Tugas Praktikum Analisis Algoritma

Disusun oleh:
Aprischa Nauva
140810180063

Program Studi S1 Teknik Informatika
Fakultas Matematika & Ilmu Pengetahuan Alam
Universitas Padjadjaran
2020

Studi Kasus 1: MERGE SORT

Setelah Anda mengetahui Algoritma Merge-Sort mengadopsi paradigma divide & conquer, lakukan Hal berikut:

1. Buat program Merge-Sort dengan bahasa C++
2. Kompleksitas waktu algoritma merge sort adalah O(n lg n). Cari tahu kecepatan komputer Anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge sort-nya adalah 20?

Jawab :

1. Program C++ Merge Sort

```
/*
 Nama : Aprischa Nauva
 NPM : 140810180063
 Kelas : A
 */

#include <iostream>
#include <chrono>
using namespace std;

void satu(int* in, int p, int q,int r){
    int n1 = q-p+1;
    int n2 = r-q;
    int L[n1+1];
    int R[n2+1];
    for (int i=1; i<=n1; i++){
        L[i-1] = in[(p-1)+i-1];
    }

    for (int j=1; j<=n2; j++){
        R[j-1] = in[(q-1)+j];
    }

    int i=0;
    int j=0;
    L[n1]=2147483647;
    R[n2]=2147483647;

    for (int k=(p-1); k<r; k++){
        if(L[i]<=R[j]){
            in[k]=L[i];
            i = i+1;
        }
        else{
            in[k]=R[j];
            j = j+1;
        }
    }
}

void msort(int* in, int p, int r){
    int q;
```

```
    if(p<r){
        q = (p+r)/2;
        msort(in, p, q);
        msort(in, q+1, r);

        satu(in, p, q, r);
    }
}

void input(int* a, int& n){
    cout << "Input banyak data: "; cin >> n;
    for (int i=0; i<n; i++){
        cout << "Input angka: "; cin >> a[i];
    }
}

int main(){
    int in[100];
    int n;
    input(in,n);
    auto start = chrono::steady_clock::now();
    msort(in,1,n);
    auto end = chrono::steady_clock::now();
    cout << "Hasil: ";
    for(int i=0; i<n; i++){
        cout << in[i] << " ";
    }

    cout<<endl;
    cout << "Elapsed time in nanoseconds : "
    << chrono::duration_cast<chrono::nanoseconds>(end - start).count()
    << " ns" << endl;

    return 0;
}
```

```
Input banyak data: 20
Input angka: 5
Input angka: 2
Input angka: 5
Input angka: 7
Input angka: 3
Input angka: 6
Input angka: 2
Input angka: 9
Input angka: 8
Input angka: 6
Input angka: 1
Input angka: 3
Input angka: 4
Input angka: 7
Input angka: 9
Input angka: 4
Input angka: 6
Input angka: 8
Input angka: 5
Input angka: 9
Hasil: 1 2 2 3 3 4 4 5 5 5 6 6 6 7 7 8 8 9 9 9
Elapsed time in nanoseconds : 2735 ns
Program ended with exit code: 0
```

2. Kompleksitas Algoritma merge sort adalah O(n lg n).

   Cari tahu kecepatan komputer Anda dalam memproses program. Hitung berapa running time yang dibutuhkan apabila input untuk merge sort-nya adalah 20?

   Untuk di program hasilnya : 2735 ns
   Tapi jika sesuai dengan O -> T($20 \log_{10} 20$) $= 26$

Studi Kasus 2: SELECTION SORT

Selection sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma selection sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma selection sort
- Tentukan T(n) dari rekurensi (pengulangan) selection sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi T(n) dengan **metode recursion-tree** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma selection sort dengan menggunakan bahasa C++

1. Program C++ Selection Sort

```
/*
 Nama : Aprischa Nauva
 NPM : 140810180063
 Kelas : A
 */

#include <iostream>
#include <conio.h>


using namespace std;

int data[100],data2[100];
int n;

void tukar(int a, int b)
{
    int t;
    t = data[b];
    data[b] = data[a];
    data[a] = t;
}
void selection_sort()
{
    int pos,i,j;
    for(i=1;i<=n-1;i++)
    {
        pos = i;
        for(j = i+1;j<=n;j++)
        {
            if(data[j] < data[pos]) pos = j;
        }
        if(pos != i) tukar(pos,i);
    }
}

int main()
{
    cout<<"\nMasukkan Jumlah Data : ";cin>>n;
    for(int i=1;i<=n;i++)
    {
        cout<<"Masukkan data ke-"<<i<<" : ";
        cin>>data[i];
        data2[i]=data[i];
    }

    selection_sort();
    cout<<"Data Setelah di Sort : "<<endl;
    for(int i=1; i<=n; i++)
    {
        cout<<" "<<data[i];
        getch();
    }
}
```

```
Masukkan Jumlah Data : 3
Masukkan data ke-1 : 10
Masukkan data ke-2 : 5
Masukkan data ke-3 : 6
Data Setelah di Sort :
 5 6 10Program ended with exit code: 0
```

2.

```
for i ← n downto 2 do {pass sebanyak n-1 kali}
    imaks ← 1
    for j ← 2 to i do
      if xj > ximaks then
        imaks ← j
      endif
    endfor
    {pertukarkan ximaks dengan xi}
    temp ← xi
    xi ← ximaks
    ximaks ← temp
endfor
```

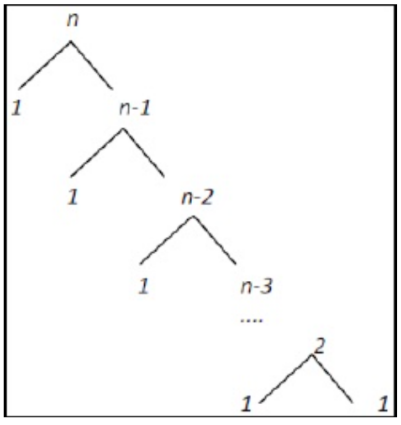
Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses pembagian = n

Waktu proses penggabungan = n

$T(n) = \{\Theta(1)\ T(n-1) + \Theta(n)$



T(n) = cn + cn-c +cn-2c + ... + 2c +cn

= c((n-1)(n-2)/2) + cn

= c((n^2-3n+2)/2) + cn

= c(n^2/2)-(3n/2)+1 + cn

=O(n^2)

T(n) = cn + cn-c +cn-2c + ... + 2c +cn

= c((n-1)(n-2)/2) + cn

= c((n^2-3n+2)/2) + cn

= c(n^2/2)-(3n/2)+1 + cn

= Ω (n^2)

T(n) = cn^2

= Θ(n^2)

Studi Kasus 3: INSERTION SORT

Insertion sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma insertion sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma insertion sort
- Tentukan T(n) dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi T(n) dengan **metode subtitusi** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma insertion sort dengan menggunakan bahasa C++

1. Program C++ Insertion Sort

```
/*
 Nama : Aprischa Nauva
 NPM : 140810180063
 Kelas : A
 */

#include <iostream>

using namespace std;

int data[100],data2[100],n;

void insertion_sort()
{
    int temp,i,j;
    for(i=1;i<=n;i++){
        temp = data[i];
        j = i -1;
        while(data[j]>temp && j>=0){
            data[j+1] = data[j];
            j--;
```

```
            }
            data[j+1] = temp;
        }
    }
    int main()
    {
        cout<<"Masukkan Jumlah Data : "; cin>>n;
        cout<<endl;
        for(int i=1;i<=n;i++)
        {
            cout<<"Masukkan data ke-"<<i<<" : ";
            cin>>data[i];
            data2[i]=data[i];
        }

        insertion_sort();
        cout<<"\nData Setelah di Sort : "<<endl;
        for(int i=1; i<=n; i++)
        {
            cout<<data[i]<<" ";
        }
    }
```

```
Masukkan Jumlah Data : 3

Masukkan data ke-1 : 6
Masukkan data ke-2 : 3
Masukkan data ke-3 : 2

Data Setelah di Sort :
2 3 6 Program ended with exit code: 0
```

2.

Algoritma

```
for i ← 2 to n do
    insert ← xi
    j ← i
    while (j < i) and (x[j-i] > insert) do
        x[j]← x[j-1]
        j←j-1
    endwhile
    x[j] = insert
endfor
```

Subproblem = 1

Masalah setiap subproblem = n-1

Waktu proses penggabungan = n

Waktu proses pembagian = n

$T(n) = \{\Theta(1)\ T(n-1) + \Theta(n)$

T(n) = cn + cn-c +cn-2c + ... + 2c +cn <= $2cn^2 + cn^2$

= c((n-1)(n-2)/2) + cn<= $2cn^2 + cn^2$

= c((n^2-3n+2)/2) + cn<= 2cn^2 + cn^2
= c(n^2/2)-c(3n/2)+c+cn <= 2cn^2 + cn^2
=O(n^2)

T(n) = cn <= cn
= Ω (n)

T(n) = (cn + cn^2)/n
= Θ(n

Studi Kasus 4: BUBBLE SORT

Bubble sort merupakan salah satu algoritma sorting yang berparadigma divide & conquer. Untuk membedah algoritma bubble sort, lakukan langkah-langkah berikut:

- Pelajari cara kerja algoritma bubble sort
- Tentukan T(n) dari rekurensi (pengulangan) insertion sort berdasarkan penentuan rekurensi divide & conquer:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{otherwise} \end{cases}$$

- Selesaikan persamaan rekurensi T(n) dengan **metode master** untuk mendapatkan kompleksitas waktu asimptotiknya dalam Big-O, Big-Ω, dan Big-Θ
- Lakukan implementasi koding program untuk algoritma bubble sort dengan menggunakan bahasa C++

1. Program C++ Bubble Sort

```
/*
 Nama : Aprischa Nauva
 NPM : 140810180063
 Kelas : A
 */

#include <iostream>

using namespace std;

int main(){
    int arr[100],n,temp;
    cout<<"Massukan banyak elemen yang akan diinputkan : ";cin>>n;

    for(int i=0;i<n;++i){
        cout<<"Masukkan Elemen ke-"<<i+1<<" : ";cin>>arr[i];
    }

    for(int i=1;i<n;i++){
        for(int j=0;j<(n-1);j++){
            if(arr[j]>arr[j+1]){
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }
```

```
    cout<<"\nHasil dari Bubble Sort : "<<endl;
    for(int i=0;i<n;i++){
        cout<<" "<<arr[i];
    }
}
```

```
Massukan banyak elemen yang akan diinputkan : 5
Masukkan Elemen ke-1 : 2
Masukkan Elemen ke-2 : 6
Masukkan Elemen ke-3 : 3
Masukkan Elemen ke-4 : 9
Masukkan Elemen ke-5 : 5

Hasil dari Bubble Sort :
 2 3 5 6 9Program ended with exit code: 0
```

2. Subproblem = 1
   Masalah setiap subproblem = n-1
   Waktu proses pembagian = n
   Waktu proses penggabungan

$$T(n) = \{\Theta(1)\ T(n-1) + \Theta(n)$$

T(n) = cn + cn-c +cn-2c + ... + 2c +c <= $2cn^2 + cn^2$
= c((n-1)(n-2)/2) + c<= $2cn^2 + cn^2$
= c((n^2-3n+2)/2) + c<= $2cn^2 + cn^2$
= c(n^2/2)-c(3n/2)+2c <= $2cn^2 + cn^2$
=O(n^2)

T(n) = cn + cn-c +cn-2c + ... + 2c +c <= $2cn^2 + cn^2$
= c((n-1)(n-2)/2) + c<= $2cn^2 + cn^2$
= c((n^2-3n+2)/2) + c<= $2cn^2 + cn^2$
= c(n^2/2)-c(3n/2)+2c <= $2cn^2 + cn^2$
= Ω (n^2)

T(n) = $cn^2 + cn^2$
= Θ(n^2)