Counting Sort

- → campersion Algo has O(nlogn)
- → but counting sort not comparison base sort but under some condition
- → count sort has → O(n)

For example

	1	2	3	4	5	6	7	8
A =	2	5	3	0	2	3	0	3
	j							

Value lie A between 0 and K

	1	2	3	4	5	6	7	8
Output B :								

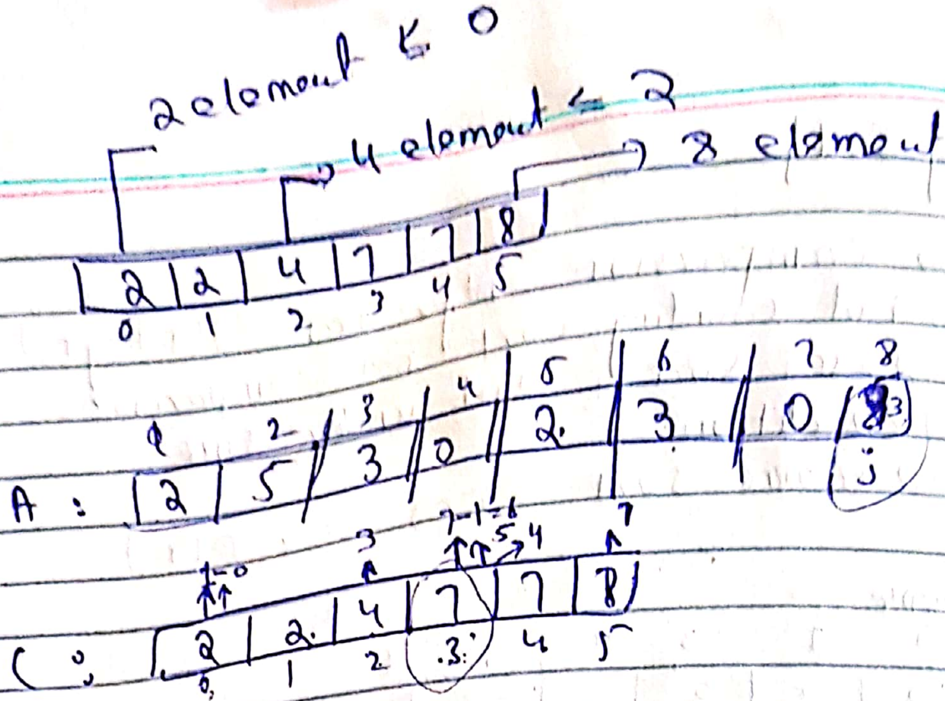
	1	2	3	4	5	
Auxiliary C :	0	0	0	0	0	→ K

C –	1+1		1+1	1+1+1		1	→ c[A[j]] = c[A[j]]+1
	0	1	2	3	4	5	

C →	2	0	2	3	0	1
	0	1	2	3	4	5 = k

Here we count variable frequency in Array C

2	2	4	7	7	8	→ cumulative sum
0	1	2	3	4	5	



~~C[A[j]]~~

Output Array

B[C[A[j]]] = A[j]

	1	2	3	4	5	6	7	8
B :								

A k last A[j] ko C Iden samaj kar C may swap karni Hai' or C wali value decreerse by one karna Hai

	1	2	3	4	5	6	7	8
B =	0	0	2	2	3	3	3	5

↓ get sorted value

$A[8] = 3 \rightarrow C[3] = 7 \rightarrow B[7] = A[8]$

$A[j] = x \quad C[j] = x \quad B[j] = A[j]$

Counting : space and time Complexity Sort

```
Count Sort (A, B, k)
    let C[0...k] be new Array
  For i = 0 to k                              } → O(k)
        C[i]  0
  For j = 1 to A.longht                       } - O(n)
        C[A[j]] = C[A[j]] + 1
  For i = 1 to k                              } O(k)
        C[i] = C[i] + C[i-1]
  For j = A.lenght down onto 1
        B[C[A[j]]] = A[j]                     } - O(n)
        C[A[j]] = C[A[j]] - 1
```

O(n + k) → O(n)

↳ If k itself O(K)