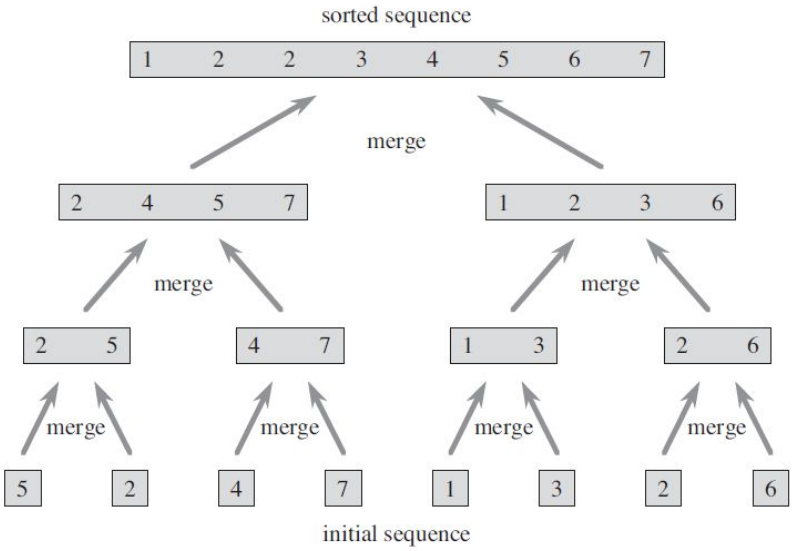
Prove: at the start of the (k+1)th iteration of the outer loop, the smallest k elements of A are in sorted order in first k indices. By the Termination argument for the inner loop, A[k-1] is the smallest element in the subarray A[k...A.length-1] and A[k] is the smallest element in the subarray A[k...A.length], which means A[k-1] < A[k]. For the smallest k-1 elements of A are in sorted order in first k-1 indices which means A[1]<...<A[k-1]<A[k], the smallest k elements of A are in sorted order in first k indices.

Termination: at the start of the (A.length+1)th iteration of the outer loop, the smallest A.length elements of A are in sorted order in first A.length indices.

c. Explain briefly why your answer to part b) proves that Selection Sort always correctly sorts its input.

It uses a general input as the input in the algorithm in part b proofs. When the proof proves that general input can work in the algorithm correctly, the algorithm would work at any input. In other words, this algorithm always correctly sorts its input.

4. Using Figure 2.4 from the textbook as a model (see below), illustrate the operation of merge sort on the array A = [19, 20, 1, 14, 12, 5, 25, 16]. You are not required to draw in arrows or boxes, you just need to somehow show the initial set of 1-element arrays, then the set of 2-element arrays that result from merging them, and so on, until you finally have the sorted 8-element array.



A = [19, 20, 1, 14, 12, 5, 25, 16]

[19], [20], [1], [14], [12], [5], [25], [16]

[19,20], [1,14], [5,12], [16,25]

[1,14,19,20], [5,12,16,25]

[1,5,12,14,16,19,20,25]