Studi kasus 1: Pencarian Nilai Maksimal

Hitung kompleksitas waktu dari algoritma pencarian nilai maksimal

```
int main(){
        int n = 10;
        int angka[n] = {1,4,3,5,7,2,12,23,9,34};
        int max = angka[0];
        int i = 1;

        while (i < n) {
            if (angka[i] > max)
                max = angka[i];
            i++;
        }

        cout << max;

}
```

1. Operasi pengisian nilai

n ← 10	1 kali
angka[n] ← {1,4,3,5,7,2,12,23,9,34}	1 kali
max ← angka[0]	1 kali
i ← 1	1 kali
max ← angka[i]	n-1 kali
i ← i + 1	n-1 kali

jumlah seluruh operasi pengisian nilai $= 1 + 1 + 1 + 1 + n + n - 2$
$= 2 + 2n$

2. Operasi penjumlahan

i + 1	n-1 kali

jumlah seluruh operasi penjumlahan $= \mathrm{n} - 1$

3. Operasi output nilai

cout << maks	1 kali

jumlah seluruh operasi output nilai $= 1$

Kompleksitas waktu algorima $= \mathrm{T(n)}$

$$\mathrm{T(n)} = t_1 + t_2 + t_3$$
$$= 2 + 2n + n - 1 + 1$$

$$= 2 + 3n$$

Studi kasus 2: Sequential search
Hitung kompleksitas waktu terbaik, terburuk, dan rata-rata

```cpp
#include <iostream>

using namespace std;

main(){
    int arr_n[10] = {1,2,5,6,7,9,12,14,16,23};
    int y = 5;
    bool found;
    int i = 0;
    int idx = 0;

    found = false;
    while (i < 10 && !found) {
        if (arr_n[i] == y)
            found = true;
        else
            i++;
    }

    if (found)
        cout << "elemen ke-" << idx;
    else
        cout << 0;

}
```

Average case scenario

y ditemukan pada elemen array pertama

Operasi assignment

arr_n[n] <- elemen array	n kali
y <- nilai yang dicari	1 kali
found <- false	1 kali
idx <- 0	1 kali
i <- 0	1 kali
i <- i + 1	n kali
found <- true	1 kali
idx <- i	1 kali

$\text{Jumlah} = 2n + 6$

Operasi aritmatika

i + 1	n kali

$\text{Jumlah} = n$

Operasi input/output

Input y	1 kali
Output hasil	1 kali

$\text{Jumlah} = 2$

$\text{T(n)} = 2n + 6 + \ n + 2$
$= 3n + 8$

$T_{avg}(n) = \frac{1}{n}(3n + 8)$

Best case scenario time complexity

y ditemukan pada elemen array pertama

Operasi assignment

arr_n[n] <- elemen array	n kali
y <- nilai yang dicari	1 kali
found <- false	1 kali
idx <- 0	1 kali
i <- 0	1 kali
found <- true	1 kali
idx <- 0	1 kali
idx <- i	1 kali

$\text{Jumlah} = 2n + 6$

Operasi input/output

Input y	1 kali
Output hasil	1 kali

$\text{Jumlah} = 2$

$T_{best}(n) = 2n + 6 + 2$
$= 2n + 8$

Worst case scenario : elemen tidak ditemukan

Operasi assignment

arr_n[n] <- elemen array	n kali
y <- nilai yang dicari	1 kali
found <- false	1 kali
idx <- 0	1 kali
i <- 0	1 kali
i <- i + 1	n kali

$\text{Jumlah} = 2n + 4$

Operasi aritmatika

i + 1	n kali

$\text{Jumlah} = n$

Operasi input/output

Input y	1 kali
Output hasil	1 kali

$\text{Jumlah} = 2$

$\text{T(n)} = 2n + 4 + n + 2$
$= 3n + 6$

$T_{worst}(n) = 3n + 6$

Studi kasus 3: Binary Search

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

int binarySearch(int arr[], int l, int r, int x)
{
    if (r >= l) {
        int mid = l + (r - l) / 2;

        if (arr[mid] == x)
            return mid;

        if (arr[mid] > x)
            return binarySearch(arr, l, mid - 1, x);
```

```
        return binarySearch(arr, mid + 1, r, x);
    }

    return -1;
}

int main()
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 10;
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = binarySearch(arr, 0, n - 1, x);
    (result == -1) ? cout << "Elemen tidak ditemukan pada array"
                   : cout << "Elemen ditemukan pada indeks ke-" << result;
    return 0;
}
```

Average case scenario

Operasi assignment

arr[n] <- elemen array	n kali
i <- 1	1 kali
j <- n	1 kali
found <- false	1 kali
mid <- (i + j) div 2	1 kali
i <- mid + 1	1 kali
j <- mid – 1	1 kali
found <- true	1 kali
idx <- mid	1 kali

$\text{Jumlah} = n + 8$

Operasi aritmatika

mid = i + j div 2	1 kali
i = mid + 1	1 kali
j = mid – 1	1 kali

$\text{Jumlah} = 3$

Operasi input/output

input elemen array	1 kali
output hasil	1 kali

$\text{Jumlah} = 2$

$T_{avg}(n) = n + 8 + 3 + 2$
$= n + 13$

Best case: ditemukan pada elemen array ke- n div 2

Operasi assignment

arr[n] <- elemen array	n kali
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

$\text{Jumlah} = \text{n} + 6$

Operasi aritmatika

mid = i + j div 2	1 kali

$\text{Jumlah} = 1$

Operasi input/output

input elemen array	1 kali
output hasil	1 kali

$\text{Jumlah} = 2$

$T_{best}(n) = n + 6 + 1 + 2$
$= n + 9$

Worst case: tidak ditemukan

Operasi assignment

arr[n] <- elemen array	n kali
i <- 1	1 kali
j <- n	1 kali
found <- false	1 kali
mid <- (i + j) div 2	1 kali
i <- mid + 1 OR j <- mid – 1	n/2 kali
idx <- mid	1 kali

$$\text{Jumlah} = \frac{3n}{2} + 4$$

Operasi aritmatika

mid = i + j div 2	1 kali
i = mid + 1 OR j = mid - 1	n/2 kali

$$\text{Jumlah} = \frac{3}{2}n$$

Operasi input/output

input elemen array	1 kali
output hasil	1 kali

$$\text{Jumlah} = 2$$

$$T_{worst}(n) = \frac{3n}{2} + 4 + \frac{3n}{2} + 2$$
$$= 3n + 6$$

Studi kasus 4: Insertion sort

```c
#include <stdio.h>
#include <math.h>

void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++)
   {
       key = arr[i];
       j = i-1;

       while (j >= 0 && arr[j] > key)
       {
           arr[j+1] = arr[j];
           j = j-1;
       }
       arr[j+1] = key;
   }
}

void printArray(int arr[], int n)
{
```

```
   int i;
   for (i=0; i < n; i++)
       printf("%d ", arr[i]);
   printf("\n");
}

int main()
{
    int arr[] = {12, 11, 13, 5, 6};
    int n = sizeof(arr)/sizeof(arr[0]);

    insertionSort(arr, n);
    printArray(arr, n);

    return 0;
}
```

worst case scenario: array terurut terbalik

Operasi assignment

i <- 2	1 kali
Insert <- xi	1 kali
j <- i	1 kali
x[j] <- x[j-1]	n kali
j <- j-1	n^2 kali
x[j] <- insert	n^2 kali
i <- i + 1	n kali

$\text{Jumlah} = 2n^2 + 2n + 3$

operasi aritmatika

i+1	n-1 kali

$\text{Jumlah} = \text{n} - 1$

Operasi input/output

Input elemen array	n kali
Output sorted array	1 kali

$\text{Jumlah} = \text{n} + 1$

$T_{worst}(n) = 2n^2 + 4n + 3$

Average case scenario: array tidak terurut

Operasi assignment

i <- 2	1 kali
Insert <- xi	1 kali
j <- i	1 kali
x[j] <- x[j-1]	n kali
j <- j-1	n^2 kali
x[j] <- insert	n^2 kali
i <- i + 1	n kali

$\text{Jumlah} = 2n^2 + 2n + 3$

operasi aritmatika

i+1	n-1 kali

$\text{Jumlah} = \text{n} - 1$

Operasi input/output

Input elemen array	n kali
Output sorted array	1 kali

$\text{Jumlah} = \text{n} + 1$

$T_{average}(n) = 2n^2 + 4n + 3$

Best case scenario: array sudah terurut

worst case scenario: array terurut terbalik

Operasi assignment

i <- 2	1 kali
Insert <- xi	1 kali
j <- i	1 kali
x[j] <- x[j-1]	0 kali
j <- j-1	0 kali
x[j] <- insert	n kali
i <- i + 1	n kali

$\text{Jumlah} = 2\text{n} + 3$

operasi aritmatika

	i+1	n-1 kali

$$\text{Jumlah} = n - 1$$

Operasi input/output

	Input elemen array	n kali
	Output sorted array	1 kali

$$\text{Jumlah} = n + 1$$

$$T_{best}(n) = 4n + 3$$

Studi kasus 5: Selection sort

```c
#include <stdio.h>

void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void selectionSort(int arr[], int n)
{
    int i, j, min_idx;

    for (i = 0; i < n-1; i++)
    {
        min_idx = i;
        for (j = i+1; j < n; j++)
        if (arr[j] < arr[min_idx])
            min_idx = j;

        swap(&arr[min_idx], &arr[i]);
    }
}

void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
```

```
    printf("\n");
}

int main()
{
    int arr[] = {64, 25, 12, 22, 11};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

Average case scenario: array belum terurut

Operasi assignment

i <- n	n kali
imaks <- 1	1 kali
j <- 2	1 kali
imaks <- j	n^2 kali
temp <- x_i	n kali
x_i <- x_{imaks}	n kali
x_{imaks} <- temp	n kali

$\text{Jumlah} = n^2 + 4n + 2$

Operasi aritmatika

i = i – 1	n-1 kali
j = j + 1	n kali

$\text{Jumlah} = 2\text{n} - 1$

Operasi input/output

Input array	n kali
Output array	n kali

$\text{Jumlah} = 2\text{n}$

$T_{avg}(\text{n}) = n^2 + 8n + 1$

Worst case scenario: array terurut terbalik

Operasi assignment

i <- n	n kali
imaks <- 1	1 kali
j <- 2	1 kali
imaks <- j	n^2 kali
temp <- x_i	n kali
x_i <- x_{imaks}	n kali
x_{imaks} <- temp	n kali

$\text{Jumlah} = n^2 + 4n + 2$

Operasi aritmatika

i = i – 1	n-1 kali
j = j + 1	n kali

$\text{Jumlah} = 2\text{n} - 1$

Operasi input/output

Input array	n kali
Output array	n kali

$\text{Jumlah} = 2\text{n}$

$T_{worst}(\text{n}) = n^2 + 8n + 1$

Average case scenario: array sudah terurut

Operasi assignment

i <- n	n kali
imaks <- 1	1 kali
j <- 2	1 kali
imaks <- j	n^2 kali
temp <- x_i	n kali
x_i <- x_{imaks}	n kali
x_{imaks} <- temp	n kali

$\text{Jumlah} = n^2 + 4n + 2$

Operasi aritmatika

i = i – 1	n-1 kali

j = j + 1	n kali

$\text{Jumlah} = 2\text{n} - 1$

Operasi input/output

Input array	n kali
Output array	n kali

$\text{Jumlah} = 2\text{n}$

$T_{best}(\text{n}) = n^2 + 8n + 1$