Exercise Sheet 1

Mengxuan Wu

September 17, 2023

Question 1

Question 1.1

(after) iteration j	$A[1]$	$A[2]$	$A[3]$	$A[4]$	$A[5]$	$A[6]$	$A[7]$	$A[8]$	$A[9]$
1	1	5	6	23	42	45	2	24	8
2	1	2	6	23	42	45	5	24	8
3	1	2	5	23	42	45	6	24	8
4	1	2	5	6	42	45	23	24	8
5	1	2	5	6	8	45	23	24	42
6	1	2	5	6	8	23	45	24	42
7	1	2	5	6	8	23	24	45	42
8	1	2	5	6	8	23	24	42	45

(Assuming the array index starts from 1)

Question 1.2

Loop invariant: After iteration j, the subarray $A[1..j]$ contains the smallest j elements of $A[1..n]$ in sorted order.

Initialisation: After the first iteration, $j = 1$. The algorithm finds the smallest element in the original array $A[1..n]$ and swap it with $A[1]$. The subarray $A[1..j]$ is now $A[1]$, which contains the smallest element in sorted order.

Maintenance: In iteration j, the algorithm swaps the smallest element in subarray $A[j..n]$ with element $A[j]$. Since the subarray $A[1..j-1]$ already contains the smallest elements of $A[1..n]$ in sorted order, the subarray $A[1..j]$ now contains the smallest j elements of $A[1..n]$ in sorted order.

Termination: The loops ends with $n-1$ iteration(s). Then the subarray $A[1..n-1]$ contains the smallest $n-1$ elements in the array $A[1..n]$ in sorted order. Therefore, the last element $A[n]$ must be the largest element in the array $A[1..n]$. The array $A[1..n]$ is now sorted.

Question 1.3

SELECTION-SORT(A)	Cost	Times
$n = A$.length	c_1	1
for $j = 1$ to $n-1$ **do**	c_2	n
smallest $= j$	c_3	$n-1$
for $i = j+1$ to n **do**	c_4	$n + (n-1) + \cdots + 2 = \frac{n^2+n-2}{2}$
if $A[i] < A$[smallest] **then** smallest $= i$	c_5	$(n-1) + (n-2) + \cdots + 1 = \frac{n^2-n}{2}$
exchange $A[j]$ with A[smallest]	c_6	$n-1$

In both the **best case** and the **worst case**, the runtime of the SELECTIONSORT is

$$\begin{aligned}
T(n) &= c_1 + c_2 n + c_3(n-1) + c_4(\frac{n^2+n-2}{2}) + c_5(\frac{n^2-n}{2}) + c_6(n-1) \\
&= (\frac{c_4+c_5}{2})n^2 + (c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} + c_6)n + (c_1 - c_3 - c_6) \\
&= n^2 + 3n - 2
\end{aligned}$$

For INSERTIONSORT, the runtime for the **best case** is $5n-4$ and the runtime for the **worst case** is $\frac{3}{2}n^2 + \frac{7}{2}n - 4$ (from lecture slides).

For the **best case**, INSERTIONSORT is better. Because INSERTIONSORT finishes in linear time, while SELECTIONSORT finishes in quadratic time.

For the **worst case**, both algorithms have quadratic runtime. However, as the coefficient of n^2 in SELECTIONSORT is smaller than that in INSERTIONSORT, SELECTIONSORT is better.