TUGAS 2

PRAKTIKUM ANALISIS ALGORITMA

Disusun oleh :

Hasna Karimah 140810160020

PROGRAM STUDI S-1 TEKNIK INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN
SUMEDANG
2019

Studi Kasus 1: Pencarian Nilai Maksimal

Buatlah programnya dan hitunglah kompleksitas waktu dari algoritma berikut:

Algoritma Pencarian Nilai Maksimal

Jawaban Studi Kasus 1

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

Jenis-jenis operasi yang terdapat di dalam Algoritma HitungRerata adalah:

- Operasi pengisian nilai/*assignment* (dengan operator "←")
- Operasi penjumlahan (dengan operator "+")

  - Operasi pengisian nilai (*assignment*)
    maks ← x_1, 1 kali
    i ← 2, 1 kali
    maks ← x_i n kali
    i ← i + 1 n -1 kali

    Jumlah seluruh operasi pengisian nilai (*assignment*) adalah
    $t_1 = 1 + 1 + n + n - 1 = 1 + 2n$

- Operasi penjumlahan
    i + 1 n-1 kali

    Jumlah seluruh operasi penjumlahan adalah
    $t_2 = n$-1

Dengan demikian, kompleksitas waktu algoritma dihitung berdasarkan jumlah operasi aritmatika dan operasi pengisian nilai adalah:

$$T(n) = t_1 + t_2 = 1 + 2n + n - 1 = 3n$$

Semua : $t_1 = 1 + 1 + n + n - 1 = 1 + 2n$
Best case : + 1 + (n-1)+(n-1) = 4n
Worst case : + (n-1)+(n-1) + (n-1) = 5n - 2

Program c++

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

int main()
{
```

```
    int n;
    float arr[100];

    cout << "Masukkan banyak angka : ";
    cin >> n;
    cout << endl;

    for(i = 2; i < n; ++i)
    {
      cout << "masukkan angka ke- " << i -  1 << " : ";
      cin >> arr[i];
    }

    for(i = 1;i < n; ++i)
    {
      if(arr[0] < arr[i])
         arr[0] = arr[i];
    }
    cout << "angka terbesar adalah = " << arr[0];

    return 0;
}
```

Studi Kasus 2: *Sequential Search*

Diberikan larik bilangan bulan $x_1, x_2, \ldots\ x_n$ 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 x1, x2, ... xn : integer,  y : integer, output idx : integer)
{  Mencari y di dalam elemen x1, x2, ... xn. Lokasi (indeks elemen) tempat y ditemukan diisi
   ke dalam idx. Jika y tidak ditemukan, makai idx diisi dengan 0.
   Input: x1, x2, ... xn
   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 xi = 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

Jumlah operasi perbandingan elemen tabel:
Kasus terbaik: ini terjadi bila a1 = x

Tmin(n) = 1

2. Kasus terburuk: bila an = x atau x tidak ditemukan.

Tmax(n) = n

Kasus rata-rata: Jika x ditemukan pada posisi ke-j, maka operasi perbandingan (ak = x) akan dieksekusi sebanyak j kali.

Tavg(n) = (1+2+3+..+n)/n = (1/2n(1+n))/n = (n+1)/2

Studi Kasus 3: *Binary Search*

Diberikan larik bilangan bulan $x_1, x_2, \ldots x_n$ 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  x1, x2, ... xn : integer, x : integer, output : idx : integer)
{  Mencari y di dalam elemen x1, x2, ... xn. Lokasi (indeks elemen) tempat y ditemukan diisi
   ke dalam idx. Jika y tidak ditemukan makai dx diisi dengan 0.
   Input: x1, x2, ... xn
   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 xmid = y then
                    found ← true
                else
                    if xmid < 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

1. Kasus terbaik : Tmin(n) = 1
2. Kasus terburuk : Tmax (n) = 2log n

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 x1, x2, ... xn : integer)
{  Mengurutkan elemen-elemen x1, x2, ... xn dengan metode insertion sort.
   Input: x1, x2, ... xn
   OutputL x1, x2, ... xn (sudah terurut menaik)
}
Deklarasi
      i, j, insert : integer
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
```

Jawaban Studi Kasus 4

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 x1, x2, ... xn : integer)
{  Mengurutkan elemen-elemen x1, x2, ... xn dengan metode selection sort.
   Input: x1, x2, ... xn
   OutputL x1, x2, ... xn (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 xj > ximaks then
                imaks ← j
              endif
            endfor
            {pertukarkan ximaks dengan xi}
            temp ← xi
            xi ← ximaks
            ximaks ← temp
        endfor
```

Jawaban Studi Kasus 5

a. Jumlah operasi perbandingan element. Untuk setiap *pass* ke-*i*,

i = 1 –> jumlah perbandingan = *n* – 1
i = 2 –> jumlah perbandingan = *n* – 2
i = 3 –> jumlah perbandingan = *n* – 3
:
i = *k* –> jumlah perbandingan = *n* – *k*
:
i = *n* – 1 –> jumlah perbandingan = 1

Jumlah seluruh operasi perbandingan elemen-elemen larik adalah $T(n) = (n-1) + (n-2) + \ldots + 1$

Ini adalah kompleksitas waktu untuk kasus terbaik dan terburuk, karena algoritma Urut tidak bergantung pada batasan apakah data masukannya sudah terurut atau acak.

b. Jumlah operasi pertukaran
Untuk setiap i dari 1 sampai $n - 1$, terjadi satu kali pertukaran elemen, sehingga jumlah operasi pertukaran seluruhnya adalah $T(n) = n - 1$.
Jadi, algoritma pengurutan maksimum membutuhkan $n(n - 1)/2$ buah operasi perbandingan elemen dan $n - 1$ buah operasi pertukaran.

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.