LAPORAN TUGAS BESAR MATA KULIAH ANALISIS KOMPLEKSITAS ALGORITMA

Analisis Kompleksitas Algoritma Dalam Perhitungan Jarak Tempuh Dengan Pola Pengisian Bahan Bakar

Kelompok:

1. Caesar Gian Indrarizky (103012300218)
2. Hanif Haidar Fathin Mumtaz (103012300072)

Program Studi S1-Informatika

Fakultas Informatika

Jl. Telekomunikasi 1, Dayeuhkolot Bandung

A. Studi kasus

Menghitung Jumlah Cara Truk Pengantar Barang Menempuh Jarak dengan Pola Pengisian Bahan Bakar

Dalam dunia logistik, efisiensi operasional menjadi salah satu faktor penting untuk memastikan pengiriman barang dapat dilakukan secara tepat waktu dan hemat biaya. Salah satu aspek yang menjadi perhatian adalah perencanaan perjalanan truk pengantar barang, khususnya dalam pengisian bahan bakar. Pada studi kasus ini, kita akan menganalisis jumlah cara yang memungkinkan sebuah truk menempuh jarak tertentu (n kilometer) dengan aturan pengisian bahan bakar pada jarak 2 km atau 3 km.

Studi ini penting karena dalam perencanaan rute logistik, memahami kemungkinan-kemungkinan ini dapat membantu pengelola armada memprediksi pola perjalanan truk secara lebih akurat.

Untuk menghitung jumlah cara truk mencapai jarak n:

1. Truk hanya dapat mengisi bahan bakar setiap 2 km atau 3 km.

2. Total jumlah cara mencapai jarak nnn dapat direpresentasikan dengan persamaan: $f(n) = f(n-2)+f(n-3)$ dengan $f(0) = 1$, $f(1) = 0$, $f(2) = 1$, dan seterusnya.

Pendekatan yang Digunakan

1. Metode Rekursif

    - Menggunakan sifat rekursif dari fungsi f(n) untuk menghitung jumlah cara.

    - Kompleksitas waktu metode ini akan bertumbuh eksponensial jika tidak menggunakan optimasi seperti memoization atau dynamic programming.

2. Metode Iteratif

    - Menggunakan loop untuk menghitung jumlah cara dari jarak 0 hingga nnn secara berurutan.

    - Metode ini memiliki kompleksitas waktu O(n), sehingga lebih efisien dibandingkan metode rekursif murni.

Eksperimen dilakukan dengan memodelkan perhitungan jumlah cara pada jarak $n=110$ kilometer dan membandingkan kinerja kedua metode. Hasil analisis akan memberikan gambaran tentang kelebihan dan kekurangan masing-masing metode dalam konteks penghitungan efisiensi logistik.

B. Fungsi iteratif

Algoritma untuk fungsi iteratif adalah sebagai berikut :

```
func countWaysIterative(n int) int64 {
        if n == 0 {
                return 1
        }
        if n < 0 {
                return 0
        }
        var dp [10000]int64
        dp[0] = 1
        for i := 1; i <= n; i++ {
                if i-2 >= 0 {
                        dp[i] += dp[i-2]
                }
                if i-3 >= 0 {
                        dp[i] += dp[i-3]
                }
        }
        return dp[n]
}
```

C. Fungsi rekursif

```
func  countWaysRecursive(n int) int {
        if n < 0 {
                return 0
        } else if n == 0 {
                return 1
        }
        return countWaysRecursive(n-2) + countWaysRecursive(n-3)
}
```

D. Mencari efektifitas waktu

a. Algoritma Iteratif

❖ Struktur Iteratif

Struktur iteratif adalah metode penyelesaian masalah yang menggunakan loop untuk menghitung hasil secara bertahap, tanpa memanggil fungsi berulang kali seperti dalam rekursi. Untuk kasus ini, kita menggunakan hubungan:

$$f(n)=f(n-2)+f(n-3)$$

Langkah Utama:

1. **Inisialisasi Kasus Dasar:**
   - Untuk n=0: f(0)=1 (hanya satu cara, yaitu tidak bergerak).
   - Untuk n=1: f(1)=1 (satu cara, yaitu 1 km).
   - Untuk n=2: f(2)=2 (dua cara: 1+1 atau 2 km).
2. Loop Iteratif:
   - Dimulai dari i=3 hingga i=110.
   - Gunakan dua variabel (prev3 untuk f(n−3) dan prev2 untuk f(n−2)) untuk menghitung nilai f(n).
3. Perhitungan:
   - Pada setiap iterasi, hitung current = prev2+prev3.
   - Perbarui prev3←prev2 dan prev2←current.
4. Hasil Akhir:
   - Setelah loop selesai, prev2 akan menyimpan nilai f(110).

❖ Pemanggilan Berulang

Pendekatan iteratif menggantikan pemanggilan fungsi rekursif dengan loop. Tidak ada tumpukan pemanggilan (call stack) seperti pada rekursif. Sebagai gantinya:

- Nilai sebelumnya (f(n−3) dan f(n−2)) disimpan dalam variabel dan diperbarui setiap iterasi.
- Hanya satu kali perhitungan dilakukan untuk setiap nilai i, sehingga mengurangi redundansi.

Contoh untuk n=110:

- Iterasi pertama (i=3):
  - current=f(1)+f(0)=1+1=2.
  - Perbarui: prev3=f(1)=1, prev2=f(2)=2.
- Iterasi kedua (i=4):
  - current=f(2)+f(1)=2+1=3.
  - Perbarui: prev3=f(2)=2, prev2=f(3)=3.
- Proses ini berlanjut hingga i=110.

❖ **Analisis Kompleksitas Waktu**

Kompleksitas waktu algoritma iteratif dihitung berdasarkan jumlah iterasi dalam loop:

Kompleksitas Waktu:

- **Loop Iteratif**:
  - Untuk n=110, loop berjalan dari i=3 hingga i=110, sehingga total iterasi adalah 110−3+1=108.
  - Setiap iterasi melakukan operasi aritmatika sederhana (prev2+prev3), yang membutuhkan waktu konstan (O(1)).
- **Total Kompleksitas**:
  - Kompleksitas waktu adalah O(n), di mana n=110.

Efisiensi:

- Algoritma iteratif jauh lebih efisien dibandingkan rekursi karena menghindari pemanggilan fungsi berulang dan penggunaan stack memori.

❖ **Kasus Dasar**

Kasus dasar adalah nilai-nilai awal yang digunakan untuk memulai iterasi. Dalam masalah ini, kasus dasar adalah:

1. f(0)=1: Hanya satu cara untuk mencapai 0 km (tidak bergerak).
2. f(1)=1: Satu cara untuk mencapai 1 km (1 langkah 1 km).
3. f(2)=2: Dua cara untuk mencapai 2 km (1+1 km atau 2 km).

Kasus dasar ini digunakan untuk menghitung nilai-nilai berikutnya dalam iterasi.

Jumlah cara untuk n=110 adalah **70492524767089125814114** atau **7 x 10^22,** dihitung dalam waktu yang sangat cepat.

b. Algoritma rekursif

Mari kita bahas secara rinci perhitungan efektivitas waktu pada fungsi rekursif countWaysRecursive(n), terutama untuk n = 110. Fungsi ini mengikuti formula rekursif:

$$f(n) = f(n-2) + f(n-3)$$

1. **Struktur Rekursif**:
   Ketika fungsi dipanggil dengan n=110, ia akan memanggil dirinya sendiri dengan n − 2 = 108 dan n - 3 = 107.
   Untuk n = 108, fungsi akan memanggil f(106) dan f(105), dan begitu seterusnya hingga = 0 atau n < 0 sebagai kasus dasar.

2. **Pemanggilan Berulang**:
   Karena fungsi ini memanggil dirinya sendiri dua kali untuk setiap nilai n, jumlah pemanggilan fungsi akan bertambah secara eksponensial. Banyak pemanggilan yang sama terjadi berulang-ulang. Misalnya:
   - f(106) dihitung dua kali: sekali dari f(108) dan sekali dari f(107).
   - Semakin besar n, semakin banyak pemanggilan yang berulang.
3. **Analisis Kompleksitas Waktu**:
   Untuk setiap n, jumlah total pemanggilan fungsi dapat direpresentasikan sebagai jumlah node dalam pohon rekursif. Pohon ini memiliki 2 cabang untuk setiap node, sehingga kompleksitas waktunya adalah O(2^n). Untuk n = 110, jumlah total pemanggilan mendekati:
   2^{110}
   Nilai ini sangat besar (sekitar 1.27×10^{33}).

4. **Kasus Dasar**:
   Kasus dasar f(0) = 1 dan f(n < 0) = 0 hanya dihitung sekali dalam setiap cabang.

Setiap pemanggilan fungsi memakan waktu tetap T_c (misalnya 1 mikrodetik). Maka total waktu eksekusi T dapat dihitung sebagai:

$$T = T_c \times (2^{n/2})$$

Untuk n = 110 : $T \approx 1\ \mu s \times 2^{55} \approx 36\times10^{15}$ mikrodetik

Konversi ke detik:

$$T \approx 36 \times 10^{9} \text{ detik} = 1.14 \text{ tahun}$$

Jadi, metode rekursif sangat tidak efisien untuk nilai nn besar karena jumlah pemanggilan fungsi tumbuh eksponensial. Bahkan untuk n=110n = 110, eksekusi membutuhkan waktu bertahun-tahun.

E. Perbandingan

Berikut adalah tabel perbandingan antara metode iteratif dan rekursif untuk n=110:

Metode	Hasil	Waktu eksekusi	Efisiensi
Iteratif	Berhasil dihitung	~0.0005	Sangat efisien
Rekursif	Tidak selesai	1.14 tahun	Tidak efisien (eksponensial)

Penjelasan:

1. **Iteratif**:
    - Menggunakan array untuk menyimpan hasil antara.
    - Kompleksitas waktu adalah O(n).
    - Waktu eksekusi sangat cepat bahkan untuk n=110.
2. **Rekursif**:
    - Menggunakan pendekatan rekursif tanpa memoization.
    - Kompleksitas waktu adalahO(2^n), sehingga tidak efisien.
    - Pada n=110, waktu eksekusi sangat besar dan sulit diselesaikan dalam waktu wajar.

F. Kesimpulan

Dalam analisis kompleksitas algoritma untuk menghitung jumlah cara truk menempuh jarak tertentu dengan pola pengisian bahan bakar setiap 2 km atau 3 km, terdapat dua pendekatan utama yang dibandingkan, yaitu metode rekursif dan metode iteratif.

1. **Efisiensi Metode Iteratif**
   Metode iteratif terbukti jauh lebih efisien dibandingkan metode rekursif. Dengan kompleksitas waktu O(n), algoritma iteratif mampu menghitung hasil untuk jarak besar (seperti n=110) dalam waktu yang sangat singkat. Hal ini dicapai dengan memanfaatkan struktur iterasi yang sederhana dan menghindari redudansi perhitungan.
2. **Kekurangan Metode Rekursif**
   Metode rekursif, meskipun sederhana dalam implementasi, memiliki kompleksitas waktu eksponensial O(2^n). Hal ini menyebabkan jumlah pemanggilan fungsi yang sangat besar, sehingga tidak praktis untuk nilai n yang besar. Misalnya, untuk n=110, metode ini memerlukan waktu bertahun-tahun untuk menyelesaikan perhitungan.
3. **Rekomendasi**
   Dalam kasus perhitungan jumlah cara dengan pola tertentu seperti pada studi ini, metode iteratif lebih disarankan untuk digunakan. Pendekatan ini tidak hanya efisien dalam waktu, tetapi juga hemat dalam penggunaan memori, terutama jika dibandingkan dengan rekursi tanpa optimasi.

Melalui analisis ini, dapat disimpulkan bahwa pemilihan algoritma yang tepat sangat berpengaruh pada efisiensi dan kinerja, khususnya dalam konteks penghitungan logistik yang membutuhkan hasil cepat dan akurat.