## PRAKTIKUM ANALISIS ALGORITMA

## **Untuk Memenuhi Tugas**



Disusun oleh:

**Mutia Karimah** 

140810170002

# TEKNIK INFORMATIKA FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM UNIVERSITAS PADJADJARAN

2019

# **Tugas 2**

# **CountingSort.cpp**

```
#include<iostream>
using namespace std;
int k=0;
/*Method to sort the array*/
void Counting_Sort(int A[],int B[],int n)
{
       int C[k];
       for(int i=0;i<k+1;i++)
       {
              /*It will initialize the C with zero*/
              C[i]=0;
       }
       for(int j=1; j<=n; j++)
       {
              /*It will count the occurence of every element x in A
               and increment it at position x in C*/
              C[A[j]]++;
       }
       for(int i=1;i<=k;i++)
       {
```

```
/*It will store the last
               occurence of the element i */
               C[i]+=C[i-1];
       }
       for(int j=n;j>=1;j--)
       {
               /*It will place the elements at their
               respective index*/
               B[C[A[j]]]=A[j];
               /*It will help if an element occurs
               more than one time*/
               C[A[j]]=C[A[j]]-1;
       }
}
int main()
{
       int n;
       cout << "Masukkan jumlah array :";</pre>
       cin>>n;
       cout << "Masukkan element : ";</pre>
       /*A stores the elements input by user */
       /*B stores the sorted sequence of elements*/
```

```
int A[n],B[n];
    for(int i=1;i<=n;i++)
     {
           cin>>A[i];
           if(A[i]>k)
           {
                  /*It will modify k if an element
                   occurs whose value is greater than k*/
                   k=A[i];
            }
     }
    Counting_Sort(A,B,n);
    /*It will print the sorted sequence on the
    console*/
    cout << "======Hasil Counting Sort=======" << endl;
    for(int i=1;i<=n;i++)
     {
           cout<<B[i]<<" ";
     }
cout<<endl;
    return 0;
```

}

#### Kompleksitas waktu dan big-O:

```
Pseudo-code

Input: A: array [1..n] of integer, k: max (A)

Output: B: array [1..n] of integer

for i = 1 to k do

C[i] = 0

for j = 1 to length(A) do

C[A[j]] = C[A[j]] + 1

for 2 = 1 to k do

C[i] = C[i] + C[i-1]

for j = 1 to length(A) do

B[C[A[j]]] = A[j]

C[A[j]] = C[A[j]] - 1

return B
```

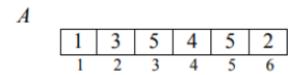
Waktu yang dibutuhkan untuk mengurutkan data menggunakan counting sort bisa didapatkan dari perhitungan sebagai berikut :

- For pertama membutuhkan waktu O(k),
- For kedua membutuhkan waktu O(n),
- For ketiga membutuhkan waktu O(k), dan
- For keempat membutuhkan waktu O(n).

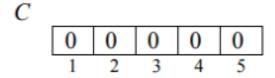
Jadi secara total membutuhkan waktu O(k+n), yang seringkali dianggap k = O(n)

#### Step by step:

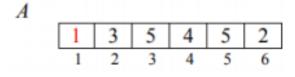
Misal array data yang akan diurutkan adalah A. Counting sort membutuhkan sebuah array C berukuran k, yang setiap elemen C[i] merepresentasikan jumlah elemen dalam A yang nilainya adalah i. Di array inilah penghitungan (counting) yang dilakukan dalam pengurutan ini disimpan. Misal kita akan melakukan pengurutan pada array A sebagai berikut, dengan n adalah 10 dan diasumsikan bahwa rentang nilai setiap A[i] adalah 1..5

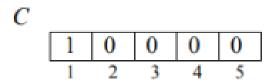


Dan array C setelah diinisialisasi adalah :

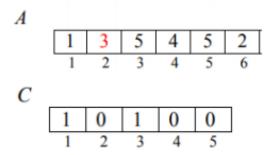


Kemudian proses penghitungan pun dimulai, proses ini linier, dilakukan dengan menelusuri array A, Langkah 1 : pembacaan pertama mendapat elemen A[1] dengan isi 1, maka C[1] ditambah 1.

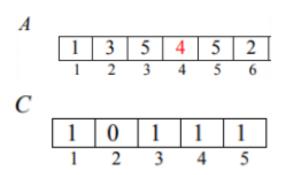




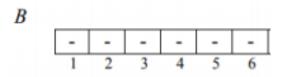
Langkah 2 : pembacaan kedua mendapat elemen A[2] dengan isi 3, maka C[3] ditambah 1.



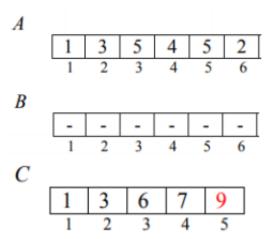
Langkah 3 : pembacaan ketiga mendapat elemen A[3] dengan isi 5, maka C[5] ditambah 1.



Demikian dilakukan terus menerus hingga semua elemen A telah diakses. Lalu array C diproses sehingga setiap elemen C, C[i] tidak lagi merepresentasikan jumlah elemen dengan nilai sama dengan i, namun setiap C[i] menjadi merepresentasikan jumlah elemen yang lebih kecil atau sama dengan i. Dalam proses ini kita mengakses elemen A[i], kemudian memposisikannya di posisi sebagaimana tercatat dalam C[A[i]], kemudian kita mengurangkan C[A[i]] dengan 1, yang dengan jelas untuk memberikan posisi untuk elemen berikutnya dengan yang isinya sama dengan A[i]. Proses ini memerlukan sebuah array bantu B yang ukurannya sama dengan array A, yaitu n. Yang pada awalnya semua B[i] diinisialisasi dengan nil



Langkah 1 : elemen A[10] adalah 2, maka karena C[5] adalah 10, maka B[6] diisi dengan 5, dan C[5] dikurangi 1.



Demikian proses dilakukan hingga elemen A[1] selesai diproses, sehingga didapatkan hasil akhir

## Contoh soal dan running time:

u	Keys Type	Average run-time	Worst case run-time	Extra Space	In Place	Stable
Insertion Sort	Any	O(n2)	O(n2)	O(1)	√.	V
Merge Sort	Any	O(nlogn)	O(nlogn)	O(n)	X	V
Heap Sort	Any	O(nlogn)	O(nlogn)	O(1)	V	X
Quick Sort	Any	O(nlogn)	O(n2)	O(1)	V	X
Counting Sort	integers [1k]	O(n+k)	O(n+k)	O(n+k)	X	4
TPS Sort	integers [1n]	O(n)	O(n)	O(n)	X	V
Radix Sort	d digits in base b	O(d(b+n))	O(d(b+n))	Depends on the stable sort used	Depends on the stable sort used	1
Bucket sort	[0,1)	O(n)	O(n2)	O(n)	X	V.

Count Sort adalah algoritma pengurutan bilangan bulat, bukan algoritma berbasis perbandingan. Sementara setiap algoritma sorting membutuhkan perbandingan  $\Omega(n \log n)$ . Sementara count sort membutuhkan O(n) untuk running timenya. Perubahan nilai k akan mempengaruhi jumlah running time.