TUGAS MANDIRI
KELOMPOK D
DECREASE-AND-CONQUER
JOSEPHUS PROBLEM
PERANCANGAN DAN ANALISIS ALGORITMA
INF1.62.4001
202223430034

DOSEN PENGAMPU:
Randi Proska Sandra, M.Sc

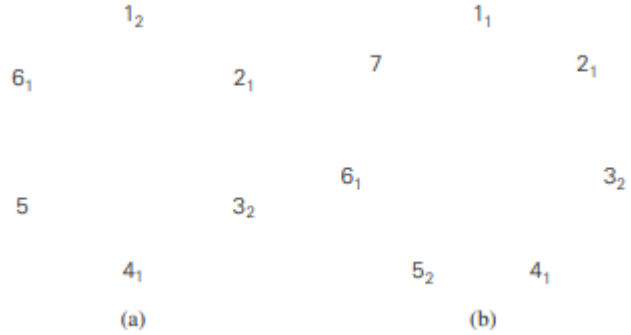
OLEH:
Muhammad Gilang Bagindo
21343030
Informatika

PROGRAM STUDI INFORMATIKA
DEPARTEMEN TEKNIK ELEKTRONIKA
FAKULTAS TEKNIK
UNIVERSITAS NEGERI PADANG
2023

A. Penjelasan Algoritma Josephus Problem

Masalah Josephus, dinamai untuk Flavius Josephus, seorang sejarawan Yahudi terkenal yang berpartisipasi dalam dan menceritakan pemberontakan Yahudi pada 66-70 Masehi melawan Romawi. Sebagai seorang jenderal, Josephus berhasil mempertahankan benteng Jotapata selama 47 hari, tetapi setelah kota itu jatuh, ia bersembunyi dengan 40 orang lainnya di dalam sebuah gua di dekatnya. Di sana, para pemberontak memutuskan untuk mati daripada menyerah. Josephus mengusulkan agar setiap orang secara bergiliran membunuh tetangganya, dengan urutan ditentukan dengan melempar koin. Josephus berhasil menarik nomor terakhir, dan sebagai salah satu dari dua orang yang selamat di dalam gua, ia berhasil membujuk orang yang seharusnya ia bunuh untuk menyerah kepada Romawi.

Jadi, ada n orang yang diberi nomor 1 hingga n berdiri dalam lingkaran. Dimulai dengan orang nomor 1, kita mengeliminasi setiap orang kedua sampai hanya satu orang yang selamat. Masalahnya adalah menentukan nomor orang yang selamat, J(n). Sebagai contoh (Gambar 4.12), jika n adalah 6, orang di posisi 2, 4, dan 6 akan dieliminasi pada putaran pertama di sekitar lingkaran, dan orang di posisi awal 3 dan 1 akan dieliminasi pada putaran kedua, meninggalkan satu orang yang selamat di posisi awal 5—maka, J(6) = 5. Sebagai contoh lain, jika n adalah 7, orang di posisi 2, 4, 6, dan 1 akan dieliminasi pada putaran pertama (lebih nyaman untuk menyertakan 1 dalam putaran pertama) dan orang di posisi 5 dan, untuk kenyamanan, 3 pada putaran kedua—maka, J(7) = 7.



Gambar 4.12 Contoh dari masalah Josephus untuk (a) n = 6 dan (b) n = 7. Angka subskrip menunjukkan putaran di mana orang pada posisi tersebut dieliminasi. Solusinya adalah J(6) = 5 dan J(7) = 7, masing-masing.

Mudah dipisahkan kasus n genap dan ganjil. Jika n genap, yaitu n = 2k, putaran pertama melalui lingkaran menghasilkan masalah yang sama persis, tetapi dengan setengah ukuran awalnya. Satu-satunya perbedaan adalah pada penomoran posisi; misalnya, orang di posisi awal 3 akan berada di posisi 2 untuk putaran kedua, orang di posisi awal 5 akan berada di posisi 3, dan seterusnya (lihat Gambar 4.12a). Mudah untuk melihat bahwa untuk mendapatkan posisi awal seseorang, kita hanya perlu mengalikan posisi barunya dengan 2 dan mengurangi 1. Hubungan ini akan berlaku, terutama untuk survivor, yaitu, J(2k) = 2J(k) - 1.

Sekarang, mari kita pertimbangkan kasus n ganjil (n> 1), yaitu n = 2k + 1. Putaran pertama menghilangkan orang pada semua posisi genap. Jika kita menambahkan penghapusan orang di posisi 1 tepat setelah itu, kita akan ditinggalkan dengan sebuah masalah berukuran k. Di sini, untuk mendapatkan

posisi awal yang sesuai dengan penomoran posisi baru, kita harus mengalikan nomor posisi baru dengan 2 dan menambahkan 1 (lihat Gambar 4.12b). Oleh karena itu, untuk nilai n ganjil, kita mendapatkan J(2k+1) = 2J(k) + 1.

Dapatkah kita mendapatkan solusi bentuk tertutup untuk rekurensi dua kasus ini dengan syarat awal J(1) = 1? Jawabannya adalah ya, meskipun memerlukan kecerdikan yang lebih daripada hanya menerapkan substitusi mundur. Faktanya, salah satu cara untuk menemukan solusinya adalah dengan menerapkan substitusi maju untuk mendapatkan, katakanlah, 15 nilai pertama J(n), menemukan polanya, dan kemudian membuktikan keabsahan umumnya dengan induksi matematika. Menariknya, bentuk yang paling elegan dari jawaban bentuk tertutup melibatkan representasi biner ukuran n: J(n) dapat diperoleh dengan geser siklik 1 bit ke kiri dari n itu sendiri! Misalnya, $J(6) = J(110_2) = 101_2 = 5$ dan $J(7) = J(111_2) = 111_2 = 7$.

B. Pseudocode Josephus Problem

Josephus Problem adalah masalah teoritis yang terkait dengan permainan hitung-keluar tertentu. Dalam masalah ini, ada n orang berdiri di sebuah lingkaran dan kita harus menghilangkan setiap orang ke-k sampai hanya tersisa satu orang. Posisi orang yang tersisa terakhir disebut posisi Josephus.

Berikut pseudocode untuk memecahkan Josephus Problem secara rekursif:

```
function josephus(n, k)
    if n = 1
        return 1
    else
        return (josephus(n - 1, k) + k-1) % n + 1
```

Fungsi di atas mengambil dua parameter: n yang mewakili jumlah orang yang berdiri di lingkaran dan k yang mewakili hitungan orang yang akan dilewati sebelum mengeliminasi satu orang. Fungsi ini mengembalikan bilangan bulat yang mewakili posisi orang yang tersisa terakhir.

Fungsi pertama memeriksa apakah hanya ada satu orang tersisa di lingkaran. Jika ya, itu akan mengembalikan posisi orang itu sebagai 1. Jika tidak, ia akan membuat pemanggilan rekursif dengan n-1 dan k sebagai argumen dan menambahkan (k-1) pada hasilnya. Hal ini dikarenakan setelah menghilangkan setiap orang ke-k, kita mulai menghitung dari posisi ke (k+1) untuk putaran penghapusan berikutnya. Akhirnya, kita ambil modulo n dari jumlah ini dan tambah 1 untuk mendapatkan posisi aktual dari orang yang tersisa terakhir.

Sebagai contoh, membuat pemanggilan josephus (7, 3) akan mengembalikan 4 sebagai posisi orang yang tersisa terakhir ketika dimulai dengan tujuh orang dan mengeliminasi setiap ketiga orang secara berputar.

Perlu diingat bahwa pseudocode ini dapat diimplementasikan dalam bahasa pemrograman apapun dengan menterjemahkannya ke sintaks kode.

C. Program Josephus Problem

Berikut adalah contoh program untuk memecahkan Josephus Problem menggunakan bahasa pemrograman Python:

```
def josephus(n, k):
    if n == 1:
        return 1
    else:
        return (josephus(n - 1, k) + k-1) % n + 1

# Contoh penggunaan
n = 7 # Jumlah orang dalam lingkaran
k = 3 # Jumlah orang yang harus dilewati sebelum satu orang
dieliminasi

print("Posisi orang terakhir yang tersisa adalah ", josephus(n,
k))
```

Program di atas akan mengeluarkan posisi orang yang tersisa terakhir ketika dimulai dengan n orang dan mengeliminasi setiap orang ke-k secara berputar. Anda dapat mengubah nilai n dan k sesuai kebutuhan.

Fungsi josephus() menerima dua parameter: n yang mewakili jumlah orang yang berdiri di lingkaran dan k yang mewakili hitungan orang yang akan dilewati sebelum mengeliminasi satu orang. Fungsi ini mengembalikan bilangan bulat yang mewakili posisi orang yang tersisa terakhir.

Fungsi pertama memeriksa apakah hanya ada satu orang tersisa di lingkaran. Jika ya, itu akan mengembalikan posisi orang itu sebagai 1. Jika tidak, ia akan membuat pemanggilan rekursif dengan n-1 dan k sebagai argumen dan menambahkan (k-1) pada hasilnya. Hal ini dikarenakan setelah menghilangkan setiap orang ke-k, kita mulai menghitung dari posisi ke (k+1) untuk putaran pengeliminasian berikutnya. Akhirnya, kita ambil modulo n dari jumlah ini dan tambah 1 untuk mendapatkan posisi aktual dari orang yg tersisa terakhir.

Berikut adalah contoh program dalam bahasa python untuk Josephus Problem dengan n dan k diinputkan dari terminal

```
def josephus(n, k):
    if n == 1:
        return 1
    else:
        return (josephus(n - 1, k) + k-1) % n + 1

# Contoh penggunaan
n = int(input('Masukkan jumlah orang dalam lingkaran: ')) # Jumlah
orang dalam lingkaran
k = int(input('Masukkan jumlah orang yang harus dilewati sebelum
satu orang dieliminasi: ')) #Jumlah orang yang harus
#dilewati sebelum satu orang dieliminasi

print("Posisi orang terakhir yang tersisa adalah ", josephus(n,
k))
```

D. Analisis Kebutuhan Waktu

1) Analisis menyeluruh dengan memperhatikan operasi/instruksi yang dieksekusi berdasarkan operator penugasan atau assignment (=, +=, *=, dan lainnya) dan operator aritmatika (+, %, * dan lainnya).
    - Operasi/intruksi seperti operator penugasan dan operator aritmatika dihitung 1, sehingga pada baris 2 kompleksitas waktunya adalah O(1), karena hanya ada 1 operator assignment.
    - Pada baris 4 terdapat 1 operasi penugasan sehingga T(n) = O(1)
    - Pada baris 5 terdapat 5 operator aritmatika, sehingga O(1) + O(1) + O(1) + O(1) + O(1) = O(1).

2) Analisis berdasarkan jumlah operasi abstrak atau operasi khas.

    Untuk program di atas, analisis jumlah operasi abstrak atau operasi khas dapat dilakukan dengan menghitung setiap operasi dalam fungsi josephus.

    Dalam fungsi josephus, terdapat dua operasi khas, yaitu pengurangan n dengan 1 dan operasi modulo, serta satu operasi penjumlahan dan satu operasi pengurangan.

    Maka, jumlah operasi khas yang diperlukan untuk menyelesaikan masalah Josephus Problem dengan pendekatan rekursif adalah 4 pada setiap panggilan fungsi.

    Karena pada setiap panggilan fungsi, nilai n berkurang satu, maka total jumlah panggilan fungsi adalah n. Sehingga, jumlah operasi khas yang diperlukan untuk menyelesaikan masalah ini dengan pendekatan rekursif adalah 4n.

    Dalam hal ini, kompleksitas waktu program adalah O(n) karena setiap panggilan fungsi mengurangi nilai n sebanyak satu. Namun, karena program ini menggunakan rekursi, maka program dapat menjadi lebih lambat untuk ukuran masukan yang besar. Oleh karena itu, untuk ukuran masukan yang besar, pendekatan iteratif dapat menjadi pilihan yang lebih baik.

3) Analisis menggunakan pendekatan best-case (kasus terbaik), worst-case (kasus terburuk), dan average-case (kasus rata-rata).
    - Best-case (kasus terbaik) dari algoritma Josephus Problem dengan pendekatan rekursif terjadi ketika nilai n sama dengan 1. Pada kondisi ini, fungsi langsung mengembalikan nilai 1 tanpa melakukan operasi tambahan apa pun. Jumlah operasi khas yang diperlukan untuk kasus terbaik adalah 0, sehingga kompleksitas waktu program pada kasus terbaik adalah O(1).

    - Worst-case (kasus terburuk) dari algoritma Josephus Problem dengan pendekatan rekursif terjadi ketika nilai n sangat besar. Pada kondisi ini, program akan melakukan panggilan fungsi secara rekursif hingga hanya satu orang yang tersisa. Setiap panggilan fungsi akan memerlukan empat operasi khas, sehingga jumlah

operasi khas yang diperlukan sebesar 4n. Oleh karena itu, kompleksitas waktu program pada kasus terburuk adalah O(n).

- Average-case (kasus rata-rata) dari algoritma Josephus Problem dengan pendekatan rekursif terjadi ketika nilai n berada di tengah-tengah atau ada dalam rentang ukuran masukan. Pada kasus rata-rata, kompleksitas waktu program adalah O(n). Namun, karena tidak ada distribusi masukan yang pasti, maka analisis kasus rata-rata bergantung pada distribusi masukan. Sebagai contoh, jika distribusi masukan terdapat di rentang kecil dan homogen, maka kompleksitas waktu program bisa menjadi lebih cepat daripada kasus terburuk. Namun, jika distribusi masukan sangat luas dan heterogen, maka kompleksitas waktu program bisa menjadi sangat lambat dan mendekati kasus terburuk.

E. Referensi

Levitin, Anany. 2012. Introduction to the Design and Analysis of Algorithms. Edisi ketiga. Boston, MA: Pearson Education, Inc.

Wikipedia. “Josephus Problem’. Wikipedia, https://en.wikipedia.org/wiki/Josephus_problem. Diakses pada 7 Maret 2023.

InterviewBit. "Josephus Problem." InterviewBit Blog, 2 September 2020, https://www.interviewbit.com/blog/josephus-problem/. Diakses pada 7 Maret 2023.

Scaler Academy. "Josephus Problem." Scaler Academy, https://www.scaler.com/topics/josephus-problem/. Diakses pada 7 Maret 2023

F. Lampiran Link Github

https://github.com/MuhammadGilangBagindo/MuhammadGilangBagindo_PerancanganAnalisisAlgoritma/tree/main/Tugas%201%20Mandiri_Perancangan%20dan%20Analisis%20Algoritma_Muhammad%20Gilang%20Bagindo_21343030