WORKSHEET 2 - PRAKTIKUM ANALISIS ALGORITMA

OLEH

JONATHAN RAFMA NANDA SIREGAR

140810180056

STUDI KASUS 1

Buat Program dan Hitunglah waktu dari Algoritma Studi Kasus 1.

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

Deklarasi

i : integer

Algoritma

```
maks <- x1
i <- 2
while i ≤ n do
    if xi > maks then
        maks <- xi
    endif
i <- i + 1 endwhile
```

Jawaban Studi Kasus 1 :

Kompleksitas waktu yang terdapat pada algoritma diatas :

a. Operator Assignment

$maks = x_1$	1 kali
$i = 2$	1 kali
$maks = x_1$	n kali
$i <- i + 1$	n kali

$t_1 = 1 + 1 + n + n = 2 + 2n$

b. Operasi Pertambahan

$i + 1$	1 kali

$t_2 = 1$

c. Operasi Perbandingan

if x_1 = > max n kali

t_3 = n

d. Total Operasi

$t_1 + t_2 + t_3 = 2 + 2n + 1 + n = 3 + 3n$

STUDI KASUS 2

Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk dan rata-rata dari algoritma pencarian beruntun dengan algoritma berikut

procedure SequentialSearch(input $x_1, x_2, \ldots x_n$: integer, y : integer, output idx : integer)

{ Mencari y di dalam elemen $x_1, x_2, \ldots x_n$. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.

Jika y tidak ditemukan, makai idx diisi dengan 0. Input: $x_1, x_2, \ldots x_n$

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:

Kompleksitas Waktu :

Kompleksitas Waktu terbaik / Best Case

a. Operasi Assignment
   i <- 1 1 kali
   found <- false 1 kali
   found <- true 1 kali
   idx <- i 1 kali
   $t_1 = 1 + 1 + 1 + 1 = 4$
b. Operasi Perbandingan
   if x_i = y then 1 kali
   If found <- true then 1 kali
   $t_2 = 1 + 1 = 2$
c. Total Operasi
   $t_1 + t_2 = 4 + 2 = 6$
   $t_{MIN} = 6$

Kompleksitas Waktu terburuk / Worst Case

a. Operasi Assignment
   i <- 1 1 kali
   found <- false 1 kali
   i <- i + 1 n kali
   idx <- 0 1 kali
   $t_1 = 1 + 1 + n + 1 = 3 + n$
b. Operasi Penjumlahan
   i + 1 n kali
   $t_2 = n$
c. Operasi Perbandingan
   if x_i = y then .. else n kali
   If found then .. else 1 kali
   $t_3 = n + 1$
d. Total Operasi
   $t_1 + t_2 + t_3 = 3 + n + n + n + 1 = 4 + 3n$
   $t_{max} = 4 + 3n$

Kompleksitas Waktu Rata-rata / Average Case

Average Case adalah rata-rata dimana waktu yang dihitung tidak masuk kedalam Best Case maupun Worst Case

$t_{AVG} = (t_{MIN} + t_{MAX})/2 = (6 + 4 + 3n)/2 = 5+3n/2$

STUDI KASUS 3

Buatlah programnnya dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata-rata dari algoritma berikut

procedure BinarySearch(input $x_1, x_2, \ldots x_n$: integer, x : integer, output : idx : integer)

{ Mencari y di dalam elemen $x_1, x_2, \ldots x_n$. Lokasi (indeks elemen) tempat y ditemukan diisi ke dalam idx.

Jika y tidak ditemukan makai dx diisi dengan 0.

Input: $x_1, x_2, \ldots x_n$

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

i <- mid + 1 else
j <- mid – 1 endif

endif endwhile

If found then

Idx <- mid
else
Idx <- 0
endif

Jawaban Studi Kasus 3 :

Kompleksitas Waktu Terbaik / Best Case

a. Operasi Assignment
   i <- 1 1 kali
   j <- n 1 kali
   found <- false 1 kali
   mid <- (i + j) div 2 1 kali
   found <- true 1 kali
   idx <- mid 1 kali
   $t_1 = 1 + 1 + 1 + 1 + 1 + 1 = 6$
b. Operasi Pertambahan, Pengurangan, dan Pembagian
   mid <- (i + j) div 2 2 kali
   $t_2 = 2$
c. Operasi Perbandingan
   if X_{mid} = y then 1 kali
   if found then 1 kali
   $t_3 = 1 + 1 = 2$

d. Total Operasi

$t_{MIN} = t_1 + t_2 + t_3 = 6 + 2 + 2 = 10$

Kompleksitas Waktu Terburuk / Worst Case

a. Operasi Assignment

i <- 1	1 kali
j <- n	1 kali
found <- false	1 kali
mid <- (i + j) div 2	n kali
i <- mid + 1 atau j <- mid - 1	n kali
idx <- 0	1 kali

$t_1 = 1 + 1 + 1 + n + n + 1 = 4 + 2n$

b. Operasi Penjumlahan, Pengurangan, dan Pembagian

mid <- (i + j) div 2	2n kali
i <- mid + 1 atau j <- mid - 1	n kali

$t_2 = 2n + n = 3n$

c. Operasi Perbandingan

if X_{mid} < Y then	n kali
if found then … else	1 kali

$t_3 = n + 1$

d. Total Operasi

$t_{MAX} = t_1 + t_2 + t_3 = 4 + 2n + 3n + n + 1 = 5 + 6n$

Kompleksitas Waktu Rata-rata / Average Case

$t_{AVG} = (t_{MIN} + t_{MAX})/2 = (10 + 5 + 6n)/2 = 15/2 + 3n$

STUDI KASUS 4

1. Buat Program Insertion Sort dengan 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 $x_1, x_2, \ldots x_n$: integer)

{ Mengurutkan elemen-elemen $x_1, x_2, \ldots x_n$ dengan metode insertion sort.

Input: $x_1, x_2, \ldots x_n$

OutputL $x_1, x_2, \ldots x_n$ (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 :

1. Operasi Assignment: 2(n - 1) + (n - 1) = 3n - 3
2. Operasi Perbandingan: 2((n - 1) + (n - 1)) = 2(2n - 2) = 4n - 4
3. Operasi Pertukaran: (n - 1)n = n^2 - n

Kompleksitas Waktu Terbaik / Best Case

t_{MIN} = 3n - 3 + 4n - 4 + 1 = 7n - 6

Kompleksitas Waktu Terburuk / Worst Case

t_{MAX} = 3n - 3 + 4n - 4 + n^2 - n = n^2 + 6n - 6

Kompleksitas Waktu Rata-rata / Average Case

t_{AVG} = (t_{MIN} + t_{MAX})/2 = (7n - 6 + n^2 + 6n - 6)/2 = (n^2 + 13n - 12)/2

STUDI KASUS 5

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 selection 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
      end
      if
    endfor
```

{pertukarkan x_{imaks} dengan x_i}
temp <- x_i
x_i <- x_{imaks}
x_{imaks} <- temp

endfor

Jawaban Studi Kasus 5 :
Operasi Perbandingan

$$\sum_{i=1}^{n-1} i = \frac{(n-1)+1}{2}(n-1) = \frac{1}{2}n(n-1) = \frac{1}{2}(n^2-n)$$

Operasi Pertukaran
n - 1
Kompleksitas Waktu Terbaik / Best Case
$t_{MIN} = 4n - 4 + ½(n^2 - n) + 1 = n^2$
Kompleksitas Waktu Terburuk / Worst Case
$t_{MAX} = ½(n^2 - n) + (n - 1) = n^2$
Kompleksitas Waktu Rata-rata / Average Case
$t_{AVG} = (t_{MIN} + t_{MAX})/2 = (n^2 + n^2)/2 = n^2$