Nama : Afie Syahrulloh Arridlo

NPM : 140810170040

COUNTINGSORT

Counting Sort adalah salah satu metode mengurutkan data yang dikenal. Ide dasarnya adalah seperti kita melakukan perhitungan pemilu yaitu dengan mencatat frekuensi atau banyaknya kemunculan data. Namun metode ini hanya cocok digunakan bila data yang digunakan bertipe integer dan dibatasi pada range tertentu.

SourceCode C++ :

```
Nama : Afie Syahrulloh Arridlo
NPM : 140810170040

#include<bits/stdc++.h>
#include<string.h>
using namespace std;
#define RANGE 255

// The main function that sort
// the given string arr[] in
// alphabatical order
void countSort(char arr[])
{
    // The output character array
    // that will have sorted arr
    char output[strlen(arr)];

    // Create a count array to store count of inidividul
    // characters and initialize count array as 0
    int count[RANGE + 1], i;
    memset(count, 0, sizeof(count));

    // Store count of each character
    for(i = 0; arr[i]; ++i)
        ++count[arr[i]];

    // Change count[i] so that count[i] now contains actual
    // position of this character in output array
    for (i = 1; i <= RANGE; ++i)
        count[i] += count[i-1];

    // Build the output character array
    for (i = 0; arr[i]; ++i)
    {
        output[count[arr[i]]-1] = arr[i];
```

```
        --count[arr[i]];
    }

    /*
    For Stable algorithm
    for (i = sizeof(arr)-1; i>=0; --i)
    {
        output[count[arr[i]]-1] = arr[i];
        --count[arr[i]];
    }

    For Logic : See implementation
    */

    // Copy the output array to arr, so that arr now
    // contains sorted characters
    for (i = 0; arr[i]; ++i)
        arr[i] = output[i];
}

// Driver  code
int main()
{
    char arr[] = "akutanpamubutirandebu";

    countSort(arr);

    cout<< "Sorted character array is " << arr;
    return 0;
}
```

Hasil dari program :

```
PS C:\Users\LENOVO\Desktop> g++ -o CountingSort CountingSort.cpp
PS C:\Users\LENOVO\Desktop> & '.\CountingSort.exe'
Sorted character array is aaaabbdeikmnnprttuuuu
```

Algoritma :

Sebagai contoh, diketahui terdapat array AA berukuran NN yang akan diurutkan.

- Lakukan inisialisasi auxillary array Aux[]Aux[] sebagai 00.
  Catatan: Ukuran dari array ini harus ≥max(A[])≥max(A[]).
- Telusuri array AA dan simpan jumlah frekuensi tiap elemen dalam indeks yang sesuai dari AuxAux array, yang artinya, Aux[A[i]]++ dieksekusi untuk tiap ii, di mana ii memiliki rentangan dari [0,N−1][0,N−1].
- Lakukan inisialisasi pada array kosong sortedA[]sortedA[]
- Telusuri array AuxAux dan salin ii ke sortedAsortedA untuk Aux[i]Aux[i] yaitu jumlah di mana 0≤i≤max(A[])0≤i≤max(A[]).

Catatan: Array AA dapat diurutkan menggunakan algoritme ini hanya jika nilai dalam array AAlebih sedikit dari ukuran maksimal array AuxAux. Normalnya, memori dapat dialokasikan hingga dalah hitungan jutaan (106)(106). Jika nilai maksimal dari AA melebihi memori maksimal (atau ukuran yang dialokasikan), penggunaan algoritme ini tidak direkomendasikan. Gunakan algoritme quick sort atau merge sort.

Contoh Soal dan Pembahasan :

Contoh:
Diketahui arr={5,2,9,5,2,3,5}. Dari algoritma diketahui Aux/count akan berukuran 9+19+1 i.e. 1010

Setelah dihitung, count={0,0,2,1,0,3,0,0,0,2}

Perlu diketahui bahwa count[2]=2 yang merepresentasikan jumlah frekeunsi 2 dalam arr[]. Begitu juga count[5]=3 yang merepresentasikan jumlah frekuensi 5 dalam arr[]

Setelah algoritma counting sort digunakan, output[] akan menjadi {2,2,3,5,5,5,9}

Kompleksitas Algoritma :

Array AA akan ditelusuri dalam O(N) waktu dan array yang sudah diurutkan juga akan dihitung dalam O(N) waktu. count[] akan ditelusuri dalam O(K) waktu. Oleh karena itu, kompleksitas waktu keseluruhan algoritma counting sort adalah O(N+K).