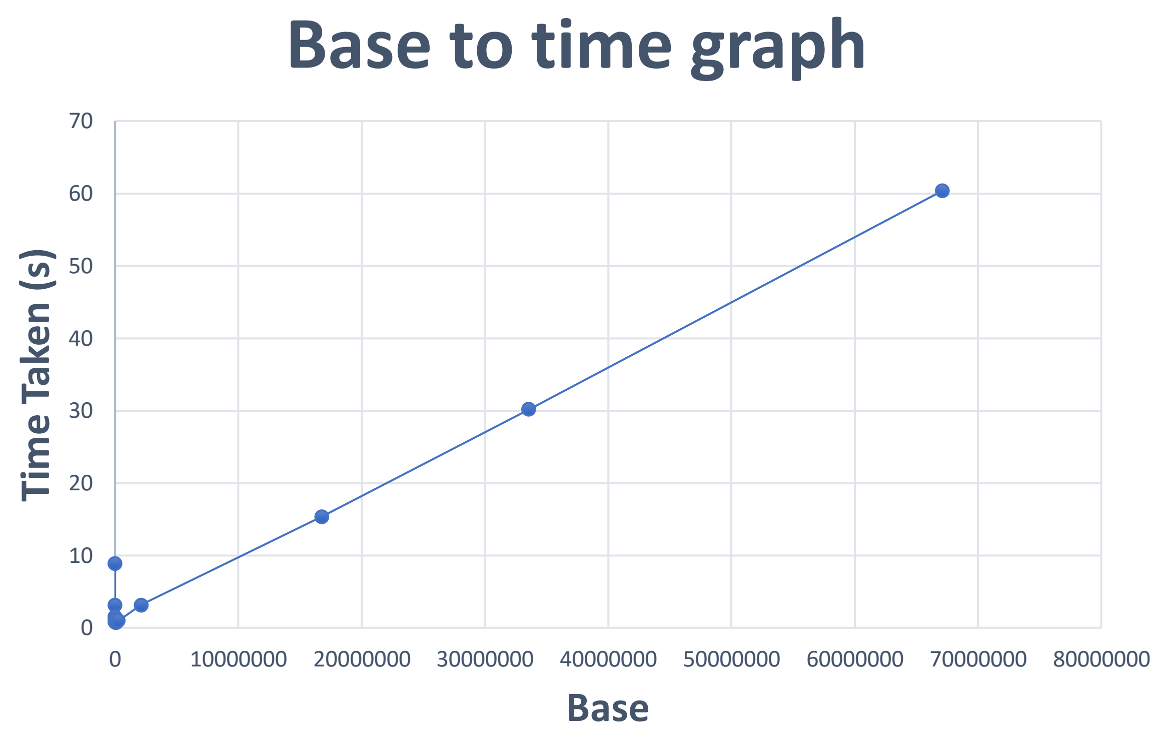
Base	**Time taken**
2	8.915262
8	3.07754
64	1.526977
256	1.133092
4096	0.853101
16384	0.725591
65536	0.735341
262144	0.974697
2097152	3.14977
16777216	15.34134
33554432	30.16528
67108864	60.3691

Reasons for the bases chosen:

I. The first 7 bases are used so we can observe the reduction in time from the smallest base to the base with the best time.

II. The 6th base's time value is very close to the best time (the 7th base), which suggests that having a base between the values 16284 and 65536 are most effective for having a low run time. These bases were chosen allows us to observe when the reduction in time is minor and barely visible.

III. The 8th to 12th base shows a drastic increase in time taken as the base continues to increase. These bases allow us to observe the drastic change in time taken. Furthermore, we also point out and estimate where the beginning of this increase starts.



Analysis

The time taken reduces with visible changes until the 6th base where the changes become minor. But the time taken will begin to drastically increase at some point.

Explanation on the time values obtained:

The time taken is directly influenced by the base used and the largest value of the given list. Radix sort uses counting sort as a subroutine and the number of times counting sort is performed is based on the base and the maximum number in the list to be sorted.

Arithmetic calculations used in radix sort:

a. First calculation

Value / Base ^ Power = Quotient + Remainder

(The remainder is not used and the power increments by 1 for each use)

b. Second calculation

(Quotient % Divisor) = Index

(Index is always a value less than the base)

The key component to remember are:

I. The larger the base, the larger the count array size
II. The number of times counting sort is performed is also the minimum power value required for the largest number in the list to have its quotient=0 in the first calculation shown above.

With this in mind, we can understand that having a base too small is bad as it would require more counting sort to be performed with a higher base power. Whereas, having a base too large is also bad as that would require a larger count array.

The first 7 bases as observed in the graph has a small count array size but requires performing counting sort more. As observed in the graph the change in time taken is not too drastic and slowly decreases.

When the base becomes larger the number of times counting sort is performed should decrease but this isn’t always the case. This is because as the base increases, so does the value difference between the maximum number in the list. There will come a point where increasing the base has only negative effects towards the performance.

Reflecting back to the graph shown, we can observe the relation stated above.