

Analisis Perbandingan Kompleksitas Algoritma Iteratif dan Rekursif dalam Pencarian Skor Tertinggi pada Data Ujian Tulis Berbasis Komputer (UTBK)

Fathan Firdaus Nuzulan - 103012400353, Rafa Ahmad Aulia - 103012400169,
Tsalitsa Khansa Aziza- 103012430131

Program Studi S1 Informatika, Fakultas Informatika, Telkom University, Jl. Telekomunikasi 1 Bandung 40257, Indonesia

Abstrak— Ujian Tulis Berbasis Komputer (UTBK) menghasilkan data nilai peserta dalam jumlah besar yang perlu diolah secara efisien. Salah satu proses dasar dalam pengolahan data tersebut adalah pencarian nilai maksimum. Pencarian nilai maksimum dapat dilakukan dengan berbagai pendekatan, di antaranya menggunakan algoritma iteratif dan algoritma rekursif. Penelitian ini akan membahas perbandingan kompleksitas kedua pendekatan tersebut dalam pencarian skor tertinggi pada data UTBK. Analisis dilakukan dengan menghitung kompleksitas waktu dan ruang dari masing-masing algoritma, serta melihat pengaruh ukuran data terhadap kinerja algoritma.

Kata Kunci— Algoritma iteratif, Algoritma rekursif, Kompleksitas algoritma, Pencarian nilai maksimum

I. PENDAHULUAN

Ujian Tulis Berbasis Komputer (UTBK) menghasilkan data skor peserta dalam jumlah yang sangat besar, sehingga memerlukan metode pengolahan yang efisien. Salah satu operasi dasar yang krusial adalah pencarian nilai tertinggi, yang menjadi fondasi dalam analisis peringkat dan evaluasi hasil ujian. Dalam konteks ini, algoritma pencarian nilai maksimum dapat diimplementasikan melalui dua pendekatan utama, yaitu iteratif dan rekursif. Perbandingan kinerja kedua pendekatan tersebut perlu dianalisis untuk menentukan solusi yang paling optimal ketika menghadapi data berskala besar.

Penelitian ini bertujuan untuk menganalisis perbedaan kompleksitas algoritma iteratif dan rekursif dalam pencarian skor tertinggi pada data UTBK. Analisis dilakukan dengan mengukur kompleksitas waktu dan ruang secara teoritis serta menguji performa eksekusinya pada berbagai ukuran data. Dengan demikian, diharapkan dapat diketahui pendekatan mana yang lebih efisien dan sesuai untuk diterapkan dalam pengolahan data UTBK yang bersifat masif dan membutuhkan kecepatan pemrosesan yang tinggi.

II. LANDASAN TEORI

A. UJIAN TULIS BERBASIS KOMPUTER (UTBK)

Ujian Tulis Berbasis Komputer (UTBK) merupakan sistem ujian nasional yang diselenggarakan sebagai bagian dari seleksi masuk perguruan tinggi negeri di Indonesia. UTBK menghasilkan data nilai peserta dalam jumlah besar yang tersimpan secara digital. Data tersebut umumnya berbentuk nilai numerik yang mempresentasikan kemampuan peserta pada berbagai komponen ujian. Dalam kajian algoritma, data nilai UTBK dapat dimodelkan sebagai sekumpulan bilangan dalam struktur data array atau list.

B. KOMPLEKSITAS ALGORITMA

Kompleksitas algoritma adalah ukuran efisiensi dari suatu algoritma dalam hal jumlah komputasi yang diperlukan untuk menyelesaikan suatu masalah, seperti waktu eksekusi dan ruang memori yang digunakan dari besar data input n . Dengan kompleksitas algoritma, dapat ditentukan seberapa cepat atau seberapa hemat memori suatu algoritma saat ukuran data meningkat.

C. METODE ITERATIF

Metode iteratif adalah algoritma yang menggunakan struktur perulangan (seperti *for* atau *while*) untuk menyelesaikan suatu permasalahan. Pada pencarian nilai maksimum, algoritma iteratif bekerja dengan menyimpan satu nilai maksimum sementara, kemudian membandingkannya secara berurutan dengan setiap elemen dalam data.

D. METODE REKURSIF

Metode Rekursif merupakan algoritma yang menyelesaikan suatu permasalahan dengan memanggil fungsi itu sendiri sehingga mencapai kondisi dasar (base case). Pada pencarian nilai maksimum, algoritma rekursif bekerja dengan membagi data menjadi bagian-bagian lebih kecil dan membandingkan hasilnya secara berulang sampai setiap bagian hanya akan memiliki satu elemen dengan nilai maksimum.

E. KOMPLEKSITAS WAKTU

Kompleksitas waktu, sering dilambangkan dengan fungsi $T(n)$, adalah jumlah tahapan komputasi yang harus dijalankan oleh sebuah algoritma dengan ukuran sebesar n . Variabel n melambangkan jumlah data yang dimasukkan ke dalam suatu algoritma. Kompleksitas waktu bisa digunakan untuk melakukan perbandingan algoritma untuk menentukan mana yang lebih efisien. Dalam konteks membandingkan algoritma, algoritma yang memiliki kompleksitas lebih kecil adalah algoritma yang lebih efisien. Kompleksitas waktu dinyatakan menggunakan notasi *Big-O*, yang menggambarkan laju pertumbuhan kebutuhan sumber daya ketika ukuran data semakin besar.

Notasi ini menunjukkan seberapa cepat kompleksitas waktu bertambah ketika ukuran masukan (n) membesar. Jika kompleksitas waktu suatu algoritma dinyatakan dengan $T(n)$ dan terdapat konstanta C yang memenuhi

$$T(n) \leq C f(n)$$

untuk $n \geq n_0$, maka

$$T(n) = O(f(n))$$

III. STUDI KASUS

A. PERMASALAHAN

Data nilai Ujian Tulis Berbasis Komputer (UTBK) disediakan dalam bentuk kumpulan data numerik dengan ukuran data yang besar. Salah satu permasalahan dasar dalam proses pengolahan data tersebut adalah pencarian skor tertinggi. Proses pencarian skor tertinggi dapat dilakukan dengan berbagai metode algoritma, di antaranya adalah algoritma iteratif dan algoritma rekursif. Permasalahan dalam studi kasus ini akan difokuskan pada penerapan kedua metode tersebut dalam pencarian skor tertinggi dari data nilai UTBK.

B. TUJUAN

Tujuan dari studi kasus ini adalah untuk mengetahui perbandingan performa dari algoritma iteratif dan algoritma rekursif dalam pencarian skor tertinggi pada data nilai UTBK, hal tersebut akan dilihat berdasarkan kompleksitas waktu dan grafik pertumbuhan (*Order of Growth*) seiring dengan bertambahnya jumlah data yang digunakan.

C. STUDI KASUS.

Pada studi kasus ini, digunakan sekumpulan data nilai Ujian Tulis Berbasis Komputer (UTBK) yang direpresentasikan dalam bentuk array. Data nilai tersebut tidak berasal dari data asli, melainkan dihasilkan secara acak (*random*) untuk keperluan dalam simulasi dan analisis algoritma. Proses tersebut dilakukan menggunakan fungsi *randint* dari library *random* pada bahasa pemrograman python, dengan rentang nilai antara 0 sampai 1000. Kedua algoritma akan diterapkan pada data yang sama untuk mencari skor tertinggi. Hasil eksekusi akan dianalisis dan dibandingkan untuk mengetahui perbedaan performa antara algoritma iteratif dan algoritma rekursif seiring dengan bertambahnya jumlah data yang digunakan.

IV. DESKRIPSI ALGORITMA

A. ALGORITMA ITERATIF

a. Konsep Iteratif

Algoritma iteratif mencari nilai maksimum dengan membandingkan setiap elemen array secara berurutan. Elemen pertama dijadikan nilai maksimum untuk sementara, kemudian setiap elemen berikutnya akan dibandingkan, jika ditemukan nilai yang lebih besar, maka nilai maksimum akan diperbaharui hingga seluruh data diperiksa.

b. Kode Algoritma Iteratif

```
def max_iteratif(A):
    maks = A[0]
    for i in range(1, len(A)):
        if A[i] > maks:
            maks = A[i]
    return maks
```

c. Kompleksitas waktu $T(n)$ (*Sum of 1*)

Jumlah data $\rightarrow n$

Operasi dasar $\rightarrow A[i] > \text{maks}$

$$T(n) = \sum_{i=1}^{n-1} 1 = n - 1 - 1 + 1 = n - 1$$

$$T(n) = n - 1 \in O(n)$$

B. ALGORITMA REKURSIF

a. Konsep Rekursif

Algoritma rekursif mencari nilai maksimum dengan membagi data menjadi dua bagian terlebih dahulu. Proses ini dilakukan secara berulang sampai setiap bagian hanya akan memiliki satu elemen dengan nilai maksimum. Nilai maksimum di bagian kiri akan dibandingkan dengan nilai maksimum di bagian kanan, dan nilai maksimum akan dikembalikan sebagai hasil.

b. Kode Algoritma Rekursif

```
def max_rekursif(A, kiri, kanan):
    if kiri == kanan:
        return A[kiri]

    mid = (kiri + kanan) // 2

    max_kiri = max_rekursif(A, kiri, mid)
    max_kanan = max_rekursif(A, mid + 1, kanan)

    if max_kiri > max_kanan:
        return max_kiri
    else:
        return max_kanan
```

c. Kompleksitas waktu $T(n)$ (Peubah variabel)

Operasi Dasar $\rightarrow \text{max_kiri} > \text{max_kanan}$

Persaman Rekurensi:

$$T(n) = \begin{cases} 0, & \text{jika } n = 1 \\ 2T\left(\frac{n}{2}\right) + 1, & \text{jika } n > 1 \end{cases}$$

semisalkan $n = 2^k$

$$T(n) = 2T\left(\frac{n}{2}\right) + 1 \rightarrow T(2^k) = 2T\left(\frac{2^k}{2}\right) + 1$$

$$T(2^k) = 2T(2^{k-1}) + 1$$

Substitusi t_k untuk $T(2^k)$

$$T(2^k) = 2T(2^{k-1}) + 1 \rightarrow t_k = 2t_{k-1} + 1$$

Menentukan persamaan karakteristik

$$t_k - 2t_{k-1} = 1 \rightarrow (r - 2)(r - 1) = 0$$

Mencari nilai akar

$$(r - 2)(r - 1) = 0 \rightarrow r_1 = 2, r_2 = 1$$

Solusi umum

$$t_k = c_1 2^k + c_2 2^k = c_1 2^k + c_2$$

Ubah 2^k ke n

$$T(2^k) = c_1 2^k + c_2 \rightarrow T(n) = c_1 n + c_2$$

Cari c_1 dan c_2

$$T(n) = c_1 n + c_2$$

$$T(1) = 0$$

$$T(2) = 2T\left(\frac{2}{2}\right) + 1 = 2T(1) + 1 = 0 + 1 = 1$$

$$T(2) = 1$$

$$T(1) = c_1 \cdot 1 + c_2 = c_1 + c_2 = 0$$

$$T(2) = c_1 \cdot 2 + c_2 = 2c_1 + c_2 = 1$$

$$2c_1 + c_2 = 1$$

$$c_1 + c_2 = 0$$

$$\begin{array}{r} c_1 + c_2 = 0 \\ - \\ c_1 = 1 \end{array}$$

$$c_1 + c_2 = 0 \rightarrow c_2 = -c_1 = -1 \rightarrow c_2 = -1$$

$$\text{Didapatkan } c_1 = 1 \text{ dan } c_2 = -1$$

Kesimpulan

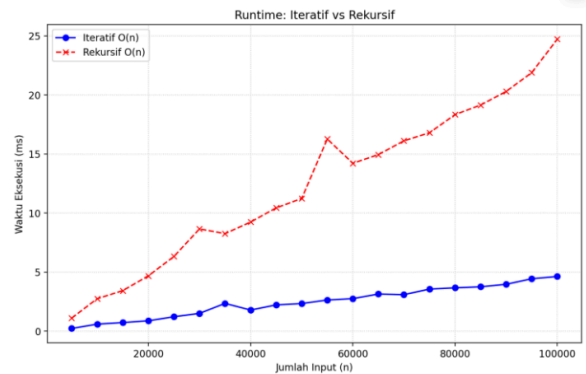
$$T(n) = c_1 n + c_2$$

$$c_1 = 1, c_2 = -1$$

$$T(n) = n - 1$$

$$T(n) = n - 1 \in \Theta(n)$$

C. GRAFIK ORDER OF GROWTH



Berdasarkan gambar diatas, terlihat bahwa waktu eksekusi kedua algoritma tersebut meningkat seiring dengan bertambahnya jumlah data. Algoritma iteratif secara konsisten memiliki waktu eksekusi yang lebih rendah dibandingkan algoritma rekursif pada seluruh ukuran data. Perbedaan ini disebabkan adanya *overhead* pemanggilan fungsi dan penggunaan *stack* rekursi pada algoritma rekursif, yang menyebabkan waktu eksekusi nya lebih besar.

V. KESIMPULAN

Berdasarkan analisis kompleksitas waktu yang dilakukan, baik algoritma iteratif maupun rekursif untuk pencarian nilai maksimum pada data UTBK memiliki kompleksitas waktu yang sama secara asimtotik, yaitu $O(n)$ atau $\Theta(n)$. Hal ini menunjukkan bahwa kedua algoritma memiliki pertumbuhan waktu eksekusi yang linear terhadap jumlah data. Namun, dari hasil simulasi dan pengamatan grafik Order of Growth, algoritma iteratif menunjukkan performa yang lebih baik secara praktis dibandingkan algoritma rekursif. Hal ini disebabkan oleh overhead pemanggilan fungsi berulang dan penggunaan stack memori pada rekursi, yang meningkatkan waktu eksekusi dan konsumsi ruang, terutama pada data yang berukuran besar.

Oleh karena itu, dalam konteks pengolahan data UTBK yang berskala besar dan membutuhkan efisiensi tinggi, algoritma iteratif lebih direkomendasikan untuk digunakan. Algoritma iteratif tidak hanya lebih sederhana dalam implementasi, tetapi juga lebih hemat dalam penggunaan memori dan waktu eksekusi. Meskipun secara teoritis kompleksitas kedua algoritma setara, pertimbangan praktis seperti overhead rekursif dan keterbatasan stack membuat pendekatan iteratif menjadi pilihan yang lebih optimal untuk pencarian nilai maksimum pada data ujian dalam volume yang masif.

VI. REFERENSI

Source Code Algoritma:

<https://github.com/kapalaut/TugasBesarAKA.git>

Febriansyah, M. F., Rhamadani, M., & Sutabri, T. (2025). PERBANDINGAN PEMANFAATAN ALGORITMA REKURSIF DAN ITERATIF DALAM PENYELESAIAN STRUKTUR DATA POHON. *Jurnal Manajemen Informatika & Teknologi*, 5(1), 46-56.
<https://journal.stiestekom.ac.id/index.php/mifortekh/article/view/742/558>

BPMPP Universitas Majalengka. (2024). *Analisis kompleksitas algoritma: Teori dan penerapannya dalam pemecahan masalah*. <https://bpmpp.uma.ac.id/2024/06/26/analisis-kompleksitas-algoritma-teori-dan-penerapannya-dalam-pemecahan-masalah/>