Page 22

2.1-3 - Consider the ***searching problem***:

Input: A sequence of n numbers $A = \langle a_1, a_2, ..., a_n \rangle$ and a value v.
Output: An index i such that $v = A[i]$ or the special value NIL if v does not appear in A.

Write pseudocode for LINEAR SEARCH, which scans the sequence, looking for v. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the necessary three properties.

i Loop Invariants: Used for showing why an algorithm is correct. Has the following properties:

- **Initialization:** It is true prior to the first iteration of the loop.
- **Maintenance:** If it is true before an iteration of the loop, it remains true before the next iteration.
- **Termination:** When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

```
LINEAR SEARCH(A, v):
1   for i = 0 to A.length:
2       if A[i] == v:
3           return cur
4   return NIL
```

Initialization: The index is not found prior to the first iteration since nothing has been searched yet.
Maintenance: At each step i is incremented and $A[i]$ is checked against v and the loop terminates if $v = A[i]$.
Termination: The loop terminates when either v is found, returning the index, or when all of A is searched, returning NIL.

Page 29

2.2-1 - Express the function $\frac{n^3}{1000} - 100n^2 - 100n + 3$ in terms of Θ notation.

$\Theta(n^3)$

2.2-2' - Write pseudocode for SELECTION SORT.

```
SELECTION SORT(A):
 1  for cur = 0 to A.length:
 2      small = A[cur]
 3      index = cur
 4      for k = cur + 1 to A.length:
 5          if A[k] < small:
 6              small = A[k]
 7              index = k
 8      temp = A[cur]                    // Swap the values
 9      A[cur] = small
10      A[index] = temp
```

Page 39

2.3-3 - Use mathematical induction to show that when n is an exact power of 2, the solution of the recurrance

$$T(n) = \begin{cases} 2 & \textbf{if } n = 2 \\ 2T(\frac{n}{2}) + n & \textbf{if } n = 2^k \textbf{ for } k > 1 \end{cases}$$

is $T(n) = n \log_2 n$.

We show that T holds for $n = 2$

$$T(2) = 2 \log_2 2 = 2 * 1 = 2$$

Assuming $T(\frac{n}{2}) = \frac{n}{2} \log_2 \frac{n}{2}$

$$\begin{aligned} T(n) &= 2\Big(\frac{n}{2} \log_2 \frac{n}{2}\Big) + n \\ &= 2\Big(\frac{n}{2}(\log_2 n - \log_2 2)\Big) + n \\ &= 2\Big(\frac{n}{2}(\log_2 n - 1)\Big) + n \\ &= (n \log_2 n - n) + n \\ &= n \log_2 n \end{aligned}$$

Therefore $T(n) = n \log_2 n$.

2.3-4 - We can express insertion sort as a recursive procedure as follows. In order to sort $A[1..n]$, we recursively sort $A[1..n-1]$ and then insert $A[n]$ into the sorted array $A[1..n-1]$. Write a recurrance for the running time of this recursive version of insertion sort.

Undefined

Induction Theorem:

$$P(n_0) \land (\forall n_i > n + o.P(n_i) \to P(n_{i+1})) \to \forall n > n_o P(n)$$

1.1-1 - Use mathematical induction to show that $T(n) = \frac{n^2+n}{2}$ for

$$T(n) = \begin{cases} 1 & \textbf{if } n = 1 \\ T(n-1) + n & \textbf{if } n > 1 \end{cases}$$

We show that T holds for $n_0 = 1$

$$T(1) = \frac{1^2 + 1}{2} = \frac{2}{2} = 1$$

Assuming $T(n-1) = \frac{(n-1)^2+(n-1)}{2}$ and $n > 1$

$$\begin{aligned} T(n-1) &= \frac{(n-1)^2 + n - 1}{2} \\ &= \frac{n^2 - 2n + 1 + n - 1}{2} \\ &= \frac{n^2 - n}{2} \end{aligned}$$

By the definition of $T(n)$

$$\begin{aligned} T(n) &= T(n-1) + n \\ &= \frac{n^2 - n}{2} + n \\ &= \frac{n^2 - n + 2n}{2} \\ &= \frac{n^2 + n}{2} \end{aligned}$$

Therefore

$$T(n) = \frac{n^2 + n}{2}$$

By the induction theorem:

$$T(n) = \frac{n^2 + n}{2} \ \forall n \in \mathbb{N}, n \neq 0$$

■