Counting Sort

For Counting Sort, the input is n integers in some known range $[0, k-1]$.

```
def CountingSort:
    C = [0]*k, B = [0] * n
    for j in [1, n]:
        C[A[j]] += 1    # C = distribution
    for i in [1, k]:
        C[i] = C[i-1] + C[i] # C = Cumulative distribution
    for j in [n, 1]:
        B[C[A[j]]] = A[j]
        C[A[j]]--
    return B
```

The algorithm calculates a cumulative frequency distribution for the numbers in the input array in the first 2 passes, so that `C[i]` is the number of elements less than or equal to `i`. In the final loop, the algorithm puts the last element from `A` into `B` based on its position within the cumulative distribution, then recurses (iteratively) on `A[:-1]`.

Counting sort has $\Theta(n+k)$ runtime and space complexity and is a stable sort. For obvious reasons, it should only be preferred if $k << n \lg n$.