Prof. Pietro S. Oliveto

Solutions for Exercise Sheet 6

Handout: Oct 24th — Deadline: Oct 31st - 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.

   Solution: the loop invariant is: *at the start of the inner loop, $A[j]$ contains a smallest element out of $A[j], \ldots, A[n]$.*

   Initialisation: $A[n]$ is the smallest element out of $A[n]$.

   Maintenance: if $A[j-1] \leq A[j]$, then $A[j-1]$ is a smallest element out of $A[j-1], \ldots, A[n]$. Decreasing j establishes the loop invariant. Otherwise, after swapping $A[j]$ and $A[j-1]$ we have $A[j-1] < A[j]$ and continue as in the previous case.

   Termination: at the end of the inner for loop, $j = i$ and $A[i]$ is the smallest element out of $A[i], \ldots, A[n]$.

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.

   Solution: the loop invariant is: *at the start of the outer loop, the subarray $A[1 \ldots i-1]$ contains the $i-1$ smallest elements in sorted order.*

   Initialisation: for $i = 1$, $A[1 \ldots 0]$ is empty and contains the 0 smallest elements in sorted order.

   Maintenance: after the end of the inner for loop, $A[i]$ contains the smallest element out of $A[i], \ldots, A[n]$. Then $A[1 \ldots i]$ contains the i smallest elements in sorted order. Incrementing i establishes the loop invariant.

   Termination: at the end of the inner for loop, $i = n$ and $A[1 \ldots n-1]$ contains the $n-1$ smallest elements in sorted order. This implies that $A[n]$ is no smaller and the whole array is sorted.

3. State the runtime of BubbleSort in asymptotic notation. Justify your answer. One iteration of the inner loop takes time $\Theta(1)$. The inner loop is executed $\sum_{i=1}^{n-1}(n-i)$ times. This is $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \Theta(n^2)$. Hence the total time is $\Theta(1)\cdot\Theta(n^2) = \Theta(n^2)$.

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?

Solution: The elements in increasing order are as follows:

$$z_1 = 2; z_2 = 4; z_3 = 5; z_4 = 6; z_5 = 8; z_6 = 9; z_7 = 10; z_8 = 12, z_9 = 25$$

So the probabilities are:

1. $A[2] = 10$ and $A[3] = 4 \iff P(z_7 \text{ and } z_2) = \frac{2}{j-i+1} = \frac{2}{7-2+1} = \frac{1}{3}$
2. $A[1] = 12$ and $A[8] = 25 \iff P(z_8 \text{ and } z_9) = \frac{2}{j-i+1} = \frac{2}{2-1+1} = 1$
3. $A[4] = 2$ and $A[8] = 25 \iff P(z_1 \text{ and } z_9) = \frac{2}{j-i+1} = \frac{2}{9-1+1} = \frac{2}{9}$
4. $A[2] = 10$ and $A[7] = 5 \iff P(z_7 \text{ and } z_3) = \frac{2}{j-i+1} = \frac{2}{7-3+1} = \frac{2}{5}$

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.)

Solution: The expected runtime of the algorithm is (from lecture)

$$E[X] = 2\sum_{i=1}^{n-1}\sum_{k=1}^{n-i}\frac{1}{k+1} > 2\sum_{i=1}^{n/2}\sum_{k=1}^{n-i}\frac{1}{k+1} \tag{1}$$

where on the right side we only sum the first $n/2$ terms. Each term of the sum is greater than:

$$\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n/2} = \sum_{k=1}^{n/2}\frac{1}{k+1} = \left(\sum_{k=1}^{n/2}\frac{1}{k}\right) - 1 \geq \ln(n/2) - 1 = \ln n - \ln 2 - 1$$

Plugging this into Eq (1):

$$E[X] = 2\sum_{i=1}^{n-1}\sum_{k=1}^{n-i}\frac{1}{k+1} > 2\sum_{i=1}^{n/2}\sum_{k=1}^{n-i}\frac{1}{k+1} \geq 2\sum_{i=1}^{n/2}(\ln n - \ln 2 - 1) = 2\frac{n}{2}(\ln n - \ln 2 - 1)$$
$$\geq n \ln n - 2n = \Omega(n \log n)$$

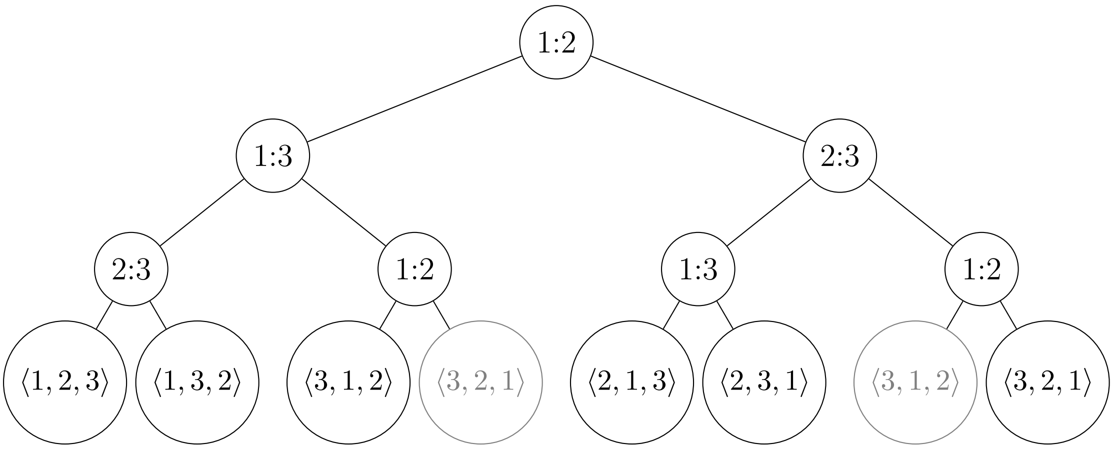
Question 6.4 (1 mark)

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]
```

Solution: In the following tree, edges going left represent the outcome "≤" and edges going right represent the outcome ">".



The leaves drawn in gray are never actually reached as the same comparison was made earlier and we wouldn't have come down the tree this way.

Question 6.5 (0.5 marks)

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

Solution: For a permutation $a_1 \leq a_2 \leq \ldots a_n$ there are $n-1$ pairs of relative sorting, thus the smallest possible depth is $n-1$.

Eg., $n = 3$: $(a_1 <= a_2), (a_2 <= a_3) => < 1, 2, 3 >$, and depth is 2.

Question 6.6 (0.25 marks)

Implement BubbleSort(A, n).