CSCE-629 Homework 1

Wang, Han-Yi
wanghy917@email.tamu.edu

August 22, 2020

2.1-3

Algorithm 1 Linear-Search(A, v)

```
1 for i = 1 to A.length
2     if A[i] == v
3         return i
4 return NIL
```

Loop invariants and the correctness of linear search

At the start of each iteration of the for loop, the subarray $A[1...i-1]$ has scanned, and v is not in the subarray. We state these properties formally as loop invariant. Following the textbook, we let n denote the length of the array in the below paragraphs, and $A.length$ is adopted in the pseudocode.

Initialization: Before the first loop ($i = 1$), the subarray $A[1...(1-0)]$ is empty, and it does not contain v. Therefore, the loop invariant holds before the first iteration of the loop.

Maintenance: In each iteration, if $A[i]$ is equals to v, the if statement in line 2 is true, then the algorithm returns i. On the contrary, if v is not equal to $A[i]$, v is not equal to the elements in $A[1...i]$. Incrementing i for the next iteration of the for loop then preserves the loop invariant.

Termination: The condition causing the for loop to terminate is that $i > n$. Because each loop iteration increases i by 1, we must have $i = n+1$ at that time. Therefore, $A[1...(n+1-1)]$ has scanned, and v is not in $A[1...n]$, so the function returns NIL as shown in line 4 which is correct.

2.2-2

Algorithm 2 Selection-Sort(A)

```
for i = 1 to A.length − 1
    k = A[i]
    for j = i + 1 to A.length
        if A[j] < A[k]
            k = j
    swap(A[i], A[k])
```

What loop invariant does this algorithm maintain?

Before each iteration, the outer loop (line 1-6) maintains $A[1...i-1]$ sorted. Besides, $A[1]$ is the 1_{st} smallest item in A, $A[2]$ is the 2_{nd} smallest item ..., and $A[i-1]$ is the $(i-1)_{th}$ smallest item. While the inner loop maintains the property that k is the index of the the smallest value in $A[i...j-1]$ before each iteration.

Why does it need to run for only the first $n-1$ elements, rather than for all n elements?

For the outer loop, the subarray $A[1..n-1]$ contains the 1_{st} to $(n-1)_{th}$ smallest item after the $(n-1)_{th}$ iteration. Therefore, $A[n]$ is the n_{th} smallest item. As a result, the algorithm only needs to run the first $n-1$ elements.

Give the best-case and worst-case running times of selection sort in Θ-notation

Let c_i denote the running time of the i_{th} line, and the total running time of selection sort is explained as below:

Best-case: $\Theta(n^2)$

For the best case, line 4 is always false, so $T(n)$ is:

$$
\begin{aligned}
T(n) &= c_1(n-1) + c_2(n-1) + (c_3 + c_4) \times \sum_{i=1}^{n-1}(n-i) + c_6(n-1) \\
&= (c_1 + c_2 + c_6)(n-1) + (c_3 + c_4) \times \frac{n \times (n-1)}{2} \\
&= \Theta(n^2)
\end{aligned}
\tag{1}
$$

Worst-case: $\Theta(n^2)$

For the worst case, line 4 is always true, c_5 has to be considered, but $T(n)$ is the same:

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