2.2-1

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

$\frac{1}{1000} n^3 + 100n^2 - 100n + 3 = \Theta(n^3)$

2.2-2

Consider sorting n numbers stored in array $A[1:n]$ by first finding the smallest element of $A[1:n]$ and exchanging it with the element in $A[1]$. Then find the smallest element of $A[2:n]$, and exchange it with $A[2]$. Then find the smallest element of $A[3:n]$, and exchange it with $A[3]$. 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 worst-case running time of selection sort in Θ-notation. Is the best-case running time any better?

SELECTION-SORT(A, n)

		cost	time
1.	for i = 1 to n - 1	c_1	n
	// Find the smallest element index in A[i:n]	0	n - 1
2.	k = i	c_2	n - 1
3.	for j = i + 1 to n	c_3	n - i + 1
4.	if A[j] < A[k]	c_4	n - i
5.	k = j	c_5	t_i
	// Swap the smalles element with key in index i	0	n - 1
6.	tmp = A[i]	c_6	n - 1
7.	A[i] = A[k]	c_7	n- 1
8.	A[k] = tmp	c_8	n - 1

Loop Invariant

- before each iteration i, A[1:i-1] is the smallest (i-1) number in A[1:n].
- before each iteration j, A[k] is the smallest number in subarr A[i:j-1]

why outer loop needs to run only for first n - 1 element ?

when outer loop terminates, i=n. A[1:n-1] is the smallest (n-1) number in A[1:n]

In other words A[n] is the greatest number in A[1:n] so A[1:n] is sorted.

Complexity Analysis

t_i is the number of times condition in line 4 is true.

$$T(n) = c_1 n + c_2(n-1) + c_3\sum_{i=1}^{n-1}(n-i+1) + c_4\sum_{i=1}^{n-1}(n-i) + c_5\sum_{i=1}^{n-1} t_i$$

$$+ c_6(n-1) + c_7(n-1) + c_8(n-1)$$

$$\sum_{i=1}^{n-1}(n-i+1) = n(n-1) - \frac{n(n-1)}{2} + n - 1 = \frac{(n-1)(2n-n+2)}{2} = \frac{(n-1)(n+2)}{2} = \frac{n^2+n-2}{2}$$

$$\sum_{i=1}^{n-1}(n-i) = n(n-1) - \frac{n(n-1)}{2} = \frac{(n-1)(2n-n)}{2} = \frac{n(n-1)}{2} = \frac{n^2-n}{2}$$

$$T(n) = c_1 n + c_2(n-1) + \frac{c_3}{2}(n^2+n-2) + \frac{c_4}{2}(n^2-n) + c_5\sum_{i=1}^{n-1} t_i$$

$$+ c_6(n-1) + c_7(n-1) + c_8(n-1)$$

In worst-case $A[1:n]$ is strictly decreasing so $t_i = n-i$.

$$T(n) = c_1 n + c_2(n-1) + \frac{c_3}{2}(n^2+n-2) + \frac{c_4}{2}(n^2-n) + \frac{c_5}{2}(n^2-n)$$

$$+ c_6(n-1) + c_7(n-1) + c_8(n-1)$$

$$= \left(\frac{c_3}{2} + \frac{c_4}{2} + \frac{c_5}{2}\right)n^2 + \left(c_1 + c_2 + \frac{c_3}{2} - \frac{c_4}{2} - \frac{c_5}{2} + c_6 + c_7 + c_8\right)n$$

$$- (c_2 + c_3 + c_6 + c_7 + c_8) = a_1 n^2 + b_1 n - c_1$$

Selection sort in worst-case is $\Theta(n^2)$

In best-case $A[1:n]$ is sorted so $t_i = 0$

$$T(n) = c_1 n + c_2(n-1) + \frac{c_3}{2}(n^2+n-2) + \frac{c_4}{2}(n^2-n) + c_6(n-1) + c_7(n-1) + c_8(n-1)$$

$$= \left(\frac{c_3}{2} + \frac{c_4}{2}\right)n^2 + \left(c_1 + c_2 + \frac{c_3}{2} - \frac{c_4}{2} + c_6 + c_7 + c_8\right)n - (c_2 + c_3 + c_6 + c_7 + c_8)$$

$$= a_2 n^2 + b_2 n - c_2$$

Selection sort in best-case is $\Theta(n^2)$

2.2-3

Consider linear search again (see Exercise 2.1-4). How many elements of the input array 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?

Using Θ-notation, give the average-case and worst-case running times of linear search. Justify your answers.

LINEAR-SEARCH(A, n, x)

```
1. i = NIL
2. for j = 1 to n
3.         if A[j] = x
4.                 i = j
5.                 break
```

	cost	time
1	c_1	1
2	c_2	t
3	c_3	$t - 1$
4	c_4	v
5	c_5	v

$T(n) = c_1 + c_2 t + c_3(t-1) + c_4 v + c_5 v$

On Average-case, $A[\lfloor n/2 \rfloor] = x$ so we need to search $\lfloor n/2 \rfloor$ element.

$\therefore t = \lfloor n/2 \rfloor + 1 \quad \therefore v = 1$

$$T(n) = c_1 + \frac{c_2}{2}n + \frac{c_2}{2} + \frac{c_3}{2}n + c_4 + c_5$$
$$= \left(\frac{c_2}{2} + \frac{c_3}{2}\right)n + \left(c_1 + \frac{c_3}{2} + c_4 + c_5\right)$$
$$= a_1 n + b_1$$

if n is odd
$\therefore \lfloor n/2 \rfloor = \frac{n-1}{2} = \frac{n}{2} - \frac{1}{2}$
and this is added constant which will combined at the end under const b_1

Linear search is $\Theta(n)$ in average-case

On Worst-case, x is not presented in the array $A[1:n]$ so we need to search all n elements.

$\therefore t = n+1 \quad \therefore v = 0$

$$T(n) = c_1 + c_2(n+1) + c_3 n = (c_2 + c_3)n + (c_1 + c_2)$$
$$= a_2 n + b_2$$

$\therefore$ Linear search is $\Theta(n)$ in worst-case and average-case also.

2.2-4

How can you modify any sorting algorithm to have a good best-case running time?

by calling IS-SORTED procedure first. If it returns 1, A[1:n] is allready sorted so no need for sorting algorithm. By making this we make the best-case for any sorting algorithm $\Theta(n)$. Otherwise, use the sorting algorithm

IS-SORTED(A, n)

```
1.  sorted = 1
2.  for i = 2 to n
3.      if A[i - 1] > A[i]
4.          sorted = 0
5.          break
6.  return sorted
```