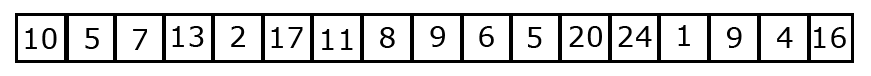
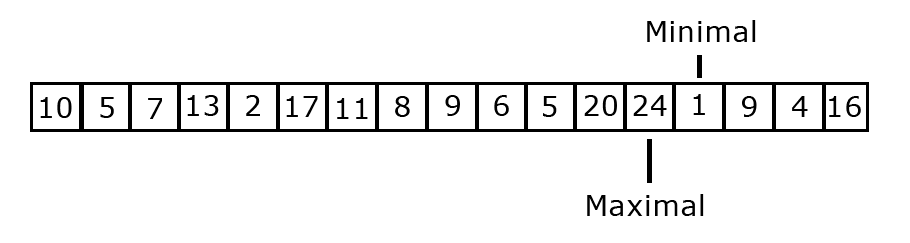
CzupajlosPercentageSort

1.How it works?

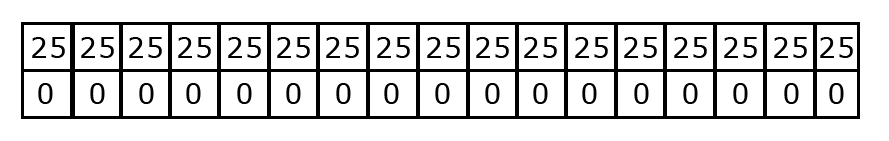
Lets say we have array of random numbers to sort.



First we will find the minimal and maximal number in that array.



Then we will create 2 dimensional array and fill it with placeholders in my case max+1 and second dimension with 0s.



Now we are going to set the proper values. The first element in the array is 10. So we are dividing it by the maximal object to get a percentage of a maximal object.

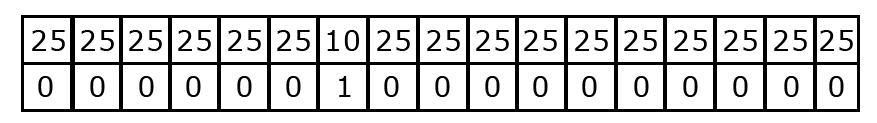
So:

10/24=41,6%

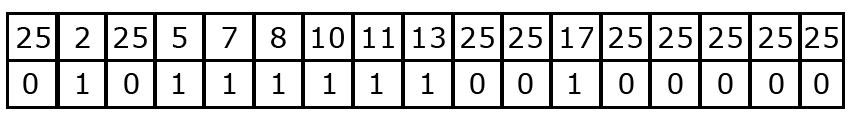
Now we are going to multiply that percentage of amount of the objects in the array minus one.

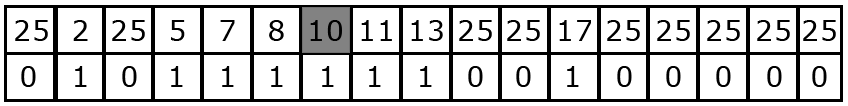
41,6% \* (17-1) = 41,6% \* 16 = 6,56

The integer is a position on which we are going to place that number. We will also and 1 in the second dimension. So it will look like this:

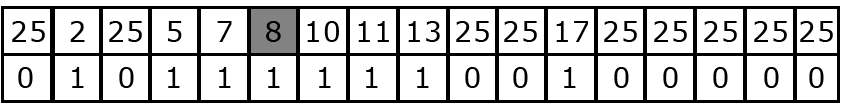


Same as this we will make for another numbers. Now our array look like this:

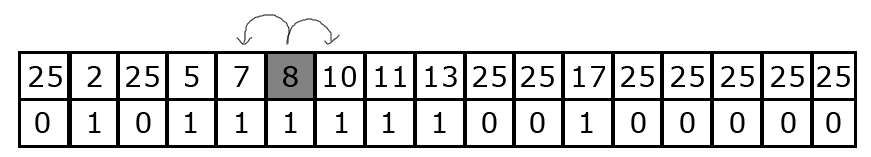
Next element is 9 but it want to be placed on position number 6. On that position is now standing 10 so we check where it could be placed.



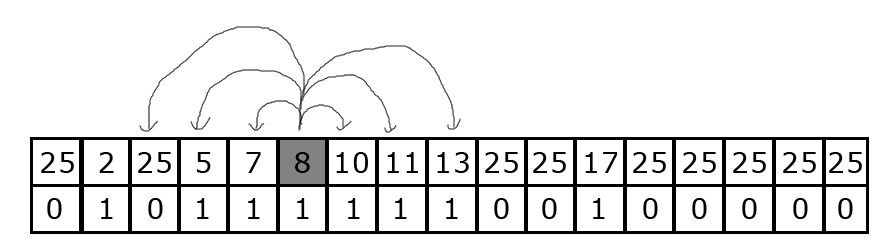
Now we are comparing if the item we want to put here is bigger, smaller or equal. Nine is smaller then ten so we go index under.

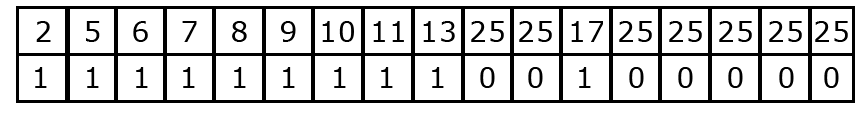


Now 8 is smaller then 9 so we stop doing that. The next thing we do is checking for nearby placeholders.

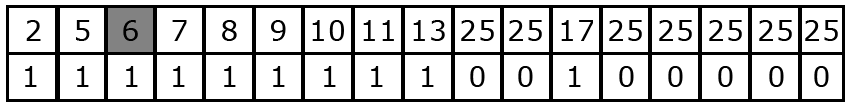


There is no placeholders so we need to move further.

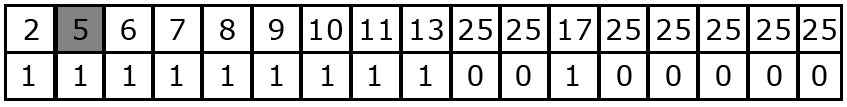


Now we see that there is placeholder on position 2 so we are going to push down these numbers between placeholder and position where our nine will be fitted. We also push down the second dimension and it will look like this. 

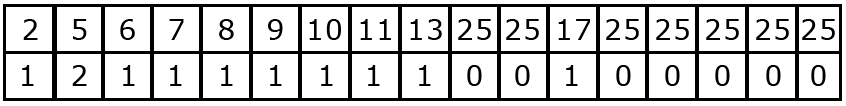
After putting here object with 6 we will get that looking array. Now we want to put another 5 so this is what we do:

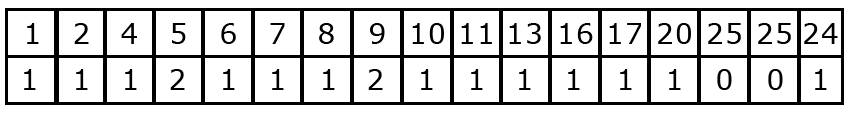


Firstly it want to be placed at position nr 3 but its already taken. So the comparison shows that new object is smaller then existing object and we will go down with the index.



Now we see that we have equal numbers so we just increment the second dimension. And it look like this.



After the every number we get something like this. 

Now we see that second dimension shows us how many of the same numbers exist in the array.

Last step to do is to decode that array so for each number in the array we add the number, number from the second dimension times. And we get sorted array.

