Sorting Array of Strings

Depending on certain circumstances, we may want to sort a given array of strings into lexicographically increasing order or into an order in which the string with the lowest length appears first.
We could make a sorting function and pass the function a flag telling it which comparison strategy to use, but that would mean that whenever we invented a new comparison strategy, we would have to define a new code or flag value and rewrite the sorting function.
But, if we observe that the final ordering depends entirely on the behaviour of the comparison function, a better implementation would be if we write our sorting function to accept a pointer to the function which we want it to use to compare each pair of strings.
Making it sort in a different order, according to a different comparison strategy, will then not require rewriting sorting at all, but instead will just involve passing it a pointer to a different comparison function.

Given an array of strings, you need to implement a **string_sort** function which sorts the strings according to the following different criterion.

In this problem, you need to implement the function :

```
void string_sort(char **arr,const int cnt, int (*cmp_func)(const char* a, const char* b)){

}
```

To this function, the arguments passed are:

- an array of strings : **`arr`**
- length of string array: **`count`**
- pointer to the string comparison function: **`cmp_func`**

You also need to implement the four string comparison functions:

1. **`int lexiographic_sort(const char*, const char*)`** to sort the strings in lexicographically non-decreasing order.
2. **`int lexiographic_sort_reverse(const char*, const char*)`** to sort the strings in lexicographically non-increasing order.
3. **`int sort_by_number_of_distinct_characters(const char*, const char*)`** to sort the strings in non-decreasing order of the number of distinct characters present in them.If two strings have the same number of distinct characters present in them, then the lexicographically smaller string should appear first.
4. **`int sort_by_length(const char*, const char*)`** to sort the strings in non-decreasing order of their lengths.If two strings have the same length, then the lexicographically smaller string should appear first.

Input Format

You just need to complete the function **`string_sort`** and implement the four string comparison functions.

Constraints

The locked code-stub will check the logic of your code.

Sample Input 0

```
4
wkue
qoi
sbv
fekls
```

Sample Output 0

```
fekls
qoi
sbv
wkue

wkue
sbv
qoi
fekls

qoi
sbv
wkue
fekls

qoi
sbv
wkue
fekls
```