The ascending sort feature is similar to the sort function of python sort(a,key=lambda a:key(a))

Here we can sort the array according to the key feature.
If nothing is given in key it sorts normally

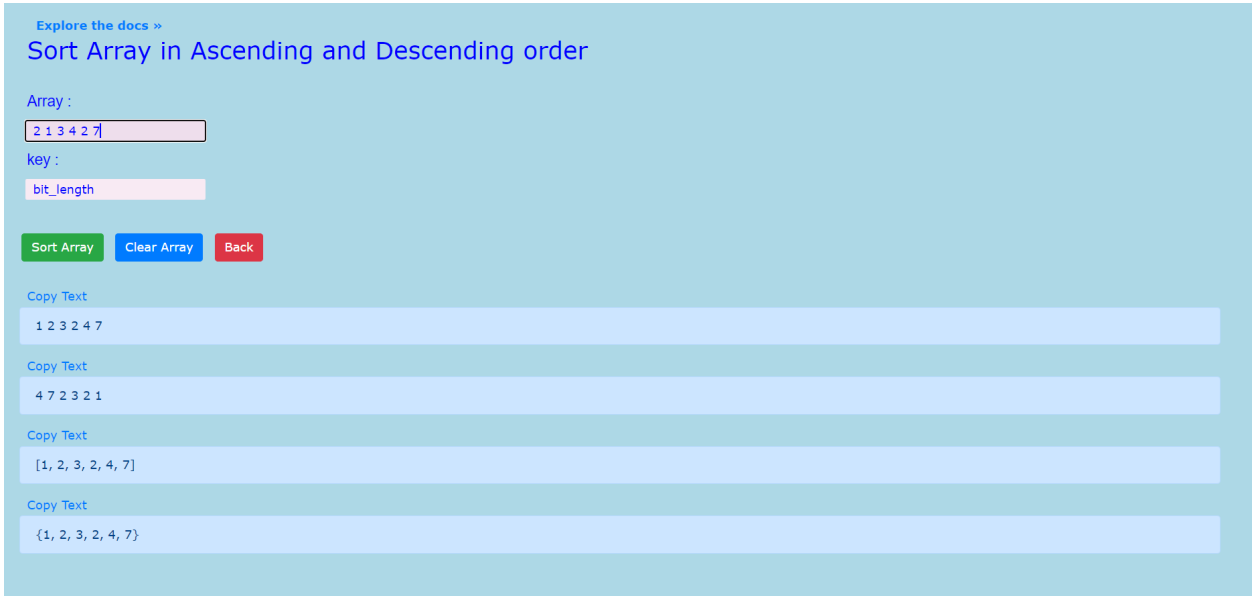
Note: When the key value is same elements are sorted according to the inplace sorting

In the key you can provide the following

1. bit_length
2. len
3. isPrime
4. `__builtin_popcountll`
5. `__builtin_parityll`
6. `__builtin_clzll`
7. `__builtin_ctzll`
8. is_sorted
9. sum
10. mex

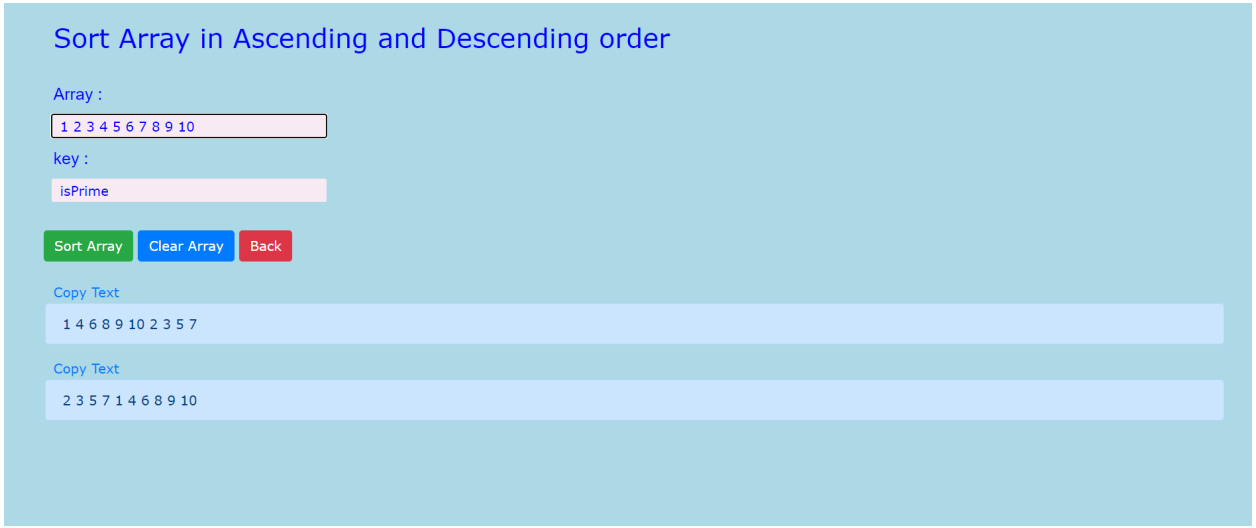
And many more…
Other key function can be made using custom javascript functions.

Examples

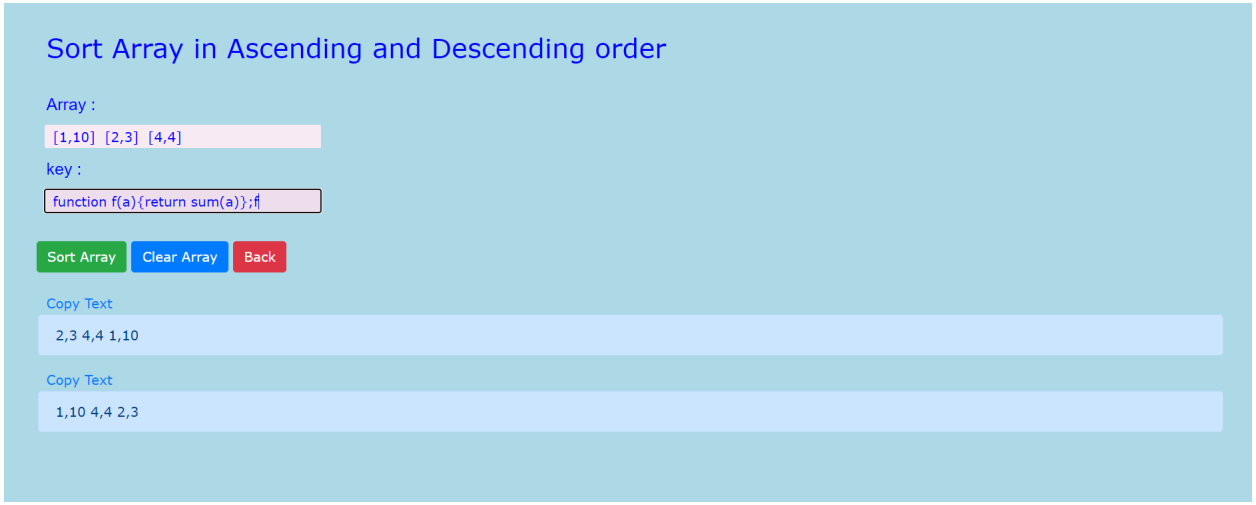



Sorted according to the bit_length
When the bit_length is same it sorts according to inplace sorting
So the order is
1 => bit length 1

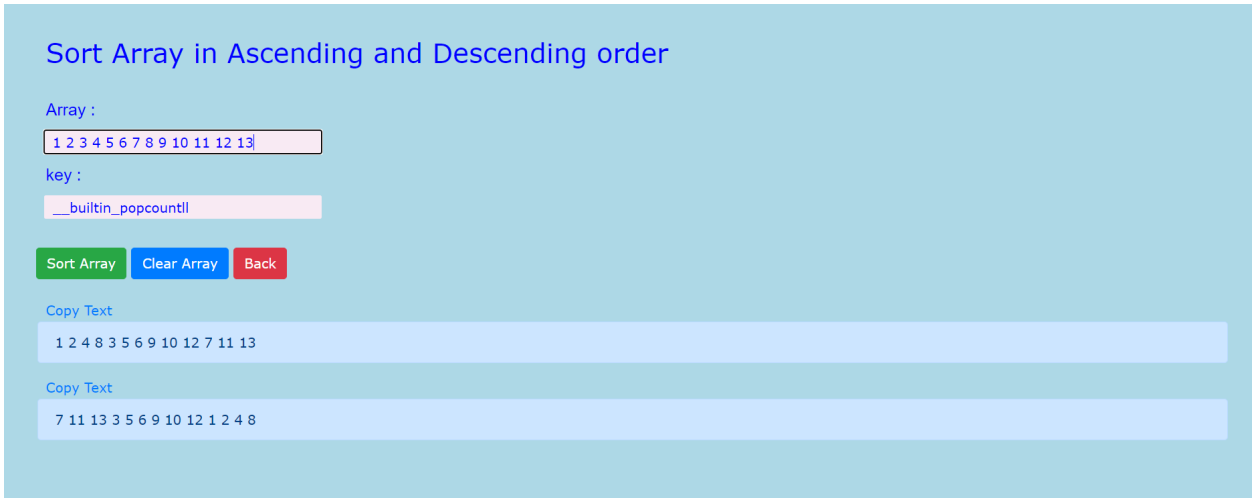
2 3 2 bit len 2
4 7 bit len 3



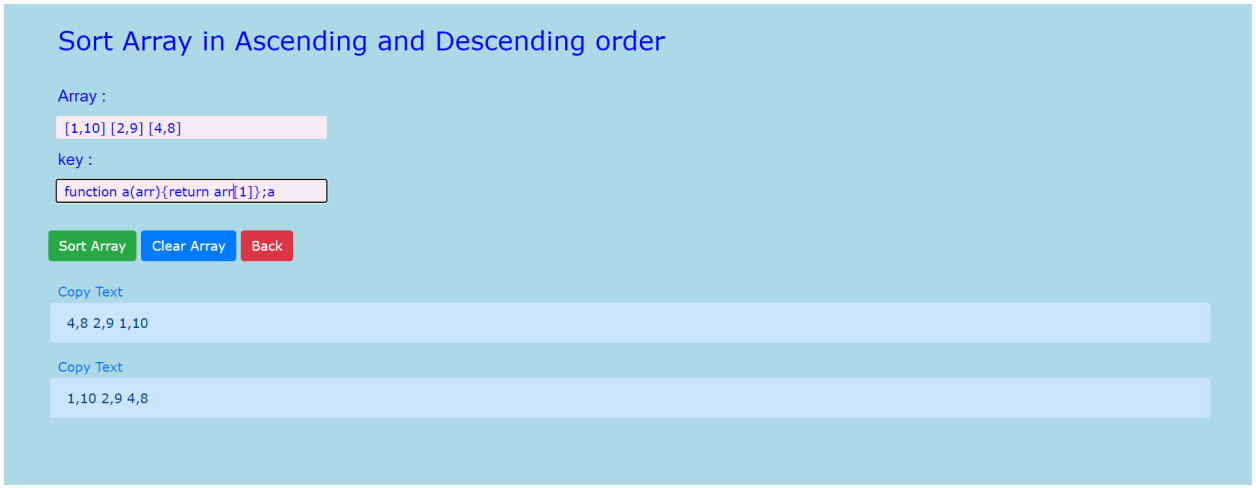

According to the primes
2 3 5 and 7 are primes so in ascending order they are at last and in descending order they are at first



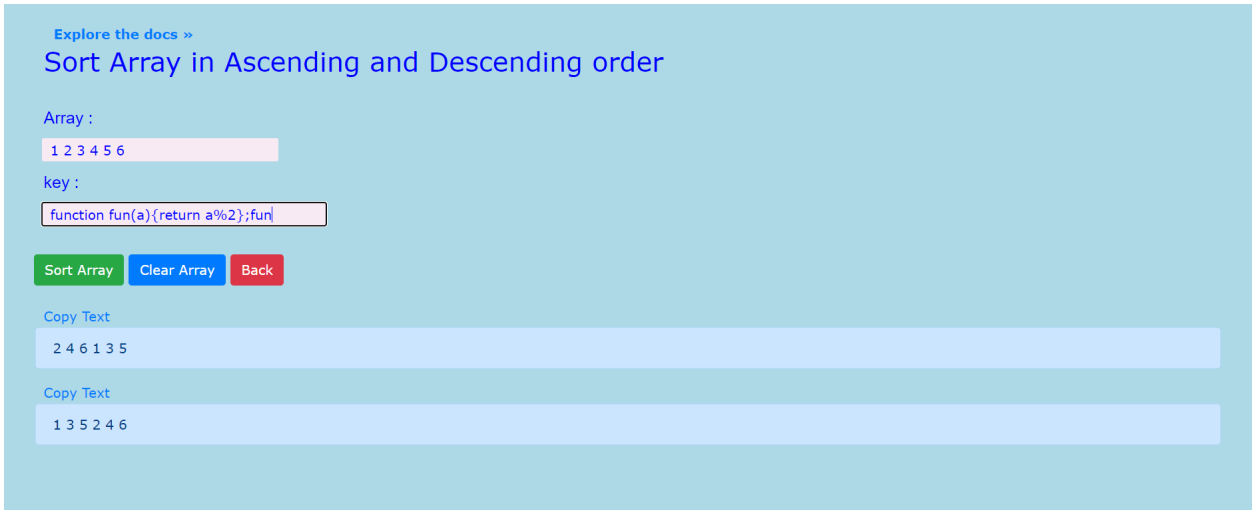

Sorting according to the sum of the array.
Sum of array [1,10] is maximum i.e. 11 so it is at last and sum of array [2,3] is minimum i.e. 2+3=5 so it is at first in ascending order.



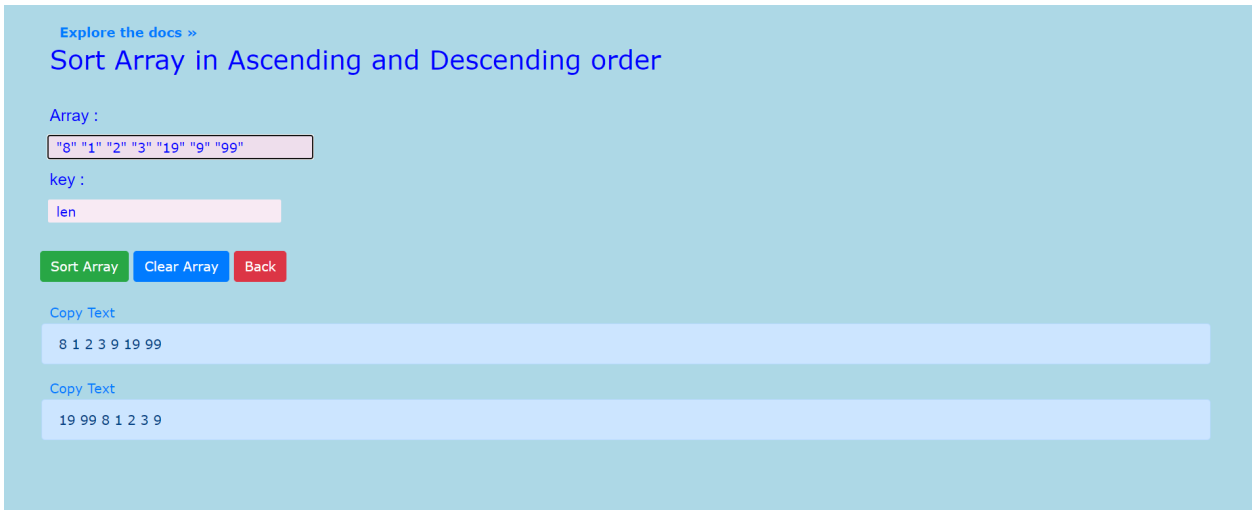

Sorting according to the number of ones in the binary representation of the number.
1 2 4 8 binary are “1” “10” “100” “1000” so the number of ones are minimum so they are at first



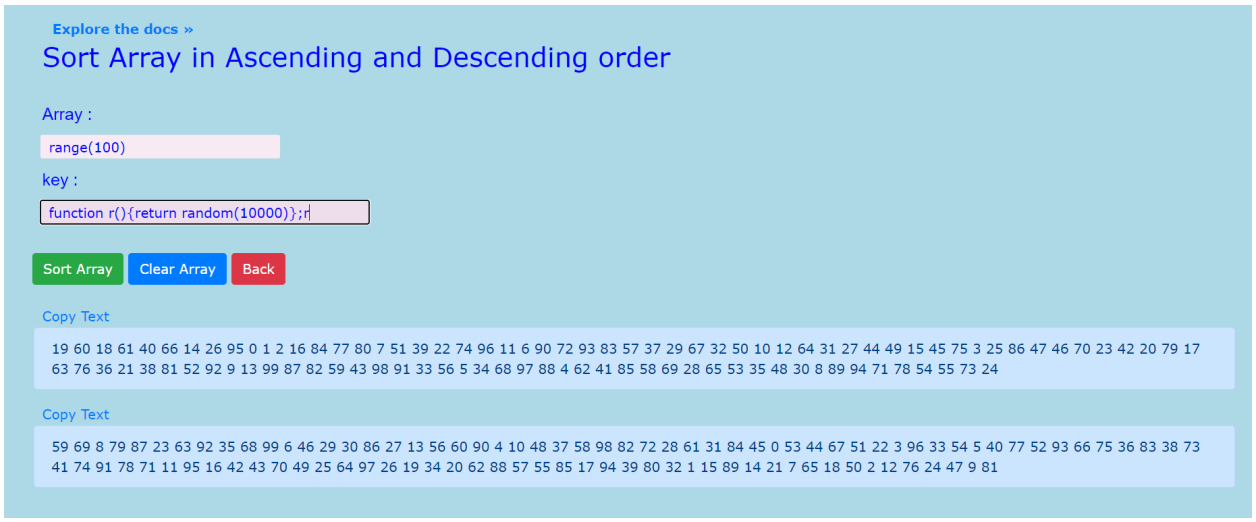

Sorting according to the second value of the array.
You can also customize the function as seen here. Simply make a function (javascript) and write its name separated by a semicolon.



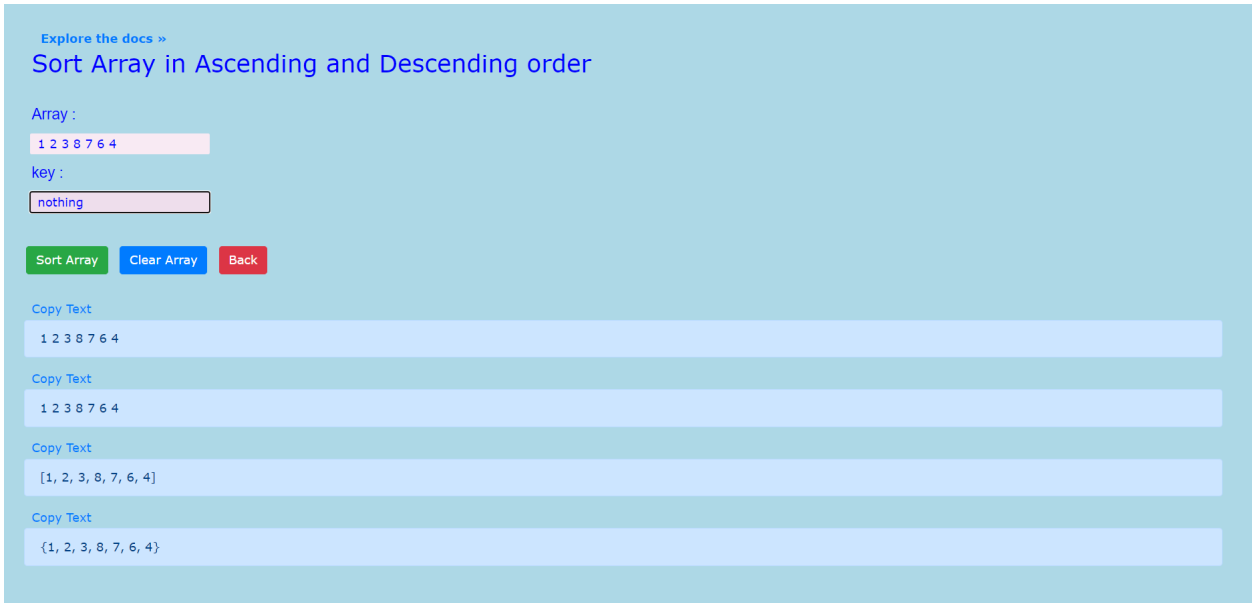

Sorting according to the a%2 custom function.




Sorting according to the length.
The strings should be entered in “” or ‘’ i.e. double quotes or single quotes.
The length of the string 19 and 99 are 2 so they are at last
8 comes before 1 because they have the same length and so they occur according to the index in input i.e. inplace sorting.




Shuffling the array randomly




Doing nothing