Counting Sort

Josh Wilcox (jw14g24@soton.ac.uk)

March 4, 2025

Contents

1 Counting Sort Algorithm

```c
/*
 * Counting Sort Algorithm in C
 * A: input array
 * B: output array
 * k: range of the input values (0 to k-1)
 */
void CountingSort(int A[], int B[], int k, int n) {
    // Create a count array to store the count of each unique object
    int C[k+1];

    // Initialize count array with all zeros
    for (int i = 0; i <= k; i++) {
        C[i] = 0;
    }

    // Store the count of each element in the input array A
    // For each element A[j], increment the count at index A[j] in C
    for (int j = 0; j < n; j++) {
        C[A[j]]++;
    }

    // Modify the count array such that each element at index i
    // contains the sum of previous counts. This step transforms
    // the count array into a prefix sum array which will help in
    // placing the elements into their correct positions in the output array.
    for (int i = 1; i <= k; i++) {
        C[i] += C[i-1];
    }

    // Build the output array B
    // Iterate over the input array A from the end to the beginning to maintain
    // the stability of the sorting algorithm (i.e., to ensure that equal elements
    // appear in the same order in the sorted array as they do in the input array).
    for (int j = n-1; j >= 0; j--) {
        // Place the element A[j] at the correct position in the output array B
        // The correct position is determined by the count array C
        B[C[A[j]]-1] = A[j];
        // Decrement the count for the element A[j] in the count array C
        C[A[j]]--;
    }
}

// Example usage
int main() {
    int A[] = {4, 2, 2, 8, 3, 3, 1};
    int n = sizeof(A) / sizeof(A[0]);
```

```
47      int k = 8; // Range of input values is 0 to 8
48      int B[n];
49
50      CountingSort(A, B, k, n);
51
52      // Print the sorted output array
53      for (int i = 0; i < n; i++) {
54          printf("%d ", B[i]);
55      }
56
57      return 0;
58  }
```

2 Steps

2.1 Intialisation

- For array A, find the range k of values
- Make a new array C, with the same number of elements as k
  - Initialise this array with all zeros

2.2 Counting

- Go through each element in A
- Every time you come accross an element that is in C, increment the count in C

2.3 Cumulative

- Make C cumulative by adding the previous cell in turn

2.4 Sorting

- Iterate over the input array A from the last element to the first element
- For each element $A[j]$:
  - Use the value of $A[j]$ to find the correct position in B using the count array C
  - Place $A[j]$ in the correct position in B
  - Decrease the count of $A[j]$ in C by 1

3 Properties

- **No Comparisons performed**
- Time complexity: $\Theta(n + k)$
- If $n >> k$, Counting Sort is $\Theta(n)$
- Not to be used if $k > n$