Counting Sort

Team JC

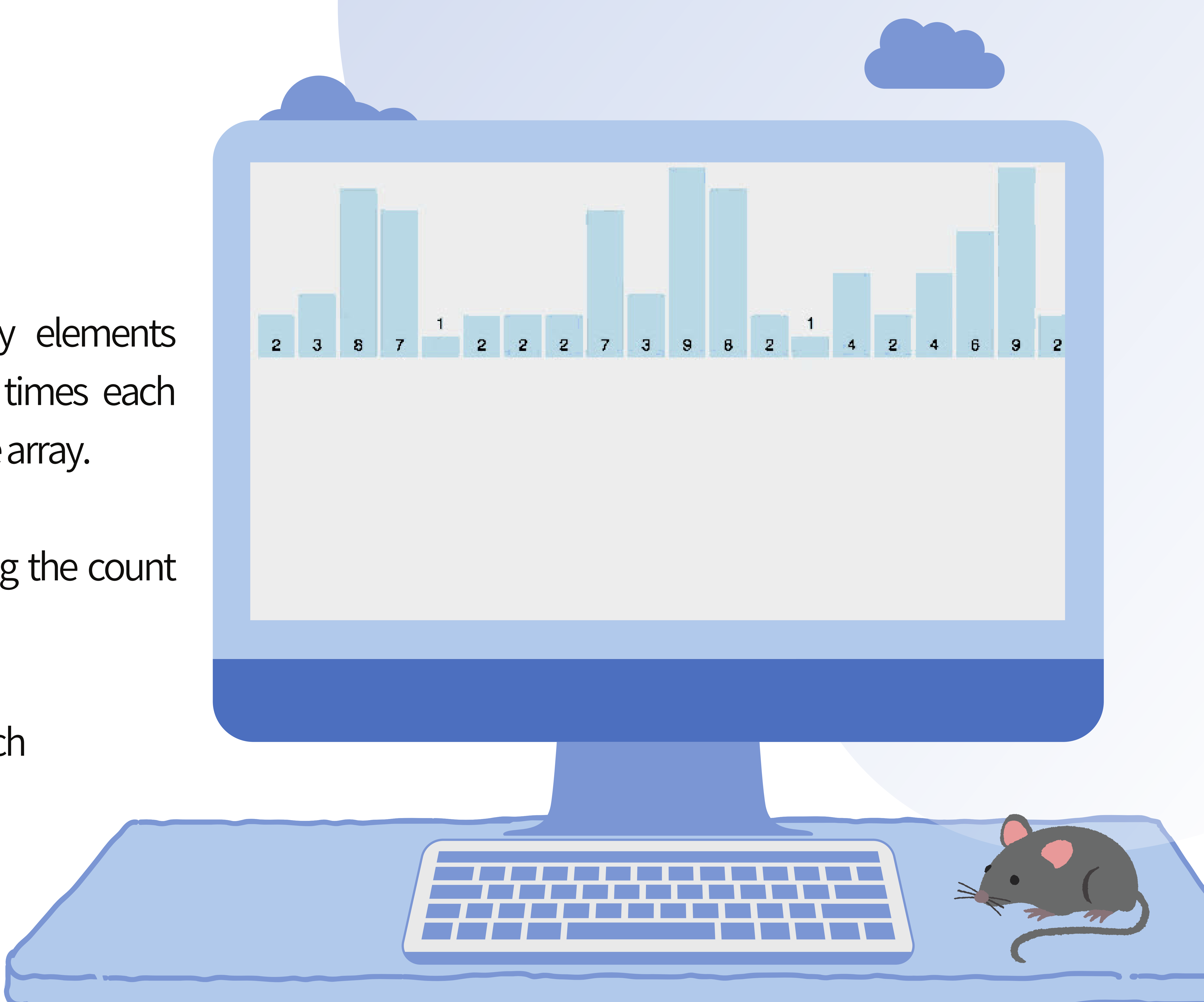
What is Counting Sort?

What is Counting Sort?

Harold Seward first developed Counting Sort (and Radix Sort) in 1954 .

Definition

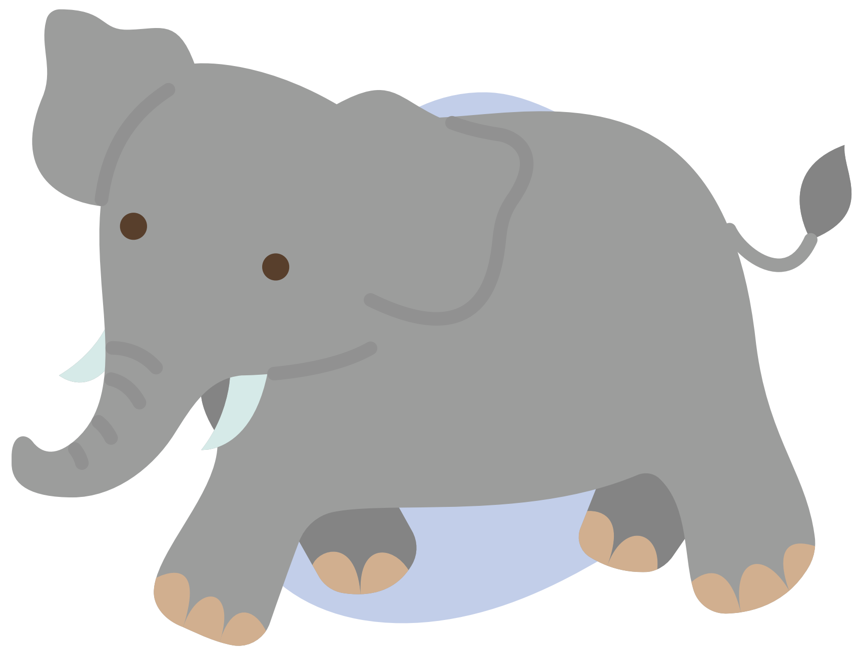
- Counting sort arranges array elements according to the number of times each distinct element appears in the array.
- It is accomplished by mapping the count as an index
- It is a non-comparison approach



When is it used?

When is it used?

Effective when used on small-scale numbers

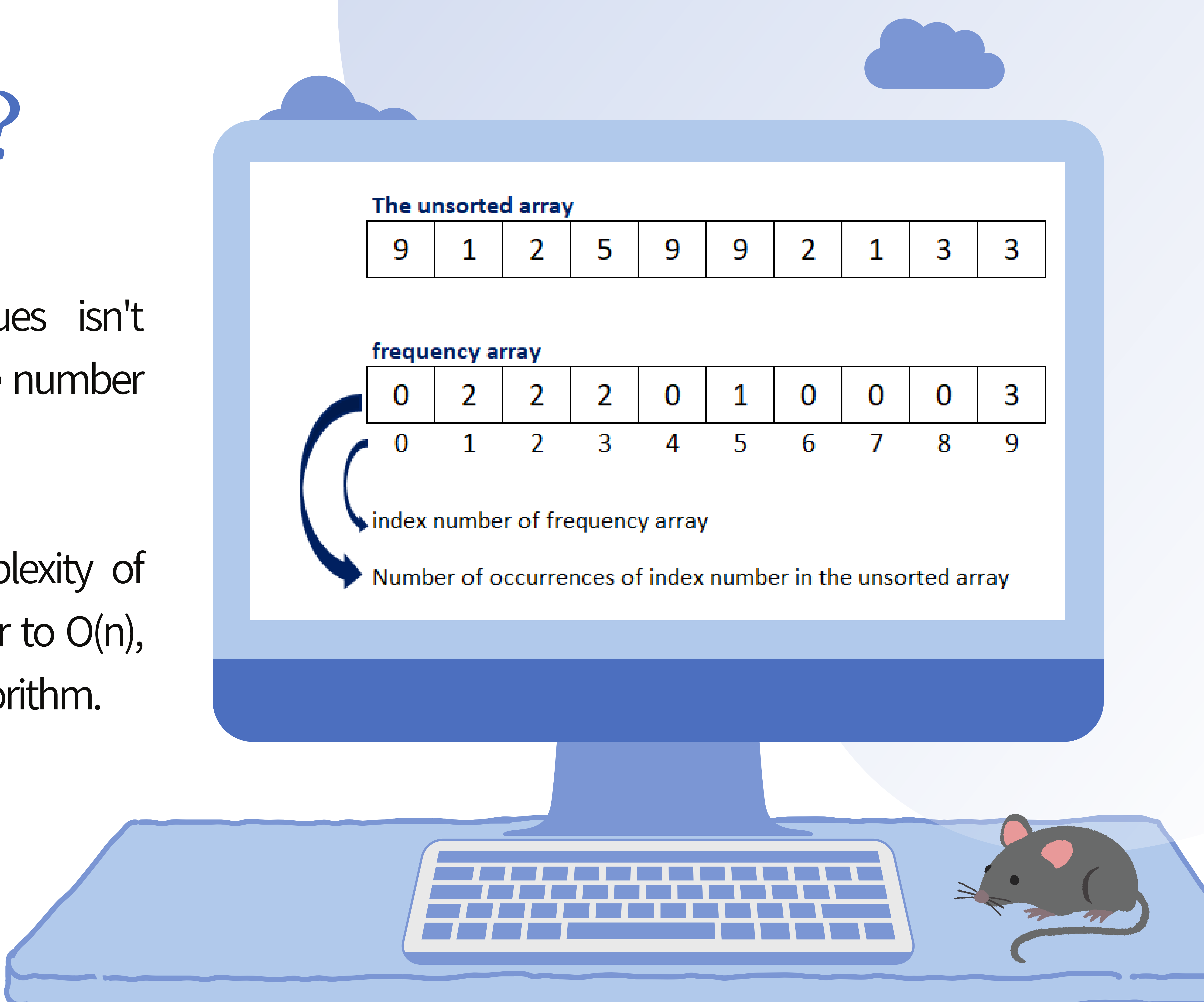


Not so when used in greater scale differences

Used when?

- The range of input values isn't signficantly greater than the number of values to be sorted.

- In that scenario, the complexity of counting sort is much closer to O(n), making it a linear sorting algorithm.



The unsorted array

9	1	2	5	9	9	2	1	3	3

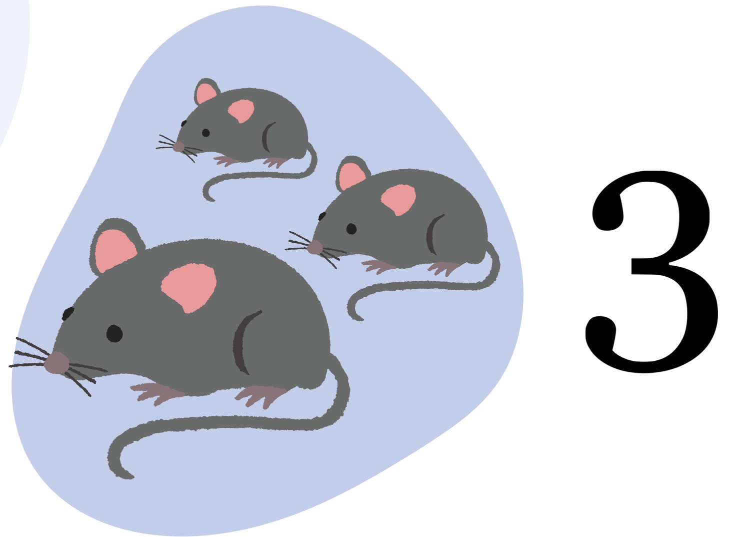
frequency array

0	2	2	2	0	1	0	0	0	3
0	1	2	3	4	5	6	7	8	9

index number of frequency array

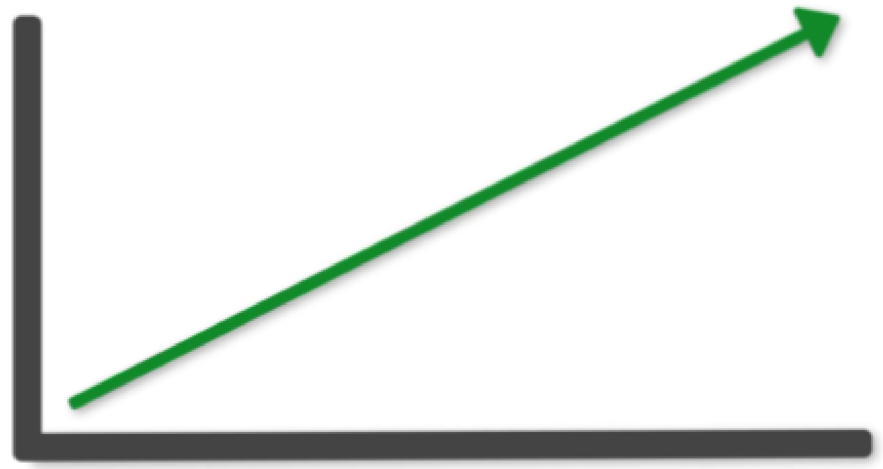
Number of occurrences of index number in the unsorted array

Conditions for Counting Sort



Must know the range of the values

Can't accept negative integers (original code)



Linear complexity only

Complexity: Time and Space

Time:

Best - O(n)
Worst - O(k)
Average - O (n+k)

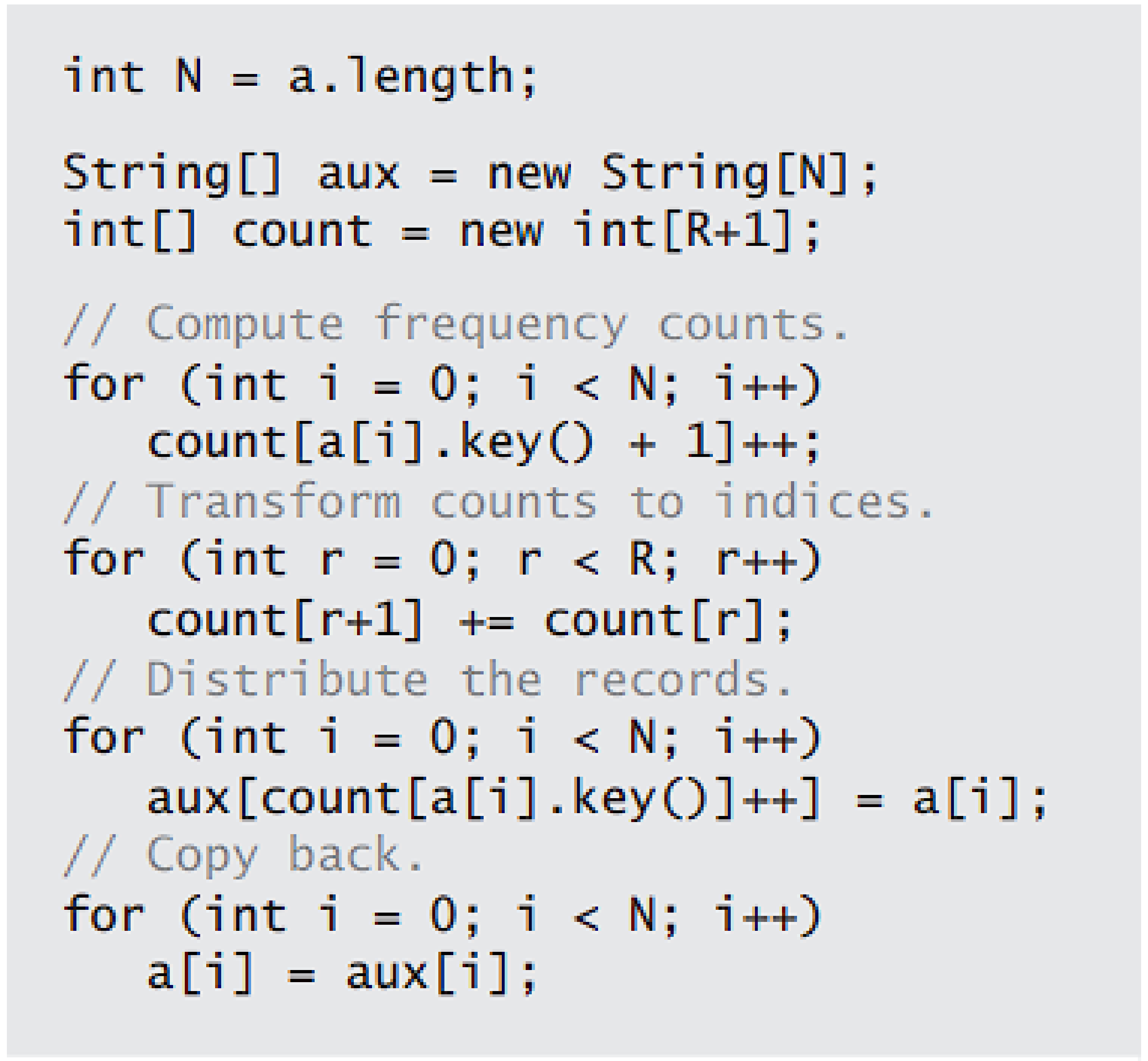
Space - O(k)

Where:

n - number of elements
k - range of the input

Original Version

from Algorithms 4th Edition by Robert Sedgewick, Kevin Wayne



```java
int N = a.length;

String[] aux = new String[N];
int[] count = new int[R+1];

// Compute frequency counts.
for (int i = 0; i < N; i++)
   count[a[i].key() + 1]++;
// Transform counts to indices.
for (int r = 0; r < R; r++)
   count[r+1] += count[r];
// Distribute the records.
for (int i = 0; i < N; i++)
   aux[count[a[i].key()]++] = a[i];
// Copy back.
for (int i = 0; i < N; i++)
   a[i] = aux[i];
```

Key-indexed counting (a[].key is an int in [0, R).

Variations

Simplified Counting Sort

Generalized Counting Sort

Simplified Counting Sort

```
int findMax(int array[], int SIZE) {
    int max = array[0], x;

    for (x = 1; x < SIZE; x++) {
        if (array[x] > max)
            max = array[x];
    }
}
```

```
int* countingSort(int arr[], int SIZE) {
     int i, j, k;
     int maxValue = findMax(arr, SIZE);
     int count[maxValue+1]={0};
     int newArr = (int)malloc(SIZE * sizeof(int));

     for(i = 0; i < SIZE; i++){
         count[arr[i]]++;
     }

     for(i = 0, k = 0; i < maxValue+1; i++) {
         for(j = 0; j < count[i]; j++, k++) {
             newArr[k] = i;
         }
     }

     return newArr;
}
```

Take Note

- Works only on primitive data types
- Input should only be non-negatives
- To allow negative values, apply offset technique

Generalized Counting Sort

```
void countSort(int arr[], int SIZE) {
    int i;
    int maxValue = findMax(arr, SIZE);
    int count[maxValue + 1]={0};
    int newArr = (int)malloc(SIZE * sizeof(int));

    for (i = 0; arr[i]; ++i)
        ++count[arr[i]];

    for (i = 1; i <= maxValue + 1; ++i)
        count[i] += count[i - 1];

    for (i = sizeof(arr)-1; i>=0; --i) {
        newArr[count[arr[i]]-1] = arr[i];
        --count[arr[i]];
    }

    for (i = 0; arr[i]; ++i)
        arr[i] = newArr[i];
}
```

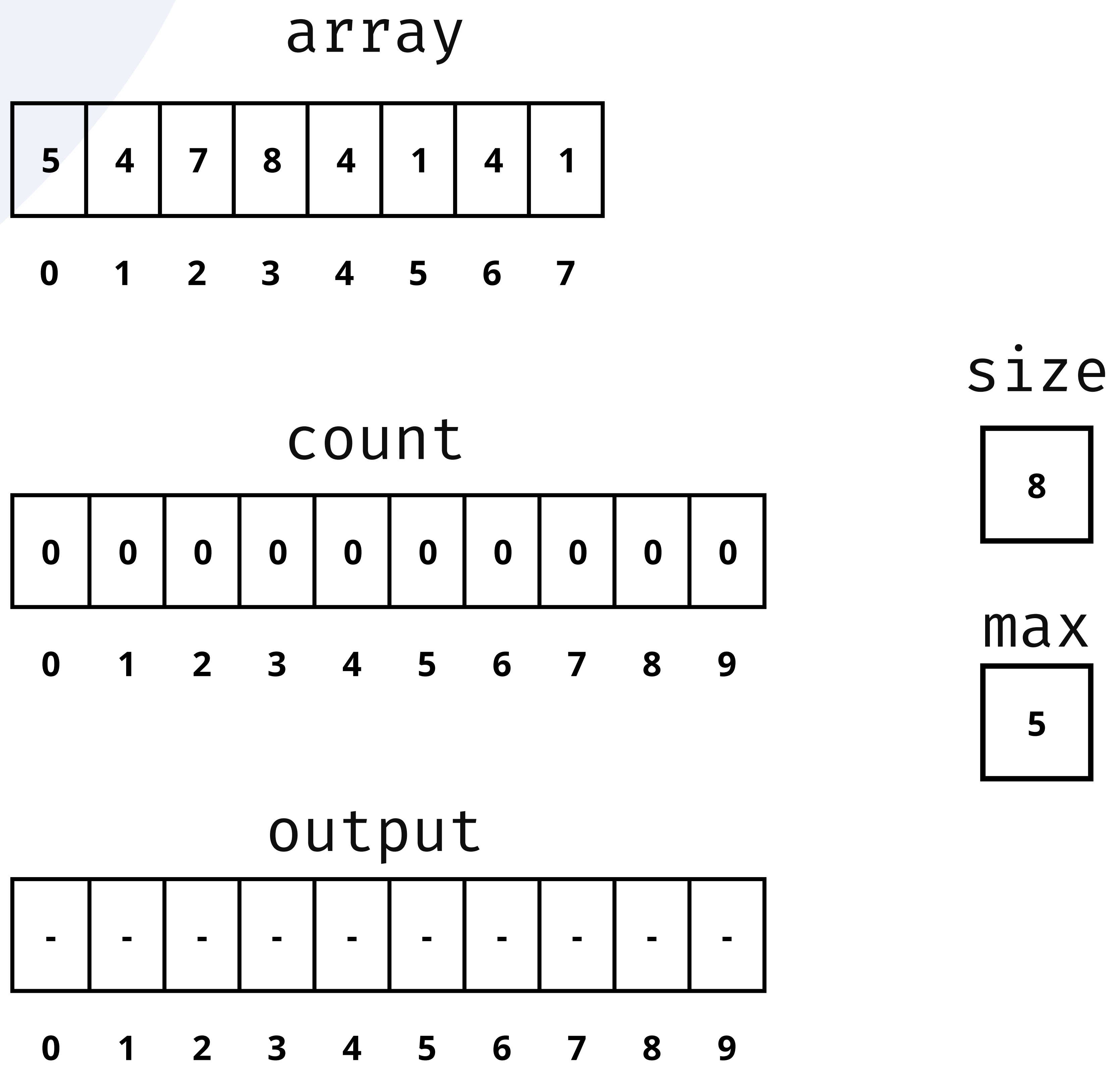
Take Note

- Works on arrays of objects
- Objects have keys determined by a certain hash function or key method
- Stable algorithm

Sample Simulation

```
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

https://www.programiz.com/dsa/counting-sort

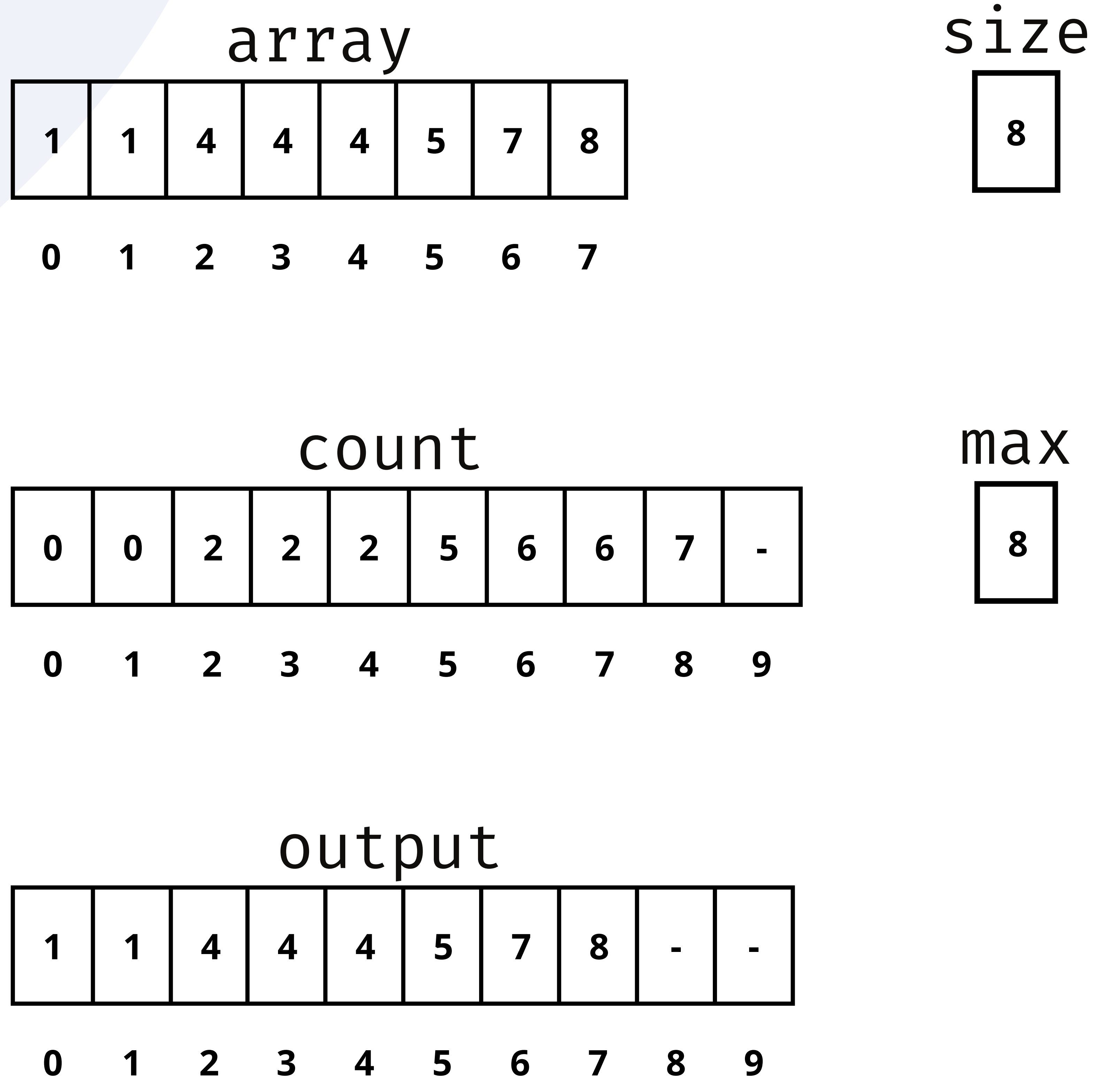


```
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

https://www.programiz.com/dsa/counting-sort

```
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

https://www.programiz.com/dsa/counting-sort



Internet Code vs. Streamline code

Internet Code

- Only accepts positive integers as array input
- Uses loop for assigning zero to array count

Streamline Code

- Accepts negative integer as input in the array variable
- Assignment of zero for count array no longer needs loop

```c
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

```c
void countingSort(int array[], int size) {
 int x, min = 0, max = 0;
 int *newArr = (int*)malloc(size * sizeof(int));
 for(x = 0; x < size; x++) {
   if(min > array[x])
     min = array[x];
 }
 for(x=0; x< size; x++){
   if(array[x] > max)
     max = arr[x];
 }
 min *= -1;
 int newRange = (max+1) + min;
 int* count = (int*)calloc(newRange, sizeof(int));
 for(x = 0; x < size; x++){
   count[array[x] + min]++;
 }
 for(x = 1; x < newRange; x++){
   count[x] += count[x - 1];
 }
 for(x = 0; x < size; x++) {
   newArr[count[array[x] + min] - 1] = array[x];
   count[array[x] + min]--;
 }
 for(x=0; x < size; x++){
  array[x] = newArr[x];
 }
 free(count);
}
```

```c
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

```c
void countingSort(int array[], int size) {
 int x, min = 0, max = 0;
 int *newArr = (int*)malloc(size * sizeof(int));
 for(x = 0; x < size; x++) {
   if(min > array[x])
     min = array[x];
 }
 for(x=0; x< size; x++){
   if(array[x] > max)
     max = arr[x];
 }
 min *= -1;
 int newRange = (max+1) + min;
 int* count = (int*)calloc(newRange, sizeof(int));
 for(x = 0; x < size; x++){
   count[array[x] + min]++;
 }
 for(x = 1; x < newRange; x++){
   count[x] += count[x - 1];
 }
 for(x = 0; x < size; x++) {
   newArr[count[array[x] + min] - 1] = array[x];
   count[array[x] + min]--;
 }
 for(x=0; x < size; x++){
  array[x] = newArr[x];
 }
 free(count);
}
```

```c
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

```c
void countingSort(int array[], int size) {
 int x, min = 0, max = 0;
 int *newArr = (int*)malloc(size * sizeof(int));
 for(x = 0; x < size; x++) {
   if(min > array[x])
     min = array[x];
 }
 for(x=0; x< size; x++){
   if(array[x] > max)
     max = arr[x];
 }
 min *= -1;
 int newRange = (max+1) + min;
 int* count = (int*)calloc(newRange, sizeof(int));
 for(x = 0; x < size; x++){
   count[array[x] + min]++;
 }
 for(x = 1; x < newRange; x++){
   count[x] += count[x - 1];
 }
 for(x = 0; x < size; x++) {
   newArr[count[array[x] + min] - 1] = array[x];
   count[array[x] + min]--;
 }
 for(x=0; x < size; x++){
  array[x] = newArr[x];
 }
 free(count);
}
```

```
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

```
void countingSort(int array[], int size) {
 int x, min = 0, max = 0;
 int *newArr = (int*)malloc(size * sizeof(int));
 for(x = 0; x < size; x++) {
   if(min > array[x])
     min = array[x];
 }
 for(x=0; x< size; x++){
   if(array[x] > max)
     max = arr[x];
 }
 min *= -1;
 int newRange = (max+1) + min;
 int* count = (int*)calloc(newRange, sizeof(int));
 for(x = 0; x < size; x++){
   count[array[x] + min]++;
 }
 for(x = 1; x < newRange; x++){
   count[x] += count[x - 1];
 }
 for(x = 0; x < size; x++) {
   newArr[count[array[x] + min] - 1] = array[x];
   count[array[x] + min]--;
 }
 for(x=0; x < size; x++){
  array[x] = newArr[x];
 }
 free(count);
}
```

```c
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

```c
void countingSort(int array[], int size) {
 int x, min = 0, max = 0;
 int *newArr = (int*)malloc(size * sizeof(int));
 for(x = 0; x < size; x++) {
   if(min > array[x])
     min = array[x];
 }
 for(x=0; x< size; x++){
   if(array[x] > max)
     max = arr[x];
 }
 min *= -1;
 int newRange = (max+1) + min;
 int* count = (int*)calloc(newRange, sizeof(int));
 for(x = 0; x < size; x++){
   count[array[x] + min]++;
 }
 for(x = 1; x < newRange; x++){
   count[x] += count[x - 1];
 }
 for(x = 0; x < size; x++) {
   newArr[count[array[x] + min] - 1] = array[x];
   count[array[x] + min]--;
 }
 for(x=0; x < size; x++){
  array[x] = newArr[x];
 }
 free(count);
}
```

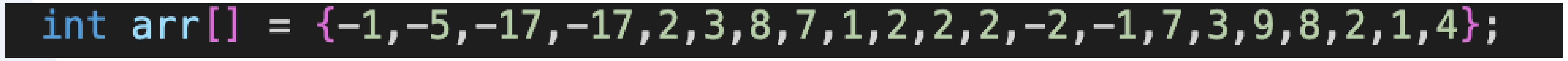
```c
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

```c
void countingSort(int array[], int size) {
 int x, min = 0, max = 0;
 int *newArr = (int*)malloc(size * sizeof(int));
 for(x = 0; x < size; x++) {
   if(min > array[x])
     min = array[x];
 }
 for(x=0; x< size; x++){
   if(array[x] > max)
     max = arr[x];
 }
 min *= -1;
 int newRange = (max+1) + min;
 int* count = (int*)calloc(newRange, sizeof(int));
 for(x = 0; x < size; x++){
   count[array[x] + min]++;
 }
 for(x = 1; x < newRange; x++){
   count[x] += count[x - 1];
 }
 for(x = 0; x < size; x++) {
   newArr[count[array[x] + min] - 1] = array[x];
   count[array[x] + min]--;
 }
 for(x=0; x < size; x++){
  array[x] = newArr[x];
 }
 free(count);
}
```

```c
void countingSort(int array[], int size) {
  int output[10];
  int max = array[0];
  for (int i = 1; i < size; i++) {
    if (array[i] > max)
      max = array[i];
  }
  int count[10];
  for (int i = 0; i <= max; ++i) {
    count[i] = 0;
  }
  for (int i = 0; i < size; i++) {
    count[array[i]]++;
  }
  for (int i = 1; i <= max; i++) {
    count[i] += count[i - 1];
  }
  for (int i = size - 1; i >= 0; i--) {
    output[count[array[i]] - 1] = array[i];
    count[array[i]]--;
  }
  for (int i = 0; i < size; i++) {
    array[i] = output[i];
  }
}
```

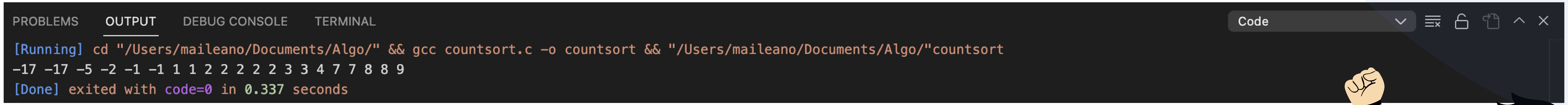
```c
void countingSort(int array[], int size) {
 int x, min = 0, max = 0;
 int *newArr = (int*)malloc(size * sizeof(int));
 for(x = 0; x < size; x++) {
   if(min > array[x])
     min = array[x];
 }
 for(x=0; x< size; x++){
   if(array[x] > max)
     max = arr[x];
 }
 min *= -1;
 int newRange = (max+1) + min;
 int* count = (int*)calloc(newRange, sizeof(int));
 for(x = 0; x < size; x++){
   count[array[x] + min]++;
 }
 for(x = 1; x < newRange; x++){
   count[x] += count[x - 1];
 }
 for(x = 0; x < size; x++) {
   newArr[count[array[x] + min] - 1] = array[x];
   count[array[x] + min]--;
 }
 for(x=0; x < size; x++){
  array[x] = newArr[x];
 }
 free(count);
}
```

INPUT

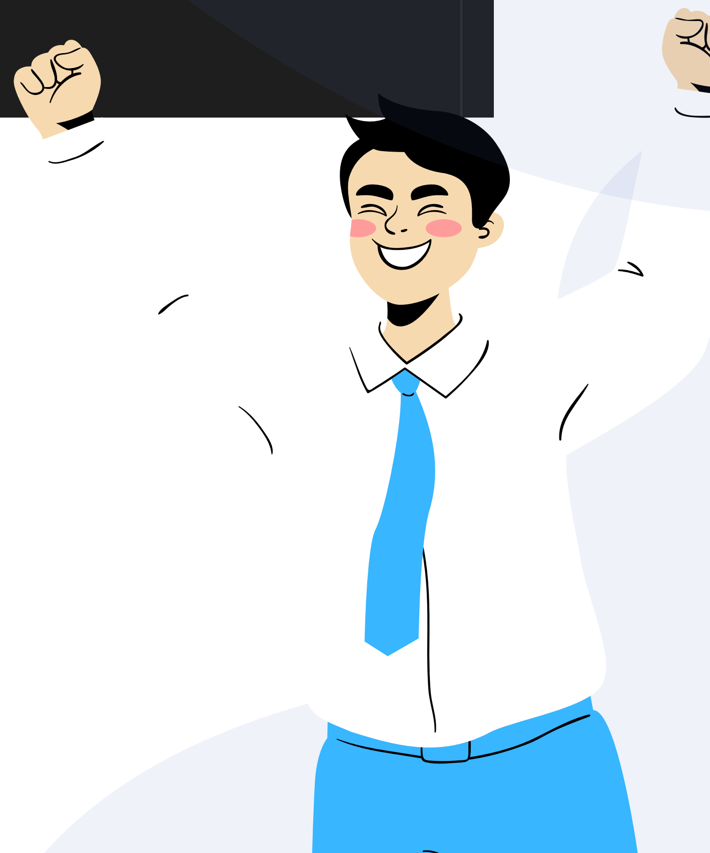


```
int arr[] = {-1,-5,-17,-17,2,3,8,7,1,2,2,2,-2,-1,7,3,9,8,2,1,4};
```

OUTPUT



```
[Running] cd "/Users/maileano/Documents/Algo/" && gcc countsort.c -o countsort && "/Users/maileano/Documents/Algo/"countsort
-17 -17 -5 -2 -1 -1 1 1 2 2 2 2 2 3 3 4 7 7 8 8 9
[Done] exited with code=0 in 0.337 seconds
```



REFERENCES

Counting Sort Algorithm. Interview Cake. (n.d.). Retrieved August 23, 2022, from https://www.interviewcake.com/concept/python/counting-sort

Counting sort algorithm. Programiz. (n.d.). Retrieved August 23, 2022, from https://www.programiz.com/dsa/counting-sort

Joshi, V. (2017, July 17). Counting linearly with counting sort. Medium. Retrieved August 23, 2022, from https://medium.com/basecs/counting-linearly-with-counting-sort-cd8516ae09b3

Sedgewick, R., & Wayne, K. D. (2017). Algorithms. W. Ross MacDonald School Resource Services Library.

Woltmann, S. (2022, July 19). Counting sort – algorithm, source code, Time Complexity. HappyCoders.eu. Retrieved August 23, 2022, from https://www.happycoders.eu/algorithms/counting-sort/