Prof. Pietro S. Oliveto

CS217 Data Structures and Algorithm Analysis
Fall term 2024

Exercise Sheet 6

Handout: Oct 22nd — Deadline: Oct 29th - 4pm

Question 6.1 (marks 0.5)

BubbleSort is a popular, but inefficient, sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order. The effect is that small elements "bubble" to the left-hand side of the array, accumulating to form a growing sorted subarray. (You might want to work out your own example to understand this better.)

```
Bubble-Sort(A)
for i = 1 to A.length − 1 do
    for j = A.length downto i + 1 do
        if A[j] < A[j − 1] then
            exchange A[j] with A[j − 1]
```

Prove the correctness of BubbleSort and analyse its running time as follows. Try to keep your answers brief.

1. The inner loop "bubbles" a small element to the left-hand side of the array. State a loop invariant for the inner loop that captures this effect and prove that this loop invariant holds, addressing the three properties initialisation, maintenance, and termination.
2. Using the termination condition of the loop invariant for the inner loop, state and prove a loop invariant for the outer loop in the same way as in part 1. that allows you to conclude that at the end of the algorithm the array is sorted.
3. State the runtime of BubbleSort in asymptotic notation. Justify your answer.

Question 6.2 (0.5 marks)

Consider the following input for Randomized-QuickSort:

12	10	4	2	9	6	5	25	8

What is the probability that:

1. The elements $A[2] = 10$ and $A[3] = 4$ are compared?
2. The elements $A[1] = 12$ and $A[8] = 25$ are compared?
3. The elements $A[4] = 2$ and $A[8] = 25$ are compared?
4. The elements $A[2] = 10$ and $A[7] = 5$ are compared?

Question 6.3 (1 mark)

Prove that the runtime of Randomized-QuickSort is $\Omega(n \log n)$.

(HINT: It may be useful to consider how long it takes to compare $n/2$ elements to achieve a lower bound on the runtime.)

Question 6.4 (1 mark)

Draw the decision tree that reflects how SelectionSort sorts $n = 3$ elements. Assume that all elements are mutually distinct.

For convenience here's the pseudocode again:

```
Selection-Sort(A)
n = A.length
for j = 1 to n − 1 do
    smallest = j
    for i = j + 1 to n do
        if A[i] < A[smallest] then smallest = i
    exchange A[j] with A[smallest]
```

Question 6.5 (0.5 marks)

What is the smallest possible depth of a leaf in a decision tree for a comparison sort?

Question 6.6 (0.25 marks)

Implement Randomized-Quicksort and BubbleSort(A, n).