Tugas Praktikum Analisis Algoritma

KOMPLEKSITAS WAKTU



Dibuat Oleh :

Adryan Luthfi Faiz 140810160049

**UNIVERSITAS PADJADJARAN**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**2019**

Mencari Nilai Max

procedure CariMaks(input x1, x2, …, xn: integer, output maks: integer)

{ Mencari elemen terbesar dari sekumpulan elemen larik integer x1, x2, …, xn. Elemen terbesar akan disimpan di dalam maks  
Input: x1, x2, …, xn  
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

{i > n}

PROGRAM C++

#include <iostream>

using namespace std;

int main(){

int n;

int x[10];

cout << "Masukkan Jumlah Data : ";

cin >> n;

for (int i = 0; i < n; i++){

cout << "Masukkan Data ke - " << i+1 << " : ";

cin >> x[i];

}

int maks = x[0];

int i = 1;

while (i <= n) {

if (x[i] > maks)

maks = x[i];

i++;

}

cout << "Maksimum Number : " << maks << endl;

return 0;

}

Kompleksitas Waktu

maks 🡨 x1 1 kali

i 🡨 2 1 kali

maks 🡨 xi n kali

i 🡨 i + 1 n kali

Sequential Search

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 xi = y then

found 🡨 true

else

i 🡨 i + 1

endif  
 endwhile

PROGRAM C++

#include <iostream>

using namespace std;

int main() {

int n;

int x[10];

cout << "Masukkan Jumlah Data : ";

cin >> n;

for (int i = 0; i < n; i++){

cout << "Masukkan Data ke - " << i+1 << " : ";

cin >> x[i];

}

int y;

cout << "Masukkan yang dicari : ";

cin >> y;

int i = 0;

bool found = false;

int idx;

while ((i < n) && (!found)){

if (x[i] == y)

found = true;

else

i++;

}

if (found)

idx = i+1;

else

idx = 0;

cout << "Yang dicari berada di urutan : " << idx << endl;

return 0;

}

Kompleksitas Waktu

* Best Case

i 🡨1 1 kali

found 🡨false 1 kali

found 🡨true 1 kali

idx 🡨 I 1 kali

* Average Case

i 🡨1 1 kali

found 🡨false 1 kali

i 🡨i + 1 ½ n kali

found 🡨true 1 kali

idx 🡨I 1 kali

* Worst Case

i 🡨1 1 kali

found 🡨false 1 kali

i 🡨i + 1 n kali

found 🡨true 1 kali

idx 🡨I 1 kali

Binary Search

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 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

{*i < n or found*}

If found then {*y ditemukan*}

idx 🡨 i

else

idx 🡨 0 {y tidak ditemukan}

endif

PROGRAM C++

#include <iostream>

using namespace std;

int main() {

int n;

int x[10];

cout << "Masukkan Jumlah Data : ";

cin >> n;

for (int i = 0; i < n; i++){

cout << "Masukkan Data ke - " << i+1 << " : ";

cin >> x[i];

}

int y;

cout << "Masukkan yang dicari : ";

cin >> y;

int i = 0;

int j = n-1;

bool found = false;

int idx;

int mid;

while ((i <= j) && (!found)){

mid = (i + j)/2;

if (x[mid] == y)

found = true;

else{

if (x[mid] < y)

i = mid + 1;

else

j = mid - 1;

}

}

if (found)

idx = mid+1;

else

idx = 0;

cout << "Yang dicari berada di urutan : " << idx << endl;

return 0;

}

Kompleksitas Waktu

* Best Case

i 🡨1 1 kali

j 🡨n 1 kali

found 🡨false 1 kali

mid 🡨(i + j) div2 1 kali

found 🡨true 1 kali

Idx 🡨mid 1 kali

* Average Case

i 🡨1 1 kali

j 🡨n 1 kali

found 🡨false 1 kali

mid 🡨(i + j) div2 ½ n + 1 kali

i 🡨mid + 1 or j 🡨mid –1 ½ n kali

found 🡨true 1 kali

Idx 🡨mid 1 kali

`

* Worst Case

i 🡨1 1 kali

j 🡨n 1 kali

found 🡨false 1 kali

mid 🡨(i + j) div2 n + 1 kali

i 🡨mid + 1 or j 🡨mid –1 n kali

found 🡨true 1 kali

Idx 🡨mid 1 kali

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 🡨 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  
 {*i < n or found*}  
 If found then {*y ditemukan*}

PROGRAM C++

#include <iostream>

using namespace std;

int main()

{

int n;

int x[10];

cout << "Masukkan Jumlah Data : ";

cin >> n;

for (int i = 0; i < n; i++)

{

cout << "Masukkan Data ke - " << i+1 << " : ";

cin >> x[i];

}

cout << "Data Sebelum di Sorting : ";

for (int i = 0; i < n; i++)

cout << x[i] << " ";

cout << endl;

int insert;

int j;

for (int i = 1; i < n; i++)

{

insert = x[i];

j = i-1;

while ((j >= 0) && (x[j] > insert))

{

x[j+1] = x[j];

j--;

}

x[j+1] = insert;

}

cout << "Data setelah di Sorting : ";

for (int i = 0; i < n; i++)

cout << x[i] << " ";

return 0;

}

Kompleksitas Waktu

* Best Case

For i 🡨2 to n do 1 kali

insert 🡨xi n kali

j 🡨i n kali

x[j] = insert n kali

* Average Case

For i 🡨2 to n do 1 kali

insert 🡨xi n kali

j 🡨I n kali

x[j]🡨x[j-1] n \* ½ n kali

j🡨j-1 n \* ½ n kali

x[j] = insert n kali

* Worst Case

For i 🡨2 to n do 1 kali

insert 🡨xi n kali

j 🡨i n kali

x[j]🡨x[j-1] n \* n kali

j🡨j-1 n \* n kali

x[j] = insert n kali

Selection 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 xj > ximaks then

imaks 🡨 j

endif

endfor

{pertukarkan ximaks dengan xi}

temp 🡨 xi

xi 🡨 ximaks

ximaks 🡨 temp

endfor

PROGRAM C++

#include <iostream>

using namespace std;

int main(){

int n;

int x[10];

cout << "Masukkan Jumlah Data : ";

cin >> n;

for (int i = 0; i < n; i++){

cout << "Masukkan Data ke - " << i+1 << " : ";

cin >> x[i];

}

cout << "Data Sebelum di Sorting : ";

for (int i = 0; i < n; i++)

cout << x[i] << " ";

cout << endl;

int imaks;

int temp;

for (int i = n-1; i >= 1; i--){

imaks = 0;

for (int j = 1; j <= i; j++){

if (x[j] > x[imaks])

imaks = j;

}

temp = x[i];

x[i] = x[imaks];

x[imaks] = temp;

}

cout << "Data setelah di Sorting : ";

for (int i = 0; i < n; i++)

cout << x[i] << " ";

return 0;

}

Kompleksitas Waktu

* Best Case

for i 🡨n downto 2 do 1 kali

imaks 🡨1 n kali

for j 🡨2 to i do n kali

imaks 🡨j n\*1 kali

temp 🡨xi n kali

xi🡨ximaks n kali

ximaks🡨temp n kali

* Average Case

for i 🡨n downto 2 do 1 kali

imaks 🡨1 nkali

for j 🡨2 to i do n kali

imaks 🡨j n \* ½ n kali

temp 🡨xi n kali

xi🡨ximaks n kali

ximaks🡨temp n kali

* Worst Case

for i 🡨n downto 2 do 1 kali

imaks 🡨1 n kali

for j 🡨2 to i do n kali

imaks 🡨j n \* n kali

temp 🡨xi n kali

xi🡨ximaks n kali

ximaks🡨temp n kali