Counting Sort

Counting sort is a sorting technique based on keys between a specific range. It works by counting the number of objects having distinct key values (kind of hashing). Then doing some arithmetic to calculate the position of each object in the output sequence.
Let us understand it with the help of an example.

For simplicity, consider the data in the range 0 to 9.
Input data: 1, 4, 1, 2, 7, 5, 2

1. Take a count array to store the count of each unique object.
   Index: 0 1 2 3 4 5 6 7 8 9
   Count: 0 2 2 0 1 1 0 1 0 0
2. Modify the count array such that each element at each index stores the sum of previous counts.
   Index: 0 1 2 3 4 5 6 7 8 9
   Count: 0 2 4 4 5 6 6 7 7 7

The modified count array indicates the position of each object in the output sequence.

3. Output each object from the input sequence followed by decreasing its count by 1.
   Process the input data: 1, 4, 1, 2, 7, 5, 2. Position of 1 is 2.
   Put data 1 at index 2 in output. Decrease count by 1 to place next data 1 at an index 1 smaller than this index.

```python
def countingSort(arr):
    mini = min([arr[i][0] for i in range(len(arr))])
    maxi = max([arr[i][0] for i in range(len(arr))])
    freqArr = [0]*(maxi-mini+1)
    # Recording the frequency of each elements
    for ele in arr:
        idx = ele[0]-mini
        freqArr[idx] = freqArr[idx]+1
    # Writing the prefix Sum in freq arr
    for i in range(1,len(freqArr)):
        freqArr[i] = freqArr[i-1]+freqArr[i]

#     Declaring a new array of size equal to arr
    ans = [0]*len(arr)
    for i in range(len(arr)-1,-1,-1):
        idxOfFreqArray = arr[i][0]-mini
        valInFreqArr = freqArr[idxOfFreqArray]
        idxInResultingArray = valInFreqArr-1
        ans[idxInResultingArray] = arr[i]
        freqArr[idxOfFreqArray] = freqArr[idxOfFreqArray]-1

    return ans


arr = [(-5,1),(5,1),(9,1),(6,1),(3,1),(1,1),(0,1),(8,1),(2,1),(4,1),(6,2),
(8,2),(5,2),(4,2),(3,2),(9,2),(7,1)]
print(countingSort(arr))
```

Basic Counting Sort:

```
def countingSort(arr):
    mini = min([arr[i] for i in range(len(arr))])
    maxi = max([arr[i] for i in range(len(arr))])
    freqArr = [0]*(maxi-mini+1)
    # Recording the frequency of each elements
    for ele in arr:
        idx = ele-mini
        freqArr[idx] = freqArr[idx]+1
    # Writing the prefix Sum in freq arr
    for i in range(1,len(freqArr)):
        freqArr[i] = freqArr[i-1]+freqArr[i]

#     Declaring a new array of size equal to arr
    ans = [0]*len(arr)
    for i in range(len(arr)-1,-1,-1):
        idxOfFreqArray = arr[i]-mini
        valInFreqArr = freqArr[idxOfFreqArray]
        idxInResultingArray = valInFreqArr-1
        ans[idxInResultingArray] = arr[i]
        freqArr[idxOfFreqArray] = freqArr[idxOfFreqArray]-1

    return ans


arr = [-5,8,6,0,4,1,8,7,3,2,5,4,6]
print(countingSort(arr))
```

My code is just proving that counting sort is stable. For an element is arr:

arr[i]===> (a,b)

a ==> Number to be sorted

b ===> Position or rank of the element.

When rank isnt an issue, don't just consider it. eliminate b.

It handles negative values as well.

Time Complexity: O(n+k) where n is the number of elements in input array and k is the range of input.
Auxiliary Space: O(n+k)

Points to be noted:
1. Counting sort is efficient if the range of input data is not significantly greater than the number of objects to be sorted. Consider the situation where the input sequence is between range 1 to 10K and the data is 10, 5, 10K, 5K.
2. It is not a comparison based sorting. **It running time complexity is O(n) with space proportional to the range of data. **
3. **It is often used as a sub-routine to another sorting algorithm like radix sort. **
4. Counting sort uses a partial hashing to count the occurrence of the data object in O(1).
5. Counting sort can be extended to work for negative inputs also.(My code works for negative numbers also). **Trick is maxi-mini**

Q.Which sorting algorithms is most efficient to sort string consisting of ASCII characters?

Counting sort algorithm is efficient when range of data to be sorted is fixed. In the above question, the range is from 0 to 255(ASCII range). Counting sort uses an extra constant space proportional to range of data.