TUGAS MANDIRI

PERANCANGAN & ANALISIS ALGORITMA

“MergeSort”

202323430048

DOSEN PENGAMPU:

Randi Proska Sandra, S.Pd, M.Sc

OLEH:

Ahmad Al-Hafidz Ramadhan

22343034

Informatika (NK)

PRODI SARJANA INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2024

A. Penjelasan Singkat MergeSort

Merge Sort adalah algoritma yang berdasarkan strategi divide-and-conquer. Strateginya adalah dengan membagi sekelompok data yang akan diurutkan menjadi beberapa kelompok kecil terdiri dari maksimal dua nilai untuk dibandingkan dan digabungkan lagi secara keseluruhan.[1]

1. Divide
   Memilah elemen-elemen dari rangkaian data menjadi dua bagian dan mengulangi pemilahan hingga satu elemen terdiri maksimal dua nilai
2. Conquer
   Mengurutkan masing-masing elemen
3. Combine.
   Menggabungkan dua bagian tersebut secara rekursif untuk mendapatkan rangkaian data berurutan. Proses rekursi berhenti jika mencapai elemen dasar.Hal ini terjadi bilamana bagian yang akan diurutkan menyisakan tepat satu elemen. Sisa pengurutan satu elemen tersebut menandakan bahwa bagian tersebut telah terurut sesuai rangkaian.

Algoritma pengurutan data merge sort dilakukan dengan menggunakan cara divide and conquer yaitu dengan memecah kemudian menyelesaikan setiap bagian kemudian menggabungkannya kembali. Pertama data dipecah menjadi 2 bagian dimana bagian pertama merupakan setengah (jika data genap) atau setengah minus satu (jika data ganjil) dari seluruh data, kemudian dilakukan pemecahan kembali untuk masing - masing blok sampai hanya terdiri dari satu data tiap blok.

Setelah itu digabungkan kembali dengan membandingkan pada blok yang sama apakah data pertama lebih besar daripada data ke - tengah+1, jika ya maka data ke - tengah+1 dipindah sebagai data pertama, kemudian data ke - pertama sampai ke - tengah digeser menjadi data ke - dua sampai ke - tengah+1, demikian seterusnya sampai menjadi satu blok utuh seperti awalnya. Sehingga metode merge sort merupakan metode yang membutuhkan fungsi rekursi untuk penyelesaiannya.[2]

B. PSEUDOCODE

Pseudocode untuk mergesort[3] :

```
Pseudocode untuk merge sort adalah sebagai berikut:
mergesort(data)
    if data memiliki setidaknya dua
    elemen
      mergesort (separuh kiri dari data);
      mergesort (separuh kanan dari data ;
    merge (kedua bagian ke dalam suatu urutan);
```

C. SOURCE CODE

Source Code Mergesort[3] :

```c
#include <stdio.h>

#define MAX_SIZE 50

int bilangan[MAX_SIZE];

void merge(int kiri, int tengah, int kanan) {
    int h, i, j, b[MAX_SIZE], k;
    h = kiri;
    i = kiri;
    j = tengah + 1;

    while ((h <= tengah) && (j <= kanan)) {
        if (bilangan[h] <= bilangan[j]) {
            b[i] = bilangan[h];
            h++;
        } else {
            b[i] = bilangan[j];
            j++;
        }
        i++;
    }

    if (h > tengah) {
        for (k = j; k <= kanan; k++) {
            b[i] = bilangan[k];
            i++;
        }
    } else {
        for (k = h; k <= tengah; k++) {
            b[i] = bilangan[k];
            i++;
        }
    }

    for (k = kiri; k <= kanan; k++)
        bilangan[k] = b[k];
}

void merge_sort(int kiri, int kanan) {
    int tengah;
    if (kiri < kanan) {
        tengah = (kiri + kanan) / 2;
```

```
        merge_sort(kiri, tengah);
        merge_sort(tengah + 1, kanan);
        merge(kiri, tengah, kanan);
    }
}

int main() {
    int n, i;
    printf("Masukkan jumlah bilangan: ");
    scanf("%d", &n);

    printf("Masukkan bilangan:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &bilangan[i]);
    }

    merge_sort(0, n - 1);

    printf("Bilangan setelah diurutkan:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", bilangan[i]);
    }
    printf("\n");

    return 0;
}
```

Hasil Output :

```
mohammedadri@Mohammeds-MacBook-Pro Semester 4 % cd "/Users/mohammedadri/Documents/Semester 4/Perancangan dan Analisis Data/" && gcc Mergesort.c -o M
ments/Semester 4/Perancangan dan Analisis Data/"Mergesort
Masukkan jumlah bilangan: 10
Masukkan bilangan:
2
3
56
123
7897
123
45342236
12313123
767
34
Bilangan setelah diurutkan:
2 3 34 56 123 123 767 7897 12313123 45342236
mohammedadri@Mohammeds-MacBook-Pro Perancangan dan Analisis Data %
```

D. ANALISIS KEBUTUHAN WAKTU

Untuk menganalisis kebutuhan waktu algoritma Merge Sort yang diberikan, kita akan menggunakan tiga pendekatan yang diminta:

1. Analisis Menyeluruh:

- Algoritma Merge Sort memiliki kompleksitas waktu O(n log n) untuk kasus terbaik, terburuk, dan rata-rata. Ini karena algoritma ini menggunakan pendekatan divide-and-conquer, yang membagi array menjadi dua bagian sama besar, mengurutkan masing-masing bagian secara rekursif, dan kemudian menggabungkan kedua bagian tersebut.

- Dalam setiap pemanggilan fungsi `merge_sort`, algoritma ini melakukan operasi aritmatika (penjumlahan dan pembagian) untuk menentukan indeks tengah.

- Fungsi `merge` melakukan operasi penugasan dan perbandingan untuk menggabungkan dua bagian yang telah diurutkan. Jumlah operasi ini tergantung pada jumlah elemen yang diurutkan.

2. Analisis Berdasarkan Jumlah Operasi Abstrak

- Dalam algoritma Merge Sort, operasi abstrak utama adalah perbandingan dan penugasan. Setiap pemanggilan fungsi `merge` melakukan perbandingan antara elemen dari dua bagian yang diurutkan dan melakukan penugasan untuk menggabungkan elemen-elemen tersebut.

- Jumlah operasi abstrak ini adalah proporsional terhadap jumlah elemen yang diurutkan, dengan kompleksitas waktu O(n log n).

3. Analisis Menggunakan Pendekatan Best-case, Worst-case, dan Average-case
    - Best-case
    - Kasus terbaik untuk Merge Sort adalah ketika array sudah diurutkan sejak awal. Namun, karena algoritma ini tidak memeriksa apakah array sudah diurutkan atau tidak, kompleksitas waktunya tetap O(n log n).

    - Worst-case

      Kasus terburuk adalah ketika array diurutkan secara terbalik. Dalam kasus ini, algoritma ini memiliki kompleksitas waktu O(n log n), karena setiap pemanggilan fungsi `merge_sort` akan membagi array menjadi dua bagian sama besar dan mengurutkan masing-masing bagian secara rekursif.

    - Average-case

Kasus rata-rata juga memiliki kompleksitas waktu O(n log n), karena algoritma ini tidak bergantung pada distribusi data input.

Dalam kesimpulan, algoritma Merge Sort memiliki kompleksitas waktu O(n log n) untuk semua kasus (best-case, worst-case, dan average-case), yang menunjukkan bahwa algoritma ini efisien dalam hal waktu eksekusi, terutama untuk data besar.

Dari analisis di atas, kita dapat menyimpulkan bahwa algoritma Merge Sort memiliki kompleksitas waktu O(n log n) dalam kasus terburuk dan average-case, serta O(1) dalam kasus terbaik. Ini menunjukkan bahwa algoritma ini efisien dalam hal waktu eksekusi, terutama untuk kasus terburuk dan average-case, meskipun tidak secepat algoritma seperti Quick Sort dalam kasus terbaik.

E. REFERENSI

[1] "Introduction to the Design and Analysis of Algorithms (3rd ed.) [Levitin 2011-10-09]".

[2] "UNIVERSITAS BUDI LUHUR FAKULTAS TEKNOLOGI INFORMASI."

[3] A. Hendra Saptadi and D. Windi Sari, "ANALISIS ALGORITMA INSERTION SORT, MERGE SORT DAN IMPLEMENTASINYA DALAM BAHASA PEMROGRAMAN C++," 2012.

F. LAMPIRAN LINK GITHUB

https://github.com/SixYellowXD/MergeSort