**TITLE**

SUBTITLE

Text

**COUNTING SORT ALGORITHM**

Basic Information

|  |  |
| --- | --- |
| **Who** | * Harold Seward |
| **When** | * 1954 |
| **What** | * Originally, counting sort is an algorithm for sorting a collection of objects according to keys that are small positive integers. * It is often used as a subroutine in radix sort, another sorting algorithm, which can handle larger keys more efficiently. * A non-comparison approach |
| **Used** | * Suitable for direct use in situations where the variation in keys is not significantly greater than the number of items |
| **Conditions** | * Non-negative Integer * Known range of values * Linear Complexity only * Stable algorithm |
| **Input** | * Collection of **n** items, maximum value of **k** |
| **Output** | * Array of ordered elements according to their keys |

Variations

|  |  |
| --- | --- |
| **Simplified Counting Sort**   * Works only on primitive data types * Input should be non-negative * Offset technique is applied to allow negative values | **Generalized Counting Sort**   * Works on arrays of objects * Objects have keys determined by a certain hash function or key method * Stable algorithm |
| **Negative Numbers Counting Sort**   * Offset technique is used to find min max values of the input array * Occupies less memory if min range is < 0 | **Parallelized Counting Sort**   * Divides input into partitions and sorts them simultaneously * Stability is no longer guaranteed |

**Simplified Counting Sort**

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;

}

int findMax(int array[], int SIZE) {

int max = array[0], x;

for (x = 1; x < SIZE; x++) {

if (array[x] > max)

max = array[x];

}

}

**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];

}

**Time and Space Complexity**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Best** | **Average** | **Worst** |
| **Time** | O (n) | O (n + k) | O (k) |
| **Space** |  |  | O (k) |

|  |  |
| --- | --- |
| **Simplified Space** | O (k) |
| **Generalized Space** | O (n + k) |

**Counting Sort Application**