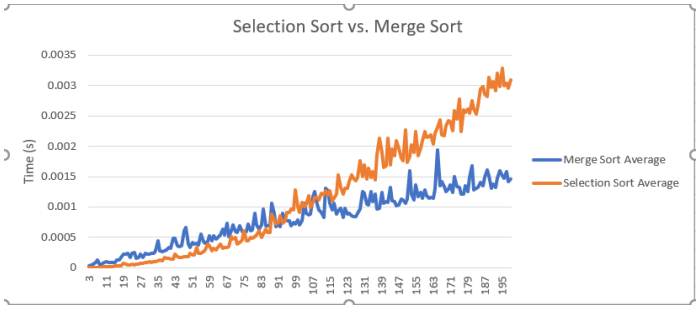
The Union of Merge Sort and Selection Sort

The experiment consisted of several key features. These included the Merge Sort and Selection Sort algorithms, a method to generate random arrays of sizes n, a way to measure the time for each sort, and the ability to easily access and manipulate the data retrieved once each sort had been completed.

This was achieved by first creating a simple function arrGenerator(size) that takes in an integer, size, and generates a random list of elements length size with entries between 0 and 1000. Next, the algorithms for Merge Sort and Selection Sort were taken from geeksforgeeks.org, listed at the end of this paper, and listed in the code. They were put in almost completely unchanged and unoptimized. The only changes necessary were for variable names and removing the redundant print statements. Following the translation of these algorithms, a way to time each sort was done using the time.perf_counter() functionality. Each sort was done for arrays of size 3 up to 199, 100 times each. This data was stored in an excel file upon completion.

With all of the hard work out of the way, it became possible to simply run the program and open the excel file to see the results. However, the data had ‘buckets’ as every 100 entries was the same size. To keep the data more manageable, every 100 entries were averaged for that size n. These data points were put on a graph (seen below), and the crossover number was around 85, reflected in the Hybrid Sort algorithm.



Computer Specifications

Processor Intel(R) Core(TM) i3-8145U CPU @ 2.10GHz 2.30 GHz

Installed RAM 4.00 GB (3.78 GB usable)

System type 64-bit operating system, x64-based processor

References

Merge sort. GeeksforGeeks. (2022, February 26).

https://www.geeksforgeeks.org/merge-sort/

Selection sort. GeeksforGeeks. (2022, February 3).

https://www.geeksforgeeks.org/selection-sort/

Writing to an Excel sheet using Python. GeeksforGeeks. (2019, July 29).

https://www.geeksforgeeks.org/writing-excel-sheet-using-python/

GenerateData.py

```
Import xlwt
from xlwt import Workbook

import random
from random import seed
from random import randint

import time

#random seed shenanigans
seed(42)

#generates random arrays of different lengths
def arrGenerator(size):

    #generate empty list
    lst = []

    #changes the ammount of inputs
    for _ in range(size):

        #appends a random number to the end of the list between 0 and 1000
        lst.append(randint(0,1000))

    return lst



def merge_sort_time(wb):

    #creates excel sheets

    counter = 0

    sheet1.write(1,1,'Merge Sort Tmes (s)')

    for size in range(3,200):

        #generates random array of size n
        lst = arrGenerator(size)
```

```
        print('Not Sorted!')
        print(lst)

        for element in range(100):

            t0 = time.perf_counter()
            mergeSort(lst)
            t1 = time.perf_counter()

            counter += 1

            timeTaken = t1 - t0
            sheet1.write(counter+1,1,timeTaken)



def selection_sort_time(wb):

    #creates excel sheets
    counter = 0

    sheet1.write(1,2,'Selection Sort Tmes (s)')

    for size in range(3,200):

        #generates random array of size n
        lst = arrGenerator(size)

        for element in range(100):

            t0 = time.perf_counter()
            selectionSort(lst)
            t1 = time.perf_counter()

            counter += 1

            timeTaken = t1 - t0
            sheet1.write(counter+1,2,timeTaken)



#selection sort taken from https://www.geeksforgeeks.org/selection-sort/
def selectionSort(arr):
```

```
    for i in range(len(arr)):

    # Find the minimum element in remaining
    # unsorted array
        min = i
        for j in range(i+1, len(arr)):
            if arr[min] > arr[j]:
                min = j

    # Swap the found minimum element with
    # the first element
        arr[i], arr[min] = arr[min], arr[i]



#merge sort taken from https://www.geeksforgeeks.org/merge-sort/
def mergeSort(arr):
    if len(arr) > 1:

        # Finding the mid of the array
        mid = len(arr)//2

        # Dividing the array elements
        L = arr[:mid]

        # into 2 halves
        R = arr[mid:]

        # Sorting the first half
        mergeSort(L)

        # Sorting the second half
        mergeSort(R)

        i = j = k = 0

        # Copy data to temp arrays L[] and R[]
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
```

```
            k += 1

        # Checking if any element was left
        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1

        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1



# Driver Code
if __name__ == '__main__':


    #Create excel sheet
    wb = Workbook()
    sheet1 = wb.add_sheet('Sheet 1', cell_overwrite_ok = True)

    #call the two time functions to record the time in excel
    merge_sort_time(wb)
    selection_sort_time(wb)

    #save the data
    wb.save('SortingData.xls')
```

HybridSort.py

```
import random
from random import randint


#generates random arrays of different lengths
def arrGenerator(size):

    #generate empty list
    lst = []

    #changes the ammount of inputs
    for _ in range(size):

        #appends a random number to the end of the list between 0 and 1000
        lst.append(randint(0,1000))

    return lst


#selection sort taken from https://www.geeksforgeeks.org/selection-sort/
def selectionSort(arr):
    for i in range(len(arr)):

    # Find the minimum element in remaining
    # unsorted array
        min = i
        for j in range(i+1, len(arr)):
            if arr[min] > arr[j]:
                min = j

    # Swap the found minimum element with
    # the first element
        arr[i], arr[min] = arr[min], arr[i]



#merge sort taken from https://www.geeksforgeeks.org/merge-sort/
def mergeSort(arr):
    if len(arr) > 1:

        # Finding the mid of the array
        mid = len(arr)//2
```

```
        # Dividing the array elements
        L = arr[:mid]

        # into 2 halves
        R = arr[mid:]

        # Sorting the first half
        mergeSort(L)

        # Sorting the second half
        mergeSort(R)

        i = j = k = 0

        # Copy data to temp arrays L[] and R[]
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                arr[k] = L[i]
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

        # Checking if any element was left
        while i < len(L):
            arr[k] = L[i]
            i += 1
            k += 1

        while j < len(R):
            arr[k] = R[j]
            j += 1
            k += 1


def hybridSort(lst):
    if(len(lst) < 86):
        selectionSort(lst)
    else:
        mergeSort(lst)
```

```
if __name__ == "__main__":



    size = int(input('Enter an integer between 3 and 200!\n'))

    #generates random array lst based on the input size
    lst = arrGenerator(size)
    print('Here is an unsorted array of size ', size,'!', sep="")
    print(lst)

    #calls the sort algorithm
    hybridSort(lst)


    if(size < 86):
        print('Here is the sorted list using Selection Sort!')
    else:
        print('Here is the sorted list using Merge Sort!')
    print(lst)
```