TUGAS 3 KELAS

RUNNING TIME

Disusun sebagai salah satu tugas

mata kuliah Analisis Algoritma



Tasya Amanda Adinegara – 140810160003

Shofiyyah Nadhiroh – 14018160057

Patricia Joanne – 140810160065

Dikumpulkan tanggal

1 April 2019

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

2019

**Soal**

1. Program counting sort.
2. Gambarkan algoritma kerja counting sort melalui kotak-kotak array 7 buah.
3. Analisis running timenya dengan mengganti data/spesifikasi hardwarenya.
4. Analisis kompleksitasnya.

**Jawaban**

1. Program counting sort: [cpp.sh/8hw46](http://cpp.sh/8hw46)

#include<iostream>

#include<string.h>

#include<chrono>

using namespace std;

using namespace std::chrono;

void countSort(char arr[]){

char output[strlen(arr)];

int count[255 + 1], i;

memset(count, 0, sizeof(count));

for(i = 0; arr[i]; ++i)

++count[arr[i]];

for (i = 1; i <= 255; ++i)

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

for (i = 0; arr[i]; ++i){

output[count[arr[i]]-1] = arr[i];

--count[arr[i]];

}

for (i = 0; arr[i]; ++i)

arr[i] = output[i];

}

int main(){

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock ::now();

char arr[] = "universitaspadjadjaran";

cout<<"Array sebelum counting sort: "<<arr<<endl;

countSort(arr);

cout<<"Array setelah counting sort: "<<arr<<endl;

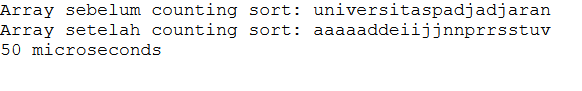
high\_resolution\_clock::time\_point t2 = high\_resolution\_clock ::now();

auto duration = duration\_cast<microseconds>( t2 - t1 ) .count();

cout<<duration<<" microseconds"<<endl;

}

Output:



1. Algoritma counting sort

for *i* ← 1 to *k* do

*c*[*i*] ← 0

for *j* ← 1 to *n* do

*c*[A[*j*]] ← *c*[A[*j*]] + 1

//c[*i*] now contains the number of elements equal to *i*

for *i* ← 2 to *k* do

    c[*i*] ← c[*i*] + c[*i*-1]

// c[*i*] now contains the number of elements ≤ *i*

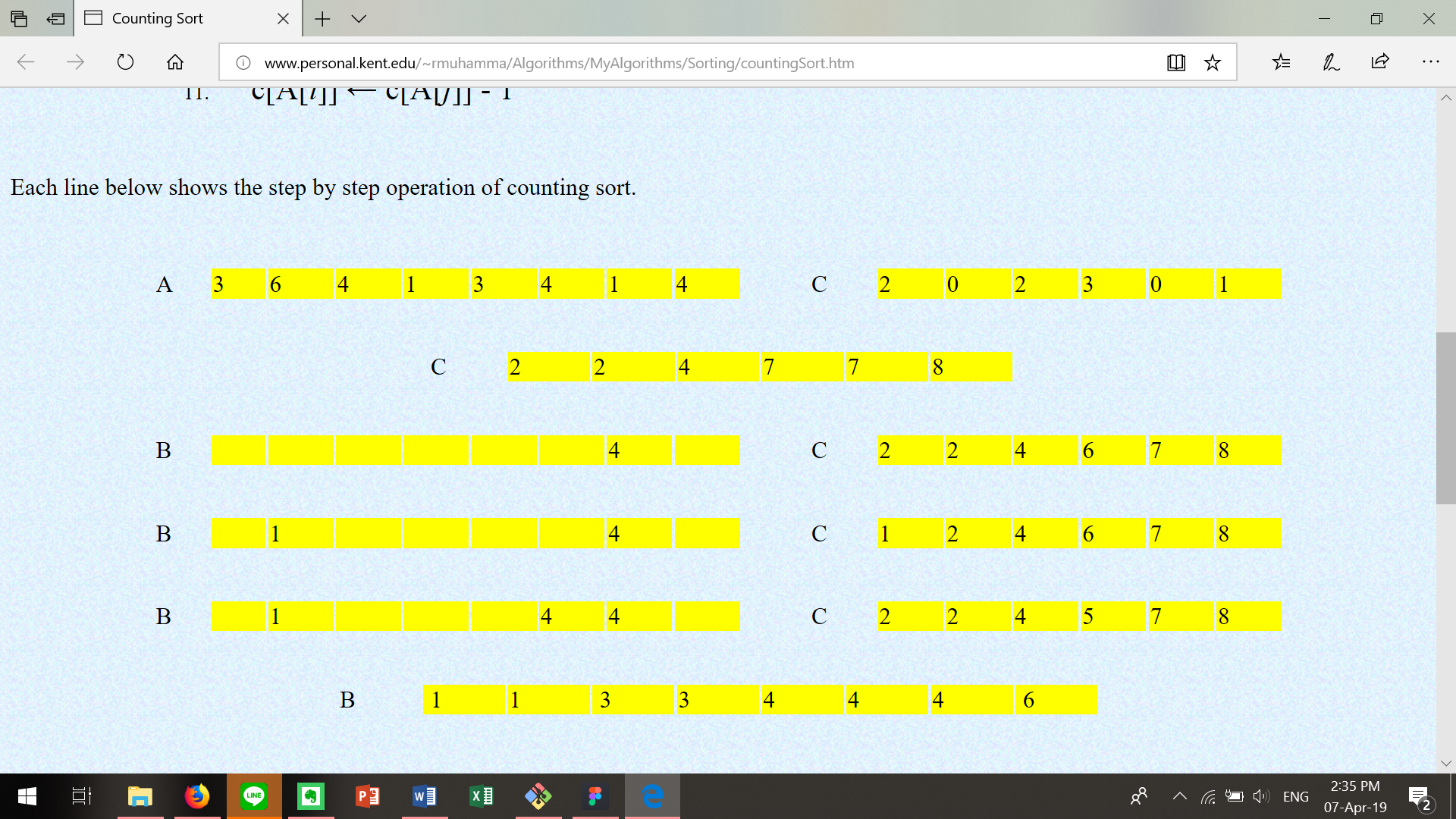
for j ← *n* downto 1 do

    B[c[A[*i*]]] ← A[*j*]

    c[A[*i*]] ← c[A[*j*]] - 1

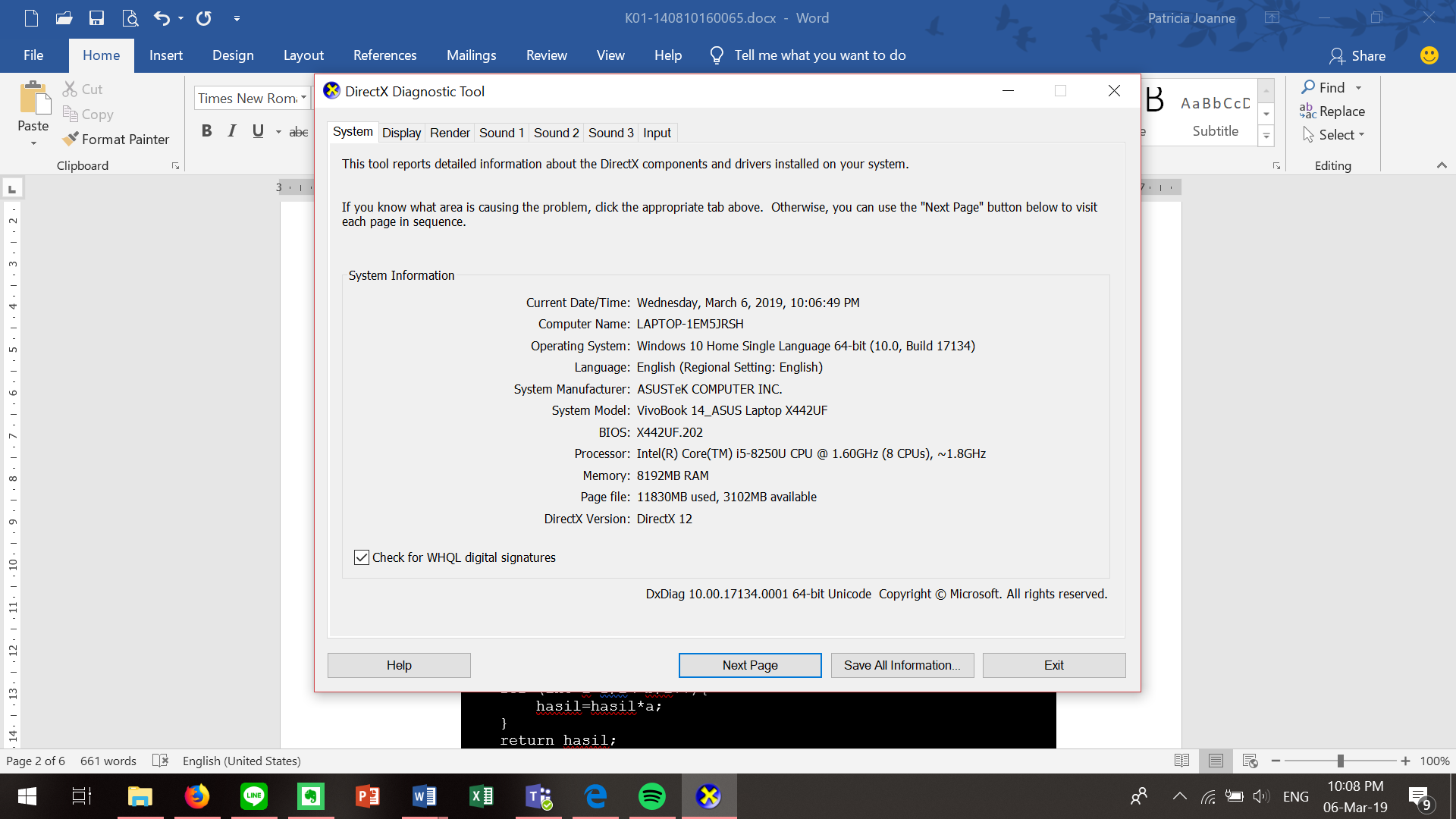
Asumsikan masing-masing elemen adalah bilangan bulat dalam rentang 1 hingga k, untuk beberapa bilangan bulat k. Ketika k = O (n), counting sort berjalan dalam waktu O (n). Ide dasar penghitungan adalah untuk menentukan jumlah elemen kurang dari x untuk setiap elemen input x,. Informasi ini dapat digunakan untuk menempatkan langsung ke posisi yang benar. Misalnya jika ada 17 elemen kurang dari x, maka x termasuk dalam posisi keluaran 18.

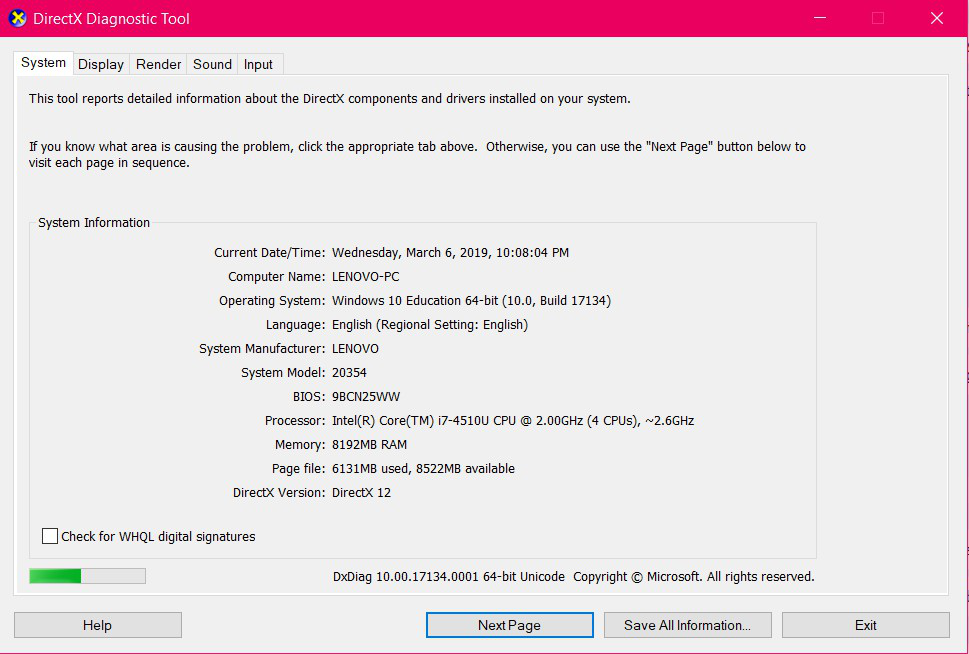
Kita diberikan array A [1. . n] panjangnya n. Kita membutuhkan dua array lagi, yaitu array B [1. . n] untuk meletakkan output yang diurutkan dan array c [1. . k] menyediakan penyimpanan sementara.

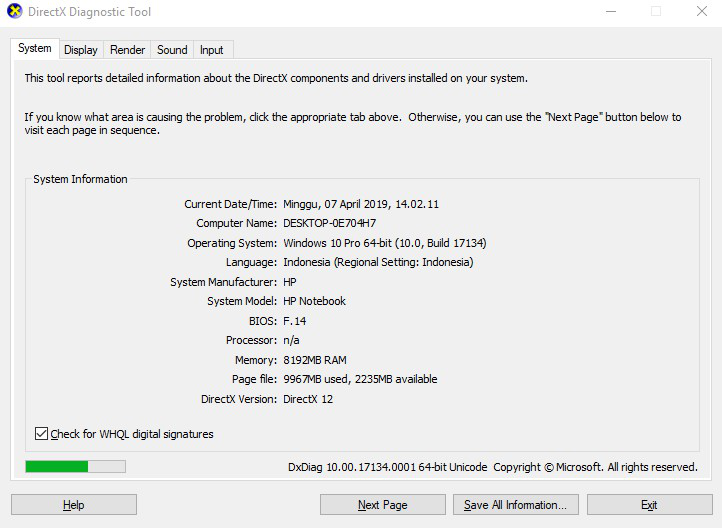


1. Analisis running time

Running time dalam program-program di atas didapatkan dari library chrono pada C++. Running time selalu berubah-ubah bahkan untuk program dengan code dan data yang sama sekalipun. Running time dipengaruhi oleh kerja CPU dan kita tidak pernah tahu apa saja proses yang sedang dikerjakan oleh CPU kita. Untuk spesifikasi dari PC yang digunakan untuk menjalankan program di atas adalah sebagai berikut:







1. Analisis kompleksitas

Karena algoritma hanya menggunakan loop sederhana tanpa panggilan rekursi atau subrutin, sehingga algoritmanya mudah untuk dianalisis. Inisialisasi array hitung dan loop yang melakukan jumlah awalan pada array hitung, masing-masing iterasi paling banyak k + 1 kali dan oleh karena itu membutuhkan waktu O(k). Dua lainnya untuk loop dan inisialisasi array output, masing-masing mengambil O(n) waktu. Oleh karena itu, waktu untuk keseluruhan algoritma adalah jumlah waktu untuk langkah-langkah ini, O(n + k). Karena menggunakan array panjang k + 1 dan n, total penggunaan ruang algoritma juga O(n + k).

Kesimpulannya:

* Kompleksitas terburuk: n+k
* Kompleksitas rata-rata: n+k
* Kompleksitas ruang: n+k