1. Problemas

2. Getting Started

2.1. Insert-Sort

2.1.1.

Using Figure 2.2 as a model, illustrate the operation of INSERTION-SORT on the array A = {31,41,59,26,41,58}

- 31,**41**,59,26,41,68 (The key is 41. This remains in place).
- 31,41,**59**,26,41,68 (The key is 59. This remains in place).
- 31,41,59,**26**,41,58 (The key is 26. This takes the position of the number 31).
- 26,31,41,49,**41**,58 (The key is 41. This takes the position of the number 59).
- 26,31,41,41,59,**58** (The key is 58. This takes the position of the number 59).
- **Result:** 26,31,41,41,58,59

2.1.2.

Rewrite the INSERTION-SORT procedure to sort into nonincreasing instead of nondecreasing order.

```
void insertSort(vector<int> &vec){
    for(int j = 1; j < vec.size(); j++){
        int key = vec[j];
        int i = j - 1;
        while(i >= 0 and vec[i] < key){
            swap(vec[i + 1], vec[i]);
            i--;
        }
    }
}
```

2.1.3.

Consider the **searching problem:**
Input: A sequence of 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 through the sequence, looking for v. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties.

```
for (int j = 0; i < A.size(); j++{
  if(v == A[j]) return j;
}
return NULL;
```

2.1.4.

Cosider the problem of adding two *n-bit* binary integers, stored in two *n-element* arrays A and B. The sum of the two integers should be stored in binaty form in an (n+1)-elemnt array C. State the problem formally and write pseudocode for adding the two integers.

```
vector<int> sumBin(vector<int> &A, vector<int> &B){
    int n = A.size();
    vector<int> C(n + 1);
    int l = 0;
    for(int i = n - 1; i >= 0; i--){
        if(!A[i] and !B[i]){
            C[i + 1]  = l;
            l = 0;
        }
        else if(A[i] and B[i]){
            C[i + 1] = l;
            if(!l) l = !l;
        }
        else{
            if((A[i] or B[i]) and l){
                C[i + 1] = 0;
                l = 1;
            }
            else C[i + 1] = 1;
        }
    }
    C[0] = l;
    return C;
}
```

2.2. Analyzing algorithms

2.2.1.

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

$$
\begin{gathered}
n^3/1000 - 100n^2 - 100n + 3 < n^3 - 100n^3 - 100n^3 + 3n^3 \\
n^3/1000 - 100n^2 - 100n + 3 < 203n^3 \\
\Theta(n^3)
\end{gathered}
$$

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 $A[1]$. The find the second smallest element of a , and exchange it with $A[2]$. Continue in this manner for the first $n-1$ element of A. Write pseudocode for this algorithm, which is know as **selection sort**.

- **What lop invariant does this algorithm mainthain?**

- **Why does it need to run only the first $n-1$ rather than for all n elements?**
  Because allways the last element is in the correct site.

- **Give the best-case and worst-case running times of selection sort int Θ-notation.**

SELECTION SORT	cost	times
n = A.length	$C1$	1
for $i = 0$ to $n-1$	$C2$	n
$menor$ = NUMERO INFINITAMENTE GRANDE	$C3$	$n-1$
$index = 0$	$C4$	$n-1$
for $j = i$ to n	$C5$	$\sum_{i=1}^{n+1} i$
if $A[j] < menor$ then	$C6$	$\sum_{i=0}^{n} i$
$menor = A[j]$	$C7$	$\sum_{i=0}^{n} i$
$index = j$	$C8$	$\sum_{i=0}^{n} i$
swap($A[index]$,$A[i]$)	$C9$	$n-1$

$T(n) = C1 + nC2 + C3(n-1) + C4(n-1) + C5(\sum_{i=1}^{n+1} i)$
$+ C6(\sum_{i=0}^{n} i) + C7(\sum_{i=0}^{n} i) + C8(\sum_{i=0}^{n} i) + C9(n-1)$

$T(n) = C1 + nC2 + nC3 + nC4 + nC9 - C3 - C4 - C9 + C5(\frac{n^2+3n+2}{2})$
$+ C6(\frac{n^2+n}{2}) + C7(\frac{n^2+n}{2}) + C8(\frac{n^2+n}{2})$

$T(n) = n^2(\frac{C6+C7+C8+C5}{2}) + n(C2 + C3 + C4 + C9 + \frac{3C5+C6+C7+C8}{2})$
$+ C1 - C3 - C4 - C9$