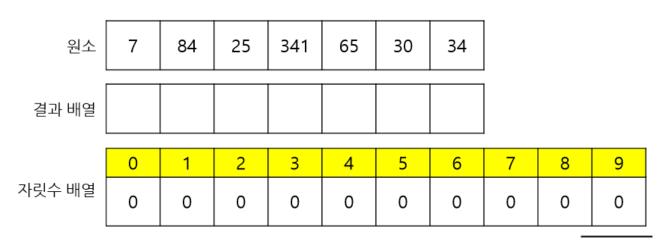
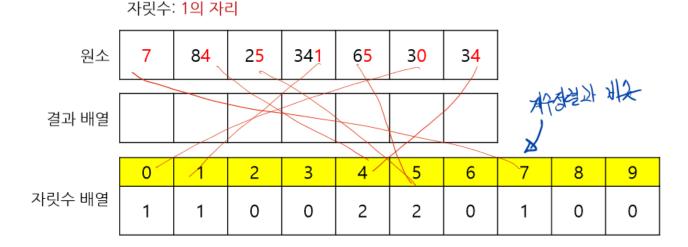
## Radix Sort (기수 정렬)

## 1. 기수 정렬이란?

기수 정렬(Radix Sort)는 자릿수를 기준으로 차례대로 데이터를 정렬하는 알고리즘입니다. 각 데이터 를 자릿수를 기준으로 분류하므로 가장 큰 자릿수를 D라고 했을 때 O(DN)의 시간 복잡도를 가집니다.

기수정렬은 낮은 자리수부터 비교하여 정렬해 간다는 것을 기본 개념으로 하는 정렬 알고리 즘입니다. 기수정렬은 비교 연산을 하지 않으며 정렬 속도가 빠르지만 데이터 전체 크기에 기수 테이블의 크기만한 메모리가 더 필요합니다.







```
#define _CRT_SECURE_NO_WARNINGS
#include "stdio.h"
#include "stdlib.h"
#include <vcruntime_string.h>
#define MAX 10000
void RadixSort(int *pDataSet)
{
       int nSize = _msize(pDataSet) / sizeof(int);
       int nMaxValue = 0;
       // Step 1 : 최대값을 찾는다.
       for (int i = 0; i < nSize; ++i)
        {
               if (pDataSet[i] >= nMaxValue)
                       nMaxValue = pDataSet[i];
       }
       int arrRes[MAX]
       int arrBucket[10] ; // 0 ~ 9
                               ; // 자릿수
       int nExp = 1
       // Step 2 : Radix Sort
       while ((nMaxValue / nExp) > 0)
        {
               memset(arrRes, 0, sizeof(int) * MAX);
               memset(arrBucket, 0, sizeof(int) * 10);
                {
                       // Step 2.1 : 자릿수 배열 처리
```

```
for (int i = 0; i < nSize; ++i)</pre>
                        {
                                arrBucket[pDataSet[i] / nExp %
10]++;
                        }
                        // Step 2.2 : 누적합 계산
                        for (int i = 1; i < 10; ++i)
                        {
                                arrBucket[i] += arrBucket[i -
1];
                        }
                        // Step 2.3 : 정렬 (같은 자릿수 끼리는 순서
를 유지)
                        for (int i = (nSize - 1); i >= 0; --i)
                        {
                                int nIDX = --arrBucket[pDataSe
t[i] / nExp % 10];
                                arrRes[nIDX] = pDataSet[i];
                        }
                        // Step 3 : 기본 배열 갱신
                        for (int i = 0; i < nSize; ++i)
                                pDataSet[i] = arrRes[i];
                }
                nExp *= 10;
        }
}
```

```
int main(void)
{
        int nSize;
        printf("Size : ");
        scanf("%d", &nSize);
        int *pDataSet = (int*)malloc(sizeof(int) * nSize);
        for (int i = 0; i < nSize; ++i)</pre>
        {
                printf("[%d] : ", i + 1);
                scanf("%d", pDataSet + i);
        }
        RadixSort(pDataSet);
        printf("After ----\n");
        for (int i = 0; i < nSize; ++i)</pre>
        {
                printf("[%d] : ", i + 1);
                printf("%d\n", *(pDataSet + i));
        }
        system("pause");
        return 0;
}
```