sorting algorithms

bubbleSorting(int\* original\_list, int numofkeys){

for I = 0 to numofkeys-1

{

For j = i to numofkeys{

If original\_list[i] > original\_list[j]{

Int temp = original\_list[i];

original\_list[i] = original\_list[j];

original\_list[j] = temp;

//swapping original\_list[i] and original\_list[j]

}

}

}

}

My sorting algorithm

void yourSecondSorting(int\* original\_list, int numofkeys)

{

// Get maximum element

int max = getMaxnum(original\_list, numofkeys);

// Apply counting sort to sort elements based on place value.

for int digit = 1 to max/digit>0 when digit \*= 10){

int result[numofkeys];

int count[10];

for int i = 0 to 9{

count[i] = 0; //initializes count array

}

for int i = 0 to numofkeys-1{

count[(original\_list[i] / digit) % 10]++; //counting each ith digit

}

for int i = 1 to 9{

count[i] += count[i - 1]; //i-th count = i-th count + i-1 - th count

}

for int i = 0 to numofkeys -1 {

result[count[(original\_list[numofkeys-i-1] / digit) % 10] - 1] = original\_list[numofkeys-i-1];

count[(original\_list[numofkeys-i-1] / digit) % 10]--;

} //sort numbers

for int i = 0 to i < numofkeys -1{

original\_list[i] = result[i]; //save result[i] on original\_list[i]

}

}

}

code

#pragma warning(disable :4996)

#include <stdio.h>

#include <time.h>

int keylist[100000];

int keylist\_sorted[100000];

char filename\_keylist[128] = "keylist.txt";

char filename\_keylist\_sorted[128] = "keylist\_sorted.txt";

int sortedlist[100000];

int check\_sorted(int\* sorting\_result);

void bubbleSorting(int\* original\_list, int numofkeys);

void yourSecondSorting(int\* original\_list, int numofkeys);

int getMaxnum(int array[], int n);

void sorting(int\* original\_list, int numofkeys);

// verify your sorting results with the validation sequence

int check\_sorted(int\* sorting\_result)

{

    int verfiedResult = 0;

    // Read the sorted result of the test sequence

    FILE\* fs;

    fs = fopen(filename\_keylist\_sorted, "r");

    if (fs == NULL)

    {

        printf("The test seqeunce file (%s) is not accessible\n", filename\_keylist\_sorted);

        return verfiedResult;

    }

    int index = 0;

    while (fscanf(fs, "%d", &keylist\_sorted[index]) == 1)

    {

        if (sorting\_result[index] == keylist\_sorted[index])

        {

            verfiedResult++;

        }

        index++;

    }

    fclose(fs);

    return verfiedResult;

}

// program your bubble sorting algorithm

void bubbleSorting(int\* original\_list, int numofkeys)

{

    for (int i = 0; i<numofkeys; i++){

        for (int j = i; j<numofkeys; j++){

            if(original\_list[i]>original\_list[j]){

                int temp = original\_list[i];

                original\_list[i] = original\_list[j];

                original\_list[j] = temp;

            }

        }

    }

}

// program your second sorting algorithm

void yourSecondSorting(int\* original\_list, int numofkeys)

{

    // Get maximum element

    int max = getMaxnum(original\_list, numofkeys);

    // Apply counting sort to sort elements based on place value.

    for (int digit = 1; max/digit > 0; digit \*= 10){

        int result[numofkeys];

        int count[10];

        for (int i = 0; i < 10; i++){

            count[i] = 0;   //initializes count array

        }

        for (int i = 0; i < numofkeys; i++){

            count[(original\_list[i] / digit) % 10]++;   //counting each ith digit

        }

        for (int i = 1; i < 10; i++){

            count[i] += count[i - 1];   //i-th count = i-th count + i-1 - th count

        }

        for (int i = 0; i < numofkeys; i++) {

            result[count[(original\_list[numofkeys-i-1] / digit) % 10] - 1] = original\_list[numofkeys-i-1];

            count[(original\_list[numofkeys-i-1] / digit) % 10]--;

        }   //sort numbers

        for (int i = 0; i < numofkeys; i++){

            original\_list[i] = result[i];   //save result[i] on original\_list[i]

        }

    }

}

int getMaxnum(int array[], int n) {

    int maxnum = array[0];

    for (int i = 1; i < n; i++){

        if (array[i] > maxnum){

            maxnum = array[i];

        }

    }

    return maxnum;

}

/\*

// Using counting sort to sort the elements in the basis of significant places

void countingSort(int array[], int size, int place) {

}

\*/

// Main function to implement radix sort

// Main function to implement radix sort

void sorting(int\* original\_list, int numofkeys)

{

    //bubbleSorting(original\_list, numofkeys);

    yourSecondSorting(original\_list, numofkeys);

    return ;

}

int main()

{

    // read the test sequesnce

    int numofkeys = 0;

    FILE\* fs = fopen(filename\_keylist, "r");

    if (fs == NULL)

    {

        printf("The test seqeunce file (%s) is not accessible\n", filename\_keylist);

        return 0;

    }

    while (fscanf(fs, "%d", &keylist[numofkeys]) == 1)

    {

        numofkeys++;

    }

    fclose(fs);

    // begin sorting with the test sequence

    clock\_t start, end;

    start = (double)clock();

    //sorting(keylist, numofkeys);

    sorting(keylist, numofkeys);

    end = (double)clock();

    // compute the time of sorting

    float computationTime = (double)(end - start); // get the total time cost

    printf("Total time cost(ms) : %lf \n", computationTime);

    // verify with the validation sequence

    int verifiedResult = check\_sorted(keylist);

    printf("=============== RESULT =============== \n");

    if (verifiedResult == numofkeys)

    {

        printf("Your sorting algorithm resulted in the correct ascending order for the given list \n\n");

    }

    else

        printf("Your sorting algorithm failed to produce the correct ascending order for the given list \n\n");

}