

# 알고리즘 과제

## Practice.04

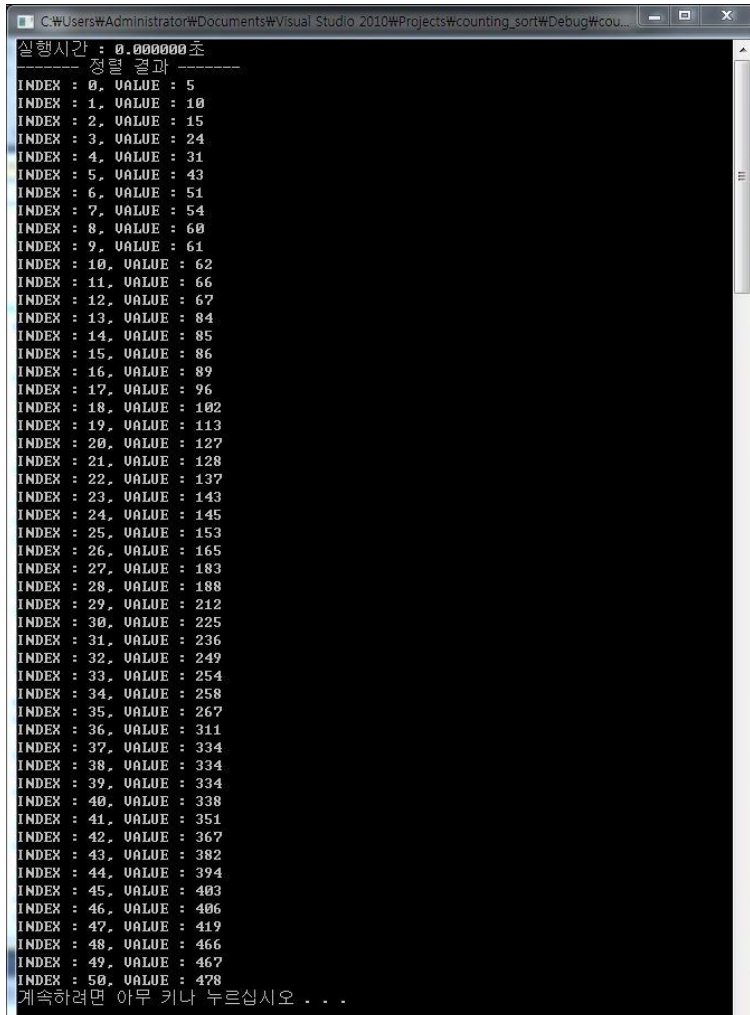
학번 : 201402432

이름 : 조디모데

# Counting Sort.

- Input\_Small

- 출력 결과



```
C:\Users\Administrator\Documents\Visual Studio 2010\Projects\counting_sort\Debug#cou...
실행시간 : 0.000000 초
----- 정렬 결과 -----
INDEX : 0, VALUE : 5
INDEX : 1, VALUE : 10
INDEX : 2, VALUE : 15
INDEX : 3, VALUE : 24
INDEX : 4, VALUE : 31
INDEX : 5, VALUE : 43
INDEX : 6, VALUE : 51
INDEX : 7, VALUE : 54
INDEX : 8, VALUE : 60
INDEX : 9, VALUE : 61
INDEX : 10, VALUE : 62
INDEX : 11, VALUE : 66
INDEX : 12, VALUE : 67
INDEX : 13, VALUE : 84
INDEX : 14, VALUE : 85
INDEX : 15, VALUE : 86
INDEX : 16, VALUE : 89
INDEX : 17, VALUE : 96
INDEX : 18, VALUE : 102
INDEX : 19, VALUE : 113
INDEX : 20, VALUE : 127
INDEX : 21, VALUE : 128
INDEX : 22, VALUE : 137
INDEX : 23, VALUE : 143
INDEX : 24, VALUE : 145
INDEX : 25, VALUE : 153
INDEX : 26, VALUE : 165
INDEX : 27, VALUE : 183
INDEX : 28, VALUE : 188
INDEX : 29, VALUE : 212
INDEX : 30, VALUE : 225
INDEX : 31, VALUE : 236
INDEX : 32, VALUE : 249
INDEX : 33, VALUE : 254
INDEX : 34, VALUE : 258
INDEX : 35, VALUE : 267
INDEX : 36, VALUE : 311
INDEX : 37, VALUE : 334
INDEX : 38, VALUE : 334
INDEX : 39, VALUE : 334
INDEX : 40, VALUE : 338
INDEX : 41, VALUE : 351
INDEX : 42, VALUE : 367
INDEX : 43, VALUE : 382
INDEX : 44, VALUE : 394
INDEX : 45, VALUE : 403
INDEX : 46, VALUE : 406
INDEX : 47, VALUE : 419
INDEX : 48, VALUE : 466
INDEX : 49, VALUE : 467
INDEX : 50, VALUE : 478
계속하려면 아무 키나 누르십시오 . . .
```

- 알고리즘 설명

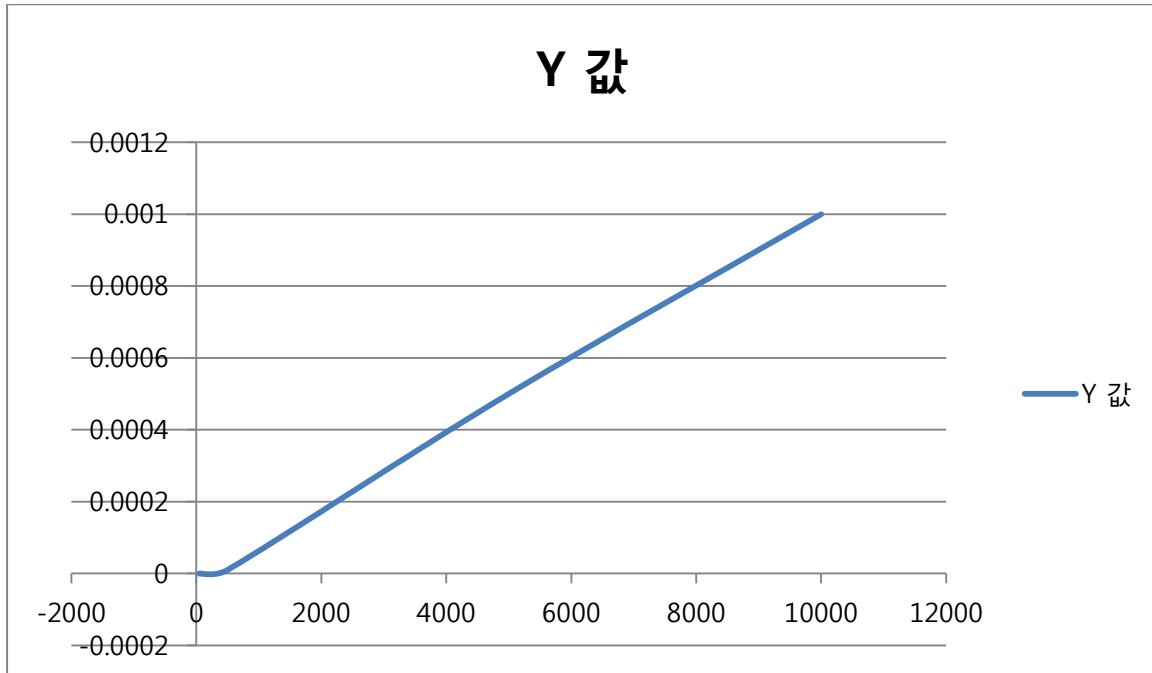
1. 배열의 가장 큰 값을 크기로 갖는 배열을 만든다.
2. 배열의 index 와 같은 값을 가진 수를 세어 해당 index 에 저장한다.
3. 배열의 index 를 증가시키며 배열(index) += 배열(index-1) 이런식으로 누적하여 더한 값을 배열에 저장한다.
4. 누적된 값의 차이를 이용하여 정렬된 배열을 구한다

- 컴파일 방법

input 폴더를 다음 위치에 넣는다

C:\Users\Administrator\Desktop\input\input50.txt

● 수행시간을 그래프



Input

50 : 0

500 : 0.00

5000 : 0.00005

10000 : 0.001

## ● Code ( .C )

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <memory.h>

int getMaxValue(int arr[], int length)
{
    int maxValue, i ;

    maxValue = 0 ;
    for(i=0 ; i<length ; i++){
        if(arr[i] > maxValue)
            maxValue = arr[i] ;
    }
    return maxValue ;
}

int* countingSort(int maxValue, int arr[], int length)
{
    int i, j, value, arrIndex, count ;
    int* bucket ;

    bucket = (int*)malloc(sizeof(int)*(maxValue+1)) ;
    memset(bucket, 0, sizeof(int)*maxValue+1) ;

    for(i=0 ; i<=length ; i++){
        bucket[ arr[i] ]++ ;
    }

    for(i=1 ; i<maxValue ; i++){
        bucket[i] += bucket[i-1] ;
    }

    arrIndex = 0 ;
    for(i=1 ; i<maxValue ; i++){
        if(bucket[i]!=bucket[i-1]){
            for(j=bucket[i] ; j>bucket[i-1] ; j--){
                arr[arrIndex] = i ;
                arrIndex ++ ;
            }
        }
    }

    return arr ;
}

void printArray(int* arr, int length)
{
    int i ;
    for(i=0 ; i<length ; i++){
        //if(arr[i]!=0 && arr[i]>=0)
```

```

        printf("INDEX : %d, VALUE : %d\n",i,arr[i]) ;
    }

}

// main function
int main(void)
{
    int i,n, maxValue, num[51] ;
    FILE *fps ;
    int temp = 0 ;
    int *bucket ;
    clock_t start,end ;

    n = 51 ;

    // input의 숫자들을 i 배열에 num에 저장하나는 부분
    fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input50.txt","rt");
    for(i=0; i <= n ; i++){
        fscanf(fps,"%d",&temp) ;
        num[i]=temp ;
    }
    fclose(fps) ;

    // 정렬의 시작 시간 i 저장
    start = clock() ;

    maxValue = getMaxValue(num, sizeof(num)/sizeof(int)) ;
    bucket = (int*)malloc(sizeof(int)*(maxValue+1)) ;
    // 정렬의 하나는 부분
    bucket = countingSort(maxValue, num, n) ;
    // 정렬 후 시간 i 저장
    end = clock() ;

    // 실행 시간 i 출력
    printf("실행시간 : %f초\n",(end-start)/(double)1000) ;

    // 정렬 결과 출력
    printf("----- 정렬 결과 ----- \n") ;
    printArray(bucket, n) ;

    system("pause") ;

    return 0;
}

```

# Radix Sort.

- Input50

- 출력 결과

- 알고리즘 설명

- 숫자의 특정 자릿수만을 보고 정렬하는 방법이다.

- 최대 자리수가 3 이라고 하면 우선 1 의 자리, 10 의 자리, 100 의 자리 순으로 오름차순 정렬을 한다.

- 컴파일 방법

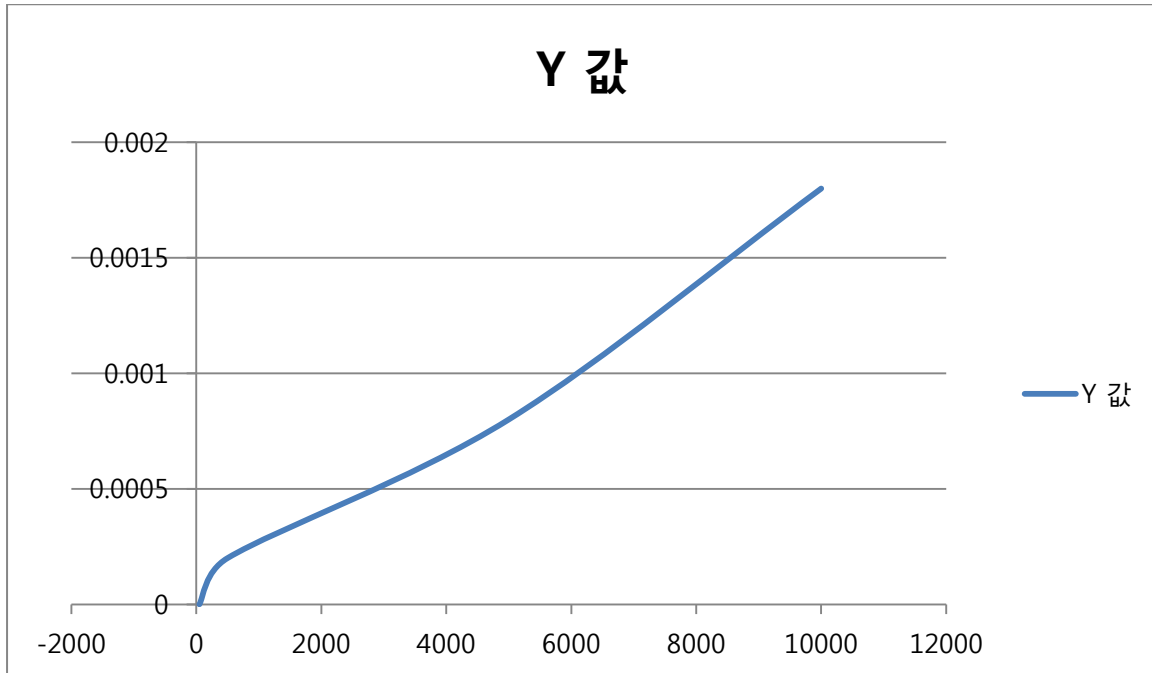
- input 폴더를 바탕화면에 넣는다

- C:\Users\Administrator\Desktop\input\input50.txt

- 원하는 데이터의 크기에 맞게 num[크기+1] 배열 초기화

- n을 원하는 크기에 맞게 설정

● 수행시간을 그래프



Input

50 : 0.0000

500 : 0.0002

5000 : 0.0008

10000 : 0.0018

## ● Code ( .C )

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <memory.h>
#include <math.h>

void radixSort(int *data, int size, int p, int k) {
    int *count, *tmp ;
    int index, pval, i, j, n;

    if ( (count=(int*)malloc(k*sizeof(int))) == NULL ){
        memset(count, 0, sizeof(int)*k) ;
        return;
    }
    if ( (tmp=(int*)malloc(size*sizeof(int))) == NULL ){
        memset(tmp, 0, sizeof(int)*size) ;
        return;
    }
    for (n=0; n<p; n++) {
        for (i=0; i<k; i++)
            count[i] = 0;
        // 위 i × 치에 계산을
        pval = (int)pow((double)k, (double)n);

        for (j=0; j<size; j++) {
            index = (int)(data[j] / pval) % k;
            count[index] = count[index] + 1;
        }

        for (i=1; i<k; i++) {
            count[i] = count[i] + count[i-1];
        }

        // 계수를 정렬 방식에
        for (j=size-1; j>=0; j--) {
            index = (int)(data[j] / pval) % k;
            tmp[count[index] - 1] = data[j];
            count[index] = count[index] - 1;
        }

        memcpy(data, tmp, size * sizeof(int));
    }

    free(count) ;
    free(tmp) ;
}

int count_large(int arr[], int n){
    int i, j, big ;
```



```

        big = arr[0] ;
        for(i=1 ; i<n ; i++){
            if(big < arr[i])
                big = arr[i] ;
        }

        j=0 ;
        while(big != 0){
            big= big/10 ;
            j++ ;
        }

        return j ;
    }
}

void printArray(int* arr,int length)
{
    int i ;
    for(i=0 ; i<length ; i++){
        printf("INDEX : %d, VALUE : %d\n",i,arr[i]) ;
    }
}

int main(void)
{
    int i,n, maxValue, num[500] ;
    FILE *fps ;
    int temp = 0 ;
    clock_t start,end ;

    n = 500 ;

    // input의 숫자들을 i 배열에 num에 저장하기는 불가능
    fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input500.txt","rt");
    for(i=0; i <= n ; i++){
        fscanf(fps,"%d",&temp) ;
        num[i]=temp ;
    }
    fclose(fps) ;

    temp = count_large(num, n) ;

    // 정렬의 시작 시간에 i 저장
    start = clock() ;

    // 정렬하기는 불가능
    radixSort(num, n, temp, 10) ; // (배열에, 배열에 크기의 a, 최대값
자리를,진수)

    // 정렬의 후 시간
    end = clock() ;

```

```

// 실행 시간 출력
printf("실행 시간 : %f초\n", (end-start)/(double)1000) ;

// 정렬 결과 출력
printf("----- 정렬 결과 ----- \n") ;
//printArray(num, n) ;

system("pause") ;

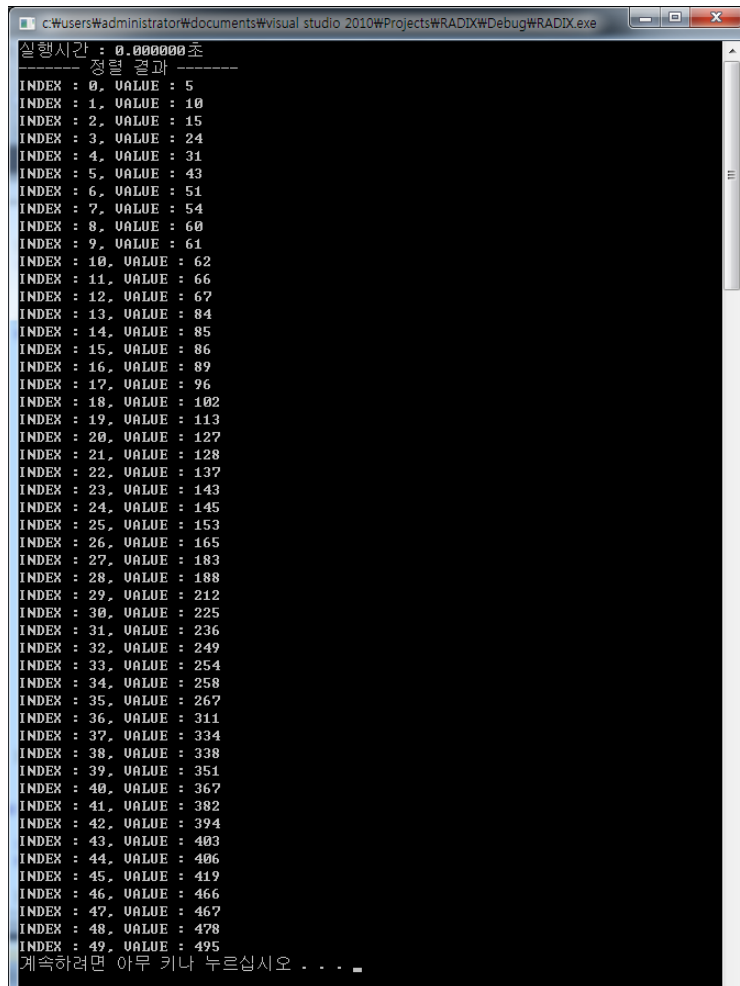
return 0;
}

```

## Bucket sort.

- Input50

- 출력 결과



```
c:\users\administrator\documents\visual studio 2010\Projects\RADIX#\Debug\RADIX.exe
실행시간 : 0.000000 초
----- 정렬 결과 -----
INDEX : 0, VALUE : 5
INDEX : 1, VALUE : 10
INDEX : 2, VALUE : 15
INDEX : 3, VALUE : 24
INDEX : 4, VALUE : 31
INDEX : 5, VALUE : 43
INDEX : 6, VALUE : 51
INDEX : 7, VALUE : 54
INDEX : 8, VALUE : 60
INDEX : 9, VALUE : 61
INDEX : 10, VALUE : 62
INDEX : 11, VALUE : 66
INDEX : 12, VALUE : 67
INDEX : 13, VALUE : 84
INDEX : 14, VALUE : 85
INDEX : 15, VALUE : 86
INDEX : 16, VALUE : 89
INDEX : 17, VALUE : 96
INDEX : 18, VALUE : 102
INDEX : 19, VALUE : 113
INDEX : 20, VALUE : 127
INDEX : 21, VALUE : 128
INDEX : 22, VALUE : 137
INDEX : 23, VALUE : 143
INDEX : 24, VALUE : 145
INDEX : 25, VALUE : 153
INDEX : 26, VALUE : 165
INDEX : 27, VALUE : 183
INDEX : 28, VALUE : 188
INDEX : 29, VALUE : 212
INDEX : 30, VALUE : 225
INDEX : 31, VALUE : 236
INDEX : 32, VALUE : 249
INDEX : 33, VALUE : 254
INDEX : 34, VALUE : 258
INDEX : 35, VALUE : 267
INDEX : 36, VALUE : 311
INDEX : 37, VALUE : 334
INDEX : 38, VALUE : 338
INDEX : 39, VALUE : 351
INDEX : 40, VALUE : 367
INDEX : 41, VALUE : 382
INDEX : 42, VALUE : 394
INDEX : 43, VALUE : 403
INDEX : 44, VALUE : 406
INDEX : 45, VALUE : 419
INDEX : 46, VALUE : 466
INDEX : 47, VALUE : 467
INDEX : 48, VALUE : 478
INDEX : 49, VALUE : 495
계속하려면 아무 키나 누르십시오 . . .
```

- 알고리즘 설명

배열의 원소들을 수의 범위 별로 구분해 버킷에 넣고 그 버킷 각각을 정렬하는 방법

- 컴파일 방법

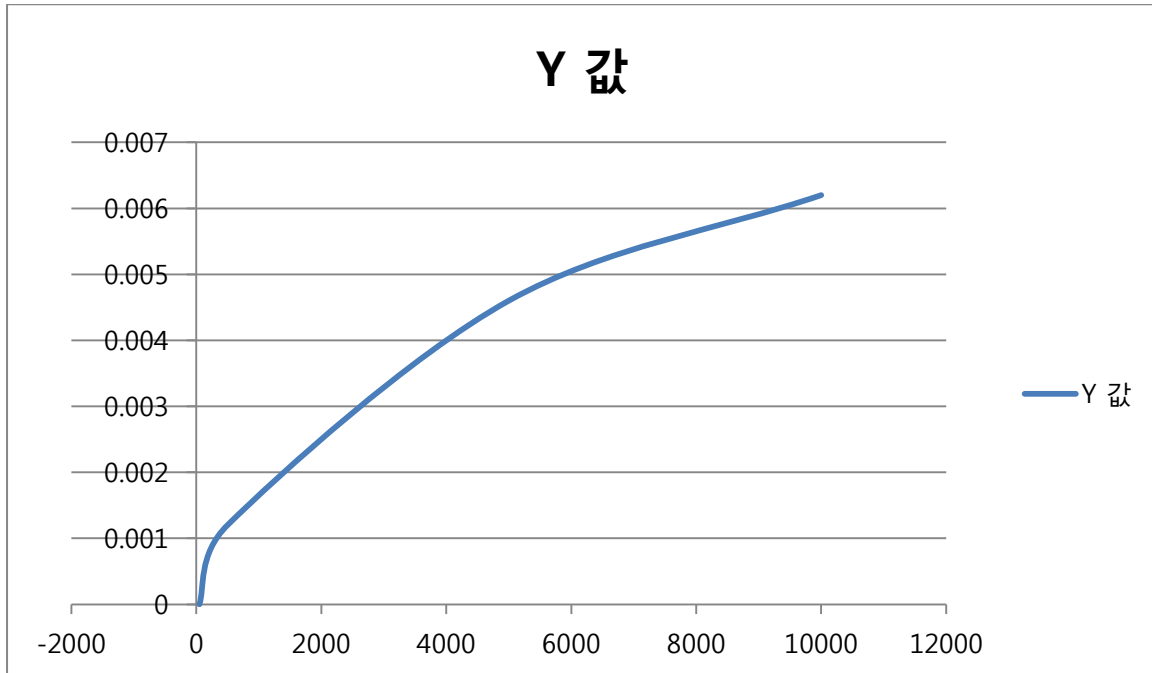
input 폴더를 바탕화면에 넣는다

C:\Users\Administrator\Desktop\input\input50.txt

원하는 데이터의 크기에 맞게 num[크기+1] 배열 초기화

n을 원하는 크기에 맞게 설정

● 수행시간을 그래프



Input

50 : 0.0000

500 : 0.0012

5000 : 0.0046

10000 : 0.0062

## ● Code ( .C )

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

// 리눅스 시스템 호출
typedef struct node {
    int value;
    struct node *link;
} node;

int* bucketSort(int *ar, int size, int max) {
    node *counter, *n2, *n1;
    int *fa, temp;
    int i, j, k=0;
    int n, a;
    fa = (int*)malloc(sizeof(int)*(size));
    memset(fa, 0, sizeof(int)*size);

    max = max*10+1; // count배열의 크기를 10배 늘림
    counter = (node*)malloc(sizeof(node)*(max));

    for(i=0; i<max; i++) { // init
        counter[i].value = 0;
        counter[i].link = 0;
    }

    for(i=0; i<size; i++) {
        // init
        n = ar[i];
        j = n * 100;
        j = j/10;

        // 버킷에 원소 추가
        if(counter[j].value == 0 && counter[j].link == 0)
            counter[j].value = ar[i];
        else {
            // 버킷의 원소 추가
            if(counter[j].link == 0 && counter[j].value != 0) {
                counter[j].link = (node *) malloc(sizeof(node));
                n2 = counter[j].link;
                n2->link = 0;
                n2->value = ar[i];
                continue;
            }

            n2 = counter[j].link;
            while(n2->link != 0) { // 맨 끝까지
                n2 = n2->link;
            }
            n2->link = (node *) malloc(sizeof(node));
        }
    }
}
```

```

        n2 = n2 -> link;
        n2 -> link=0;
        n2 -> value = ar[i];
    }
}

// 버퍼를 i 순서대로 채워 정렬
for(i=0; i<max ; i++) {
    // 버퍼에 노드가 없으면 i경로로 우회
    if(counter[i] . link ==0 && counter[i] . value == 0)
        continue;
    else {
        n1 = &counter[i];
        n2 = &counter[i] ;
        // 버퍼의 원소 개수가 2이상인 경우
        if(n2 -> link != 0) {
            // 버퍼별 정렬
            while(n1!=0) {
                while(n2!= 0) {
                    if(n1 -> value > n2 -> value) {
                        temp =n1 -> value;
                        n1 -> value =n2 -> value;
                        n2 -> value =temp;
                    }
                    n2 = n2 -> link;
                }
                n2 = n1 -> link;
                n1 = n1 -> link;
            }
            n1 = &counter[i];
            // 차이를 대역으로 담기
            for(; n1!=0; k++) {
                fa[k] = n1 -> value;
                n1 = n1 -> link;
            }
        }
        // 버퍼에 하나도 없는 노드가 있는 경우
        else {
            fa[k] = counter[i].value;
            k=k+1;
        }
    }
}
return fa;
}

int getMaxValue(int *arr, int size){
    int max = 0, i ;
    for(i=0 ; i<size ; i++){
        if(arr[i]>max)
            max = arr[i] ;
    }
}

```

```

        return max ;
    }

void printArray(int* arr, int length)
{
    int i ;
    for(i=0 ; i<length ; i++){
        printf("INDEX : %d, VALUE : %d\n",i,arr[i]) ;
    }
}

int main(void) {
    int i,size , maxValue, num[51] ;
    FILE *fps ;
    int temp = 0 ;
    int *result ;
    clock_t start,end ;

    size = 50 ;

    // input의 숫자들을 i 배열에 num에 저장하기는 부분
    fps = fopen("C:\\Users\\Administrator\\Desktop\\input\\input50.txt","rt");
    for(i=0; i <= size ; i++){
        fscanf(fps,"%d",&temp) ;
        num[i]=temp ;
    }
    fclose(fps) ;

    // 정렬의 시작시간에 i 저장
    start = clock() ;

    temp = getMaxValue(num, size) ;
    // 정렬하기는 부분
    result = bucketSort(num, size, temp) ;
    // 정렬 후 시간애 i 저장
    end = clock() ;

    // 실행 시간애 출력
    printf("실행시간애 i : %lf초\n", (end-start)/(double)1000) ;

    // 정렬 결과 출력
    printf("----- 정렬 결과 ----- \n") ;
    printArray(result, size) ;

    free(result);
    system("pause") ;

    return 0;
}

```