Chapter 2

November 14, 2022

1 Exercise 2.2-1

Express the function $n^3/1000 - 100n^2 - 100n + 3$ in terms of ,Θ notation.

$\Theta(n^3)$

2 Exercise 2.2-2

Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in A[1]. Then find the second smallest element of A, and exchange it with A[2]. Continue in this manner for the first $n - 1$ elements of A. Write pseudocode for this algorithm, which is known as selection sort. What loop invariant does this algorithm maintain? Why does it need to run for only the first $n - 1$ elements, rather than for all n elements? Give the best-case and worst-case running times of selection sort in ,Θ notation.

```
for i=1 to n-1 do
    min=i
    for j=i+1 to n do
        // find the smallest index ith smallest element
        if A[j]<A[min] then
            min=j
        end if
    end for
    swap A[min] and A[j]
end for
```

This yield's a running time of :

$$\theta(n^2)$$

3 Exercise 2.2-3

Consider linear search again (see Exercise 2.1-3). How many elements of the input sequence need to be checked on the average, assuming that the element being searched for is equally likely to be any element in the array? How about in the worst case? What are the average-case and worst-case running times of linear search in ,Θnotation? Justify your answers.

3.0.1 pseudo code

```
for i = 2 to n
    key = A [ i ]
    j = i- 1
    while j > 0 and A [ j ] < key
        A [ j + 1] = A [ j ]
        j = j - 1
    A [ j + 1] = key
```

4 Exercise 2.2-4

How can we modify almost any algorithm to have a good best-case running time?

For a good best-case running time, modify an algorithm to first randomly produce output and then check whether or not it satisfies the goal of the algorithm. If so, produce this output and halt. Otherwise, run the algorithm as usual. It is unlikely that this will be successful, but in the best-case the running time will only be as long as it takes to check a solution. For example, we could modify selection sort to first randomly permute the elements of A, then check if they are in sorted order. If they are, output A. Otherwise run selection sort as usual. In the best case, this modified algorithm will have running time (n).