```
#include <bits/stdc++.h>
using namespace std;

void merge(int gArray[], int low, int mid1, int mid2, int high, int destArray[]){
	int i = low, j = mid1, k = mid2, l = low;
	//Choose smaller of the smallest in the three ranges
	while((i < mid1) && (j < mid2) && (k < high)){
		if(gArray[i] < gArray[j]){
			if(gArray[i] < gArray[k]){
				destArray[l++] = gArray[i++];
			}
			else{
				destArray[l++] = gArray[k++];
			}
		}
		else{
			if(gArray[j] < gArray[k]){
				destArray[l++] = gArray[j++];
			}
			else{
				destArray[l++] = gArray[k++];
			}
		}
	}
	while((i < mid1) && (j < mid2)){
		if(gArray[i] < gArray[j]){
			destArray[l++] = gArray[i++];
		}
		else{
			destArray[l++] = gArray[j++];
		}
	}
	while((j < mid2) && (k < high)){
		if(gArray[j] < gArray[k]){
			destArray[l++] = gArray[j++];
		}
		else{
			destArray[l++] = gArray[k++];
		}
	}
	while((i < mid1) && (k < high)){
		if(gArray[i] < gArray[k]){
```

```
                destArray[l++] = gArray[i++];
            }
            else{
                destArray[l++] = gArray[k++];
            }
        }
        while (i < mid1)
            destArray[l++] = gArray[i++];
        while (j < mid2)
            destArray[l++] = gArray[j++];
        while (k < high)
            destArray[l++] = gArray[k++];
}

void mergeSort3WayRec(int gArray[], int low, int high, int destArray[]){
        if (high - low < 2) return; //If array size is 1 then do nothing

        int mid1 = low + ((high - low) / 3);
        int mid2 = low + 2 * ((high - low) / 3) + 1;
        // Sorting 3 arrays recursively
        mergeSort3WayRec(destArray, low, mid1, gArray); //T(n/3)
        mergeSort3WayRec(destArray, mid1, mid2, gArray); //T(n/3)
        mergeSort3WayRec(destArray, mid2, high, gArray); //T(n/3)
        //Merge the sorted arrays
        merge(destArray, low, mid1, mid2, high, gArray); //n
}

void mergeSort3Way(int gArray[], int n){
        if (n == 0) return;

        int fArray[n];
        for (int i = 0; i < n; i++) //n
            fArray[i] = gArray[i];

        mergeSort3WayRec(fArray, 0, n, gArray);

        for (int i = 0; i < n; i++) //n
            gArray[i] = fArray[i];
}

int main(){
        int data[] = {45, -2, -45, 78, 30, -42, 10, 19, 73, 93};
```

```
    mergeSort3Way(data,10);
    cout << "Sorted: ";
    for (int i = 0; i < 10; i++){ //n
        cout << data[i] << " ";
    }

    return 0;
}
```

n is the size of the list

Space complexity

陣列 data 需要 n 個空間，陣列 fArray 需要 n 個空間，其他變數各 1 個空間，共 c 個空間。

O(n+n+c) = n

Time complexity

Best case:

當 list 裡的資料只有一筆的時候，時間複雜度為 O(1)

Wosrt case

如果三個小陣列加起來有 n 個數字，總共就需要 n 個步驟。

每回合的 merge 需要花：O(n)，總共需要回合數：O(log_3^n)。

Divide 的步驟數為 n-1，合併的步驟數為$n * log_3^n$，相加後可以得知總共的步驟數為 $n - 1 + nlog_3^n$。

$O(n - 1 + nlog_3^n + C) = nlog_3^n$