NAME:	Prithvi Singh
UID:	2022301014
SUBJECT	DAA
EXPERIMENT NO:	2b
AIM:	Understanding more concepts regarding quick sort algorithm
Algorithm:	• Quick Sort Algorithm: 1. Pick a pivot element from the array (usually the first or last element). 2. Partition the array around the pivot element by rearranging the elements so that all elements smaller than the pivot come before it, and all elements larger than the pivot come after it. 3. Recursively apply steps 1 and 2 to the sub-array of elements smaller than the pivot, and the sub-array of elements larger than the pivot. 4. The base case of the recursion is when the sub-array has fewer than two elements, in which case it is already sorted.

Code:	`#include <stdio.h>`
	`#include <stdlib.h>`
	`#include <time.h>`
	`long SWAP = 0;`
	
	`int quicksort(int a[], int start, int end)`
	`{`
	`int pivot = a[end];`
	`//int pivot = a[start];`
	`//int random = start + rand() % (end - start);`
	`//int pivot = a[random];`
	`//int mid = start + (end - start)/2;`
	`//int pivot = a[mid];`
	`int i = (start - 1);`
	`for (int j = start; j <= end - 1; j++)`
	`{`
	`if (a[j] < pivot)`
	`{`
	`i++;`

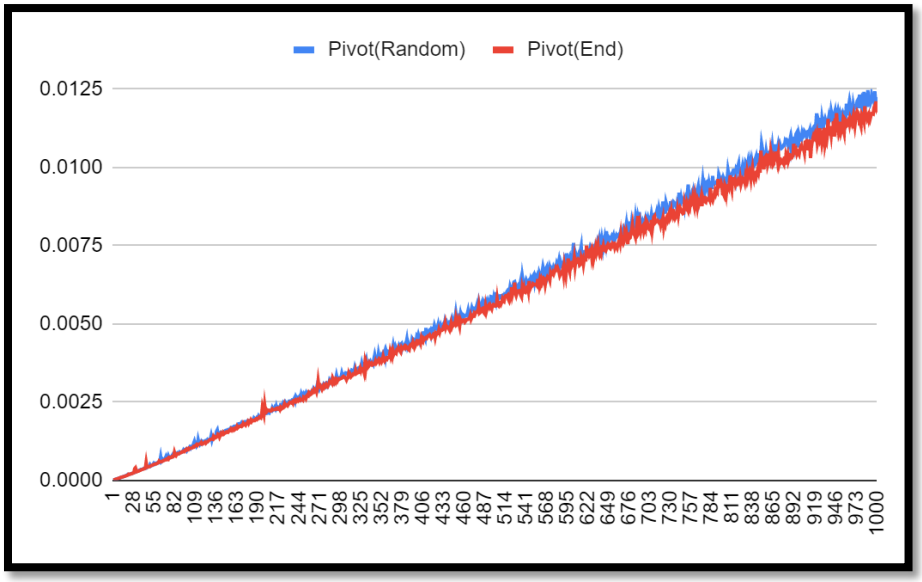
```
int t = a[i];
a[i] = a[j];
a[j] = t;
SWAP++;
}
}
int t = a[i + 1];
a[i + 1] = a[end];
a[end] = t;
SWAP++;
return (i + 1);
}
double quick(int a[], int start, int end)
{
if (start < end)
{
int p = quicksort(a, start, end);
quick(a, start, p - 1);
quick(a, p + 1, end);
}
}
int main()
{
double qust,mest;
srand(time(0));

FILE *fp,*file;
fp = fopen("random.txt", "w");
for (int i = 0; i < 100000; i++)
{
fprintf(fp, "%d\n", rand() % 900001 + 100000);
}
int upper_limit = 100;
fclose(fp);
file = fopen("output.txt","w");
fprintf(file,"Block\tQuickSort\tSwaps\n");
for (int i = 0; i < 1000; i++)
{
fp = fopen("random.txt", "r");
int arr1[upper_limit], arr2[upper_limit], temp_num;
for (int j = 0; j < upper_limit; j++)
{
fscanf(fp, "%d", &temp_num);
arr1[j] = temp_num;
```

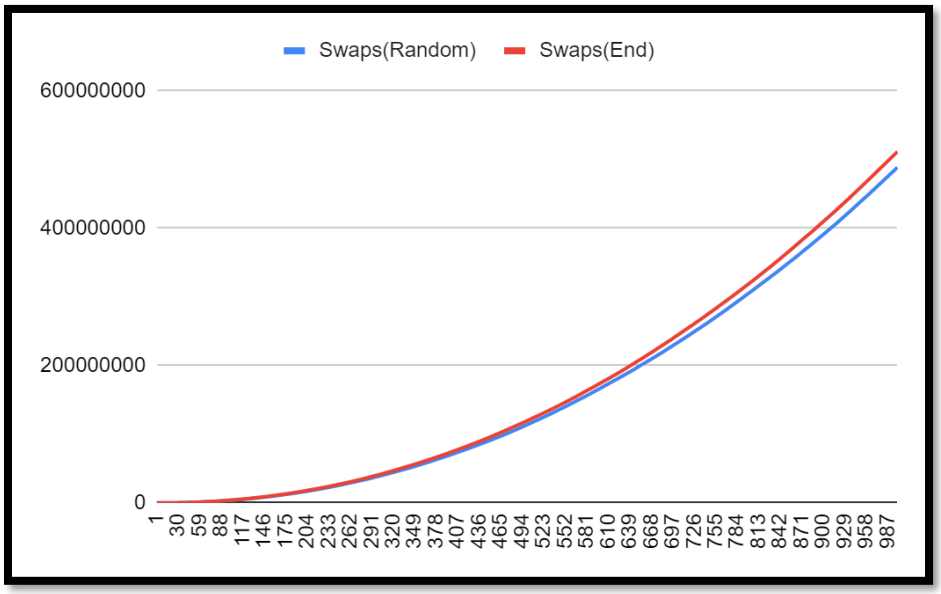
```
arr2[j] = temp_num;
}
fclose(fp);
clock_t t1;
t1 = clock();
qust=quick(arr1,0,upper_limit-1);
t1 = clock() - t1;
qust = ((double)t1) / CLOCKS_PER_SEC;
fprintf(file,"%d\t%lf\t%lf\t%ld\n",i+1, mest, qust,SWAP);
fflush(stdout);
upper_limit += 100;
}
return 0;
}
```

Graphs and Observation:

- **<u>Now considering Different Pivot Positions:</u>**



Here we can understand that even after considering different pivot positions the time complexity of quick sort is almost same. We can clearly accumulate that quick sort when pivot is at random position takes more time than compared to when pivot is at end position at the end of execution. Although both executions are completed within 0.1 seconds.

	• **Count of swaps considering different pivot positions:**  Here we can understand that Number of swaps required for quick sort when pivot is at random position is less than for quick sort when pivot is at end position. The average number of swaps are almost more than 500000000. The number of swaps is constantly increasing throughout the whole execution.
Conclusion:	Thus, we have provided observations for different pivots for quick sort algorithms.