# two bit add algorithm

Consider the two bit add problem:

- Input:
  Two $n-$bit binary integers, stored in two $n-$element array $A$ and $B$.
- Output:
  The sum of the two integers should be stored in binary form in an $(n+1)-$element array $C$.

You can find the implementation **here** or go to the next url: https://github.com/DiegoMendezMedina/C_Algorithms/blob/master/Search/linear_searching/implementations/searching_problem.c.

## Pseudocode

**two_bit_add**

1. $aux = 0$
2. **for** $i = 1$ **to** $n$
3. $\quad C[i] = binary\_add(A[i], B[i], aux)$
4. $C[n+1] = aux$

**binary_add**

1. $sum = a + b + aux$
2. **if** $sum >= 2$
3. $\quad aux = 1$
4. **else** $aux = 0$
5. **if** $sum\%2 == 0$
6. $\quad$ **return** 0
7. **return** 1

## Proof

**Loop invariant:**
At the start of each iteration of the **for** loop of lines 1-3, v was not found on the previous **i** values. **if** v == A[i], **i** is returned and the **for** loop breaks. Otherwhise at the end of the loop **'N'** is returned.

**Initialization:**
When i = 0, since i = 0 there are no previous **i** values.**if** A[0] = v; then **i** is return and the **for** loop breaks.

**Maintenance:**
There's another iteration which means that for all the previous value of **i** v was found. If for the current value of **i** happens that A[i] = v then **i** is returned and the **for** loop breaks.

**Termination:**
When the loop finishes **i** had browsed all the possible positions of the array and v was not found then **'N'** is returned.