**PENERAPAN DAN PERBANDINGAN KOMPLEKSITAS WAKTU ALGORITMA *INSERTION SORT* DAN *SELECTION SORT***



**REZA IRFAN WIJAYA**

**19102149**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

**2023**

1. **Dasar Teori**

* *Algorithm Selection Sort*

*Algorithm selection sort* didasarkan pada gagasan untuk menemukan elemen minimum atau maksimum dalam larik yang tidak diurutkan dan kemudian meletakkannya di posisi yang benar dalam larik yang diurutkan.

Asumsikan bahwa array A=[7, 5, 4, 2] perlu diurutkan dalam urutan menaik.

Elemen minimum dalam array yaitu 2 dicari dan kemudian ditukar dengan elemen yang saat ini terletak di posisi pertama, yaitu 7. Sekarang elemen minimum di sisa array yang tidak disortir dicari dan diletakkan di posisi kedua, dan seterusnya.

* *Algorithm Insertion Sort*

*Algorithm Insertion Sort* didasarkan pada gagasan bahwa satu elemen dari elemen input dikonsumsi di setiap iterasi untuk menemukan posisinya yang benar, yaitu posisi yang dimilikinya dalam array yang diurutkan. Ini mengulangi elemen input dengan menumbuhkan array yang diurutkan pada setiap iterasi. [1]

Ini membandingkan elemen saat ini dengan nilai terbesar dalam array yang diurutkan. Jika elemen saat ini lebih besar, maka ia meninggalkan elemen di tempatnya dan pindah ke elemen berikutnya jika tidak menemukan posisi yang benar dalam array yang diurutkan dan memindahkannya ke posisi itu. Ini dilakukan dengan menggeser semua elemen, yang lebih besar dari elemen saat ini, dalam array yang diurutkan ke satu posisi di depan. [2]

1. **Implementasi**

* Spesifikasi *Hardware*
* RAM : 12288MB
* CPU : Inter® Core™ i3-5005U CPU @ 2.00Ghz (4 CPUs), ~2.0Ghz
* Spesifikasi *Software*
* *Operating System* : Windows 10
* Bahasa Pemrograman : Golang v1.17
* IDE : Visual Studio Code v1.66
* *Selection Sort*
* Pesudecode

procedure SelectionSort(arr : array of integer, i,j : integer);

{Mengurutkan array arr dari i sampai j dengan algoritma Selection Sort

Masukan : arr = array yang akan diurutkan, i = indeks awal, j = indeks akhir

Luaran : arr = array yang telah diurutkan

}

START

*// ketika nilai i kurang dari j maka lakukan semua proses dalam block ini*

if i < j then

begin

*// lakukan partisi*

p <- Partisi3(arr, i, j, k);

*// urutkan bagian kiri dari partisi*

arr <- SelectionSort(arr, i, p-1);

*// urutkan bagian kanan dari partisi*

arr <- SelectionSort(arr, p+1, j);

end

*// kembalikan nilai arr*

return arr;

END;

procedure Partisi3(arr : array of integer, i,j,: integer);

{Membagi array arr menjadi dua bagian, yaitu bagian kiri dan bagian kanan yang tidak sama panjangnya}

{Masukan : arr = array yang akan dipartisi, i = indeks awal, j = indeks akhir

Luaran : arr = array yang telah dipartisi

}

START

*// lakukan perulangan sebanyak jumlah elemen array*

for k <- i to j do

begin

*// inisialisasi nilai minimal*

min <- i

*// jika nilai dari arr[k] lebih kecil dari arr[min] maka nilai minimal akan diubah*

if arr[k] < arr[min] then

begin

min <- k;

end;

*//lakukan pertukaran posisi*

arr[min], arr[i] <- arr[i], arr[min];

end;

return i;

END;

* Screenshot Program *Selection Sort*



* *Insertion Sort*
* Pseudecode

prosedure InsertionSort (var arr: array of integer; i,j: integer);

{Mengurutkan element array arr dengan Insertion Sort

i dan j adalah indeks awal dan akhir array arr

Masukan : arr[i..j]

Luaran : arr[i..j] terurut

}

START

*// ketika i lebih kecil dari j*

if i < j then

begin

*// inisiasi nilai awal*

k <- i

*// rekursif*

InsertionSort (arr, k+1, j)

*// merge*

Merge(arr, i, k, j)

end

return arr

END

prosedure Merge(var arr: array of integer; i,k,j: integer);

{Menggabngkan element array arr dengan Merge Sort

i, k, dan j adalah indeks awal, tengah, dan akhir array arr

Masukan : arr[i..k] dan arr[k+1..j]

Luaran : arr[i..j] terurut

}

START

*// looping element array arr*

for i <= k && k <= j do

begin

*// jika element array arr[i] lebih besar dari element array arr[k]*

if arr[i] > arr[k] then

begin

*// tukar element*

arr[i], arr[k] = arr[k], arr[i]

*// increment i*

i <- i + 1

*// incerment k*

k <- k + 1

end

else

begin

*// increment k*

k <- k + 1

end

end

return arr

END

* Screenshot Program *Insertion Sort*
* 

**Referensiii**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | [1] | hackerearth, "hackerarth," [Online]. Available: <https://www.hackerearth.com/practice/algorithms/sorting/selection-sort/tutorial/>. [Accessed 18 April 2022]. |  |  |  | | --- | --- | | [2] | hackerearth, "hackerearth," [Online]. Available: <https://www.hackerearth.com/practice/algorithms/sorting/insertion-sort/tutorial/>. [Accessed 18 April 2022]. | |  |  | |