## Tugas Kelompok ke-3

## Week 9

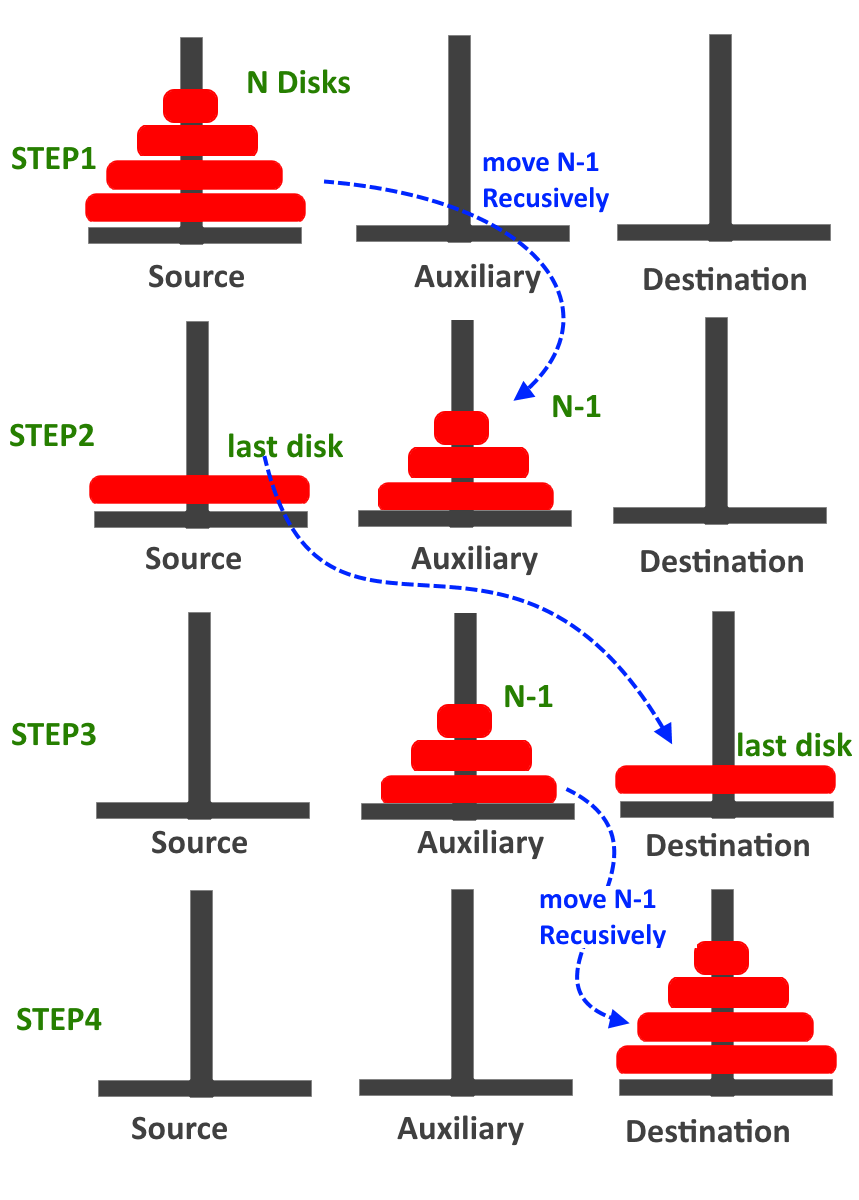
**Anggota Kelompok 6**

|  |  |
| --- | --- |
| NIM | Nama |
| 2802548713 | **MUHAMAD TOHA** |
| 2802527532 | **CINDI TIA ARDANESWARI. H** |
| 2802550541 | **SUGENG WAHYUDI** |
| 2802567126 | **USAMA WARDANA** |

## Soal 1: Desain dan Analisis Recursive Function

### Deskripsi Soal

Anda diminta untuk merancang sebuah fungsi rekursif yang menyelesaikan masalah "***Towers of Hanoi***" untuk 4 buah disk. Fungsi harus dapat mengindikasikan langkah-langkah pemindahan disk dari tiang A ke tiang C dengan menggunakan tiang B sebagai perantara.



Gambar 1. Ilustrasi Algoritma Towers of Hanoi

### Instruksi Pengerjaan

1. Buat pseudo code atau flowchart untuk fungsi rekursif yang menyelesaikan Towers of Hanoi.
2. Jelaskan secara terperinci setiap langkah rekursi yang dilakukan oleh fungsi Anda.
3. **Pseudo Code/Flowchart**: Harus menggambarkan pemanggilan rekursif yang jelas dan kondisi berhenti rekursi.
4. **Penjelasan Langkah**: Harus mencakup base case dan cara fungsi memecah masalah menjadi sub-problem yang lebih kecil.
5. **Analisis Kompleksitas**: Berikan analisis tentang kompleksitas waktu dari algoritma berdasarkan jumlah disk.

**Soal 2: Analisis Kesalahan pada Implementasi Recursive Function**

### Deskripsi Soal:

Diberikan pseudo code dari implementasi rekursif yang dirancang untuk menghitung faktorial suatu bilangan tetapi mengandung kesalahan logis. Tugas Anda adalah menganalisis dan menentukan kesalahan dalam pseudo code tersebut dan memberikan solusi yang benar.

### Cuplikan Pseudo Code:

|  |
| --- |
| Function factorial(n):  if n == 0:  return 1  else:  return n \* factorial(n) |

### Instruksi Pengerjaan:

1. Identifikasi dan jelaskan kesalahan dalam pseudo code yang diberikan.
2. Berikan perbaikan yang benar pada pseudo code.
3. Gambarkan flowchart yang menunjukkan fungsi rekursif yang benar untuk menghitung faktorial.

### Catatan Penting

* Pastikan untuk mencantumkan referensi jika menggunakan sumber dari internet atau bantuan GPT.
* Cantumkan referensi dengan format: Nama sumber, Tautan (jika ada), diakses pada [Tanggal Akses].
* Pekerjaan harus asli dan setiap bentuk plagiarisme akan ditindak sesuai dengan kebijakan akademik.

**Jawaban:**

**Soal 1:** **Desain dan Analisis Recursive Function**

**1. Pseudo Code**

FUNCTION hanoi(n, from\_peg, to\_peg, aux\_peg)

IF n == 1 THEN

PRINT "Pindahkan disk 1 dari", from\_peg, "ke", to\_peg

RETURN

END IF

// Pindahkan n-1 disk dari tiang asal ke tiang bantu

CALL hanoi(n - 1, from\_peg, aux\_peg, to\_peg)

// Pindahkan disk teratas dari tiang asal ke tiang tujuan

PRINT "Pindahkan disk", n, "dari", from\_peg, "ke", to\_peg

// Pindahkan n-1 disk dari tiang bantu ke tiang tujuan

CALL hanoi(n - 1, aux\_peg, to\_peg, from\_peg)

END FUNCTION

// Program utama

FUNCTION main()

DECLARE n AS INTEGER

SET n = 4 // Jumlah disk

PRINT "Langkah-langkah pemindahan untuk", n, "disk:"

CALL hanoi(n, 'A', 'C', 'B') // Memindahkan disk dari A ke C dengan B sebagai bantu

END FUNCTION

**2. Langkah – Lagkah Rekursi yang dilakukan adalah sebagai berikut:**

* + 1. Memanggil hanoi(4, 'A', 'C', 'B'):
* Tujuan: Memindahkan 4 disk dari A ke C.
* Langkah: Pertama, pindahkan 3 disk dari A ke B menggunakan C sebagai tiang bantu.
* Rekursi: Panggil hanoi(3, 'A', 'B', 'C').
  + 1. Memanggil hanoi(3, 'A', 'B', 'C'):
* Tujuan: Memindahkan 3 disk dari A ke B.
* Langkah: Pertama, pindahkan 2 disk dari A ke C menggunakan B sebagai tiang bantu.
* Rekursi: Panggil hanoi(2, 'A', 'C', 'B').
  + 1. Memanggil hanoi(2, 'A', 'C', 'B'):
* Tujuan: Memindahkan 2 disk dari A ke C.
* Langkah: Pertama, pindahkan 1 disk dari A ke B menggunakan C sebagai tiang bantu.
* Rekursi: Panggil hanoi(1, 'A', 'B', 'C').
  + 1. Memanggil hanoi(1, 'A', 'B', 'C'):
* Dasar Rekursi: Hanya ada 1 disk. Pindahkan disk 1 dari A ke B.
* Output: "Pindahkan disk 1 dari A ke B".
* Kembali ke: hanoi(2, 'A', 'C', 'B').
  + 1. Kembali ke hanoi(2, 'A', 'C', 'B'):
* Setelah memindahkan disk 1, sekarang pindahkan disk 2 dari A ke C.
* Output: "Pindahkan disk 2 dari A ke C".
* Sekarang pindahkan disk 1 dari B ke C menggunakan A sebagai tiang bantu.
* Rekursi: Panggil hanoi(1, 'B', 'C', 'A').
  + 1. Memanggil hanoi(1, 'B', 'C', 'A'):
* Dasar Rekursi: Pindahkan disk 1 dari B ke C.
* Output: "Pindahkan disk 1 dari B ke C".
* Kembali ke: hanoi(3, 'A', 'B', 'C'**).**
  + 1. Kembali ke hanoi(3, 'A', 'B', 'C'):
* Setelah memindahkan 2 disk ke C, sekarang pindahkan disk 3 dari A ke B.
* Output: "Pindahkan disk 3 dari A ke B".
* Selanjutnya, pindahkan 2 disk dari C ke B menggunakan A sebagai tiang bantu.
* Rekursi: Panggil hanoi(2, 'C', 'B', 'A').
  + 1. Memanggil hanoi(2, 'C', 'B', 'A'):’
* Tujuan: Memindahkan 2 disk dari C ke B.
* Langkah: Pertama, pindahkan 1 disk dari C ke A menggunakan B sebagai tiang bantu.
* Rekursi: Panggil hanoi(1, 'C', 'A', 'B').
  + 1. Memanggil hanoi(1, 'C', 'A', 'B'):
* Dasar Rekursi: Pindahkan disk 1 dari C ke A.
* Output: "Pindahkan disk 1 dari C ke A".
* Kembali ke: hanoi(2, 'C', 'B', 'A').
  + 1. Kembali ke hanoi(2, 'C', 'B', 'A'):
* Setelah memindahkan disk 1 ke A, sekarang pindahkan disk 2 dari C ke B.
* Output: "Pindahkan disk 2 dari C ke B".
* Sekarang pindahkan disk 1 dari A ke B menggunakan C sebagai tiang bantu.
* Rekursi: Panggil hanoi(1, 'A', 'B', 'C').
  + 1. **Memanggil hanoi(1, 'A', 'B', 'C'):**
* **Dasar** Rekursi**: Pindahkan disk 1 dari A ke B.**
* **Output: "**Pindahkan disk 1 dari A ke B".
* Kembali ke: hanoi**(4, 'A', 'C', 'B').**
  + 1. **Kembali ke hanoi(4, 'A', 'C', 'B'):**
* Setelah memindahkan 3 disk ke B, sekarang pindahkan disk 4 dari A ke C.
* Output: "Pindahkan disk 4 dari A ke C".
* Terakhir, kita perlu memindahkan 3 disk dari B ke C menggunakan A sebagai tiang bantu.
* Rekursi: Panggil **hanoi(3, 'B', 'C', 'A').**
  + 1. Memanggil hanoi(3, 'B', 'C', 'A'):
* Tujuan: Memindahkan 3 disk dari B ke C.
* Langkah: Pertama, kita perlu memindahkan 2 disk dari B ke A menggunakan C sebagai tiang bantu.
* Rekursi: Panggil hanoi(2, 'B', 'A', 'C').
  + 1. Memanggil hanoi(2, 'B', 'A', 'C'):
* Tujuan: Memindahkan 2 disk dari B ke A.
* Langkah: Pertama, kita perlu memindahkan 1 disk dari B ke C menggunakan A sebagai tiang bantu.
* Rekursi: Panggil hanoi(1, 'B', 'C', 'A')
  + 1. Memanggil hanoi(1, 'B', 'C', 'A'):
* Dasar Rekursi: Pindahkan disk 1 dari B ke C.
* Output: "Pindahkan disk 1 dari B ke C".
* Kembali ke: hanoi(2, 'B', 'A', 'C').
  + 1. Kembali ke hanoi(2, 'B', 'A', 'C'):
* Pindahkan disk 2 dari B ke A.
* Output: "Pindahkan disk 2 dari B ke A".
* Sekarang kita pindahkan disk 1 dari C ke A menggunakan B sebagai tiang bantu.
* Rekursi: Panggil hanoi(1, 'C', 'A', 'B').
  + 1. Memanggil hanoi(1, 'C', 'A', 'B'):’
* Dasar Rekursi: Pindahkan disk 1 dari C ke A.
* Output: "Pindahkan disk 1 dari C ke A".

A diagram of a computer

Description automatically generated with medium confidenceBerikut alur eksekusi metode rekursif pada teka-teki Tower of Hanoi 4 disk:

gambar disertakan dengan nama file hanoi.png

**3. Analisis Kompleksitas**

Kompleksitas waktu dari algoritma ini dapat dianalisis dengan mempertimbangkan cara pemindahan disk yang dilakukan. Berikut penjelasan langkah-langkah untuk memindahkan 4 disk dari satu tiang ke tiang lainnya dengan lebih sederhana.

Tujuan dari Tower of Hanoi dari soal ini, yaitu memindahkan 4 disk dari tiang A ke tiang C dengan bantuan tiang B.

Langkah-langkah untuk memindahkan 4 disk:

1. Pindahkan 3 disk dari tiang A ke tiang B (menggunakan tiang C sebagai bantuan).
2. Pindahkan disk ke-4 dari tiang A ke tiang C.
3. Pindahkan 3 disk dari tiang B ke tiang C (menggunakan tiang A sebagai bantuan).

Untuk **poin a** terdapat beberapa tahapan proses untuk memindahkan 3 disk dari tiang A ke tiang B, yaitu:

1. Pindahkan 2 disk dari tiang A ke tiang C (menggunakan tiang B sebagai bantuan).

Untuk melakukan ini:

* Pindahkan disk 1 dari A ke B (disk terkecil, tidak ada yang lebih kecil di atasnya).
* Pindahkan disk 2 dari A ke C (sekarang hanya disk 2 di C).
* Pindahkan disk 1 dari B ke C (disk 1 sekarang ada di atas disk 2 di C).

Total: 3 langkah.

Setelah langkah ini, disk 1 berada di C (di atas disk 2), disk 2 berada di C, dan disk 3 masih di A.

1. Pindahkan disk ke-3 dari tiang A ke tiang B.

Total: 1 langkah.

Sekarang, disk 3 dapat dipindahkan dari A ke B karena disk 1 dan 2 tidak ada di B.

1. Pindahkan 2 disk dari tiang C ke tiang B (menggunakan tiang A sebagai bantuan).

Untuk melakukan ini:

* Pindahkan disk 1 dari C ke A (disk terkecil).
* Pindahkan disk 2 dari C ke B (disk kedua).
* Pindahkan disk 1 dari A ke B (disk 1 sekarang berada di atas disk 2 di B).

Total: 3 langkah.

Jadi, total langkah untuk memindahkan 3 disk pada **poin a** adalah:

3 (untuk 2 disk) + 1 (disk ke-3) + 3 (untuk 2 disk) = 7 langkah

Setelah **poin a** selesai, lanjut ke **poin b**, yaitu memindahkan disk ke-4 dari tiang A ke tiang C, ini memerlukan 1 langkah.

Setelah itu, lanjut ke **poin c** untuk memindahkan 3 disk dari B ke C, yang memerlukan 7 langkah (sama seperti pada **poin a**).

Jadi, untuk memindahkan 4 disk dari tiang A ke tiang C menggunakan tiang B sebagai bantuan, total langkah yang dibutuhkan adalah 15 langkah (7 langkah + 1 langkah + 7 langkah).

Kompleksitas waktu dari algoritma Tower of Hanoi ini menunjukkan waktu yang diperlukan untuk menyelesaikan pemindahan disk meningkat seiring bertambahnya jumlah disk. Dalam algoritma ini, ketika ingin memindahkan n disk, perlu dilakukan beberapa langkah yang melibatkan pemindahan disk yang lebih kecil terlebih dahulu. Setiap kali ada tambahan disk, jumlah langkah yang diperlukan untuk memindahkan disk tersebut menjadi lebih banyak.

Jika hanya ada satu disk, cukup satu langkah untuk memindahkannya. Namun, saat ada dua disk, dibutuhkan tiga Langkah. Ketika jumlah disk bertambah menjadi tiga, diperlukan tujuh langkah, dan ketika ada empat disk, jumlah langkah meningkat menjadi lima belas.

Dari analisis tersebut, dapat disimpulkan bahwa kompleksitas waktu dari algoritma Tower of Hanoi adalah O(2^n). Notasi ini menandakan bahwa waktu yang diperlukan untuk menyelesaikan pemindahan disk akan tumbuh secara eksponensial seiring dengan bertambahnya jumlah disk yang harus dipindahkan. Dimana untuk menghitung banyak langkah dapat menggunakan T(n) = 2^n – 1 (n adalah jumlah disk).

**Soal 2:** **Analisis Kesalahan pada Implementasi Recursive Function**

**Cuplikan pseudocode dan code**

|  |
| --- |
| Function factorial(n):  if n == 0:  return 1  else:  return n \* factorial(n) |
|  |

1. **Identifikasi dan jelaskan kesalahan dalam pseudo code yang diberikan.**

Potongan pseudo-code diatas merupakan sebuah fungsi dengan nama factorial dengan argumen n dengan block code yang berisi

* Baris ke-3 berisi conditional if untuk melakukan pengecekan nilai dari n apakah n sama dengan nol
* Baris ke-4 memberikan return angka 1 jika n sama dengan nol karena faktorial dari 0 adalah 1.
* Baris ke-5-6 berisi else statement yang merupakan operasi perkalian n dengan faktorial(n) yang merupakan fungsi itu sendiri sehingga terjadi rekursif.
* Kesalahan dalam program tersebut adalah pada baris ke-6 saat melakukan pemanggilan ke fungsi itu sendiri atau rekursi nilai argumen yang di passing adalah n. Hal ini tidak tepat karena akan menyebakan infinite loop karena program akan memanggil fungsi itu sendiri dengan nominal n terus menerus sebagai argumen sehingga tidak ada jalan untuk program itu selesai. Oleh karena itu hal ini sangat fatal karena dapat menyebabkan program crash dan stack overflow karena kehabisan memori.

1. **Berikan perbaikan yang benar pada pseudo code.**

Untuk memperbaiki kode diatas kita dapat mengubah argumen n menjadi n-1 karena memang secara logika untuk menghitung faktorial adalah dengan melakukan perkalian dengan angka yang lebih kecil selisih satu dan seterusnya sampai dengan 1. Berikut adalah cuplikan source code yang sudah dibenarkan yang kami sampaikan dalam file faktorial.c

A screen shot of a computer program

Description automatically generated

1. **Gambarkan flowchart yang menunjukkan fungsi rekursif yang benar untuk menghitung faktorial.**

A screenshot of a diagram

Description automatically generated

Sumber referensi:

ChatGPT pada link <https://chatgpt.com/share/67107bd6-6974-800c-a877-e3b450b9368b> diakses tanggal 26 Oktober 2024.

GeeksforGeeks - Tower of Hanoi Algorithm. <https://www.geeksforgeeks.org/c-program-for-tower-of-hanoi/> diakses tanggal 27 Oktober 2024.

A Complete Recursive Visualization. <https://www.youtube.com/watch?v=rf6uf3jNjbo> diakses tanggal 27 Oktober 2024.