Value:	0	1	2	3	4	5	6	7	8	9
Frequency:	0	1	1	2	0	0	1	2	1	1

A = { 6, 1, 8, 3, 7, 2, 3, 9, 7}

S = { 1, 2, 3, 3, 6, 7, 7, 8, 9 }

Time complexity:

O(R) + O(n) + O(n+R)
= O(2n + 2R)
= O(n + R)
= O(max(n, R))

F[i] = 0 -> O(1)
F[i] > 0 -> O(F[i])

R * O(1) + Sum_i { F[i] } = O(n+R)

R * O(1) + O(n) = O(n+R)

```
CountSort( int[] A, int n, int a, int b )
   // Input: An array of integers of length n,
             where the values are in [a, b]
   // Output: The sorted version of A

   R = b-a+1
   int F[R]
   for i = 0 to (R-1)
       F[i] = 0

   for i = 0 to (n-1)
       tmp = A[i] - a
       F[tmp] = F[tmp] + 1

   int S[n]
   k = 0
   for i = 0 to (R-1)
        freq = F[i]
        for j = 1 to freq
            S[k] = i + a
            k = k+1
   return S
```