

LAPORAN PRAKTIKUM STRUKTUR DATA

LAPORAN AWAL PRAKTIKUM

Pertemuan ke-08

Sorting Lanjut-1



Disusun Oleh:

Nama Lengkap : Nova Ardiansyah

NIM : 211011401309

Kelas : 04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. RANGKUMAN MATERI

Sorting adalah proses mengatur atau menyusun elemen-elemen dalam satu kumpulan data dengan aturan tertentu. Dalam program C++, terdapat beberapa algoritma sorting yang digunakan untuk mengurutkan array atau struktur data lainnya, seperti Bubble Sort, Selection Sort, dan Insertion Sort. Keuntungan menggunakan algoritma sorting yaitu menghemat waktu dan memudahkan dalam mencari nilai tertentu pada suatu array yang sudah terurut.

Algoritma Bubble Sort merupakan salah satu metode pengurutan sederhana dalam pemrograman C++. Pengurutan dilakukan dengan cara membandingkan 2 angka dalam sebuah array, jika angka pertama lebih besar dari angka kedua, maka mereka akan ditukar posisinya. Proses ini berlangsung terus-menerus hingga tidak ada nilai yang ditukar lagi. Algoritma Bubble Sort sangat mudah dipahami dan diimplementasikan oleh pemula, tetapi memiliki kompleksitas waktu yang tinggi saat jumlah data semakin banyak.

Selain itu, algoritma Selection Sort juga sering digunakan dalam pemrograman C++ untuk mengurutkan data. Cara kerja Selection Sort yaitu mencari angka terkecil dalam array dan menukar posisi angka tersebut dengan angka pertama dalam array. Kemudian proses diulang untuk subarray yang belum terurut sampai seluruh array tersusun dengan benar. Meskipun lebih cepat daripada Bubble Sort, Selection Sort masih sulit diimplementasikan untuk array yang sangat besar. Oleh karena itu, untuk data yang bersifat dinamis dan kompleks, diperlukan algoritma sorting yang lebih efisien seperti Merge Sort atau Quick Sort.

B. TUGAS PENDAHULUAN

1. Jelaskan kekurangan menggunakan metode Maksimum/Minimum Sort dengan metode-metode Sorting lainnya!

Jawab :

- a) Kinerja yang buruk pada data yang besar: Metode Maksimum/Minimum Sort memiliki kompleksitas waktu yang cukup tinggi. Dalam setiap iterasi, metode ini mencari elemen maksimum atau minimum, yang membutuhkan waktu linear terhadap jumlah elemen. Oleh karena itu, pada data yang

besar, metode ini akan bekerja lebih lambat dibandingkan dengan metode sorting lain yang memiliki kompleksitas waktu yang lebih efisien.

- b) Ketidakstabilan: Metode Maksimum/Minimum Sort cenderung tidak menjaga urutan relatif antara elemen-elemen dengan nilai yang sama. Hal ini berarti jika terdapat elemen-elemen yang sama dalam data yang diurutkan, posisi relatif mereka dapat berubah setelah proses pengurutan. Ini dapat menjadi masalah dalam beberapa kasus di mana urutan relatif elemen-elemen yang sama harus dipertahankan.
- c) Keterbatasan penggunaan: Metode Maksimum/Minimum Sort hanya cocok untuk pengurutan data dalam jumlah kecil atau kasus sederhana. Pada data yang lebih besar atau kompleks, metode sorting lain yang lebih efisien seperti Quicksort, Mergesort, atau Heapsort umumnya lebih disarankan.

Dalam banyak kasus, metode Maksimum/Minimum Sort digunakan untuk tujuan pendidikan atau sebagai langkah sederhana dalam implementasi algoritma sorting yang lebih kompleks. Untuk pengurutan data yang efisien dan stabil, sebaiknya menggunakan metode sorting yang lebih canggih dan dioptimalkan.

2. Jelaskan perbedaan program Sorting dengan menggunakan antara metode Maksimum Sort dan Minimum Sort

Jawab :

Perbedaan antara metode Maksimum Sort dan Minimum Sort terletak pada cara mereka mencari dan menggeser elemen selama proses pengurutan. Berikut adalah penjelasan perbedaannya:

Metode Maksimum Sort:

- a) Pada metode Maksimum Sort, dalam setiap iterasi, elemen maksimum dicari dalam data yang belum diurutkan.

- b) Setelah elemen maksimum ditemukan, elemen tersebut dipindahkan ke posisi terakhir atau ke posisi yang sesuai sesuai dengan urutan pengurutan yang diinginkan.
- c) Proses ini berlanjut dengan mengurangi rentang data yang belum diurutkan dan mencari elemen maksimum dalam rentang yang lebih kecil, hingga seluruh data terurut.

Metode Minimum Sort:

- a) Pada metode Minimum Sort, dalam setiap iterasi, elemen minimum dicari dalam data yang belum diurutkan.
- b) Setelah elemen minimum ditemukan, elemen tersebut dipindahkan ke posisi awal atau ke posisi yang sesuai sesuai dengan urutan pengurutan yang diinginkan.
- c) Proses ini berlanjut dengan mengurangi rentang data yang belum diurutkan dan mencari elemen minimum dalam rentang yang lebih kecil, hingga seluruh data terurut.

Dengan demikian, perbedaan utama antara metode Maksimum Sort dan Minimum Sort adalah dalam cara mereka menentukan elemen yang akan dipindahkan selama proses pengurutan. Metode Maksimum Sort mencari elemen maksimum, sementara Metode Minimum Sort mencari elemen minimum dalam setiap iterasinya.

3. Jelaskan tahapan-tahapan Sorting menggunakan metode Maximum Sort!

Jawab :

Berikut adalah tahapan-tahapan dalam sorting menggunakan metode Maximum Sort:

- a) Inisialisasi: Mulailah dengan sekelompok data yang akan diurutkan. Misalnya, kita memiliki array dengan n elemen yang perlu diurutkan.

- b) Iterasi: Lakukan iterasi sebanyak $(n-1)$ kali, di mana n adalah jumlah elemen dalam data. Iterasi ini akan mengurangi jumlah elemen yang belum diurutkan setiap kali.
- c) Cari elemen maksimum: Pada setiap iterasi, cari elemen maksimum dalam rentang data yang belum diurutkan. Rentang data yang belum diurutkan akan berkurang setiap iterasi.
- d) Pindahkan elemen maksimum: Setelah elemen maksimum ditemukan, pindahkan elemen tersebut ke posisi yang sesuai. Posisi yang sesuai bisa menjadi posisi terakhir dalam rentang data yang belum diurutkan atau posisi yang diinginkan sesuai dengan urutan pengurutan yang diinginkan.
- e) Ulangi iterasi: Setelah elemen maksimum dipindahkan, ulangi iterasi dengan rentang data yang belum diurutkan yang lebih kecil dari iterasi sebelumnya. Kembali ke langkah 3 dan lakukan langkah-langkah yang sama hingga seluruh data terurut.
- f) Selesai: Setelah semua iterasi selesai dan seluruh data terurut, proses sorting menggunakan metode Maximum Sort selesai.

Perlu diingat bahwa metode Maximum Sort merupakan metode pengurutan sederhana dan kurang efisien dibandingkan dengan metode sorting yang lebih canggih seperti Quicksort, Mergesort, atau Heapsort.

4. Jelaskan tahapan-tahapan Sorting menggunakan metode Minimum Sort!

Jawab :

Berikut adalah tahapan-tahapan dalam sorting menggunakan metode Minimum Sort:

- a) Inisialisasi: Mulailah dengan sekelompok data yang akan diurutkan. Misalnya, kita memiliki array dengan n elemen yang perlu diurutkan.
- b) Iterasi: Lakukan iterasi sebanyak $(n-1)$ kali, di mana n adalah jumlah elemen dalam data. Iterasi ini akan mengurangi jumlah elemen yang belum diurutkan setiap kali.
- c) Cari elemen minimum: Pada setiap iterasi, cari elemen minimum dalam rentang data yang belum diurutkan. Rentang data yang belum diurutkan akan berkurang setiap iterasi.
- d) Pindahkan elemen minimum: Setelah elemen minimum ditemukan, pindahkan elemen tersebut ke posisi yang sesuai. Posisi yang sesuai bisa menjadi posisi awal dalam rentang data yang belum diurutkan atau posisi yang diinginkan sesuai dengan urutan pengurutan yang diinginkan.
- e) Ulangi iterasi: Setelah elemen minimum dipindahkan, ulangi iterasi dengan rentang data yang belum diurutkan yang lebih kecil dari iterasi sebelumnya. Kembali ke langkah 3 dan lakukan langkah-langkah yang sama hingga seluruh data terurut.
- f) Selesai: Setelah semua iterasi selesai dan seluruh data terurut, proses sorting menggunakan metode Minimum Sort selesai.

Perlu diingat bahwa metode Minimum Sort merupakan metode pengurutan sederhana dan kurang efisien dibandingkan dengan metode sorting yang lebih canggih seperti Quicksort, Mergesort, atau Heapsort.

C. TUGAS PRAKTIKUM

- Lat8_1

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

int main()
{
    cout << "Nama \t: Nova Ardiansyah\n";
    cout << "NIM \t: 211011401309\n";
    cout << "=====\n\n";

    int Nilai[20];
    int i, j, N, l;
    int temp, U, lmaks;

    cout << "Masukkan banyaknya bilangan : ";
    cin >> N;

    for (i = 0; i < N; i++) {
        cout << "Masukkan nilai ke-" << i + 1 << " : ";
        cin >> Nilai[i];
    }

    // * Proses Cetak Sebelum Diurutkan
    cout << "\nData sebelum diurutkan : ";
    for (i = 0; i < N; i++) {
        cout << Nilai[i] << " ";
    }

    // * Proses Pengurutan
    U = N - 1;
    for (i = 0; i <= N - 2; i++) {
        lmaks = 0;

        for (j = 1; j <= U; j++) {
            if (Nilai[j] > Nilai[lmaks]) {
                lmaks = j;
            }
        }

        temp = Nilai[U];
        Nilai[U] = Nilai[lmaks];
        Nilai[lmaks] = temp;
        U--;
    }

    cout << "\nData setelah diurutkan : ";
    for (i = 0; i < N; i++) {
        cout << Nilai[i] << " ";
    }

    return 0;
}
```

```

Nama      : Nova Ardiansyah
NIM       : 211011401309
=====

Masukkan banyaknya bilangan : 4
Masukkan nilai ke-1 : 21
Masukkan nilai ke-2 : 13
Masukkan nilai ke-3 : 5
Masukkan nilai ke-4 : 10

Data sebelum diurutkan : 21 13 5 10
Data setelah diurutkan : 5 10 13 21

```

- Lat8_2

```

#include <iostream>
#include <string>
using namespace std;

int main()
{
    cout << "Nama \t: Nova Ardiansyah\n";
    cout << "NIM \t: 211011401309\n";
    cout << "=====\n\n";

    int Nilai[20];
    int i, j, N, l;
    int temp, U, lmaks;

    cout << "Masukkan banyaknya bilangan : ";
    cin >> N;

    for (i = 0; i < N; i++) {
        cout << "Masukkan nilai ke-" << i + 1 << " : ";
        cin >> Nilai[i];
    }

    // * Proses Cetak Sebelum Diurutkan
    cout << "\nData sebelum diurutkan : ";
    for (i = 0; i < N; i++) {
        cout << Nilai[i] << " ";
    }

    // * Proses Pengurutan
    U = N - 1;
    for (i = 0; i <= N - 2; i++) {
        lmaks = i;

        for (j = i + 1; j <= U; j++) {
            if (Nilai[j] > Nilai[lmaks]) {
                lmaks = j;
            }
        }

        temp = Nilai[i];
        Nilai[i] = Nilai[lmaks];
        Nilai[lmaks] = temp;
    }

    cout << "\nData setelah diurutkan : ";
    for (i = 0; i < N; i++) {
        cout << Nilai[i] << " ";
    }
}

```



```

}

return 0;
}

```

```

Nama      : Nova Ardiansyah
NIM       : 211011401309
=====

Masukkan banyaknya bilangan : 4
Masukkan nilai ke-1 : 21
Masukkan nilai ke-2 : 13
Masukkan nilai ke-3 : 5
Masukkan nilai ke-4 : 10

Data sebelum diurutkan : 21 13 5 10
Data setelah diurutkan : 21 13 10 5

```

- Lat8_3

```

#include <iostream>
#include <string>
using namespace std;

int main()
{
    cout << "Nama \t: Nova Ardiansyah\n";
    cout << "NIM \t: 211011401309\n";
    cout << "=====\n\n";

    int Nilai[20];
    int i, j, N, l;
    int temp, lmin;

    cout << "Masukkan banyaknya bilangan : ";
    cin >> N;

    for (i = 0; i < N; i++) {
        cout << "Masukkan nilai ke-" << i + 1 << " : ";
        cin >> Nilai[i];
    }

    // * Proses Cetak Sebelum Diurutkan
    cout << "\nData sebelum diurutkan : ";
    for (i = 0; i < N; i++) {
        cout << Nilai[i] << " ";
    }

    // * Proses Pengurutan
    for (i = 0; i < N-1; i++) {
        lmin = i;

        for (j = i + 1; j < N; j++) {
            if (Nilai[j] < Nilai[lmin]) {
                lmin=j;
            }
        }

        temp = Nilai[i];
        Nilai[i] = Nilai[lmin];
        Nilai[lmin] = temp;
    }

    cout << "\nData setelah diurutkan : ";

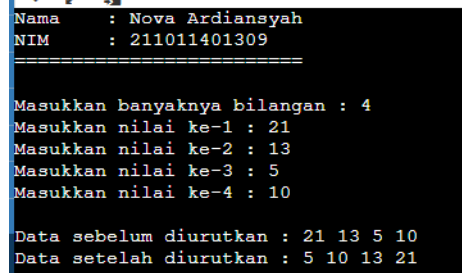
```

```

for (i = 0; i < N; i++) {
    cout << Nilai[i] << " ";
}

return 0;
}

```



```

Nama      : Nova Ardiansyah
NIM       : 211011401309
=====

Masukkan banyaknya bilangan : 4
Masukkan nilai ke-1 : 21
Masukkan nilai ke-2 : 13
Masukkan nilai ke-3 : 5
Masukkan nilai ke-4 : 10

Data sebelum diurutkan : 21 13 5 10
Data setelah diurutkan : 5 10 13 21

```

- Lat8_4

```

#include <iostream>
#include <string>
using namespace std;

int main()
{
    cout << "Nama \t: Nova Ardiansyah\n";
    cout << "NIM \t: 211011401309\n";
    cout << "=====\n\n";

    int Nilai[20];
    int i, j, N, l;
    int temp, lmin;

    cout << "Masukkan banyaknya bilangan : ";
    cin >> N;

    for (i = 0; i < N; i++) {
        cout << "Masukkan nilai ke-" << i + 1 << " : ";
        cin >> Nilai[i];
    }

    // * Proses Cetak Sebelum Diurutkan
    cout << "\nData sebelum diurutkan : ";
    for (i = 0; i < N; i++) {
        cout << Nilai[i] << " ";
    }

    // * Proses Pengurutan
    for (i = 0; i < N-1; i++) {
        lmin = i;

        for (j = i + 1; j < N; j++) {
            if (Nilai[j] < Nilai[lmin]) {
                lmin = j;
            }
        }

        temp = Nilai[i];
        Nilai[i] = Nilai[lmin];
        Nilai[lmin] = temp;
    }
}

```

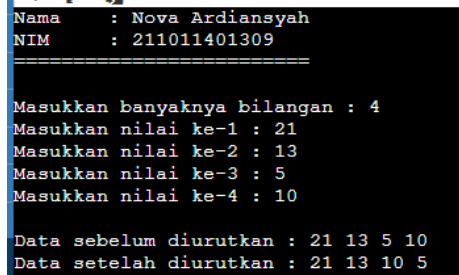
```

}

cout << "\nData setelah diurutkan : ";
for (i = 0; i < N; i++) {
    cout << Nilai[i] << " ";
}

return 0;
}

```



```

Nama      : Nova Ardiansyah
NIM       : 211011401309
=====

Masukkan banyaknya bilangan : 4
Masukkan nilai ke-1 : 21
Masukkan nilai ke-2 : 13
Masukkan nilai ke-3 : 5
Masukkan nilai ke-4 : 10

Data sebelum diurutkan : 21 13 5 10
Data setelah diurutkan : 21 13 10 5

```

- La8_5

```

#include <iostream>
#include <string>
using namespace std;

void maximumSort(bool ascending = true)
{
    int Nilai[20];
    int i, j, N, l;
    int temp, U, lmaks;

    cout << "Masukkan banyaknya bilangan : ";
    cin >> N;

    for (i = 0; i < N; i++) {
        cout << "Masukkan nilai ke-" << i + 1 << " : ";
        cin >> Nilai[i];
    }

    // * Proses Cetak Sebelum Diurutkan
    cout << "\nData sebelum diurutkan : ";
    for (i = 0; i < N; i++) {
        cout << Nilai[i] << " ";
    }

    // * Proses Pengurutan
    if (ascending == true) {
        U = N - 1;
        for (i = 0; i <= N - 2; i++) {
            lmaks = 0;

            for (j = 1; j <= U; j++) {
                if (Nilai[j] > Nilai[lmaks]) {
                    lmaks = j;
                }
            }

            temp = Nilai[U];

```

```

        Nilai[U] = Nilai[lmaks];
        Nilai[lmaks] = temp;
        U--;
    }
} else {
    U = N - 1;
    for (i = 0; i <= N - 2; i++) {
        lmaks = 0;

        for (j = 1; j <= U; j++) {
            if (Nilai[j] < Nilai[lmaks]) {
                lmaks = j;
            }
        }

        temp = Nilai[U];
        Nilai[U] = Nilai[lmaks];
        Nilai[lmaks] = temp;
        U--;
    }
}

cout << "\nData setelah diurutkan : ";
for (i = 0; i < N; i++) {
    cout << Nilai[i] << " ";
}
}

void minimumSort(bool ascending = true)
{
    int Nilai[20];
    int i, j, N, l;
    int temp, lmin;

    cout << "Masukkan banyaknya bilangan : ";
    cin >> N;

    for (i = 0; i < N; i++) {
        cout << "Masukkan nilai ke-" << i + 1 << " : ";
        cin >> Nilai[i];
    }

    // * Proses Cetak Sebelum Diurutkan
    cout << "\nData sebelum diurutkan : ";
    for (i = 0; i < N; i++) {
        cout << Nilai[i] << " ";
    }

    // * Proses Pengurutan
    if (ascending == true) {
        for (i = 0; i < N-1; i++) {
            lmin = i;

            for (j = i + 1; j < N; j++) {
                if (Nilai[j] < Nilai[lmin]) {
                    lmin=j;
                }
            }

            temp = Nilai[i];
            Nilai[i] = Nilai[lmin];
            Nilai[lmin] = temp;
        }
    }
}

```

```

    } else {
        for (i = 0; i < N-1; i++) {
            lmin = i;

            for (j = i + 1; j < N; j++) {
                if (Nilai[j] > Nilai[lmin]) {
                    lmin=j;
                }
            }

            temp = Nilai[i];
            Nilai[i] = Nilai[lmin];
            Nilai[lmin] = temp;
        }
    }

    cout << "\nData setelah diurutkan : ";
    for (i = 0; i < N; i++) {
        cout << Nilai[i] << " ";
    }
}

int main()
{
    cout << "Nama \t: Nova Ardiansyah\n";
    cout << "NIM \t: 211011401309\n";
    cout << "=====\n\n";

    int pilihan = 1;

    string metode[4] = {
        "Maksimum Sort: Menaik",
        "Maksimum Sort: Menurun",
        "Minimum Sort: Menaik",
        "Minimum Sort: Menurun"
    };

    for (int i = 0; i < 4; i++) {
        cout << i + 1 << ". " << metode[i] << endl;
    }

    cout << "\nPilih metode pengurutan: ";
    cin >> pilihan;

    cout << "=====\n\n";

    if (pilihan == 1) {
        maximumSort(true);
    } else if (pilihan == 2) {
        maximumSort(false);
    } else if (pilihan == 3) {
        minimumSort(true);
    } else if (pilihan == 4) {
        minimumSort(false);
    } else {
        cout << "Pilihan tidak tersedia!";
    }

    return 0;
}

```

```
Nama      : Nova Ardiansyah  
NIM       : 211011401309  
=====
```

1. Maksimum Sort: Menaik
2. Maksimum Sort: Menurun
3. Minimum Sort: Menaik
4. Minimum Sort: Menurun

```
Pilih metode pengurutan: 5  
=====
```

```
Pilihan tidak tersedia!
```

LAPORAN PRAKTIKUM STRUKTUR DATA

LAPORAN AKHIR PRAKTIKUM

Pertemuan ke-08

Sorting Lanjut-1



Disusun Oleh:

Nama Lengkap : Nova Ardiansyah

NIM : 211011401309

Kelas : 04-TPLE008

TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PAMULANG

Jl. Raya Puspitek No. 11 Buaran, Serpong Telp. (021) 7412566, Fax. (021) 7412566
Tangerang Selatan - Banten

A. TUGAS AKHIR

1. Buatlah tambahan program menggunakan sistem menu pada program yang telah dipraktekkan!

Jawab :

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

void bubbleSort(std::vector<int>& arr) {
    int n = arr.size();
    bool swapped;

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

        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] % 2 == 0 && arr[j + 1] % 2 != 0) {
                std::swap(arr[j], arr[j + 1]);
                swapped = true;
            }
        }

        if (!swapped) {
            break;
        }
    }
}

int main()
{
    cout << "Nama \t: Nova Ardiansyah\n";
    cout << "NIM \t: 211011401309\n";
    cout << "=====\n\n";

    int pilihan = 1;

    string metode[5] = {
        "Maksimum Sort: Menaik",
        "Maksimum Sort: Menurun",
        "Minimum Sort: Menaik",
        "Minimum Sort: Menurun",
        "Ganjil ke genap: Menaik (Tambahan)"
    };

    for (int i = 0; i < 5; i++) {
        cout << i + 1 << ". " << metode[i] << endl;
    }

    cout << "\nPilih metode pengurutan: ";
    cin >> pilihan;

    cout << "=====\n\n";

    if (pilihan == 1) {
        cout << "Pilihan tidak tersedia!";
    }
}
```



```

    } else if (pilihan == 2) {
        cout << "Pilihan tidak tersedia!";
    } else if (pilihan == 3) {
        cout << "Pilihan tidak tersedia!";
    } else if (pilihan == 4) {
        cout << "Pilihan tidak tersedia!";
    } else if (pilihan == 5) {
        vector<int> numbers = {5, 2, 7, 10, 8, 3, 6, 1, 4, 9};

        // * Data sebelum diuruskan
        cout << "Data sebelum diurutkan : ";
        for (int i = 0; i < numbers.size(); i++) {
            cout << numbers[i] << " ";
        }

        bubbleSort(numbers);

        // * Data setelah diurutkan
        cout << "\nData setelah diurutkan : ";

        for (int number : numbers) {
            cout << number << " ";
        }
    } else {
        cout << "Pilihan tidak tersedia!";
    }

    return 0;
}

```

```

Nama      : Nova Ardiansyah
NIM       : 211011401309
=====

1. Maksimum Sort: Menaik
2. Maksimum Sort: Menurun
3. Minimum Sort: Menaik
4. Minimum Sort: Menurun
5. Ganjil ke genap: Menaik (Tambahan)

Pilih metode pengurutan: 5
=====

Data sebelum diurutkan : 5 2 7 10 8 3 6 1 4 9
Data setelah diurutkan : 5 7 3 1 9 2 10 8 6 4

```

2. KESIMPULAN

Metode sorting Maksimum dan Minimum memiliki beberapa persamaan dan perbedaan. Kedua metode ini merupakan metode pengurutan sederhana yang melibatkan pencarian elemen maksimum atau minimum dalam setiap iterasi. Mereka mengurutkan data dengan menggeser elemen yang ditemukan ke posisi yang sesuai. Namun, perbedaan utama terletak pada cara pencarian elemen yang akan dipindahkan. Metode Maksimum Sort mencari elemen maksimum, sementara Metode Minimum Sort mencari elemen minimum.

Keduanya memiliki kekurangan yang serupa, yaitu kinerja yang buruk pada data yang besar dan ketidakstabilan. Keduanya memiliki kompleksitas waktu yang tinggi karena membutuhkan pencarian linier dalam setiap iterasi. Oleh karena itu, pada data yang besar, metode ini akan bekerja lebih lambat dibandingkan dengan metode sorting lain yang lebih efisien. Selain itu, keduanya cenderung tidak menjaga urutan relatif antara elemen-elemen dengan nilai yang sama, yang dapat menyebabkan perubahan posisi relatif elemen-elemen tersebut setelah proses pengurutan.

Dalam prakteknya, metode Maksimum dan Minimum Sort umumnya digunakan untuk tujuan pendidikan atau dalam kasus sederhana. Untuk pengurutan data yang efisien dan stabil, disarankan menggunakan metode sorting yang lebih canggih seperti Quicksort, Mergesort, atau Heapsort. Metode tersebut telah dioptimalkan dan memiliki kinerja yang lebih baik dalam pengurutan data yang lebih besar dan kompleks.