Example Class Project 1

By Jordan, Ethan, and Jared

Content

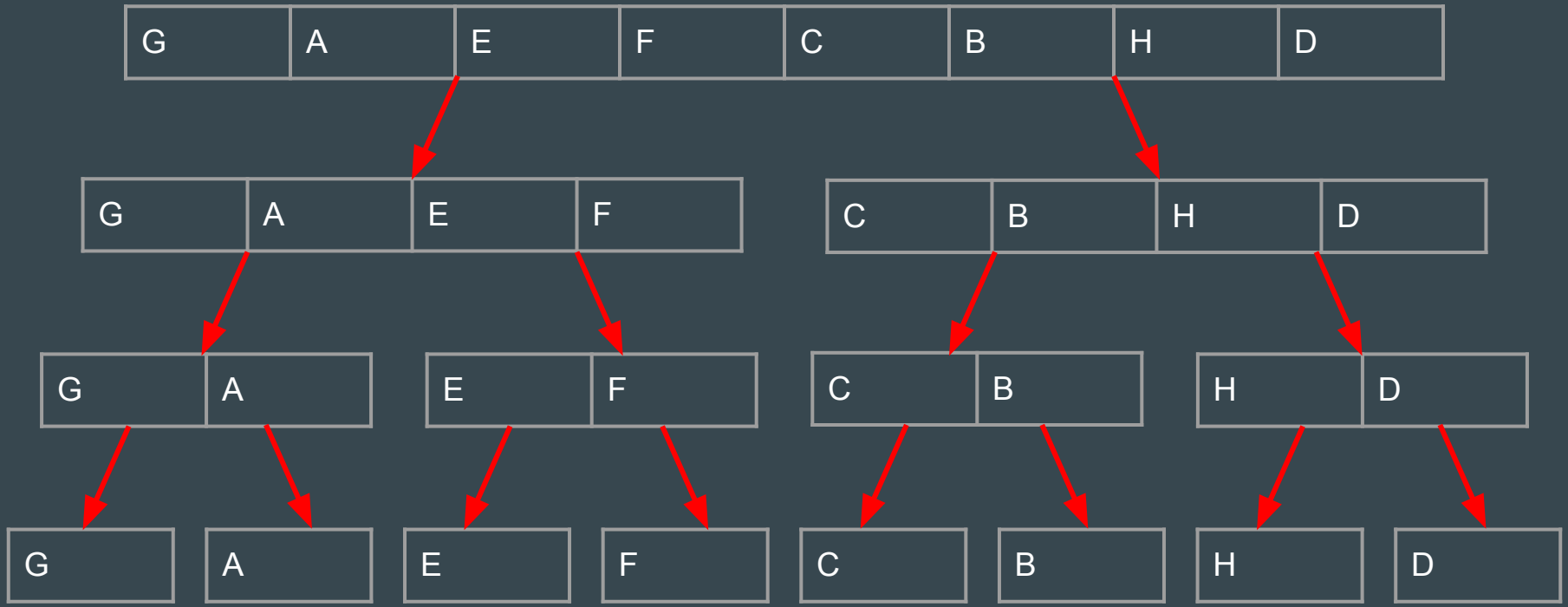
- Implementation
  - Screenshot + Pseudocode?
- Generate randomised list
- Time complexity
  - Same S (size of smallest subarray), diff n (size of list)
  - Same n, diff S
  - Using diff input datasets (random, reversed, etc)
  - Find the optimal value of S
- Compare w original mergesort
  - Performance of merge sort
  - Comparison between mergesort and hybrid algorithm

Implementation

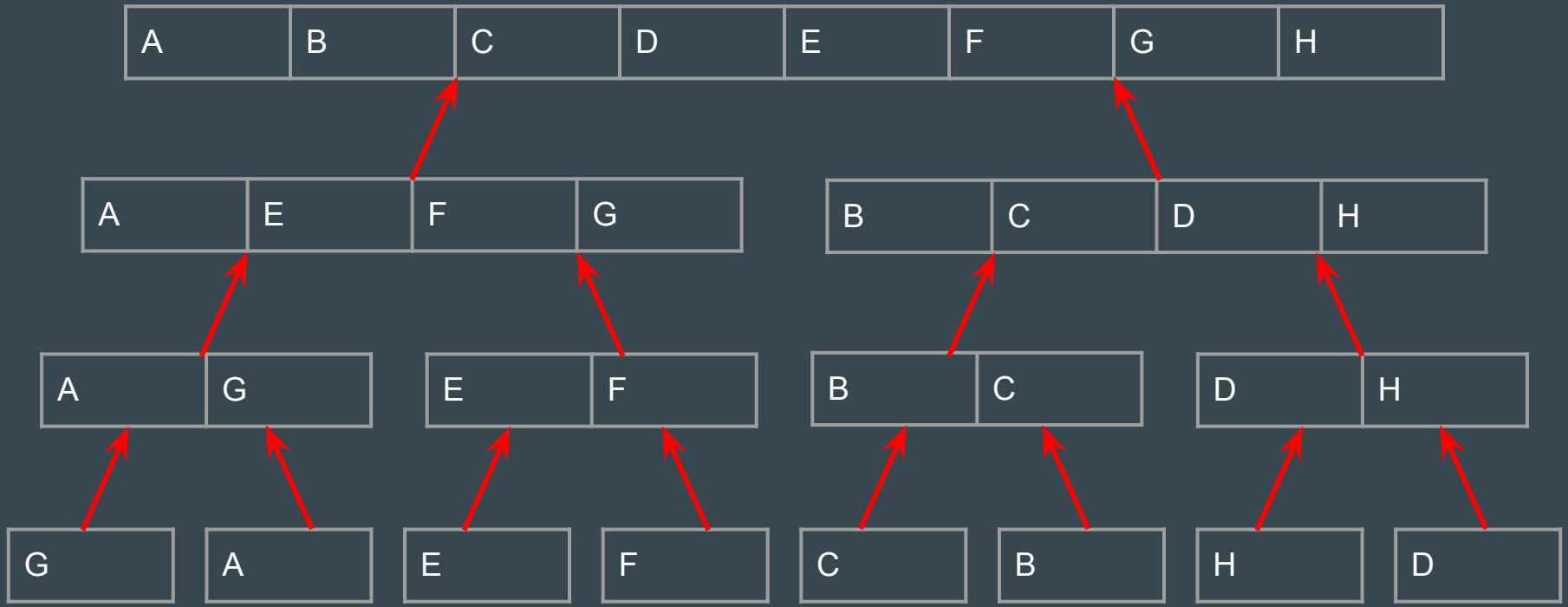
```
HybridSort(list, start, end):
    If sizeOfList < S:
        InsertionSort(list)
    Else:
        HybridSort(list, start, mid)
        HybridSort(list, mid, end)
        merge(list, start, mid, end)
```

```
MergeSort(list, start, end):
    if sizeOfList <= 1:
        return
    Else:
        MergeSort(list, start, mid)
        MergeSort(list, mid, end)
        merge(list, start, mid, end)
```

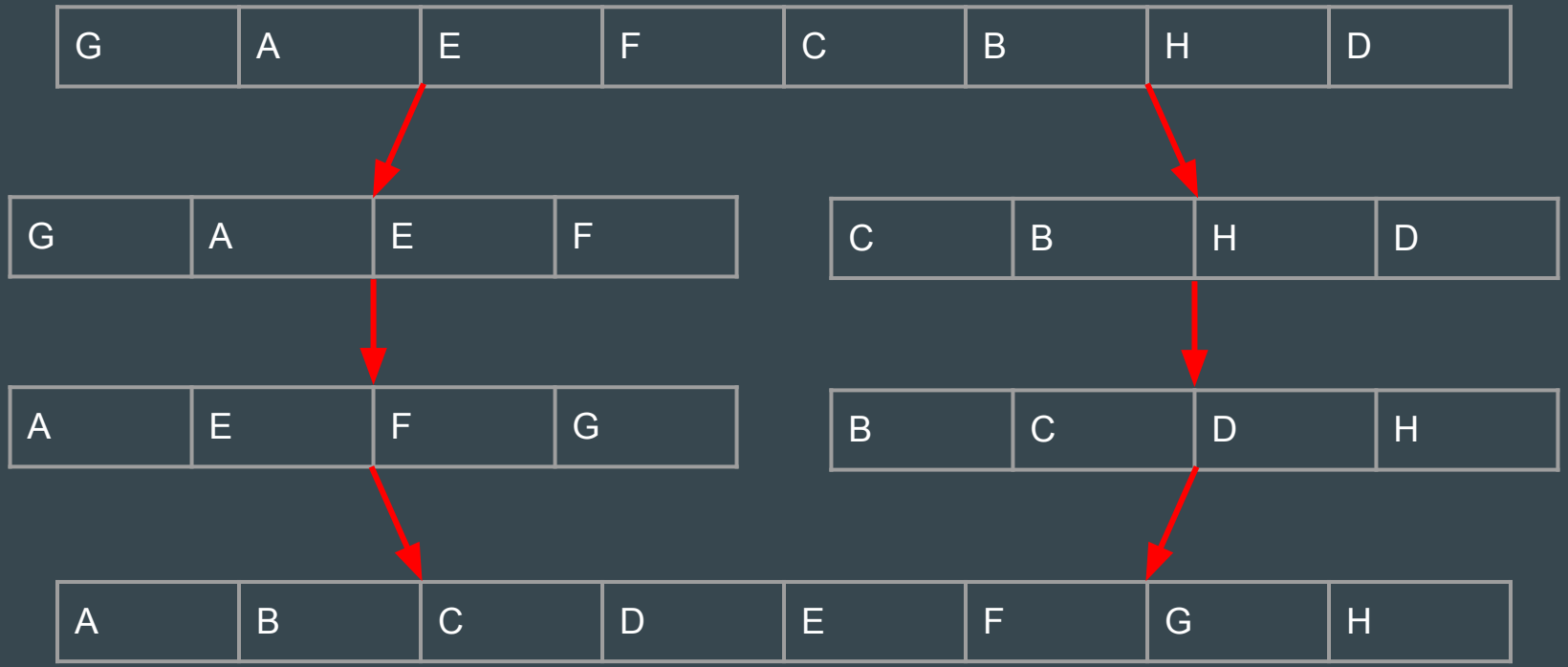
Merge Sort



Merge Sort



Merge Sort



Implementation

```
HybridSort(list, start, end):
    If sizeOfList < S:
        InsertionSort(list)
    Else:
        HybridSort(list, start, mid)
        HybridSort(list, mid, end)
        merge(list, start, mid, end)
```

```
InsertionSort(list, sizeOfList):
    //Incremental Approach
    for (int i = 1; i < sizeOfList; ++i):
        For (int j = i; j > 0; j--)
            if(list[j] < list[j-1])
                swap(list[j], list[j-1]
            Else break;
```

Implementation

```
HybridSort(list, start, end):
    If sizeOfList < S:
        InsertionSort(list)
    Else:
        HybridSort(list, start, mid)
        HybridSort(list, mid, end)
        merge(list, start, mid, end)
```

```
merge(list, start, mid, end) {
    // Split the list into left and right
    left = list[start:mid]
    right = list[mid:end]
    int i, j = 0
    int k = start;
    //Add smaller item of the 2 sublists to the
list
    while(i < sizeOfLeft && j < sizeOfRight):
        if left[i] < right[j]: append left to
    list
        else: append right to list
    // Add the remaining items to the list
    while (i < sizeOfLeft)
        append left[i] to list
        i++
    while (j < sizeOfRight)
        append right[j] to list
        j++
```

Time Complexity Theoretical Analysis

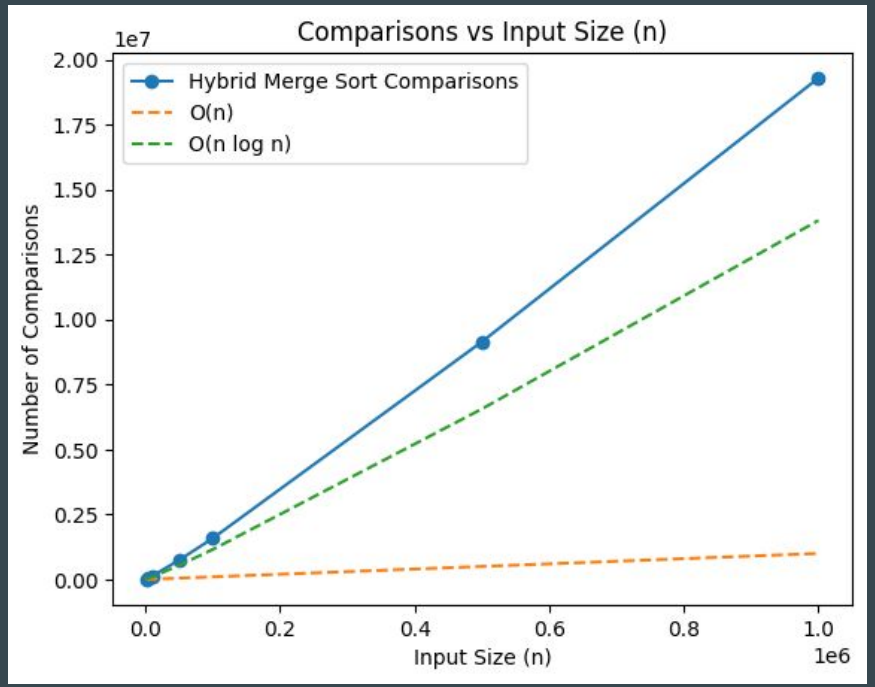
In the worst case

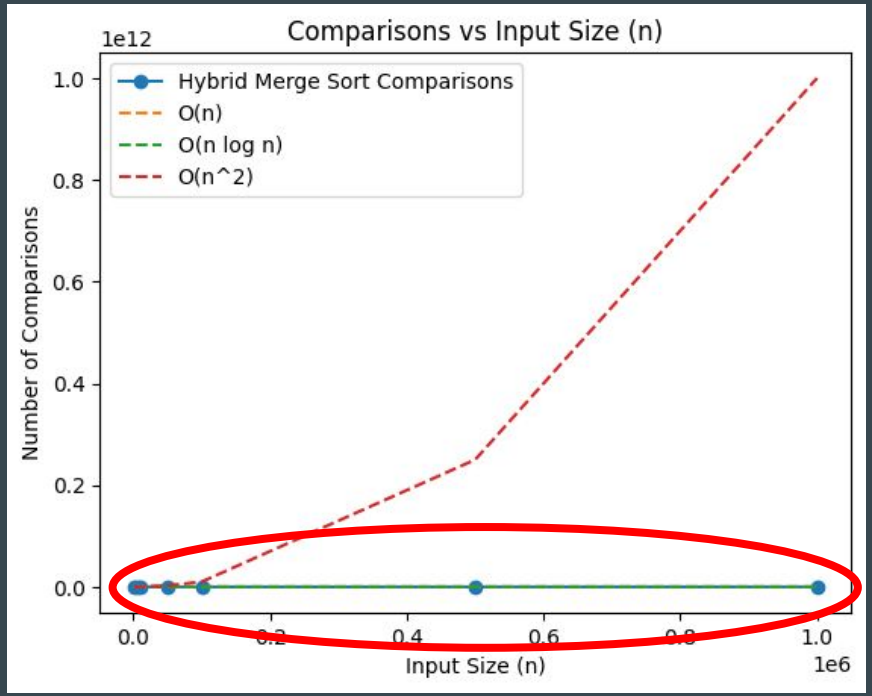
- For small S, behaves more like Merge Sort: O(nlogn)
- For large S, behaves more like Insertion Sort: O(n^2)

Assuming an optimal value of S is used,

the algorithm has **O(n log n)** Complexity

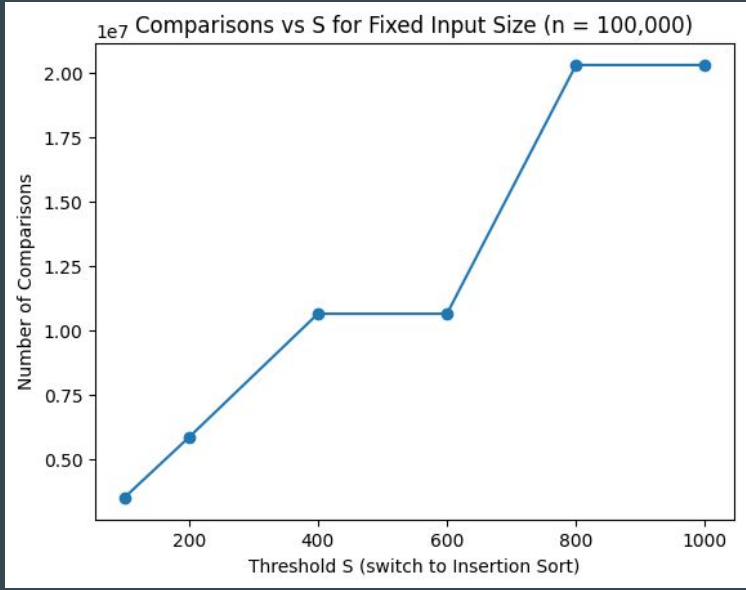
Empirical Results (Fixed S)

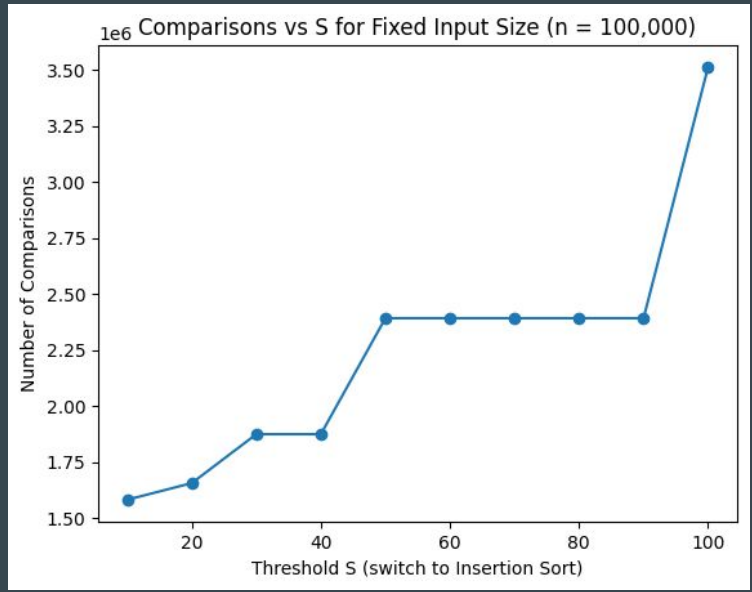




O(n log

Empirical results (fixed n)

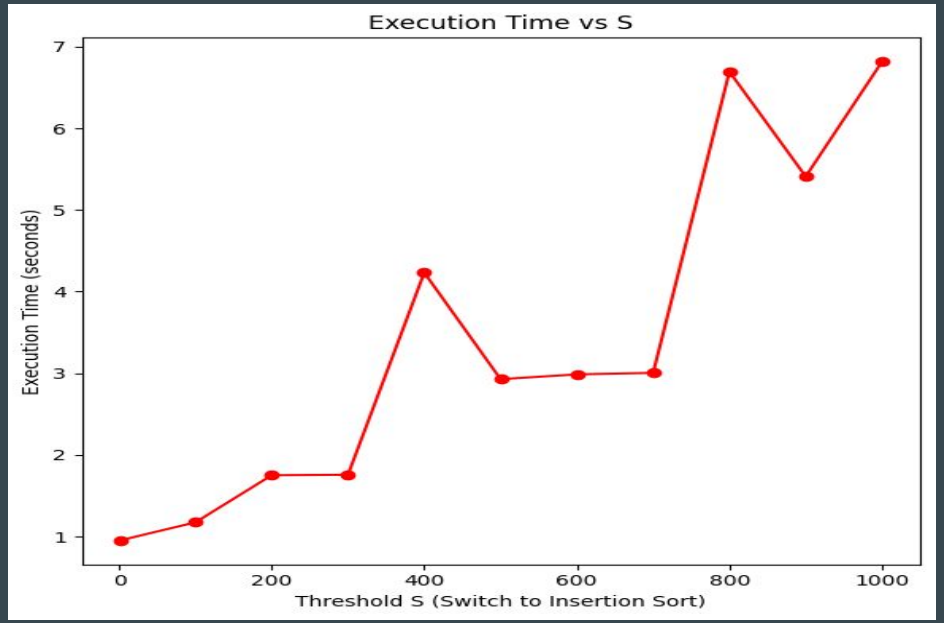




Smaller S = less comparisons = better runtime?

Optimal S value

Somewhat linear relationship between S and runtime



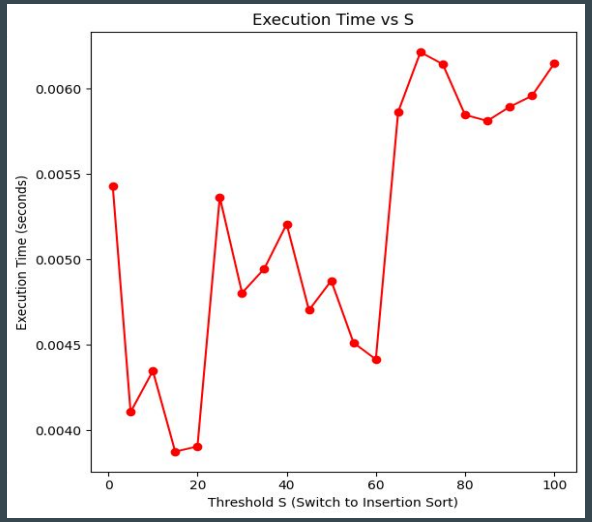
Less comparisons =/= faster!!!
Why?

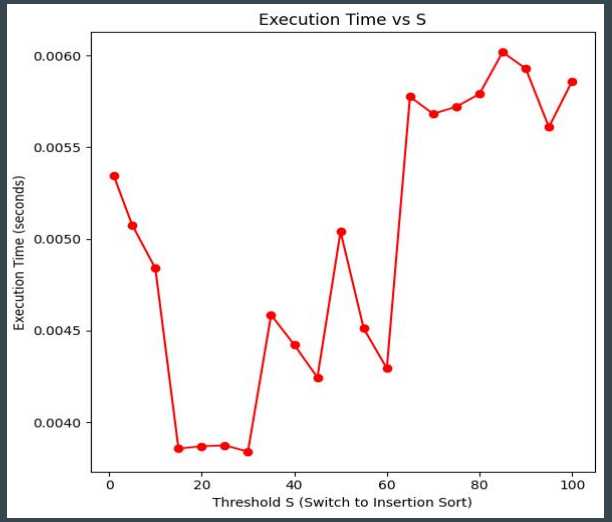
High recursion overhead

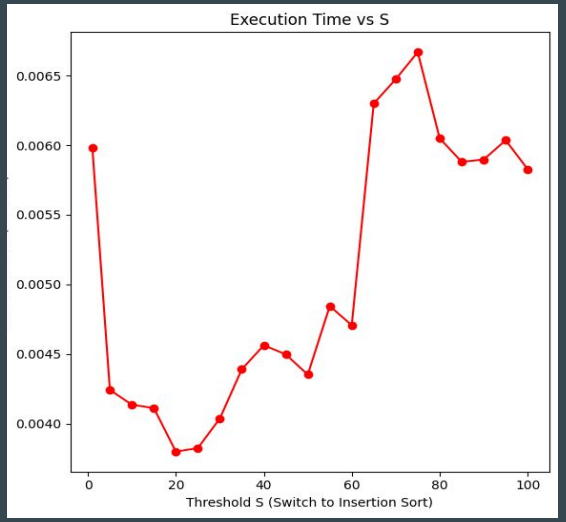
Prevalence of Insertion sort O(n^2)

Size of S

n = 1,000

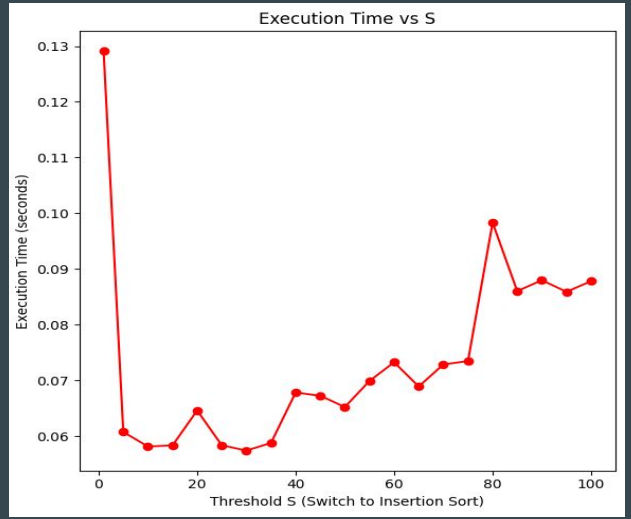

Execution Time vs S
Execution Time (seconds)
0.0060
0.0055
0.0050
0.0045
0.0040
0
20
40
60
80
100
Threshold S (Switch to Insertion Sort)

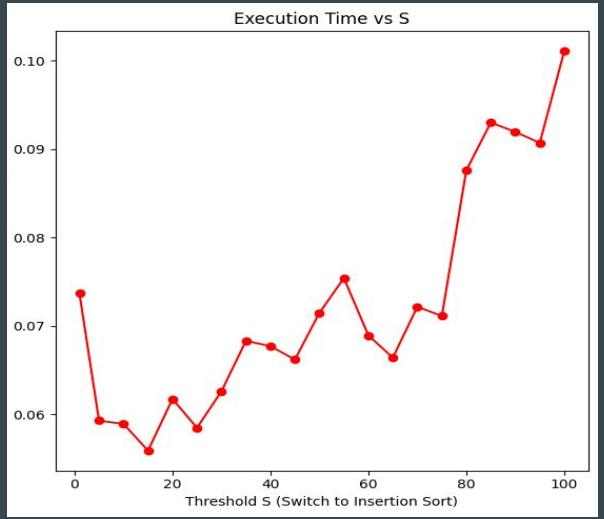

Execution Time vs S
Execution Time (seconds)
0.0060
0.0055
0.0050
0.0045
0.0040
0
20
40
60
80
100
Threshold S (Switch to Insertion Sort)

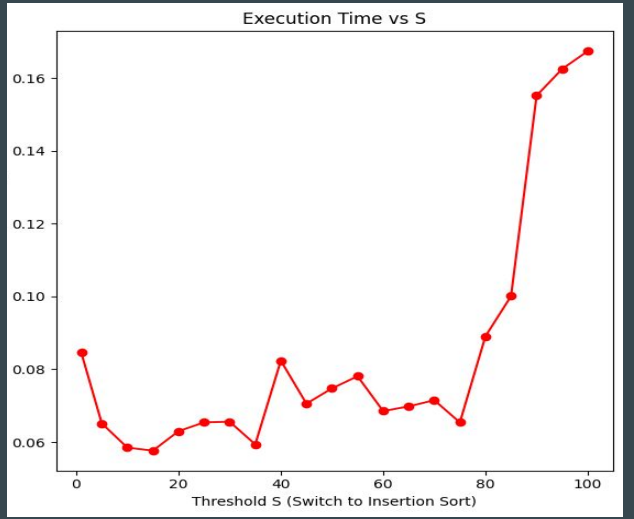

Execution Time vs S
0.0065
0.0060
0.0055
0.0050
0.0045
0.0040
0
20
40
60
80
100
Threshold S (Switch to Insertion Sort)

15 < S < 30

n = 10,000

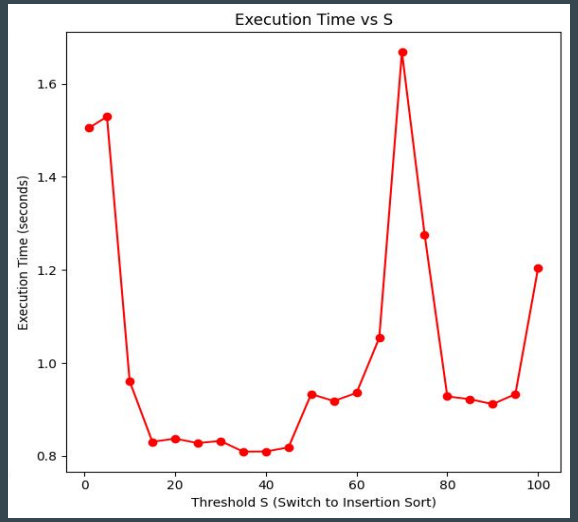


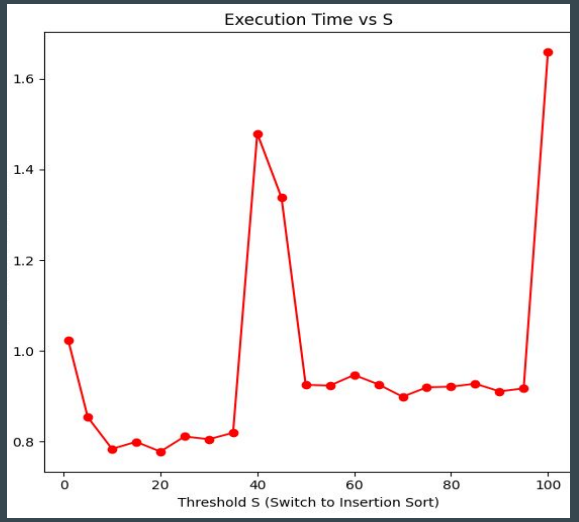


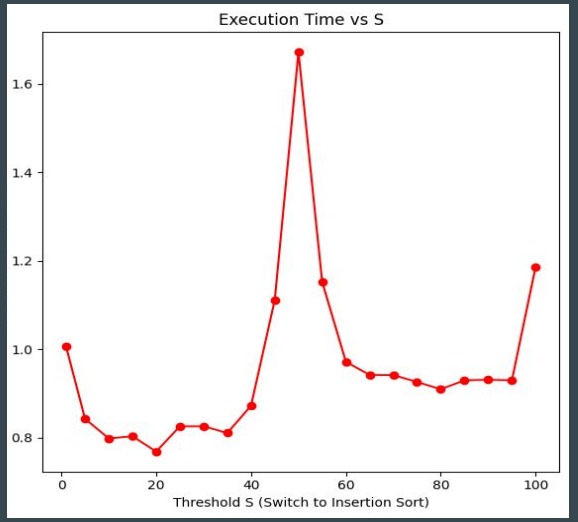


10 < S < 35

n = 100,000

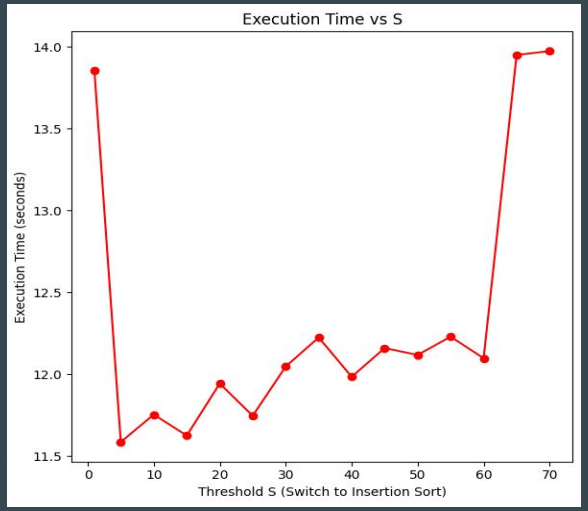


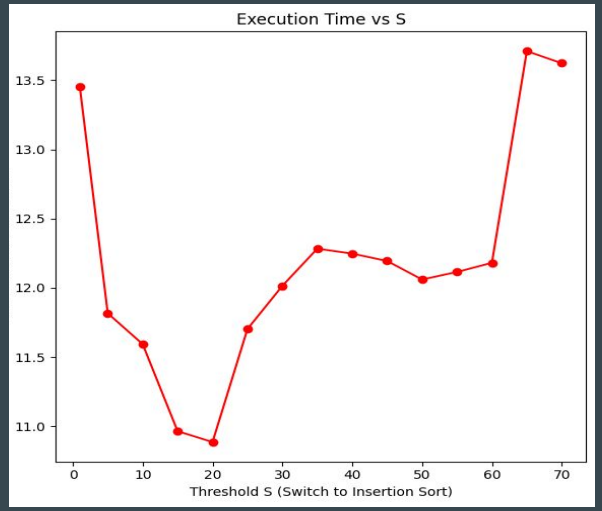


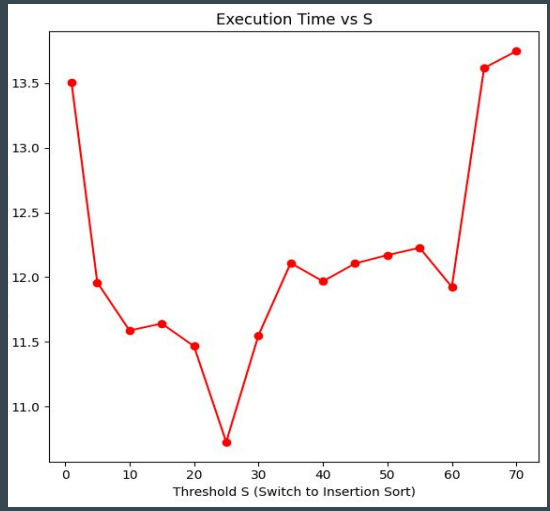


10 < S < 40

n = 1,000,000







10 < S < 30

Conclusion:

Optimal S: 15 < S < 25

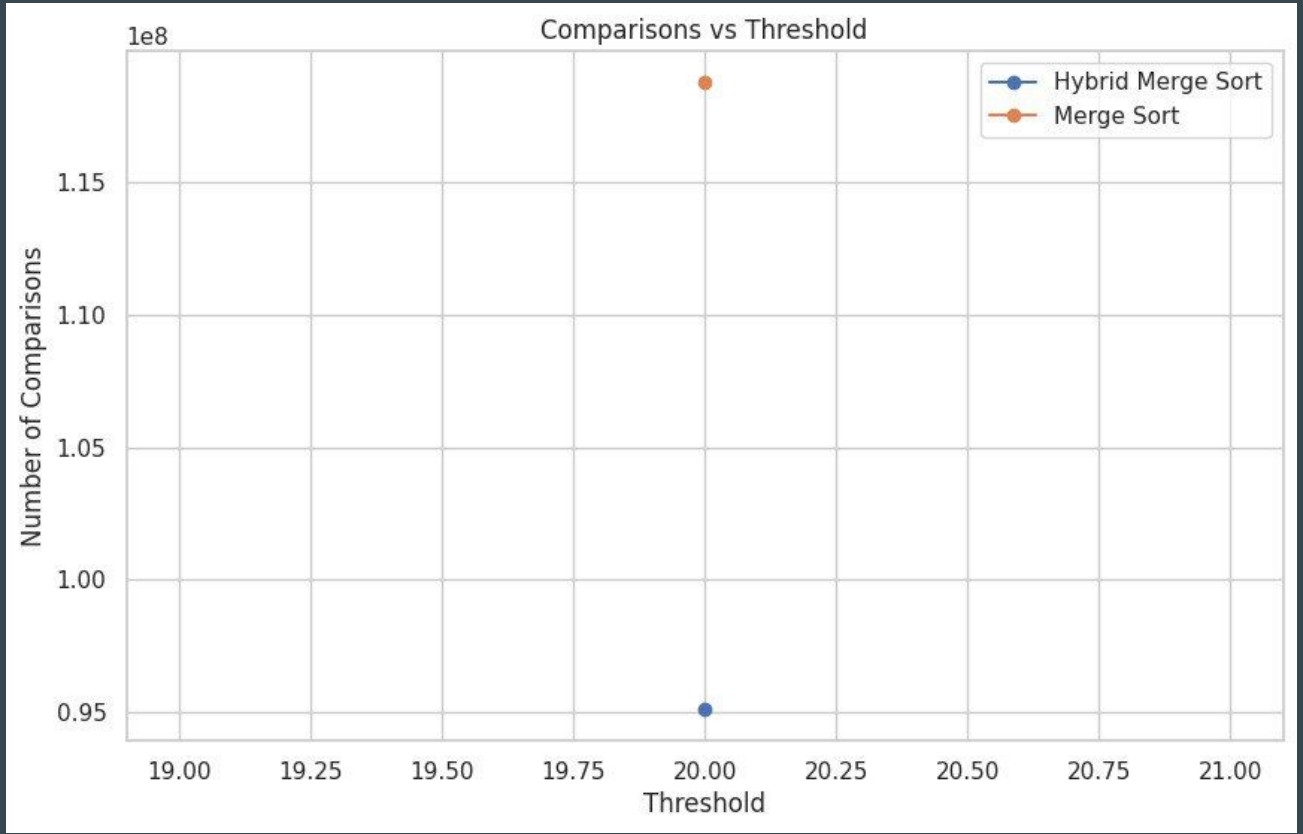
Comparing w mergesort

S = 20, Size= 10 million

	Hybrid sort algorithm	Merge sort algorithm
No. of comparisons	95,079,616	118,788,160
Time taken(seconds)	157.6165	197.6796

No. of Comparisons

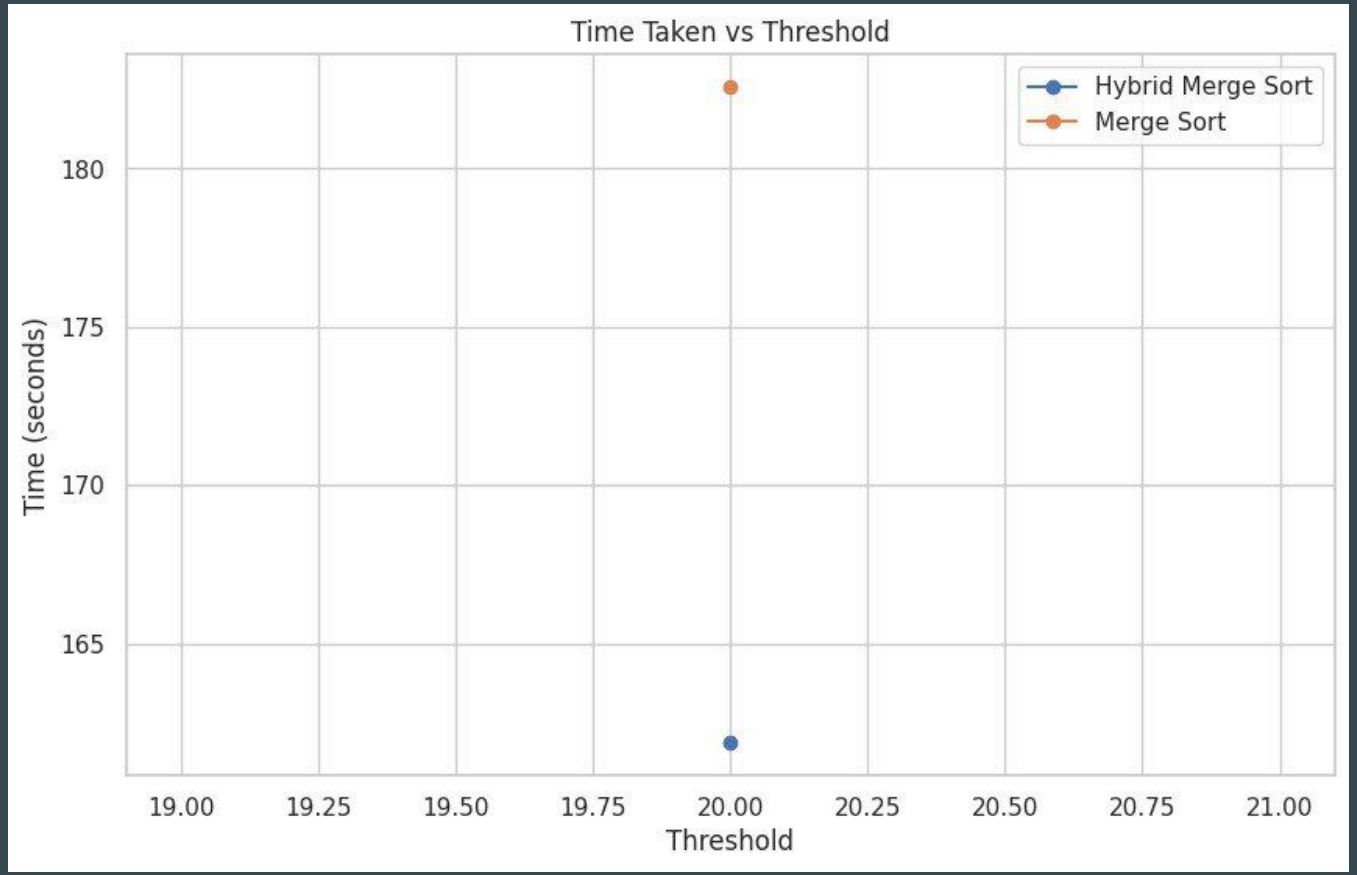
-Hybrid more efficient in smaller subarrays



CPU Time

-Lesser comparison = faster execution time

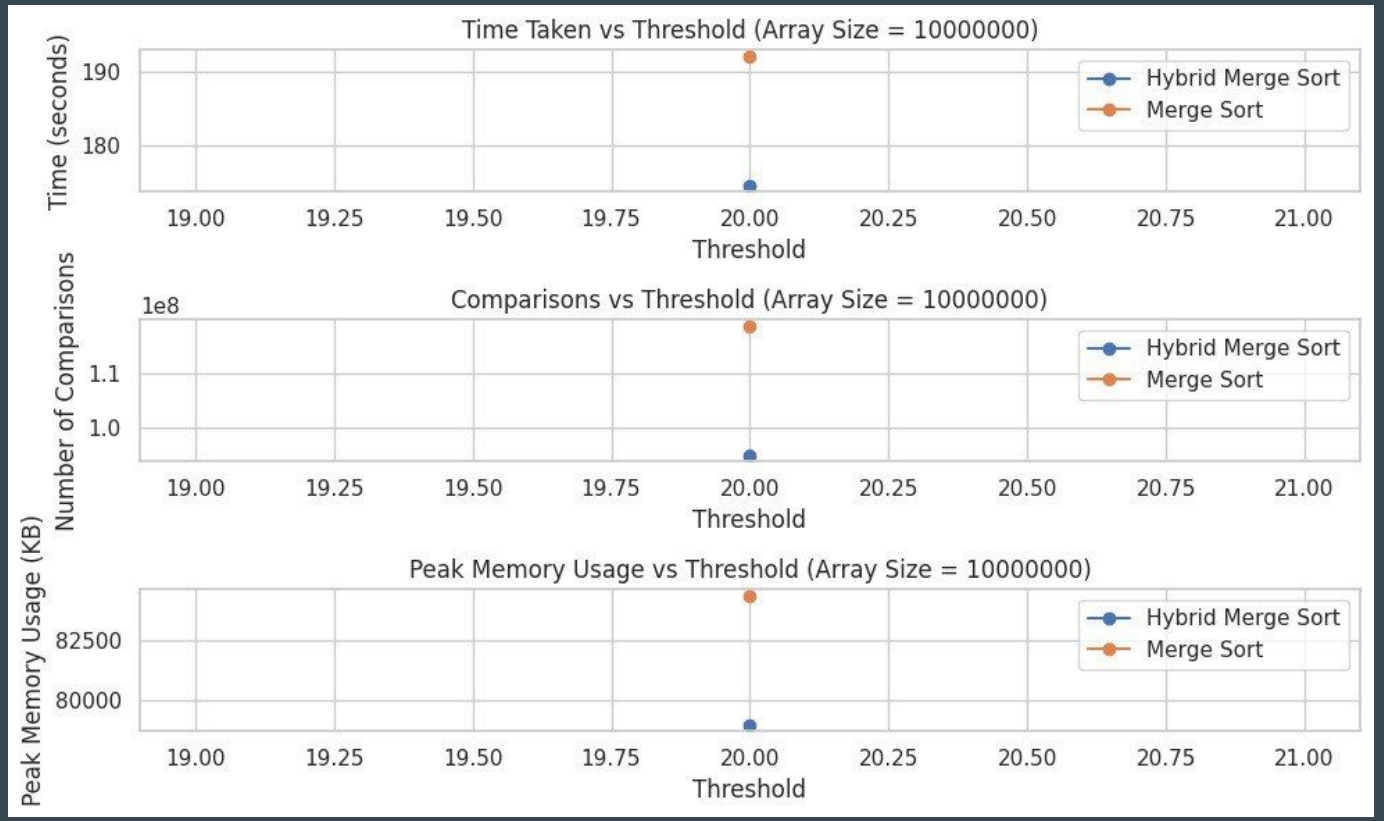
-Lesser recursive calls and merge operations



Impact of optimal threshold value(S)

-Balances overhead of recursive calls and efficiency of insertion sort for small subarrays

-Optimal threshold value improved hybrid's practical performance in efficiency



Thank you