

2.1-3 - Consider the *searching problem*:

Input: A sequence of n numbers $A = \langle a_1, a_2, \dots, a_n \rangle$ 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.

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 - Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in $A[0]$. Then finding the second smallest element of A , and exchange it with $A[1]$. Continue in this manner for the first $n - 1$ elements of A . This algorithm is known as SELECTION SORT.

A) Write pseudocode for SELECTION SORT.

B) What [loop invariants](#) does this algorithm maintain?

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

D) Give the best-case and worst-case running times of SELECTION SORT in Θ -notation.

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 recurrence

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

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

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

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 recurrence for the running time of this recursive version of insertion sort.

Undefined

Induction Theorem:

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

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

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

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:

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

■

1.1-2 - Proof that $x * 2 = x + x$ using only the following identities:

$$2 = 1 - 1 \quad 1.$$

$$a * 1 = a \quad 2.$$

$$a * b = (a * (b - 1)) + a \quad 3.$$

$$x * 2 = x + x$$

$$\Leftrightarrow (\text{id } 3)$$

$$(x * (2 - 1)) + x = x + x$$

$$\Leftrightarrow (\text{id } 1)$$

$$(x * 1) + x = x + x$$

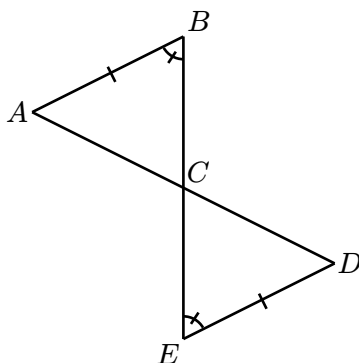
$$\Leftrightarrow (\text{id } 2)$$

$$x + x = x + x$$

Tautology, therefore we're done. ■

1.1-3 - Proof that $\triangle ABC$ is congruent to $\triangle CDE$ using the following identities:

- If two triangles have two congruent angles and one congruent side in the same order relative to each other they are congruent. (Usually called "AAS Congruent").
- When two straight lines cross, their opposite angles are congruent to each other.



When \overline{AD} crosses with \overline{BE} the following angles become congruent:

$$\angle ACB \cong \angle DCE$$

$$\angle BCD \cong \angle ACE$$

For an Angle-Angle-Side congruent triangle we need 2 congruent angles and one congruent side.

$$\overline{DE} \cong \overline{AB}$$

$$\angle CED \cong \angle ABC$$

$$\angle ACB \cong \angle DCE$$

By AAS Congruency:

$$\triangle ABC \cong \triangle CDE$$

1.1-4 - Proof that the inversion of $\neg A \wedge ((B < C) \vee D)$ is $A \vee ((B \geq C) \wedge \neg D)$.
List all identities you used.

Given the following identities:

$$\neg(\neg A) = A \quad (1)$$

$$\neg(A \wedge B) = \neg A \vee \neg B \quad (2)$$

$$\neg(A \vee B) = \neg A \wedge \neg B \quad (3)$$

$$\neg(A < B) = A \geq B \quad (4)$$

We can proof

$$\neg(\neg A \wedge ((B < C) \vee D)) = A \vee ((B \geq C) \wedge \neg D)$$

$$\Leftrightarrow (\text{id } 2)$$

$$\neg(\neg A) \vee \neg((B < C) \vee D) = A \vee ((B \geq C) \wedge \neg D)$$

$$\Leftrightarrow (\text{id } 1)$$

$$A \vee \neg((B < C) \vee D) = A \vee ((B \geq C) \wedge \neg D)$$

$$\Leftrightarrow (\text{id } 3)$$

$$A \vee (\neg(B < C) \wedge \neg D) = A \vee ((B \geq C) \wedge \neg D)$$

$$\Leftrightarrow (\text{id } 4)$$

$$A \vee ((B \geq C) \wedge \neg D) = A \vee ((B \geq C) \wedge \neg D)$$

Tautology, therefore we're done. ■