MERGE SORT

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void merge(int a[], int low, int mid, int high);
void merge_sort(int a[], int low, int high);

int main() {
    int i, n;
    clock_t start, end;
    double time_taken;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int *a = (int *)malloc(n * sizeof(int));
    if (a == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }

    printf("Enter the array elements: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    start = clock();
    merge_sort(a, 0, n - 1);
    end = clock();

    time_taken = (double)(end - start) / CLOCKS_PER_SEC;

    printf("Sorted array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");

    printf("Time taken to sort: %f seconds\n", time_taken);

    free(a);
    return 0;
}

void merge_sort(int a[], int low, int high) {
    if (low < high) {
        int mid = (low + high) / 2;
        merge_sort(a, low, mid);
```

```
        merge_sort(a, mid + 1, high);
        merge(a, low, mid, high);
    }
}

void merge(int a[], int low, int mid, int high) {
    int i = low, j = mid + 1, k = 0;
    int *c = (int *)malloc((high - low + 1) * sizeof(int));
    if (c == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }

    while (i <= mid && j <= high) {
        if (a[i] < a[j]) {
            c[k++] = a[i++];
        } else {
            c[k++] = a[j++];
        }
    }

    while (i <= mid) {
        c[k++] = a[i++];
    }

    while (j <= high) {
        c[k++] = a[j++];
    }

    for (i = 0; i < k; i++) {
        a[low + i] = c[i];
    }

    free(c);
}
```

OUTPUT:

```
Enter the number of elements: 5
Enter the array elements: 1 3 7 67 0
Sorted array: 0 1 3 7 67
Time taken to sort: 0.000000 seconds
```

GRAPH:

	A	B	C	D	E	F	G	H	I	J	K
1	array size	time									
2	500	0									
3	1500	0.001									
4	2500	0.002									
5	3500	0.005									
6	4500	0.007									
7	5500	0.011									
8	6500	0.015									
9	7500	0.02									
10	8500	0.024									
11	9500	0.031									
12	10500	0.038									
13	11500	0.047									
14	12500	0.055									
15	13500	0.065									
16	14500	0.075									
17											
18											
19											
20