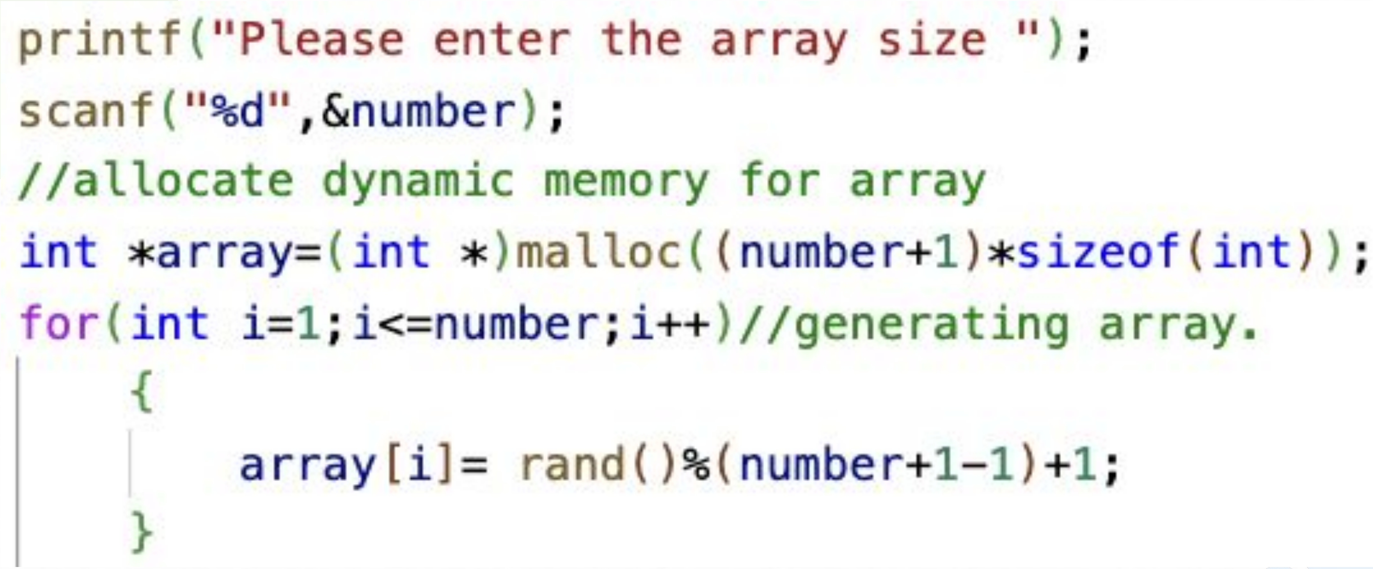
Team 5

Presented by:
Loke Yong Jian
Lim Yao Xian
Lim Zi Suan

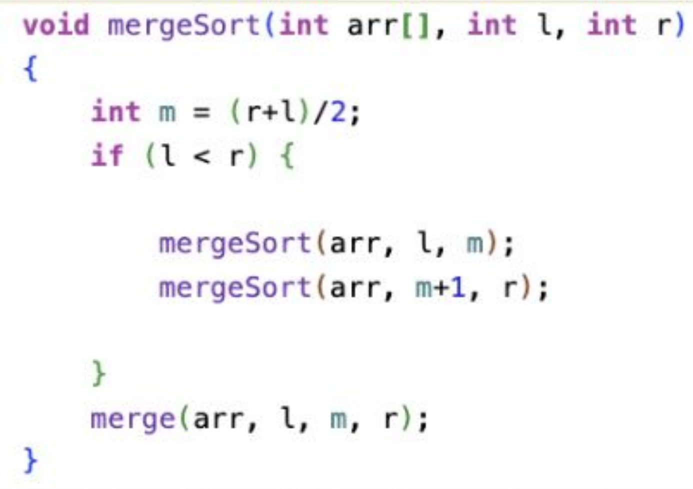
Table of Contents

Generate random Input data for array size of 1k to 10 million



```
printf("Please enter the array size ");
scanf("%d",&number);
//allocate dynamic memory for array
int *array=(int *)malloc((number+1)*sizeof(int));
for(int i=1;i<=number;i++)//generating array.
    {
        array[i]= rand()%(number+1-1)+1;
    }
```

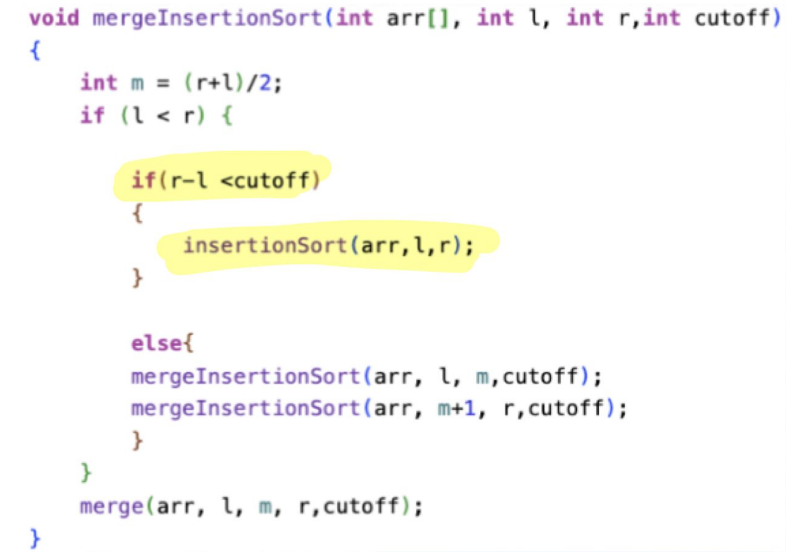
How was Hybrid Sort implemented



```
void mergeSort(int arr[], int l, int r)
{
    int m = (r+l)/2;
    if (l < r) {

        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

    }
    merge(arr, l, m, r);
}
```



```
void mergeInsertionSort(int arr[], int l, int r,int cutoff)
{
    int m = (r+l)/2;
    if (l < r) {

        if(r-l <cutoff)
        {
            insertionSort(arr,l,r);
        }

        else{
        mergeInsertionSort(arr, l, m,cutoff);
        mergeInsertionSort(arr, m+1, r,cutoff);
        }
    }
    merge(arr, l, m, r,cutoff);
}
```

Time Analysis

Time Analysis

- **Merge sort**

The worst case, best case, and the average case time complexity of merge sort is O(N*log(N))).

- **Insertion sort**

The best case time complexity of insertion sort is O(N), i.e the array is already sorted.

The worst case time complexity of insertion sort is O(N*N)

- **Insertion sort + Merge sort**

There will be N/S rows comparison.
Merge sort will sort after length S to Length N.
Mergesort -> O(N*log(N/S))
Insertion sort will sort up to length S. Hence N -> S
Insertion sort -> O(S*S*(N/S)) -> O(NS)

Best Case : O(N + N*log(N/S))

Worst Case: O(NS + N*log(N/S))

Overall Formula: O(N/S * no of insertion Sort + Nlog(N/S))

Derive from here

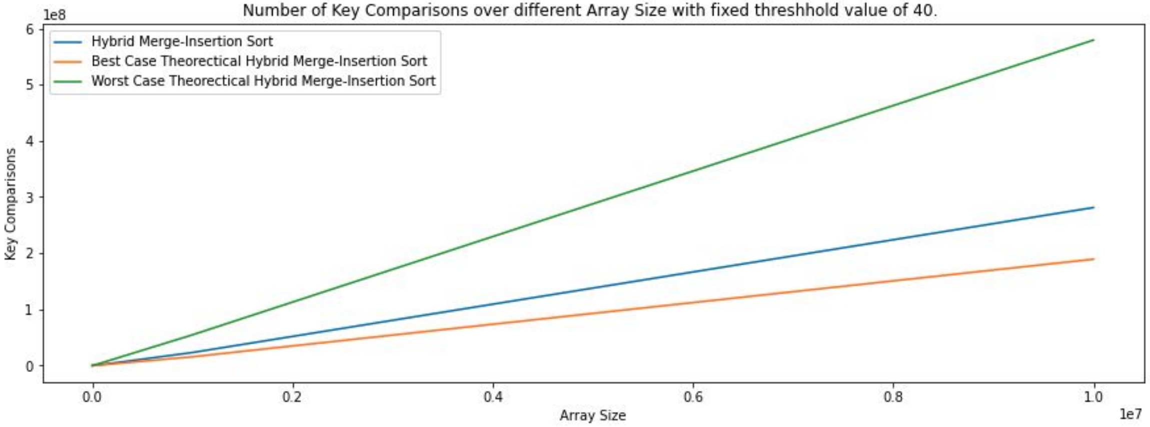
Optimal S Value

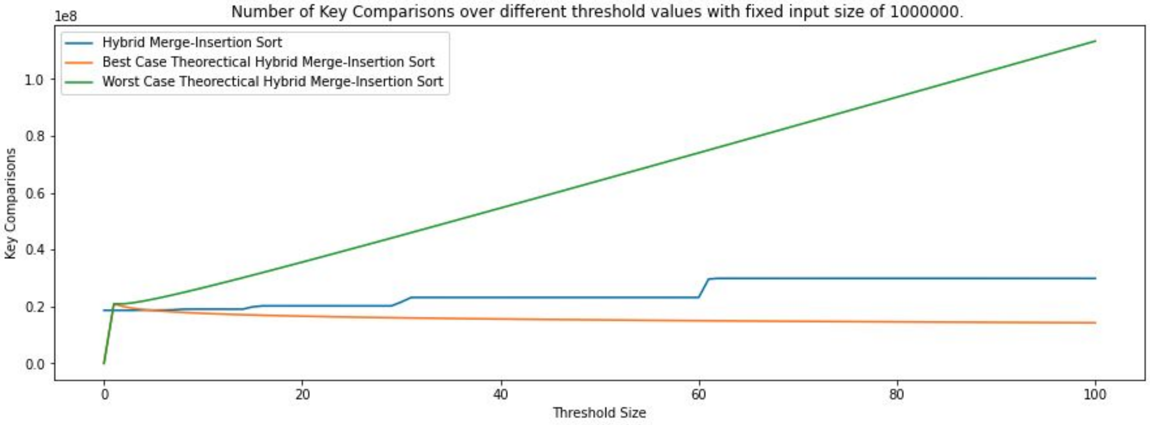
Theoretical

Best Case: N+Nlog(N/S)

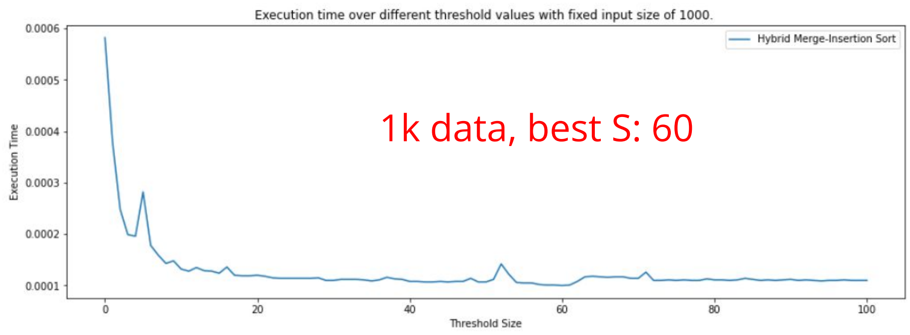
Worst Case: NS + Nlog(N/S)

Theoretical vs Empirical


Number of Key Comparisons over different Array Size with fixed threshhold value of 40.

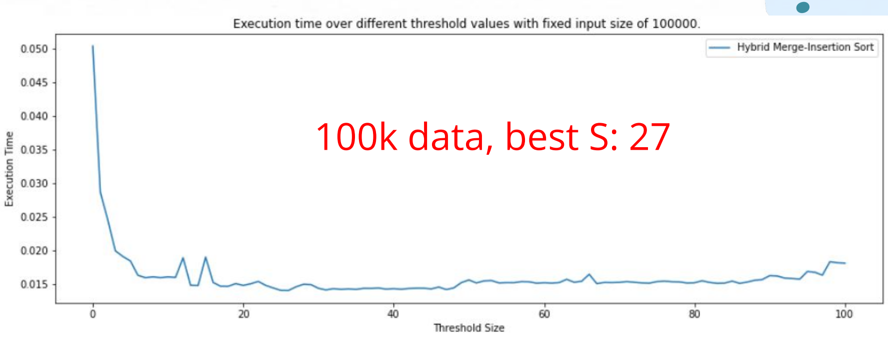

Number of Key Comparisons over different threshold values with fixed input size of 1000000.

Optimal S Value for different input size



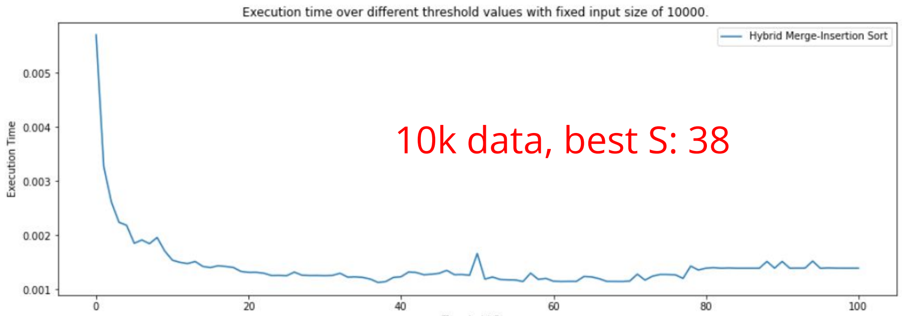
Execution time over different threshold values with fixed input size of 1000.

1k data, best S: 60



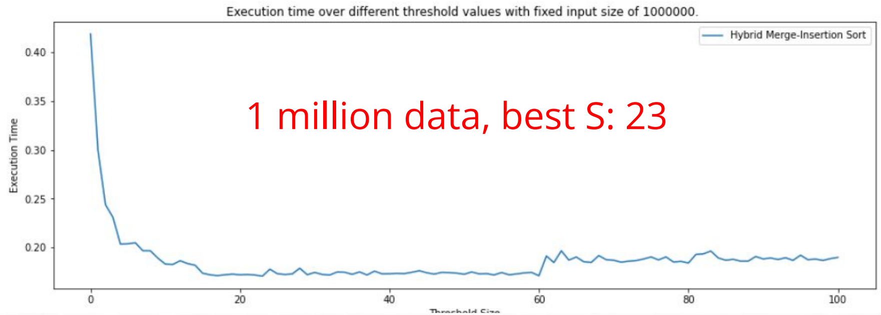
Execution time over different threshold values with fixed input size of 100000.

100k data, best S: 27



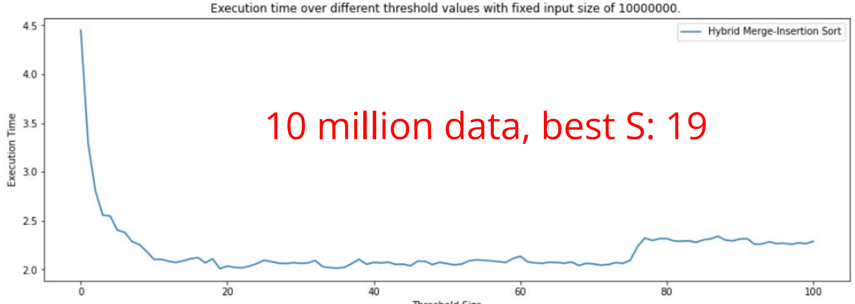
Execution time over different threshold values with fixed input size of 10000.

10k data, best S: 38



Execution time over different threshold values with fixed input size of 1000000.

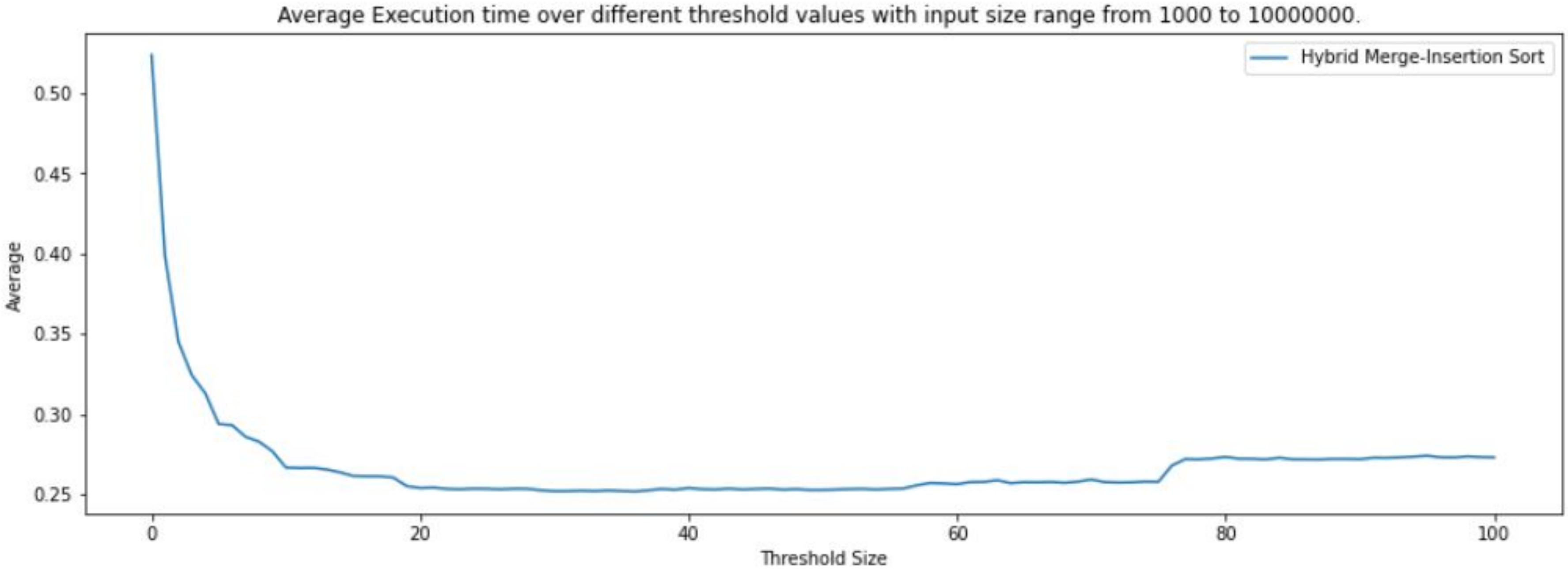
1 million data, best S: 23



Execution time over different threshold values with fixed input size of 10000000.

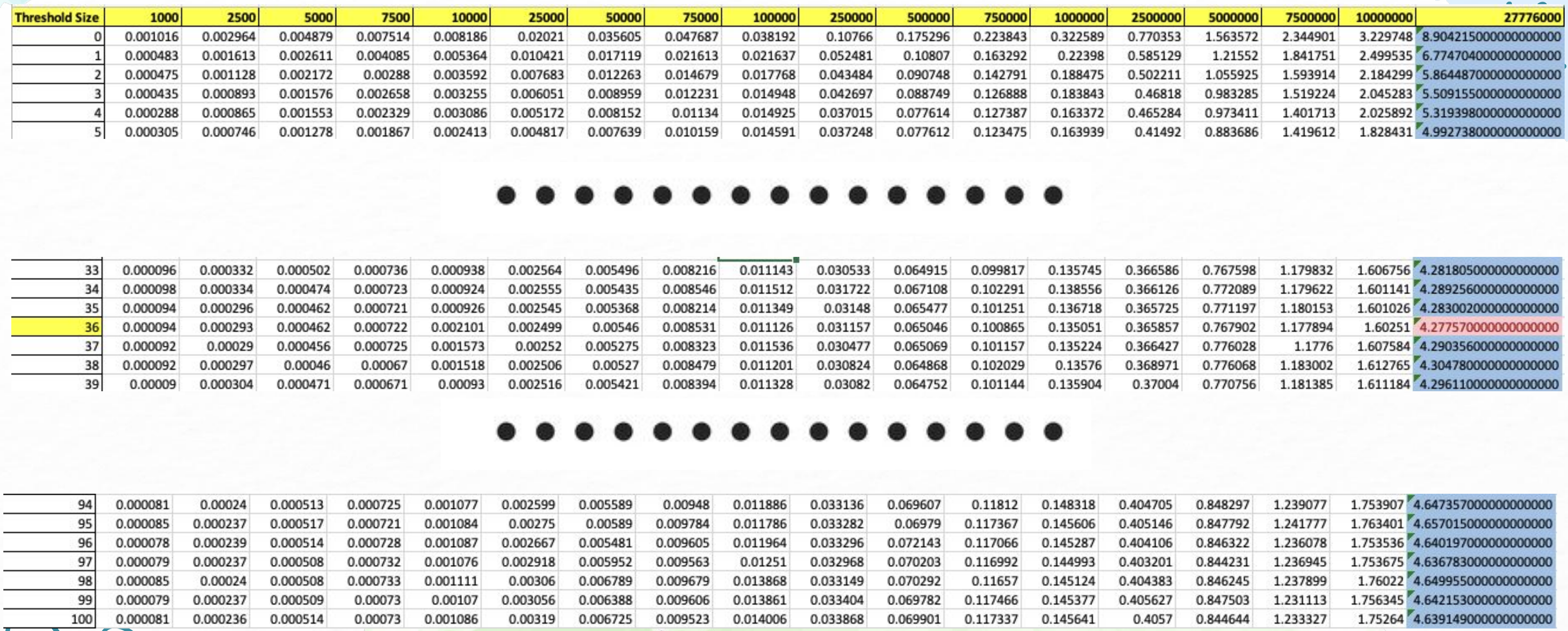
10 million data, best S: 19

Average Optimal S Value



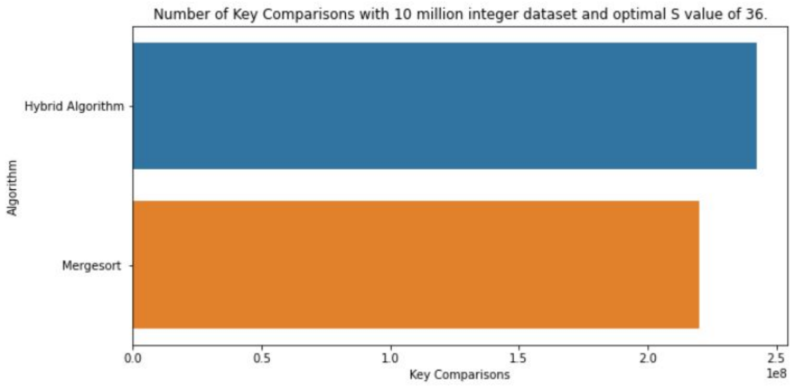
Through our data, we found out that the optimal value of S for the best performance of this hybrid algorithm is 36!

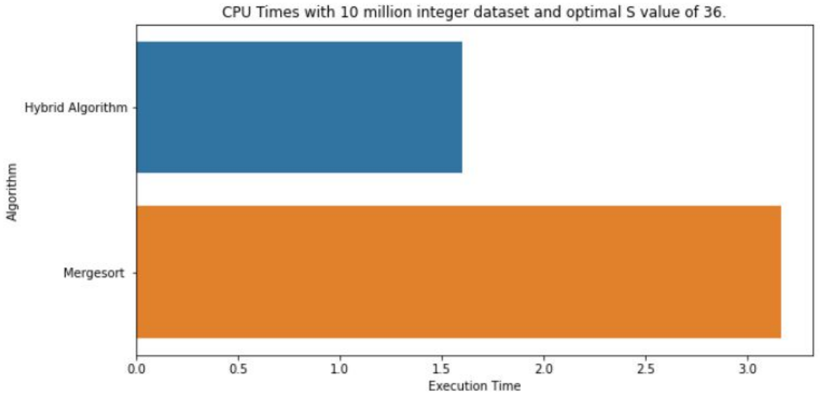
Average Optimal S Value



Threshold Size	1000	2500	5000	7500	10000	25000	50000	75000	100000	250000	500000	750000	1000000	2500000	5000000	7500000	10000000	27776000
0	0.001016	0.002964	0.004879	0.007514	0.008186	0.02021	0.035605	0.047687	0.038192	0.10766	0.175296	0.223843	0.322589	0.770353	1.563572	2.344901	3.229748	8.904215000000000000
1	0.000483	0.001613	0.002611	0.004085	0.005364	0.010421	0.017119	0.021613	0.021637	0.052481	0.10807	0.163292	0.22398	0.585129	1.21552	1.841751	2.499535	6.774704000000000000
2	0.000475	0.001128	0.002172	0.00288	0.003592	0.007683	0.012263	0.014679	0.017768	0.043484	0.090748	0.142791	0.188475	0.502211	1.055925	1.593914	2.184299	5.864487000000000000
3	0.000435	0.000893	0.001576	0.002658	0.003255	0.006051	0.008959	0.012231	0.014948	0.042697	0.088749	0.126888	0.183843	0.46818	0.983285	1.519224	2.045283	5.509155000000000000
4	0.000288	0.000865	0.001553	0.002329	0.003086	0.005172	0.008152	0.01134	0.014925	0.037015	0.077614	0.127387	0.163372	0.465284	0.973411	1.401713	2.025892	5.319398000000000000
5	0.000305	0.000746	0.001278	0.001867	0.002413	0.004817	0.007639	0.010159	0.014591	0.037248	0.077612	0.123475	0.163939	0.41492	0.883686	1.419612	1.828431	4.992738000000000000
33	0.000096	0.000332	0.000502	0.000736	0.000938	0.002564	0.005496	0.008216	0.011143	0.030533	0.064915	0.099817	0.135745	0.366586	0.767598	1.179832	1.606756	4.281805000000000000
34	0.000098	0.000334	0.000474	0.000723	0.000924	0.002555	0.005435	0.008546	0.011512	0.031722	0.067108	0.102291	0.138556	0.366126	0.772089	1.179622	1.601141	4.289256000000000000
35	0.000094	0.000296	0.000462	0.000721	0.000926	0.002545	0.005368	0.008214	0.011349	0.03148	0.065477	0.101251	0.136718	0.365725	0.771197	1.180153	1.601026	4.283002000000000000
36	0.000094	0.000293	0.000462	0.000722	0.002101	0.002499	0.00546	0.008531	0.011126	0.031157	0.065046	0.100865	0.135051	0.365857	0.767902	1.177894	1.60251	4.277570000000000000
37	0.000092	0.00029	0.000456	0.000725	0.001573	0.00252	0.005275	0.008323	0.011536	0.030477	0.065069	0.101157	0.135224	0.366427	0.776028	1.1776	1.607584	4.290356000000000000
38	0.000092	0.000297	0.00046	0.00067	0.001518	0.002506	0.00527	0.008479	0.011201	0.030824	0.064868	0.102029	0.13576	0.368971	0.776068	1.183002	1.612765	4.304780000000000000
39	0.00009	0.000304	0.000471	0.000671	0.00093	0.002516	0.005421	0.008394	0.011328	0.03082	0.064752	0.101144	0.135904	0.37004	0.770756	1.181385	1.611184	4.296110000000000000
94	0.000081	0.00024	0.000513	0.000725	0.001077	0.002599	0.005589	0.00948	0.011886	0.033136	0.069607	0.11812	0.148318	0.404705	0.848297	1.239077	1.753907	4.647357000000000000
95	0.000085	0.000237	0.000517	0.000721	0.001084	0.00275	0.00589	0.009784	0.011786	0.033282	0.06979	0.117367	0.145606	0.405146	0.847792	1.241777	1.763401	4.657015000000000000
96	0.000078	0.000239	0.000514	0.000728	0.001087	0.002667	0.005481	0.009605	0.011964	0.033296	0.072143	0.117066	0.145287	0.404106	0.846322	1.236078	1.753536	4.640197000000000000
97	0.000079	0.000237	0.000508	0.000732	0.001076	0.002918	0.005952	0.009563	0.01251	0.032968	0.070203	0.116992	0.144993	0.403201	0.844231	1.236945	1.753675	4.636783000000000000
98	0.000085	0.00024	0.000508	0.000733	0.001111	0.00306	0.006789	0.009679	0.013868	0.033149	0.070292	0.11657	0.145124	0.404383	0.846245	1.237899	1.76022	4.649955000000000000
99	0.000079	0.000237	0.000509	0.00073	0.00107	0.003056	0.006388	0.009606	0.013861	0.033404	0.069782	0.117466	0.145377	0.405627	0.847503	1.231113	1.756345	4.642153000000000000
100	0.000081	0.000236	0.000514	0.00073	0.001086	0.00319	0.006725	0.009523	0.014006	0.033868	0.069901	0.117337	0.145641	0.4057	0.844644	1.233327	1.75264	4.639149000000000000

HybridSort vs MergeSort

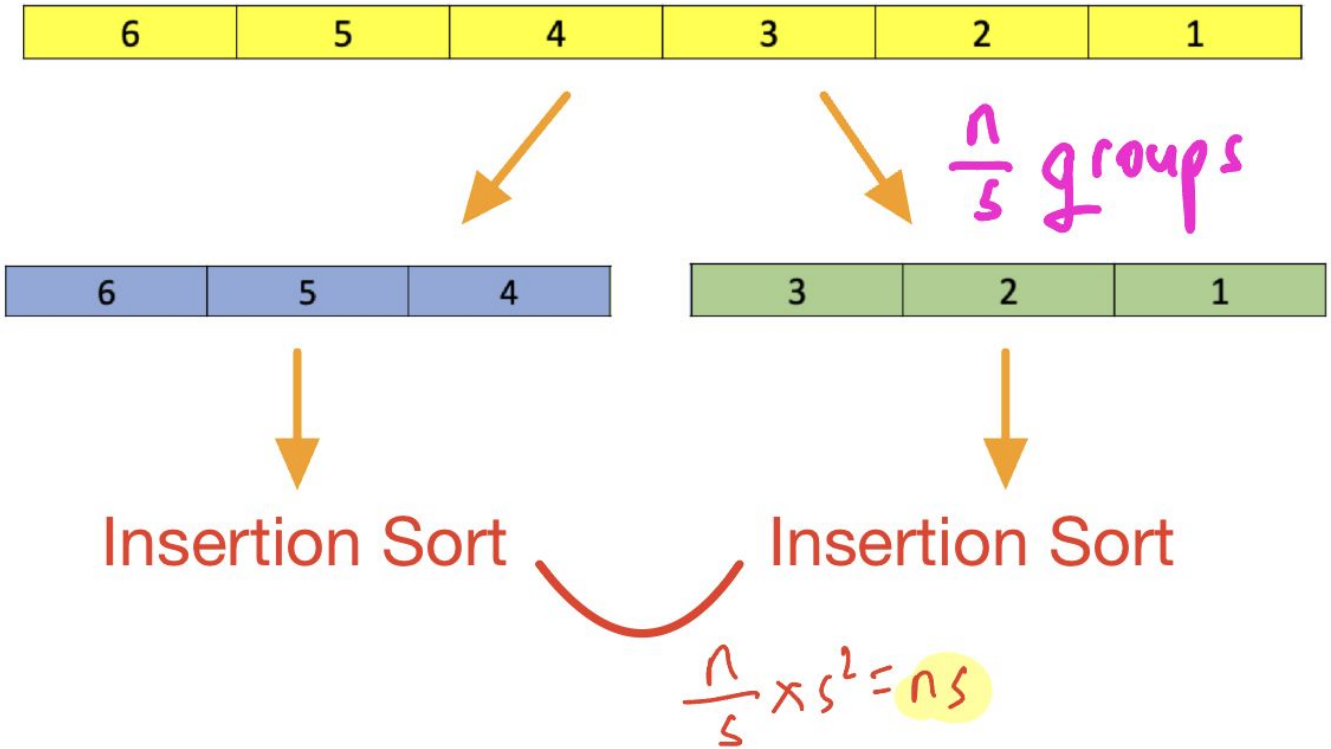




THANKS

CREDITS: This presentation template was created **by Slidesgo**, including icons **by Flaticon**, infographics & images **by Freepik**

Please keep this slide for attribution



6	5	4	3	2	1

$\frac{n}{5}$ groups

6	5	4

3	2	1

Insertion Sort

Insertion Sort

$\frac{n}{5} \times 5^2 = n5$