

ANALISIS PERBANDINGAN KOMPLEKSITAS ALGORITMA ITERATIF DAN REKURSIF DALAM PENCARIAN SKOR TERTINGGI PADA DATA UJIAN TULIS BERBASIS KOMPUTER (UTBK)

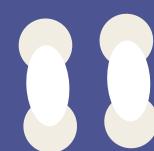
ANGGOTA KELOMPOK

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



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.



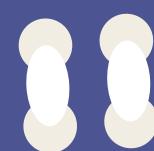
TUJUAN

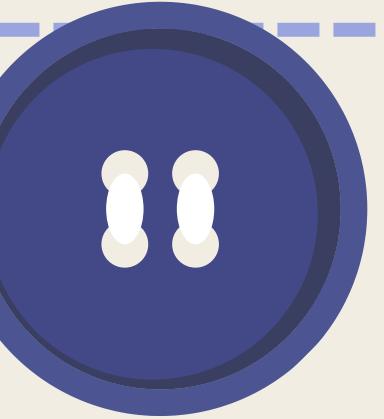
Tujuan dari studi kasus ini adalah untuk mengetahui perbandingan performa dari algoritma iteratif dan algoritma rekursif dalam pencarian skor tertinggi pada data nilai UTBK, hal tersebut akan dilihat berdasarkan kompleksitas waktu dan grafik pertumbuhan (Order of Growth) seiring dengan bertambahnya jumlah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



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 interatif dan algoritma rekursif seiring dengan bertambahnya jumlah data yang digunakan





DESKRIPSI ALGORITMA

ALGORITMA 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.

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

Jumlah data → n
Operasi dasar → $A[i] > \text{maks}$

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

ALGORITMA REKURSIF

Algoritma rekursif mencari nilai maksimum dengan membagi data menjadi dua bagian terlebih dahulu. Proses ini dilakukan secara berulangan 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.

```
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
```

ALGORITMA REKURSIF

Operasi Dasar → max_kiri > max_kanan
Persamaan Rekurensi:

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

semisalkan $n = 2^k$

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + 1 \rightarrow T(2^k) = 2T\left(\frac{2^k}{2}\right) + 1 \\ T(2^k) &= 2T(2^{k-1}) + 1 \end{aligned}$$

Subtitusi t_k untuk $T(2^k)$

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

Menentukan persamaan karakteristik

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

Mencari nilai akar

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

Solusi umum

$$t_k = c_1 2^k + c_2 1^k = c_1 2^k + c_2$$

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

$$\begin{aligned} T(n) &= c_1 n + c_2 \\ T(1) &= 0 \\ T(2) &= 2T\left(\frac{2}{2}\right) + 1 = 2T(1) + 1 = 0 + 1 = 1 \\ T(2) &= 1 \end{aligned}$$

$$\begin{aligned} T(1) &= c_1 \cdot 1 + c_2 = c_1 + c_2 = 0 \\ T(2) &= c_1 \cdot 2 + c_2 = 2c_1 + c_2 = 1 \\ 2c_1 + c_2 &= 1 \\ c_1 + c_2 &= 0 \\ \hline c_1 &= 1 \\ c_1 + c_2 &= 0 \rightarrow c_2 = -c_1 = 1 \rightarrow c_2 = -1 \end{aligned}$$

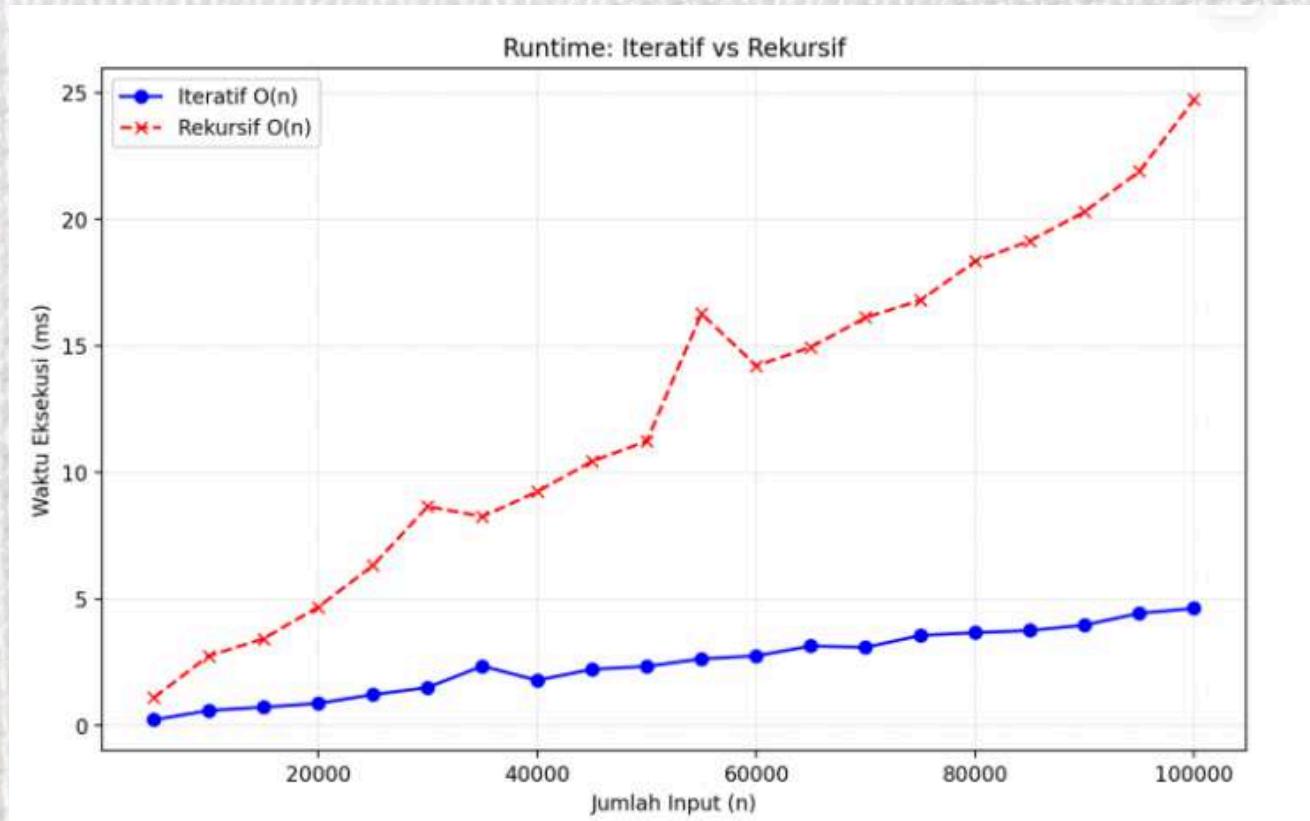
Didapatkan $c_1 = 1$ dan $c_2 = -1$

Kesimpulan

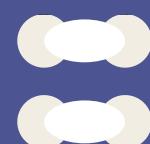
$$\begin{aligned} T(n) &= c_1 n + c_2 \\ c_1 &= 1, c_2 = -1 \\ T(n) &= n - 1 \\ T(n) &= n - 1 \in \Theta(n) \end{aligned}$$



GRAFIK ORDER OF GROWTH

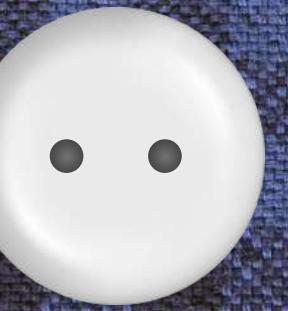


Berdasarkan gambar disamping, terlihat bahwa waktu eksekusi kedua algoritma tersebut meningkat seiring dengan bertambahnya jumlah data. Algoritma iteratif secara konsisten memiliki waktu eksekusi yang lebih rendah dibandingkan algoritma rekursif pada seluruh ukuran data. Perbedaan ini disebabkan adanya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



KESIMPULAN

Berdasarkan analisis kompleksitas waktu secara teoritis, baik algoritma iteratif maupun rekursif dalam pencarian nilai maksimum menunjukkan kompleksitas yang sama, yaitu linear $O(n)$. Namun, evaluasi praktis melalui simulasi data UTBK mengungkapkan bahwa algoritma iteratif secara konsisten memiliki waktu eksekusi yang lebih rendah dibandingkan rekursif. Perbedaan ini terutama disebabkan oleh adanya overhead pemanggilan fungsi berulang dan alokasi stack memori pada pendekatan rekursif, yang meningkatkan beban komputasi terutama pada data berukuran besar. Oleh karena itu, dalam konteks pengolahan data UTBK yang memerlukan efisiensi tinggi dan skala masif, algoritma iteratif lebih direkomendasikan karena kombinasi kemudahan implementasi, kecepatan eksekusi, serta optimasi penggunaan sumber daya memori.



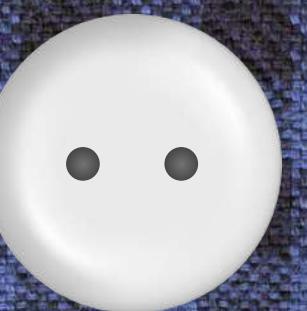
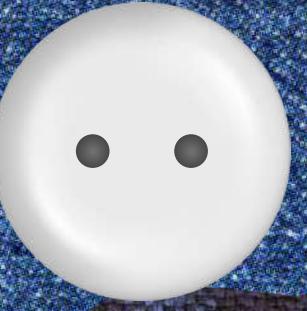
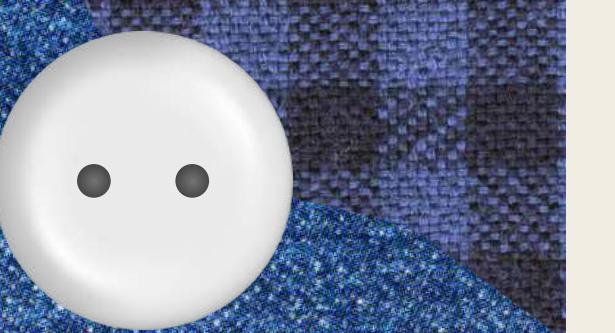
X

X

X

X





**TERIMA
KASIH**

