Laporan Praktikum

Struktur Data



Disusun Oleh :

**Ahda Rindang Al-Amin (2311531003)**

Dosen Pengampu : Dr. Wahyudi, S.T, M.T

Departemen Informatika

Fakultas Teknologi Informasi

Universitas Andalas

Tahun 2024

**Algoritma Sorting 1**

1. **Tujuan Praktikum**
2. Memahami algoritma Insertion Sort pada struktur data.
3. Memahami algoritma Selection Sort pada struktur data.
4. Memahami algoritma Bubble Sort pada struktur data.
5. **Pendahuluan**

Algoritma pengurutan (sorting) meletakkan elemen data kedalam kumpulan data urutan tertentu, proses pengurutan yang sebelumnya data disusun acak menjadi tersusun teratur menurut aturan tertentu. Pengurutan merupakan hal yang tidak bisa dipisahkan dari dunia komputer. Sekarang ini Google dikenal sebagai mesin pencari terbesar di dunia. Dalam hitungan detik dapat diperoleh informasi yang diinginkan. Adanya kebutuhan terhadap proses pengurutan memunculkan bermacam-macam metode pengurutan yang bertujuan untuk memperoleh metode pengurutan yang optimal.

Pengurutan data (sorting) didefinisikan sebagai proses untuk menyusun kembali himpunan objek dengan menggunakan aturan tertentu. Tujuannya adalah untuk mendapatkan kemudahan dalam pencarian dari suatu himpunan. Kelebihan suatu data yang terurut adalah mudah untuk dicek apabila ada data yang hilang.

Data terkadang berada dalam bentuk yang tidak berpola ataupun dengan pola tertentu yang diinginkan. Tidak ada algoritma terbaik untuk semua keadaan, kadang kala sebuah algortitma sangat efisien ketika jumlah datanya sedikit, namun kinerjanya menjadi berkurang ketika jumlah data ditambahkan atau meningkat. Meskipun memiliki kemampuan komputasi yang lebih tinggi, namun jika menggunakan algoritma yang kurang efisien, maka akan membutuhkan waktu lebih lama. Sehingga untuk memecahkan permasalahan diperlukan sebuah algoritma yang efektif dan efisien agar persoalan komputasi serta terbatasnya alokasi memori dapat diatasi.

Secara umum ada dua jenis pengurutan data yaitu: 1) Model urut naik (ascending) yang mengurutkan data dari yang mempunyai nilai terkecil sampai terbesar. 2) Model urut turun (descending) yang mengurutkan data dari yang mempunyai nilai terbesar sampai terkecil.

Algoritma pengurutan data yang sering ditemukan dalam literatur komputer antara lain bubble sort, selection sort, insertion sort, heap sort, shell sort, quick sort, merge sort, radix sort, dan tree sort. Semua algoritma pengurutan selalu melakukan operasi perbandingan data untuk menemukan posisi urutan yang tepat. Berdasarkan tempat penyimpanan data, sorting dibedakan antara external sort dan internal sort. External sort bila datanya berada dalam media external, atau external storage seperti hardisk. Internal sort bila datanya ada dalam internal storage atau memory komputer.

1. **Metode Praktikum**
2. **Algoritma Insertion Sort**

Algoritma insertion sort, adalah metode pengurutan dengan cara menyisipkan elemen data pada posisi yang tepat. Pencarian posisi yang tepat dilakukan dengan melakukan pencarian berurutan didalam barisan elemen, selama pencarian posisi yang tepat dilakukan pergeseran elemen. Pengurutan insertion sort sangat mirip dengan konsep permainan kartu, bahwa setiap kartu disisipkan secara berurutan dari kiri ke kanan sesuai dengan besar nilai kartu tersebut, dengan syarat apabila sebuah kartu disisipkan pada posisi tertentu kartu yang lain akan bergeser maju atau mundur sesuai dengan besaran nilai yang dimiliki.

Berikut method yang menggunakan algoritma Insertion sort beserta method main yang menerapkannya :

public class InsertionSort {

//function to sort array using insertion sort

void sort(int arr[]) {

int n = arr.length;

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

int key = arr[i];

int j= i-1;

/\*Move element of arr[0..i-1],

\* that are greater than the key,

\* to one position ahead of their

\* current position

\*/

while (j>=0 && arr[j] > key) {

arr[j+1] = arr[j];

j=j-1;

}

arr[j+1] = key;

}

}

static void printArray(int arr[]) {

int n = arr.length;

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

System.***out***.print(arr[i]+" ");

} System.***out***.println();

}

//Driver Method

public static void main(String[] args) {

int arr[] = {3,10,4,6,8,9,7,2,1,5};

InsertionSort ob = new InsertionSort();

ob.sort(arr);

*printArray*(arr);

}

}

Method sort pada kelas InsertionSort berfungsi mengurutkan isi dari sebuah array dengan prinsip algoritma insertion sort. Method printArray berfungsi untuk memberikan output print elemen-elemen dari array parameter.

Output dari program tersebut adalah isi dari array arr yang telah terurut :

1 2 3 4 5 6 7 8 9 10

Bisa juga program tersebut tidak menggunakan method printArray, tetapi perintah print diletakkan di method main. Berikut kelas yang tidak menggunakan method printArray :

public class InsertionSort2 {

public static void InsertionSort(int[] arr) {

int n = arr.length;

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

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j+1] = arr[j];

j--;

}

arr[j+1] = key;

}

}

public static void main(String[] args) {

int arr[]= {23,78,45,8,32,56,1};

int n = arr.length;

System.***out***.printf("array yang belum terurut:\n");

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

System.***out***.print(arr[i] + " ");

System.***out***.println();

*InsertionSort*(arr);

System.***out***.printf("array yang belum terurut:\n");

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

System.***out***.print(arr[i] + " ");

System.***out***.println();

}

}

Output yang dihasilkan menampilkan elemen-elemen array arr sebelum dan sesudah diurutkan :

array yang belum terurut:

23 78 45 8 32 56 1

array yang belum terurut:

1 8 23 32 45 56 78

1. **Algoritma Selection Sort**

Algoritma selection sort sering juga disebut dengan metode maksimum atau minimum. Metode maksimum karena didasarkan pada pemilihan data atau elemen maksimum sebagai dasar pengurutan. Konsepnya dengan memilih elemen maksimum kemudian mempertukarkan elemen maksimum tersebut dengan elemen paling akhir untuk urutan ascending dan elemen pertama untuk descending.

Algoritma selection sort disebut juga dengan metode minimum karena didasarkan pada pemilihan elemen minimum sebagai dasar pengurutan. Konsepnya dengan memilih elemen minimum kemudian mempertukarkan elemen minimum dengan elemen pertama untuk urutan ascending dan elemen paling akhir untuk urutan descending. Proses yang dilakukan oleh algoritma selection sort adalah mengambil nilai terbesar dari susunan data dan menggantikannya dengan data yang paling kanan.

Berikut method yang menggunakan algoritma selection sort beserta method main yang menerapkannya ;

public class SelectionSort {

public static void SelectionSort (int[] arr) {

int n = arr.length;

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

int minIndex = i;

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

if(arr[j] < arr[minIndex]) {

minIndex = j;

}

}

int temp = arr[i];

arr[i] = arr[minIndex];

arr[minIndex] = temp;

}

}

public static void main(String[] args) {

int arr[] = { 23, 78, 45, 8, 32, 56, 1};

int n= arr.length;

System.***out***.printf("array yang belum terurut:\n");

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

System.***out***.print(arr[i] + " ");

System.***out***.println();

*SelectionSort*(arr);

System.***out***.printf("array yang terurut:\n");

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

System.***out***.print(arr[i] + " ");

System.***out***.println();

}

}

Berikut output dari program tersebut:

array yang belum terurut:

23 78 45 8 32 56 1

array yang terurut:

1 8 23 32 45 56 78

1. **Algoritma Bubble Sort**

Bubble Sort merupakan cara pengurutan yang sederhana. Konsep dari ide dasarnya adalah seperti “gelembung air” untuk elemen struktur data yang semestinya berada pada posisi awal. Cara kerjanya adalah dengan berulang-ulang melakukan traversal (proses looping) terhadap elemen-elemen struktur data yang belum diurutkan. Di dalam traversal tersebut, nilai dari dua elemen struktur data dibandingkan. Jika ternyata urutannya tidak sesuai dengan “pesanan”, maka dilakukan pertukaran (swap). Algoritma sorting ini disebut juga dengan comparison sort dikarenakan hanya mengandalkan perbandingan nilai elemen untuk mengoperasikan elemennya.

Berikut program yang menggunakan algoritma bubble sort :

public class BubbleSort {

public static void bubbleSort(int[] arr) {

int n=arr.length;

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

for (int j=0; j<n-i-1; j++) {

if (arr[j] > arr[j+1]) {

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

public static void main(String[] args) {

int arr[]= {23,78,45,8,32,56,1};

int n = arr.length;

System.***out***.printf("array yang belum terurut:\n");

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

System.***out***.print(arr[i] + " ");

System.***out***.println();

*bubbleSort*(arr);

System.***out***.printf("array yang terurut:\n");

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

System.***out***.print(arr[i] + " ");

System.***out***.println();

}

}

Berikut output dari program tersebut:

array yang belum terurut:

23 78 45 8 32 56 1

array yang terurut:

1 8 23 32 45 56 78

1. **Kesimpulan Praktikum**

Pengurutan data (sorting) didefinisikan sebagai proses untuk menyusun kembali himpunan objek dengan menggunakan aturan tertentu. Tujuannya adalah untuk mendapatkan kemudahan dalam pencarian dari suatu himpunan. Beberapa contoh algoritma sorting yang populer dalam literatur komputer diantaranya insertion sort, selection sort dan bubble sort.

Algoritma *insertion sort* meengurutkan data dengan cara menyisipkan elemen data pada posisi yang tepat. Sedangkan algoritma *selection sort*, konsepnya adalah dengan memilih elemen minimum kemudian mempertukarkan elemen minimum dengan elemen pertama untuk urutan *ascending* dan elemen paling akhir untuk urutan *descending*. Selain itu, algoritma *Bubble sort* adalah algoritma pengurutan sederhana yang bekerja dengan cara membandingkan dan menukar elemen-elemen bersebelahan jika berada dalam urutan yang salah. Proses ini diulangi hingga tidak ada lagi elemen yang perlu ditukar.

Pemilihan algoritma pengurutan yang tepat tergantung pada berbagai faktor seperti ukuran data, kebutuhan stabilitas pengurutan, dan kompleksitas implementasi. Inilah mengapa banyak algoritma sorting dikembangkan, masing-masing dengan keunggulan dan kelemahan yang berbeda untuk memenuhi berbagai kebutuhan pengurutan data. Dengan algoritma yang baik dapat dihasilkan sebuah program yang efisien dari segi waktu dan hasil yang dicapai. Kompleksitas waktu atau kecepatan eksekusi dari algoritma sorting menjadi faktor penting dalam memilih algoritma yang tepat untuk aplikasi tertentu. Selain itu, sebuah algoritma juga menjadi sangat efisien ketika jumlah datanya sedikit, yang kinerjanya akan berkurang ketika jumlah data ditambahkan atau meningkat.