Tugas Besar Analisis Kompleksitas Algoritma

Perbandingan Algoritma Iteratif dan Rekursif Pada Reverse String

Telkom University

Disusun Oleh:

Muhammad Irgiansyah(103012300039)

Dandy Fadilla(103012300180)

IF-47-08

S1 Informatika

Analisis Kompleksitas Algoritma

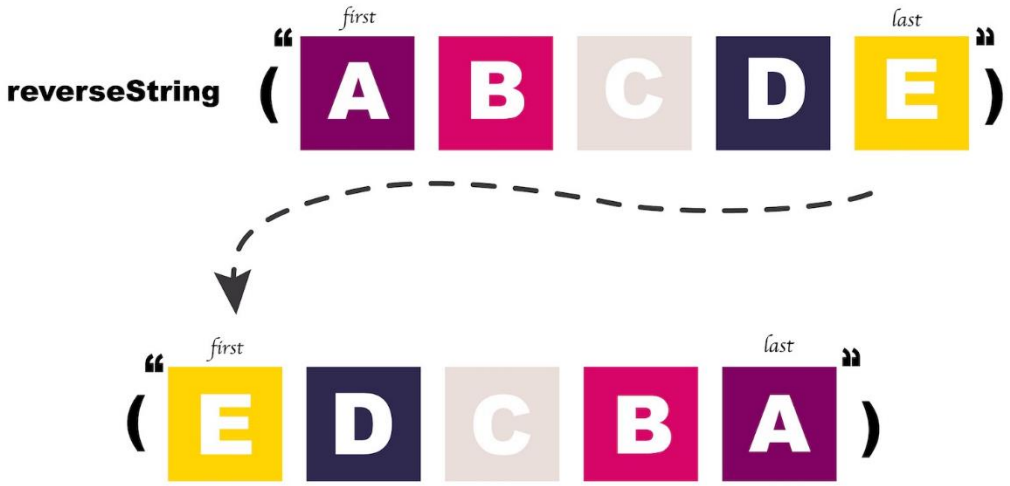
2024

A. Definisi Studi Kasus Algoritma Reverse String pada Iteratif dan Rekursif

Studi kasus yang diambil dalam tugas ini adalah proses membalikkan string. Reverse string merupakan salah satu permasalahan dasar dalam algoritma dan pemrograman yang sering digunakan untuk mempelajari berbagai pendekatan penyelesaian masalah. Dalam konteks ini, diberikan sebuah string input, dan tujuan utamanya adalah:

1. Mengembalikan string dengan karakter yang urutannya dibalik (dari belakang ke depan).
2. Membandingkan efisiensi antara dua pendekatan penyelesaian, yaitu algoritma iteratif dan rekursif, untuk menyelesaikan masalah ini.

Studi kasus ini dipilih karena dapat dengan jelas menggambarkan perbedaan antara pendekatan iteratif dan rekursif, baik dari segi kompleksitas waktu (time complexity) maupun penggunaan memori (space complexity). Selain itu, permasalahan ini cukup sederhana untuk memahami konsep dasar algoritma, namun tetap relevan untuk aplikasi yang lebih kompleks, seperti pemrosesan string dalam pengolahan data atau pengembangan perangkat lunak.



Sumber: https://algodaily.com/challenges/reverse-a-string

Reverse String adalah algoritma yang digunakan untuk membalikkan sebuah string atau kata di dalam bahasa pemrograman. Misalnya, string "hello" jika dibalik akan menjadi "olleh". Terdapat banyak cara untuk melakukan Reverse String, mulai dari menggunakan algoritma sendiri atau menggunakan method bawaan. Namun yang akan kita gunakan adalah dengan menggunakan algoritma secara iterasi dan juga rekursif.

B. Kode Algoritma Reverse String

Algoritma yang kami tampilkan berbentuk Pseudocode agar lebih mudah dipahami secara universal. Berikut adalah algoritma reverse string menggunakan iteratif dalam pseudocode

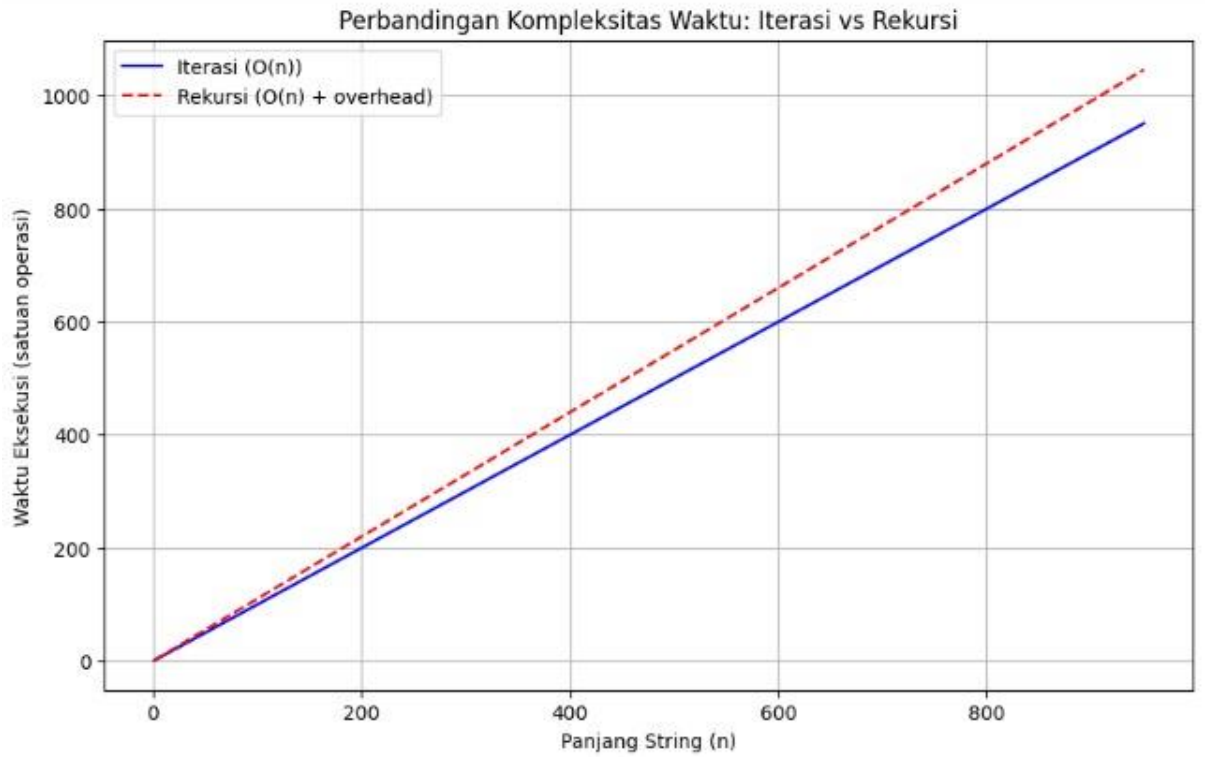
```
procedure reverseStringIterative(str : string)
kamus
   i : int
algoritma
   for i = length(str) downto 1 do
      output(str[i])
   endfor
endprocedure
```

Berikutnya adalah algoritma reverse string yang menggunakan rekursif dalam pseudocode

```
procedure reverseStringRecursive(str : string, idx : int)
kamus
algoritma
   if idx <= 0 then
      return
   else
      output (str[idx])
      reverseStringRecursive(str, idx - 1)
   endif
endprocedure
```

C. Grafik Kompleksitas Algoritma

Berdasarkan kedua algoritma baik secara iteratif, maupun rekursif, algoritma ini tidak memiliki *best-case* dan *worst-case,* karena dalam membalikan posisi srting / *reverse string,* lamanya kompleksitas waktunya tergantung kepada jumlah string yang di eksekusi. Berikut ilustrasi grafik untuk kompleksitas algoritma nya untuk algoritma *reverse string* menggunakan metode iterasi dan rekursif.



Dapat dilihat dari grafik tersebut, perbedaan waktu antara algoritma iterasi dan rekursif tidak terlalu signifikat perbedaannya, dan waktu eksekusinya cenderung menaik mengikuti jumlah string yang dieksekusi. Sehingga algoritma *reverse string* ini tidak memiliki *best-case* ataupun *worst-case*. Namu kompleksitas waktu eksekusinya bisa lebih rendah jika di support dengan device yang memadai untuk mengeksekusi algoritma *reverse string* baik secara iterasi maupun rekursif.

Pada algoritma iteratif dalam setiap iterasi, satu karakter diproses, dan total iterasi adalah n, di mana n adalah panjang string. Operasi di setiap iterasi adalah konstan (O(1)).

Kompleksitas Waktu:

O(n) ,Iterasi linearnya membuat algoritma ini efisien untuk string berukuran besar.

Pada algoritma rekursif setiap pemanggilan fungsi memproses satu karakter, dan ada n karakter di string. Operasi penggabungan string dalam Python (atau bahasa lain) memiliki kompleksitas waktu O(k), di mana k adalah panjang string yang digabung. Jika implementasi menggunakan pendekatan optimasi (seperti menyusun daftar dan menggabungkannya sekali di akhir), penggabungan string dapat dilakukan dalam waktu konstan, sehingga totalnya menjadi:

O(n)

Algoritma rekursif membutuhkan ruang tambahan untuk setiap pemanggilan fungsi, yaitu O(n) untuk tumpukan rekursi.

D. Kesimpulan

Kesimpulan yang dapat diambil adalah bahwa baik algoritma iteratif maupun rekursif memiliki kompleksitas waktu yang sama, yaitu O(n), karena keduanya melakukan iterasi atau pemanggilan fungsi sebanyak elemen dalam string (n). Namun, pendekatan iteratif

menunjukkan efisiensi yang lebih tinggi dalam hal penggunaan memori karena hanya memerlukan memori konstan (O(1)) tanpa adanya overhead pemanggilan fungsi yang berulang. Hal ini menjadikannya lebih cocok untuk kasus dengan input besar atau skenario yang membutuhkan performa tinggi, seperti aplikasi real-time atau pengolahan data skala besar. Sementara itu, algoritma rekursif menawarkan kelebihan dalam kesederhanaan penulisan kode dan lebih intuitif untuk memahami struktur masalah, terutama jika masalahnya memiliki sifat yang secara alami rekursif. Namun, pendekatan ini cenderung kurang efisien untuk input besar karena memerlukan memori tambahan untuk setiap pemanggilan fungsi, yang dapat menyebabkan stack overflow pada kondisi tertentu. Oleh karena itu, pendekatan iteratif lebih disarankan untuk skenario produksi, sedangkan pendekatan rekursif lebih cocok untuk pembelajaran, eksplorasi algoritma, atau kasus dengan input yang relatif kecil.

E. Referensi

https://algodaily.com/challenges/reverse-a-string
https://www.digitalocean.com/community/tutorials/reverse-string-c-plus-plus
https://medium.com/aufabillah/reverse-string-javascript-d51a6cc1a1cd
https://interviewing.io/questions/reverse-string