Analisa Algoritma Search dan Sort

Naufal Aulia - 140810180074

Kasus 1 Pencarian Nilai Maksimal

```
int max = input[0];          // 2 operasi
int i =  1;                  // 2 operasi

while (i <= n)               // n-1 operasi
{
    if (input[i] > max)      // n-2 operasi
    max = input[i];          // 0 s.d. n-2 operasi
    i = i +  1;              // 2(n-1) operasi
}
```

Kompleksitas Waktu

$$T_{min}(n) = 2 + 2 + (n-1) + (n-2) + 0 + 2(n-1) = 3n - 1$$

$$T_{avg}(n) = 2 + 2 + (n-1) + (n-2) + (n-2)/2 + 2(n-1) = 3.5n - 2$$

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

Kasus 2 *Sequential Search*

```
int idx;                     // 1 operasi
int i =  0;                  // 2 operasi
bool found =  false;         // 2 operasi

while (i < n &&  !found)     // 2(3) s.d. 3(n-1) operasi
{
    if (input[i] == y)       // 1 s.d. n-1 operasi
    {
        found =  true;       // 0 atau n operasi
    }
    else
```

```
    {
        i++;                // 0 s.d. n-1 operasi
    }
}

if (found)                  // 1 operasi
    idx = i;                // 0 atau 1 operasi
else
    idx =  -1;              // 0 atau 1 operasi
```

Kompleksitas Waktu

$$T_{min}(n) = 1 + 2 + 2 + 3(n-1) + 1 + 1 + 0 + 1 + 1 = 3n + 6$$

$$T_{avg}(n) = 1 + 2 + 2 + 6 + (n-1)/2 + 1 + (n-1)/2 + 1 + 1 = 2n + 11$$

$$T_{max}(n) = 1 + 2 + 2 + 3(n-1) + (n-1) + 0 + (n-1) + 1 + 1 = 5n + 2$$

Kasus 3 *Binary Search*

```
bool found =  false;            // 2 operasi
int i =  0;                     // 2 operasi
int j = n;                      // 2 operasi
int mid;                        // 1 operasi

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

```
    }
}
```

```
if (found)                  // 1 operasi
    idx = i;                // 0 atau 1 operasi
else
    idx =  -1;              // 0 atau 1 operasi
```

Kompleksitas Waktu

$T_{min}(n) = O(1)$

$T_{avg}(n) = O(logn)$

$T_{max}(n) = O(logn)$

Kasus 4 *Insertion Sort*

```
int i, j, insert;               // 3 operasi

for (i =  1; i < n; i++)        // n operasi
{
    insert = input[i];          // n-1 operasi
    j = i;                      // n-1 operasi
    while (j >  0  && input[j -  1] > insert)
    {
        input[j] = input[j -  1];
        j = j -  1;
    }
    input[j] = insert;
}
```

Kompleksitas Waktu

$T_{min}(n) = n$

$T_{avg}(n) = n^2$

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

Kasus 5 *Selection Sort*

```
int i, j, imax, temp;                    // 4 operasi

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

Kompleksitas Waktu

$T_{min}(n) = n^2$

$T_{avg}(n) = n^2$

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