CS217 DSAA Homework6

HONGLI YE 12311501

October 29^{th}2024

Contents

1 Question 1

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.)

Algorithm 1 Bubble-Sort

```
for i = 1 to n − 1 do
    for j = n down to i + 1 do
        if A[j] < A[j − 1] then
            Exchange A[j] with A[j − 1]
        end if
    end for
end for
```

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.

Answer:

1. (a) **Initialization:**
   Let the loop invariant after the loop of $j = k$ be L_k. L_n satisfy that $a[j-1]$ is the smallest element in the subarray in the right side of j.

   (b) **Maintenance:**
   Suppose the ”Right minimum” property holds for L_k.
   In the loop when $j = k - 1$,

   $$A[j-1] = min\{A[j-1], A[j]\} = min\{A[j-1], A[m]\} \text{ where } m \in [j, n].$$

   So, after doing the loop when $j = k - 1$, we get L_{k-1} which still holds the ”Right Minimum” property.

   (c) **Termination:**
   The loop ends when $j = i + 1$, which means $A[i]$ is the minimum of $A[i + 1, n]$

2. (a) **Initialization:**
   Let the loop invariant after the loop of $i = k$ be L_k. L_1 satisfy that $a[1]$ is the smallest element in the subarray in the right side of 1.

   (b) **Maintenance:**
   Suppose the ”Right minimum” property holds for L_k.
   In the loop when $i = k + 1$,

   $$A[k] = min\{A[k+1], A[m]\}, \text{ where } m \in [k+2, n-1]$$
   $$A[k] \leq \text{all elements with index bigger than k}$$
   $$A[k+1] = min\{A[j-1], A[j]\} = min\{A[j-1], A[m]\} \text{ where } m \in [j, n].$$

   So, after doing the loop when $j = k + 1$, we get L_{k+1} which still holds the ”Right Minimum” property.

(c) **Termination:**
The loop ends when $i = n-1$, we get L_{n-1} which satisfy that:

$$a_1 \leq a_2 \leq \dots a_{n-1}$$

Since $A[n-1] \leq$ all elements with index bigger than k, so we have:

$$a_{n-1} \leq a_n$$

$$a_1 \leq a_2 \leq \dots a_{n-1} \leq a_n$$

3. Let c_i be the cost of the code in line i and $T(n)$ be the time cost function for BubbleSort. Then we have:

$$T(n) = nc_1 + \Sigma_{j=2}^{n}(n-(j+1)+1) + \Sigma_{j=2}^{n}(c_{34} \times (n-(j+1)))$$

So:

$$T(n) = \Theta(n^2)$$

2 Question 2

Consider the following input for Randomized-QuickSort:

12	10	4	2	9	6	5	25	8

Table 1: Question 2

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?

Answer:

The array after sorting is:

$$\text{Array:}\{2, 4, 5, 6, 8, 9, 10, 12, 25\}$$

We let P_i be the i^{th} problem's solution. By the formula given on the slides,

$$Pr(z_i \text{ is compared to } z_j) = \frac{2}{j-i+1}$$

1. $4 = z_2$ and $10 = z_7$, so:
$$Pr(4 \text{ is compared to } 10) = \frac{2}{7-2+1} = \frac{1}{3}$$
2. $12 = z_8$ and $25 = z_9$, so:
$$Pr(12 \text{ is compared to } 25) = \frac{2}{9-8+1} = 1$$
3. $2 = z_1$ and $25 = z_9$, so:
$$Pr(4 \text{ is compared to } 10) = \frac{2}{9-1+1} = \frac{2}{9}$$
4. $5 = z_3$ and $10 = z_7$, so:
$$Pr(5 \text{ is compared to } 10) = \frac{2}{7-3+1} = \frac{2}{5}$$

3 Question 3

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

Answer:

Since we need to prove in Ω, so we consider the best case of randomised-quicksort. Let $Z = \{z_1, z_2, \ldots z_n\}$ be the array after being sorted. We already know that the best case is when we pick the $z_{[n/2]}$ number.

Suppose for all partitions we all take the middle element. Then we have:

$$T(n) = 2T(n/2) + \Theta(n).$$

By master theorem, we know the best case's time cost is $\Theta(nlogn)$. So we prove that:

$$T(n) = \Omega(nlogn)$$

4 Question 4

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:

Algorithm 2 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
    end if
  end for
  exchange A[j] with A[smallest]
end for
```

Answer:

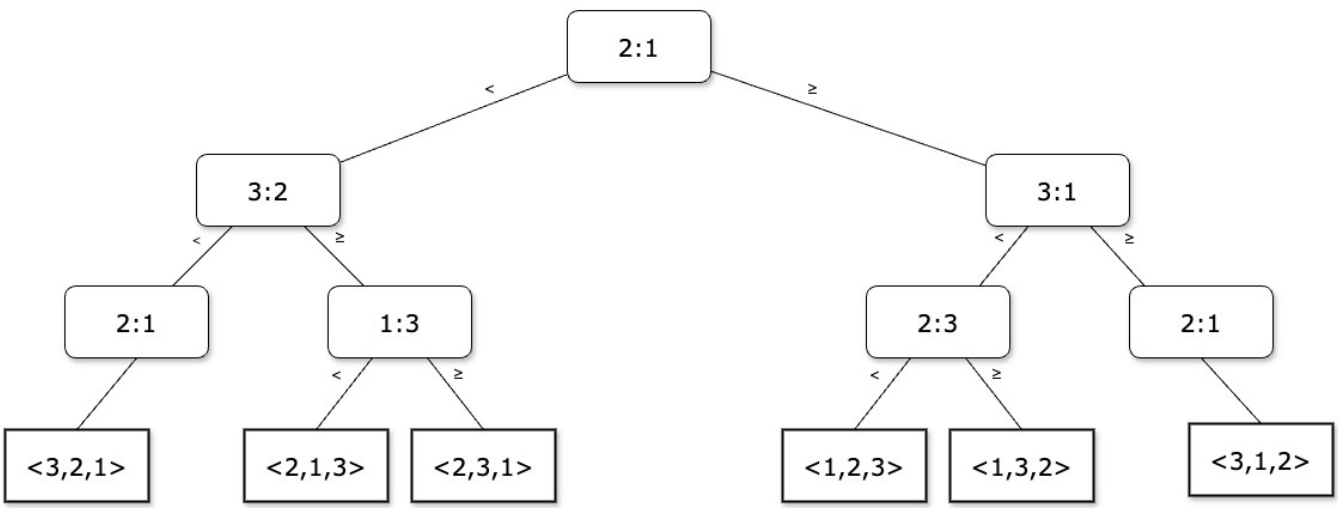

Figure 1: Question 4

5 Question 5

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

Answer:

Since we need to consider the smallest possible depth of a leaf in a decision tree. So we consider the theoretical best situation for a comparison sort. Let d_{min} be the smallest possible depth.
If we need to sort an array, we need at least $n-1$ comparisons for n elements, so we know that:

$$d_{min} \geq n-1$$

Now we construct an array A with n elements, and it satisfy the property that:

$$a_1 > a_n > \cdots > a_2$$

Then after $n-1$ comparisons from 1 to n, we have a new array

$$A' = \{a_2, a_3, \ldots a_n, a_1\}$$

So it is possible that $d_{min} = n-1$.

6 Question 6

Implement $Randommized - QuickSort$ and $BubbleSort(A, n)$

Answer:

I Already submitted it to OnlineJudge.