LAPORAN PRAKTIKUM

ANALISIS ALGORITMA

Disusun Oleh:

Kefilino Khalifa Filardi

140810180028

PROGRAM STUDI S-1 TEKNIK INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS PADJADJARAN

JATINANGOR

2020

Studi Kasus 1: Pencarian Nilai Maksimal

Buatlah programnya dan hitunglah kompleksitas waktu dari algoritma berikut: **Algoritma Pencarian Nilai Maksimal**

procedure CariMaks(input x_1, x_2, ..., x_n: integer, output maks: integer) {
Mencari elemen terbesar dari sekumpulan elemen larik integer x_1, x_2, ..., x_n. Elemen terbesar akan disimpan di dalam maks
Input: x_1, x_2, ..., x_n
Output: maks (nilai terbesar)
}

Deklarasi i : integer

Algoritma maks ← x^1
i ← 2
while i ≤ n do
if x_i > maks then
maks ← x_i
endif
i ← i + 1
endwhile

Jawaban Studi Kasus 1

Program :

```
/*
* Nama      : Kefilino Khalifa Filardi
* NPM       : 140810180028
* Kelas     : B
* Program   : Pencari Bilangan Maksimal
* Tanggal   : 4 Maret 2020
* Desc      : Program ini mengimplemen algoritma untuk mencari bilangan
maksimal.
*/

#include <iostream>
using namespace std;

int CariMaks(int arr[], int n) {
   int maks = arr[0], i = 1;

   while (i < n) {
       if (arr[i] > maks)
           maks = arr[i];
       i++;
   }
```

```
    return maks;
}

int main()
{
    int n, maks;

    cout << "Banyak Bilangan : ";
    cin >> n;
    int arr[n];

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

    maks = CariMaks(arr, n);

    cout << "\nBilangan Terbesar adalah " << maks << endl;
}
```

Kompleksitas Waktu :

1. Pengisian Nilai

   $maks \leftarrow x_1$ = 1

   $i \leftarrow 2$ = 1

   $maks \leftarrow x_i$ = n - 1

   $i \leftarrow i + 1$ = n - 1

   $T_1(n) = 1 + 1 + (n - 1) + (n - 1) = 2n$
2. Perbandingan

   $x_i > maks$ = n - 1

   $T_2(n) = n - 1$
3. Penjumlahan

   i + 1 = 1

   $T_3(n) = n - 1$

$T(n) = 2n + (n - 1) + (n - 1) = 4n - 2$

Studi Kasus 2: *Sequential Search*

Diberikan larik bilangan bulan , ,... yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata-rata dari algoritma pencarian beruntun (*sequential search*). Algoritma *sequential search* berikut menghasilkan indeks elemen yang bernilai sama dengan y. Jika y tidak ditemukan, indeks 0 akan dihasilkan.

procedure SequentialSearch(input , ,... : integer, y : integer, output idx : integer) { Mencari di dalam elemen , ,... . Lokasi (indeks elemen) tempat ditemukan diisi ke dalam idx.
Jika tidak ditemukan, makai idx diisi dengan 0. Input: , ,... Output: idx }

Deklarasi i : integer

found : boolean {bernilai true jika y ditemukan atau false jika y tidak ditemukan}

Algoritma i ← 1

found ← false while (i ≤ n) and (not found) do

if x_i = y then

found ← true else i ← i + 1 endif endwhile {*i < n or found*}

If found then {*y ditemukan*}

idx ← i else

idx ← 0 {y tidak ditemukan} endif

Jawaban Studi Kasus 2

Program :

```
/*
* Nama      : Kefilino Khalifa Filardi
* NPM       : 140810180028
* Kelas     : B
* Program   : Pencari Bilangan Maksimal
* Tanggal   : 4 Maret 2020
* Desc      : Program ini mengimplemen algoritma untuk mencari bilangan
maksimal.
*/

#include <iostream>
using namespace std;

int SequentialSearch(int x[], int y, int size) {
   int i = 0;
   bool found = false;
```

```
    while (i < size && !found) {
        if (x[i] == y)
            found = true;
        else
            i++;
    }

    if (found)
        return i;
    else
        return 0;
}

int main()
{
    int x[10] = {1,2,3,4,5,6,7,8,9,10};

    cout << "Index dari 5 = " << SequentialSearch(x, 5, 10) << endl;
}
```

Kompleksitas Waktu :

1. Pengisian Nilai

   $i \leftarrow 1 = 1$

   found ← false = 1

   found ← true = 1 (Average Case & Best Case), 0 (Worst Case)

   $i \leftarrow i + 1$ = n (Worst Case), (n+1)/2 (Average Case), 0 (Best Case)

   idx ← i = 1 (Average Case & Best Case), 0 (Worst Case)

   idx ← 0 = 1 (Worst Case), 0 (Average Case & Best Case)

   $T_{1min}(n) = 1 + 1 + 1 + 0 + 1 + 0 = 4$
   $T_{1avg}(n) = 1 + 1 + 1 + (n+1)/2 + 1 + 0 = (n+9)/2$
   $T_{1max}(n) = 1 + 1 + 0 + n + 0 + 1 = n + 3$

2. Perbandingan
   if xi = y then = n (Worst Case), (n+1)/2 (Average Case), 0 (Best Case)
   If found then {y ditemukan} = 1

   $T_{2min}(n) = 0 + 1 = 1$
   $T_{2avg}(n) = (n+1)/2 + 1 = (n+3)/2$
   $T_{2max}(n) = n + 1$

3. Penjumlahan

   $i \leftarrow i + 1$ = n (Worst Case), (n+1)/2 (Average Case), 0 (Best Case)

   $T_{3min}(n) = 0$
   $T_{3avg}(n) = (n+1)/2$
   $T_{3max}(n) = n$

   Best Case (langsung ketemu) :
   $T_{min}(n) = T_{1min}(n) + T_{2min}(n) + T_{3min}(n) = 4 + 1 + 0 = 5$

   Average Case (ketemu di tengah) :
   $T_{avg}(n) = T_{1avg}(n) + T_{2avg}(n) + T_{3avg}(n) = (n+9)/2 + (n+3)/2 + (n+1)/2 = (3n+13)/2$

   Worst Case (tidak ketemu) :
   $T_{max}(n) = T_{1max}(n) + T_{2max}(n) + T_{3max}(n) = n + 3 + n + 1 + n = 3n + 4$

Studi Kasus 3: *Binary Search*

Diberikan larik bilangan bulan , ,... yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata- rata dari algoritma pencarian bagi dua (*binary search*). Algoritma *binary search* berikut menghasilkan indeks elemen yang bernilai sama dengan y. Jika y tidak ditemukan, indeks 0 akan dihasilkan.

procedure BinarySearch(input , ,... : integer, x : integer, output : idx : integer) { Mencari y di dalam elemen , ,... . Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.

Jika y tidak ditemukan makai dx diisi dengan 0. **Input:** , ,... **Output: idx** } **Deklarasi**

i, j, mid : integer found : Boolean **Algoritma**

i ← 1 j ← n found ← false while (not found) and (i ≤ j) do mid ← (i + j) div 2 if x_{mid} = y then found ← true else

if x_{mid} < y then {*mencari di bagian kanan*}

i ← mid + 1 else {*mencari di bagian kiri*}

j ← mid – 1 endif endif endwhile {*found or i > j* }

If found then

Idx ← mid else

Idx ← 0 endif

Jawaban Studi Kasus 3

Program :

```
/*
* Nama      : Kefilino Khalifa Filardi
* NPM       : 140810180028
* Kelas     : B
* Program   : Pencari Index dari Bilangan
* Tanggal   : 4 Maret 2020
* Desc      : Program ini mencari index dari bilangan yang dicari dengan
Menggunakan Binary Search.
*/

#include <iostream>
using namespace std;

int BinarySearch(int x[], int y, int size) {
   int i = 0, j = size, mid;
   bool found = false;

   while (!found && i < j) {
       mid = (i + j + 1)/2;
       if (x[mid] == y)
           found = true;
       else if (x[mid] < y)
           i = mid + 1;
       else
           j = mid - 1;
   }

   if (found)
       return mid;
   else
       return 0;
}

int main()
{
   int x[10] = {1,2,3,4,5,6,7,8,9,10};

   cout << "Index dari 5 = " << BinarySearch(x, 5, 10) << endl;
}
```

Kompleksitas Waktu :

1. Pengisian Nilai

   i ← 1 = 1
   j ← n = 1
   found ← false = 1
   mid ← (i + j) div 2 = 1 (best case), ((n+1)/2)/2 (average case), n/2 (worst case)
   found ← true = 1 (best case & average case), 0 (worst case)
   i ← mid + 1 = 0 (best case), ((n+1)/2)/2 (average case), n/2 (worst case)
   j ← mid – 1 = 1 (best case), ((n+1)/2)/2 (average case), n/2 (worst case)
   i ← mid + 1 & j ← mid – 1 diambil salah satu ketika average case & worst case
   Idx ← mid = 1 (best case & average case), 0 (worst case)
   Idx ← 0 = 0 (worst case), 1 (best case & average case)

   $T_{1min}(n) = 1 + 1 + 1 + 1 + 1 + 0 + 1 + 1 + 0 = 7$
   $T_{1avg}(n) = 1 + 1 + 1 + (n+1)/4 + 1 + (n+1)/4 + 0 + 1 + 1 + 0 = (n+1)/2 + 6$
   $T_{1max}(n) = 1 + 1 + 1 + n/2 + 0 + n/2 + 0 + 0 + 1 = n + 4$

2. Penjumlahan

   mid ← (i + j) div 2 = 1 (best case), ((n+1)/2)/2 (average case), n/2 (worst case)
   i ← mid + 1 = 0 (best case), ((n+1)/2)/2 (average case), n/2 (worst case)

   $T_{2min}(n) = 1 + 0 = 1$
   $T_{2avg}(n) = (n+1)/2)/2 + (n+1)/2)/2 = (n+1)/2$
   $T_{2max}(n) = n/2 + n/2 = n$

3. Pembagian

   mid ← (i + j) div 2 = 1 (best case), ((n+1)/2)/2 (average case), n/2 (worst case)

   $T_{3min}(n) = 1$
   $T_{3avg}(n) = ((n+1)/2)/2$
   $T_{3max}(n) = n/2$

4. Pengurangan

   j ← mid – 1 = 1 (best case), ((n+1)/2)/2 (average case), n/2 (worst case)

   $T_{4min}(n) = 1$
   $T_{4avg}(n) = (n+1)/2)/2$
   $T_{4max}(n) = n/2$

5. Perbandingan

   if x_{mid} = y then = 1 (best case), ((n+1)/2)/2 (average case), n/2 (worst case)
   if x_{mid} < y then = 1 (best case), ((n+1)/2)/2 (average case), n/2 (worst case)
   If found then = 1

   $T_{5min}(n) = 1 + 1 + 1 = 3$
   $T_{5avg}(n) = (n+1)/2)/2 + (n+1)/2)/2 + 1 = (n+1)/2 + 1 = (n+3)/2$
   $T_{5max}(n) = n/2 + n/2 + 1 = n + 1$

   $T_{min}(n) = T_{1min}(n) + T_{2min}(n) + T_{3min}(n) + T_{4min}(n) + T_{5min}(n)$
   $= 7 + 1 + 1 + 1 + 3 = 13$
   $T_{avg}(n) = T_{1avg}(n) + T_{2avg}(n) + T_{3avg}(n) + T_{4avg}(n) + T_{5avg}(n)$
   $= (n+1)/2 + 6 + (n+1)/2 + ((n+1)/2)/2 + ((n+1)/2)/2 + (n+3)/2$
   $= n + 1 + 6 + n + 2 = 2n + 9$
   $T_{max}(n) = T_{1max}(n) + T_{2max}(n) + T_{3max}(n) + T_{4max}(n) + T_{5max}(n)$
   $= n + 4 + n + n/2 + n/2 + n + 1 = 4n + 5$

Studi Kasus 4: Insertion Sort

1. Buatlah program insertion sort dengan menggunakan bahasa C++ 2. Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma insertion sort. 3. Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.

procedure InsertionSort(input/output , ,... : integer) { Mengurutkan elemen-elemen , ,... dengan metode insertion sort.

Input: , ,... OutputL , ,... (sudah terurut menaik) } **Deklarasi** i, j, insert : integer **Algoritma** for i ← 2 to n do insert ← x_i j ← i while (j < i) and (x[j-i] > insert) do

x[j]← x[j-1] j←j-1 endwhile x[j] = insert endfor

Jawaban Studi Kasus 4

Program :

```
/*
* Nama      : Kefilino Khalifa Filardi
* NPM       : 140810180028
* Kelas     : B
* Program   : Pengurut Bilangan Larik
* Tanggal   : 4 Maret 2020
* Desc      : Program ini mengurut bilangan larik menggunakan insertion
sort.
*/

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

```
void InsertionSort(int x[], int size) {
   int i, j, insert;

   for (i = 1; i < size; i++) {
       insert = x[i];
       j = i;

       do {
           x[j] = x[j-1];
           j--;
       } while (j < i && x[j-i] > insert);

       x[j] = insert;
   }
}

int main()
{
   int x[10] = {10,9,8,7,6,5,4,3,2,1};
   InsertionSort(x, 10);

   for (int i = 0; i < 10; i++)
   {
       cout << x[i] << " ";
   }

}
```

Kompleksitas Waktu :

1. Pengisian Nilai

   for i ← 2 = 1
   insert ← x_i = n - 1
   j ← i = n - 1
   x[j]← x[j-1] = 1 * (n-1) best, (n+1)/2 * (n-1) avg, n * (n-1) worst
   j←j-1 = 1 * (n-1) best, (n+1)/2 * (n-1) avg, n * (n-1) worst
   x[j] = insert = n - 1

   $T_{1min}(n) = 1 + n-1 + 1 + n-1 + n-1 + n-1 = 4n - 2$
   $T_{1avg}(n) = 1 + n-1 + 1 + (n^2-1)/2 + (n^2-1)/2 + n-1 = n^2 + n - 1$
   $T_{1max}(n) = 1 + n-1 + 1 + n^2-n + n^2-n + n-1 = 2n^2 - n$

2. Pengurangan

x[j]← x[j-1] = 1 * (n-1) best, (n+1)/2 * (n-1) avg, n * (n-1) worst

j←j-1 = 1 * (n-1) best, (n+1)/2 * (n-1) avg, n * (n-1) worst

$T_{2min}(n) = n-1 + n-1 = 2n - 2$

$T_{2avg}(n) = (n^2 - 1)/2 + (n^2 - 1)/2 = n^2 - 1$

$T_{2max}(n) = n^2-n + n^2-n = 2n^2 - 2n$

$T_{min}(n) = T_{1min}(n) + T_{2min}(n) = 4n-2 + 2n-2 = 6n - 4$

$T_{avg}(n) = T_{1avg}(n) + T_{2avg}(n) = n^2+n-1 + n^2-1 = 2n^2 + n - 2$

$T_{max}(n) = T_{1max}(n) + T_{2max}(n) = 2n^2-n +2n^2-2n = 4n^2 - 3n$

Studi Kasus 5: Selection Sort

1. Buatlah program selection sort dengan menggunakan bahasa C++ 2. Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma selection sort. 3. Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.

procedure SelectionSort(input/output , ,... : integer) { Mengurutkan elemen-elemen , ,... dengan metode selection sort.

Input: , ,... OutputL , ,... (sudah terurut menaik) } **Deklarasi** i, j, imaks, temp : integer **Algoritma** for i ← n downto 2

do {*pass sebanyak n-1 kali*}

imaks ← 1 for j ← 2 to i do

if $x_j > x_{imaks}$ then

imaks ← j endif endfor {pertukarkan x_{imaks} dengan x_i} temp ← x_i x_i ← x_{imaks} x_{imaks} ← temp endfor

Jawaban Studi Kasus 5

Program :

```
/*
* Nama      : Kefilino Khalifa Filardi
* NPM       : 140810180028
* Kelas     : B
* Program   : Pengurut Bilangan Larik
* Tanggal   : 4 Maret 2020
* Desc      : Program ini mengurut bilangan larik menggunakan selection
sort.
*/

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

```
void SelectionSort(int x[], int size) {
   int i, j, imaks, temp;

   for (i = size; i > 0; i--) {
       imaks = 0;

       for (j = 1; j < size; j++) {
           if (x[j] > x[imaks])
               imaks = j;
       }

       temp = x[i];
       x[i] = x[imaks];
       x[imaks] = temp;
   }
}

int main()
{
   int x[10] = {10,9,8,7,6,5,4,3,2,1};
   SelectionSort(x, 10);

   for (int i = 0; i < 10; i++)
   {
       cout << x[i] << " ";
   }

}
```

Kompleksitas Waktu :

1. Pengisian Nilai

   for i ← n downto 2 do = 1
   imaks ← 1 = n - 1
   for j ← 2 to i do = n - 1
   imaks ← j = 0 * (n-1) best, (n+1)/2 * (n-1) avg, n * (n-1)
   temp ← x_i = n - 1
   x_i ← x_{imaks} = n - 1
   x_{imaks} ← temp = n - 1

   $T_{1min}(n) = 1 + n-1 + n-1 + 0 + n-1 + n-1 + n-1 = 5n + 4$
   $T_{1avg}(n) = 1 + n-1 + n-1 + (n^2-1)/2 + n-1 + n-1 + n-1 = (n^2-1)/2 + 5n + 4$
   $T_{1max}(n) = 1 + n-1 + n-1 + n^2-n + n-1 + n-1 + n-1 = n^2 + 4n + 4$

2. Perbandingan

   if $x_j > x_{imaks}$ then = n * (n-1)

   $T_2(n) = n^2-n$

   $T_{min}(n) = T_{1min}(n) + T_2 = 5n + 4 + n^2-n = n^2 + 4n + 4$
   $T_{avg}(n) = T_{1avg}(n) + T_2 = (n^2-1)/2 + 5n + 4 + n^2-n = 3n^2/2 + 4n + 7/2$
   $T_{max}(n) = T_{1max}(n) + T_2 = n^2 + 4n + 4 + n^2-n = 2n^2 + 3n + 4$

Teknik Pengumpulan

• Lakukan push ke github/gitlab untuk semua program dan laporan hasil analisa yang berisi jawaban dari pertanyaan-pertanyaan yang diajukan. Silahkan sepakati dengan asisten praktikum.

Penutup

• Ingat, berdasarkan Peraturan Rektor No 46 Tahun 2016 tentang Penyelenggaraan Pendidikan, mahasiswa wajib mengikuti praktikum 100%

• Apabila tidak hadir pada salah satu kegiatan praktikum segeralah minta tugas pengganti ke asisten praktikum

• Kurangnya kehadiran Anda di praktikum, memungkinkan nilai praktikum Anda tidak akan dimasukkan ke nilai mata kuliah.