ALGORITMA TURNAMEN UNTUK MASALAH MUTUAL EXCLUSION PADA SISTEM TERDISTRIBUSI MEMORI BERSAMA

Skripsi

Oleh :

M. AHSAN. SM

H 111 02 019

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2009

ALGORITMA TURNAMEN UNTUK MASALAH MUTUAL EXCLUSION PADA SISTEM TERDISTRIBUSI MEMORI BERSAMA

S K R I P S I

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Sains pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin, Makassar

Oleh:

M. AHSAN. SM
H 111 02 019

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR
2009

P E R N Y A T A A N

Saya yang bertanda tangan di bawah ini menyatakan dengan sesungguh-sungguhnya bahwa skripsi yang saya buat dengan judul :

"ALGORITMA TURNAMEN UNTUK MASALAH MUTUAL EXCLUSION PADA SISTEM TERDISTRIBUSI MEMORI BERSAMA"

adalah benar hasil kerja saya sendiri, bukan hasil plagiat dan belum pernah dipublikasikan dalam bentuk apapun.

Makassar, 22 Januari 2009

M. AHSAN. SM
NIM : H 111 02 019

ALGORITMA TURNAMEN UNTUK MASALAH MUTUAL EXCLUSION PADA SISTEM TERDISTRIBUSI MEMORI BERSAMA

Disetujui Oleh :

Pembimbing Utama

Dr. Armin Lawi, M.Eng

NIP. 132 133 693

Pembimbing Pertama

Andi Galsan Mahie, S.Si, M.Si

NIP. 132 314 941

Pembimbing Kedua

Drs. Muh. Hasbi, M.Sc

NIP. 131 846 397

Pada tanggal : 22 Januari 2009

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

Pada hari ini, Jumat tanggal 22 Januari 2009, Panitia Ujian Skripsi menerima dengan baik skripsi yang berjudul :

ALGORITMA TURNAMEN UNTUK MASALAH MUTUAL EXCLUSION PADA SISTEM TERDISTRIBUSI MEMORI BERSAMA

yang diajukan untuk memenuhi salah satu syarat guna memperoleh gelar Sarjana Sains pada Program Studi Matematika Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

Makassar, 22 Januari 2009

PANITIA UJIAN SKRIPSI

		Tanda Tangan
1. Ketua	: **Drs. Diaraya, M.Ak**	(.....)
2. Sekretaris	: **Dr. Loeky Haryanto, MS.MSc.MA**	(.....)
3. Anggota	: **Dr. Armin Lawi, M.Eng**	(.....)
4. Anggota	: **Andi Galsan Mahie, S.Si, M.Si**	(.....)
5. Anggota	: **Drs. Muh. Hasbi, M.Sc**	(.....)

KATA PENGANTAR

Sistem terdistribusi memori bersama adalah sistem yang terdiri dari satu atau beberapa memori bersama yang digunakan oleh beberapa prosesor untuk saling berkomunikasi dan tidak ada pengontrol yang bersifat terpusat. *Mutual exclusion* adalah mekanisme mensinkronkan prosesor-prosesor yang berkompetisi untuk mengakses sumber daya yang hanya dapat dipakai oleh paling banyak satu prosesor secara bersama pada setiap satu kesatuan waktu. Algoritma yang dirancang untuk masalah ini harus memenuhi 3 sifat berikut : paling banyak satu prosesor memiliki izin untuk mengaksesi sumber daya; semua permintaan untuk mengakses sumber daya akhirnya akan dikabulkan; dan, jika terdapat beberapa prosesor yang ingin mengakses sumber daya dan tidak ada prosesor lain yang mengakses sumber daya tersebut maka satu diantaranya boleh mengakses sumber daya pada waktu tersebut. Masalah *mutual exclusion* pada sistem terdistribusi memori bersama yg dibahas pada skripsi ini menggunakan Algoritma Turnamen.

UCAPAN TERIMA KASIH

Alhamdulillah Rabbil Alamin. Segala puji hanyalah milik **Allah SWT** semata, Rabb seluruh alam semesta. Hanya kepada-Nyalah kita memohon pertolongan dan ampunan. Shalawat beriring salam semoga tercurah kepada **Nabi Muhammad SAW** pembawa kabar gembira dan peringatan, penyeru kepada jalan kebenaran dan pelita yang terang benderang, juga kepada sanak keluarga, para Sahabat, serta umatnya yang senantiasa taat hingga hari kebangkitan.

Skripsi ini merupakan salah satu persyaratan dalam menyelesaikan Studi di Program Studi Matematika Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin Makassar. Dalam penyelesaian skripsi ini memerlukan proses yang panjang dan berliku. Namun berkat rahmat dan hidayah-Nya, dan atas bantuan dari berbagai pihak akhirnya skripsi ini dapat diselesaikan.

Oleh sebab itu, penulis menyampaikan ucapan terima kasih dan penghargaan yang setinggi-tingginya kepada Ayahanda dan Ibunda tercinta **Drs. Syarafuddin Malik** dan **Hj. St. Hukman Rachim** atas limpahan cinta, kasih sayang dan pengorbanan dalam membesarkan dan mendidik penulis, dukungan moril dan materil serta doanya yang senantiasa dipanjatkan dan tak ternilai harganya demi keberhasilan ananda selama menjalani proses pendidikan. Untuk kakakku tersayang **Malyanah SM. SP** dan **Encep Kusmana SP**, serta

adikku yang tercinta **Ahmad Irfan SM** dan **St. Hardiyanti SM** terima kasih yang tak terhingga atas segala pengertian, nasehat, bantuan , serta doanya.

Demikian pula dengan penuh rasa syukur penulis mengucapkan penghargaan dan terima kasih yang setinggi-tingginya kepada:

1. Bapak **Dr. Armin Lawi, M.Eng** selaku Pembimbing Utama, Bapak **Andi Galsan Mahie, S.Si, M.Si** selaku Pembimbing Pertama dan Bapak **Drs. Muh Hasbi, M.Sc** selaku Pembimbing Kedua, atas waktu yang telah diluangkan, ilmu, petunjuk dan bimbingannya dalam penyusunan hingga penyelesaian skripsi ini.
2. Bapak **Drs. Diaraya, M.Ak** selaku Ketua Penguji dan Bapak **Dr. Loeky Haryanto, MS, M.Sc, MA** selaku Sekretaris Penguji yang telah memberikan saran dan pertanyaan yang bermanfaat dalam penyusunan skripsi ini.
3. Bapak **Drs. Muh Zakir, M.Si** selaku Ketua Jurusan Matematika dan para **Dosen Jurusan Matematika.**
4. Bapak **Andi Kresna Jaya S.Si M.Si** selaku Penasehat Akademik atas saran, nasehat, arahan dan dorongan selama ini.
5. Bapak **Sutamin S.Sos**, Bapak **Nasir**, Bapak **Akbar** selaku Staf Jurusan Matematika yang telah memberikan bantuan selama perkuliahan hingga ujian akhir.
6. **Hj Marma, Drs Abdullah Rahim, M.Si** dan seluruh keluarga lainnya atas bantuan, perhatian, dan dukungannya.

7. **ZzztQQ, Rumble Crew, Blacklist, Tala' Salapang Crew** dan **PS Crew** atas nasehat, saran, motivasi dan doanya.
8. Seluruh warga **Himatika** (Aritmatika 02, Geometri 03, Dimensi 04, Ordinat 05, Epsilon 06, Ekspansi 07, Ekspektasi 08).
9. Seluruh warga **Himafi, Himbio** dan **HMK** atas persaudaraan **KMF MIPA.**

Semoga segala partisipasinya bernilai ibadah, menjadi sebentuk amal jariyah dan mendapat pahala kebaikan disisi Allah SWT.

Akhir kata, semoga tulisan ini dapat memberikan manfaat kepada semua pihak yang membutuhkan dan terutama bagi penulis.

Makassar, 20 Januari 2009

Penulis

ABSTRAK

Sistem terdistribusi memori bersama merupakan sistem komputer yang terdiri atas beberapa prosesor yang saling berkomunikasi satu sama lain menggunakan memori bersama. Dalam sistem terdistribusi memori bersama, terdapat sumberdaya yang hanya dapat diakses secara saling-bebas (atau mutual exklusif); yaitu, sumberdaya hanya boleh diakses paling banyak satu proses pada satu kesatuan waktu. Permasalahan akan terjadi ketika dua atau lebih proses ingin mengeksekusi sumberdaya ini secara bersamaan. Masalah sinkronisasi dalam mengatasi konflik akses seperti ini disebut masalah mutual exclusion (atau masalah saling lepas terdistribusi). Algoritma Turnamen merupakan salah satu algoritma sinkronisasi yang dapat memecahkan masalah mutual exclusion dengan kompleksitas running time untuk kasus terburuk adalah $(n-1)c+O(n^2\ell)$, dimana n, c dan l masing-masing menyatakan jumlah proses dalam sistem, batas waktu maksimum proses dalam mengakses sumberdaya dan banyaknya langkah yang ditempuh dalam prosedur algoritma. Running time ini menyatakan batas atas dari waktu sejak proses mengakses trying protocol hingga memasuki daerah kritis.

Kata kunci: **Sistem terdistribusi memori bersama, masalah saling bebas terdistribusi, algoritma turnamen, kompleksitas running time.**

ABSTRACT

Distributed shared memory systems consists of processors communicated with each other using shared memory (or register). In many distributed shared memory systems, there are some resources that can only be accessed by processes in a mutually exclusive way (i.e., at most one process may access the resource at a time). A problem arises when two or more processes competing to execute the resource concurrently, while the resource can only be used bya at most one process at a time. This kind of synchronization problem is called mutual exclusion. Tournament Algorithm is one of synchronization algorithms which can resolve the mutual exclusion problem with the worst-case running time complexity, $(n-1)c+O(n^2\ell)$, where n, c and l, respectively, denote the number of processes in the system, the upper-bound of the process spend its time in the resource and the number of steps of the process when executing the algorithm. This running time is an upper bound time of the process when it accesses the trying protocol until it successfully enters the critical region.

Keywords: **Distributed shared memory system, mutual exclusion, tournament algorithm, running time complexity.**

DAFTAR ISI

DAFTAR GAMBAR

BAB I

PENDAHULUAN

I.1. LATAR BELAKANG

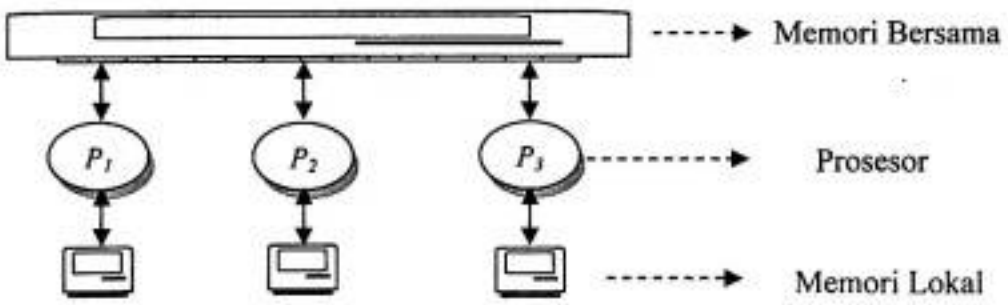
Sekalipun didukung oleh teknologi prosesor yang berkembang sangat pesat, komputer sekuensial tetap akan mengalami keterbatasan dalam hal kecepatan pemrosesannya. Hal ini menyebabkan lahirnya konsep komputasi paralel untuk menangani masalah dan aplikasi yang membutuhkan kecepatan pemrosesan yang sangat tinggi, seperti misalnya prakiraan cuaca, simulasi pada reaksi kimia, perhitungan aerodinamika dan lain-lain.

Komputer sekuensial adalah komputer yang hanya mempunyai satu unit prosesor untuk menentukan instruksi yang akan dieksekusi. Pada setiap satuan waktu hanya satu instruksi yang dapat dieksekusi, dimana kecepatan akses ke memori dan kecepatan piranti masukan dan keluaran dapat memperlambat proses komputasi. Beberapa metoda dibangun untuk menghindari masalah tersebut, seperti penggunaan *cache memory* atau memori lokal. Namun komputer sekuensial ini tetap mengalami keterbatasan jika menangani masalah yang memerlukan perhitungan yang banyak dan membutuhkan waktu lama. Hal-hal tersebut di atas pada akhirnya melatarbelakangi lahirnya sistem komputer paralel (Lester,1993).

Komputer paralel mempunyai lebih dari satu unit prosesor dalam sebuah komputer yang sama. Sebuah komputer dengan banyak prosesor disebut sebagai

komputer paralel karena seluruh prosesor tersebut dapat beroperasi secara simultan. Jika tiap-tiap prosesor dapat mengerjakan satu juta operasi tiap detik, maka sepuluh prosesor dapat mengerjakan sepuluh juta operasi tiap detik, seratus prosesor akan dapat mengerjakan seratus juta operasi tiap detiknya sehingga komputasi dapat dilakukan dengan performa tinggi.

Komputer paralel dengan banyak prosesor yang bekerja secara simultan memerlukan kemampuan untuk mendistribusi memori bersama atau *distributed shared memory* (DSM) dan berkomunikasi antar prosesor. Pada sistem terdistribusi memori bersama, setiap prosesor umumnya memiliki memori lokal dimana tiap prosesor dapat menulis dan membaca di memori lokalnya masing-masing namun hanya dapat membaca memori lokal prosesor lainnya. Setiap prosesor juga dapat mengakses memori besar dan menggunakan isi data dan struktur data yang disimpan dalam memori bersama (*shared memory*). Prosesor berkomunikasi dengan prosesor lain dengan menulis dan membaca pesan pada memori besar. Ilustrasi DSM pada komputer paralel dapat dilihat pada gambar 1.1.1 (Maryuni,2008).



Gambar 1.1.1 Sistem Distribusi Komputer Paralel

Salah satu kesulitan utama dari sistem distribusi memori bersama adalah kemungkinan adanya tabrakan akses dalam mengeksekusi sumber daya. Peristiwa ini terjadi ketika beberapa prosesor mencoba untuk mengakses sumber daya dalam periode waktu yang sangat singkat, sehingga sumber daya tidak akan dapat menampung dan mengkoordinir semua permintaan secara simultan. Akibatnya beberapa prosesor akan harus menunggu sampai prosesor lainnya dilayani. Kemungkinan terjadinya tabrakan akses ini berhubungan dengan bertambahnya jumlah prosesor.

Dalam kebanyakan sistem terdistribusi,akses *mutually exclusive* sering diperlukan dalam mengakses sumber daya secara simultan. Akses ini diperlukan ketika sebuah sumber daya hanya dapat diakses secara *mutually exclusive* ; yaitu, sebuah sumber daya hanya dapat dieksekusi oleh paling banyak satu prosesor pada satu waktu, sehingga konflik dari akses-akses ini perlu untuk disinkronkan pencapaiannya dalam mengeksekusi sumber daya secara bersamaan. Salah satu cara sinkronisasi akses dari prosesor-prosesor yang mengeksekusi sumber daya secara simultan menggunakan algoritma *mutual exclusion.*

Berdasarkan uraian diatas, penulis tertarik untuk mengkaji masalah *mutual exclusion* dengan menggunakan algoritma turnamen, yang akan dipaparkan dalam bentuk skripsi dengan judul :

"Algoritma Turnamen untuk Masalah Mutual Exclusion pada Sistem Terdistribusi Memori Bersama"

I.2. RUMUSAN MASALAH

Rumusan masalah yang dibahas dalam tulisan ini adalah

1. Bagaimana prosesor-prosesor dalam sebuah sistem terdistribusi memori bersama atau *distributed shared memory system* (DSM) berkomunikasi satu sama lainnya tanpa ada pengontrol yang bersifat terpusat (*central conroller*)?
2. Bagaimana mekanisme sinkronisasi prosesor-prosesor yg berkompetisi dalam mengakses sumber daya yang hanya dapat dipakai oleh paling banyak satu prosesor secara bersama pada setiap satu kesatuan waktu (masalah *mutual exclusion*) dalam sistem DSM?
3. Bagaimana abstraksi kerja sebuah algoritma untuk menyelesaikan masalah *mutual exclusion* pada sistem DSM?
4. Bagaimana kinerja *running time* algoritma turnamen dalam menyelsaikan masalah *mutual exclusion* pada sistem DSM

I.3. TUJUAN PENULISAN

Tujuan penulisan skripsi ini adalah

1. Mengkaji masalah *mutual exclusion* pada sistem DSM

2. Menyelesaikan masalah *mutual exclusion* pada sistem DSM dengan menggunakan algoritma turnamen

3. Menganalisis kompleksitas running time algoritma turnamen dalam menyelesaikan masalah *mutual exclusion* pada DSM

I.4. BATASAN MASALAH

Dalam penulisan tugas akhir ini masalah dibatasi pada algoritma turnamen untuk menyelesaikan masalah *mutual exclusion* pada sistem terdistribusi memori bersama (DSM).

I.5. ORGANISASI PENULISAN

Sistematika penulisan skripsi ini diorganisasikan sebagai berikut.

Pada Bab II Landasan Teori akan dijelaskan mengenai sistem terdistribusi memori bersama, masalah mutual exclusion, pohon turnamen, kompleksitas algoritma.

Pada Bab III Algoritma Pohon Turnamen akan dipaparkan mengenai presentasi algoritma, skenario kerja algoritma.

Pada Bab IV akan dijelaskan mengenai analisis running time algoritma.

Pada Bab V Penutup akan diberikan kesimpulan dans saran dalam penulisan skripsi ini.

BAB II

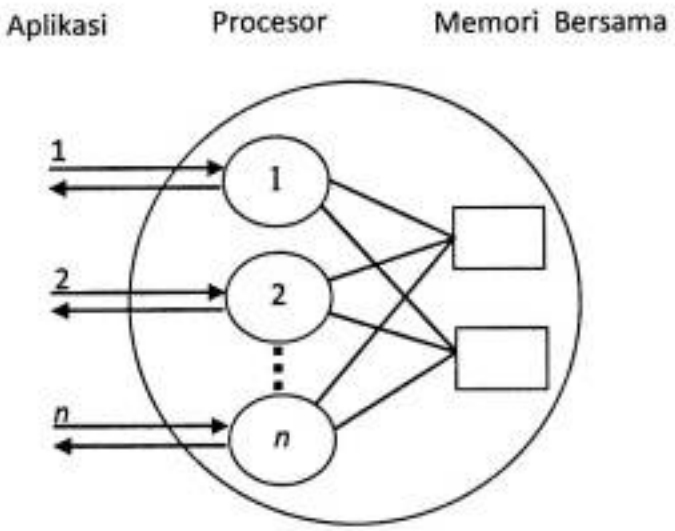
LANDASAN TEORI

II.1. SISTEM TERDISTRIBUSI MEMORI BERSAMA

Sistem terdistribusi memori bersama adalah sistem yang hanya terdiri dari satu memory bersama yang digunakan oleh beberapa prosesor untuk berkomunikasi tanpa adanya pengontrol yang bersifat terpusat. Manfaat utama sistem terdistribusi memori bersama adalah agar sejumlah prosesor dapat bekerja sama untuk memecahkan sebuah masalah komputasi besar (sebagai masalah bersama) sehingga proses komputasi dapat dilakukan lebih cepat. Pada sistem yang melibatkan banyak prosesor dalam mengakses sumber daya bersama selalu membutuhkan mekanisme akses *mutual exclusion.* (Masalah *mutual exclusion* akan dibahas pada sub bab II.2).

Sistem terdistribusi memori bersama dimodelkan sebagai kumpulan beberapa prosesor dan memori bersama, seperti yang di ilustrasikan pada gambar 2.1.1. Ada beberapa proses yang terjadi dalam sistem terdistribusi bersama yang dapat di klasifikasikan sebagai *input*, *output* dan *internal action.* Dalam gambar 2.1.1, anak panah yang menuju prosesor disebut *input* dan anak panah yang meninggalkan prosesor di sebut *output*. Terdapat dua macam *internal action* yaitu; proses yang hanya melibatkan memori bersama dan proses yang melibatkan memori lokal. Jika sebuah proses hanya melibatkan memori bersama, maka dapat diasumsikan bahwa hanya terdapat satu memori bersama. Lain halnya pada system jaringan, pada system

DSM tidak terdapat fungsi pembangkit pesan karena tidak ada pesan pada system ini. Semua komunikasi antara prosesor dilakukan melalui memori bersama (Lynch,1996).



Gambar 2.1.1 Sistem Terdistribusi Memori Bersama

II.2. MUTUAL EXCLUSION

Mutual Exclusion adalah mekanisme mensinkronkan prosesor-prosesor yg berkompetisi dalam mengakses sumber daya yang hanya dapat dipakai oleh paling banyak satu prosesor secara bersamaan pada setiap satu kesatuan waktu. Masalah *mutual exclusion* terjadi ketika prosesor-prosesor yang terdapat dalam sistem saling bersaing untuk mengakses sumber daya yang hanya boleh dipakai oleh paling banyak satu prosesor secara simultan.

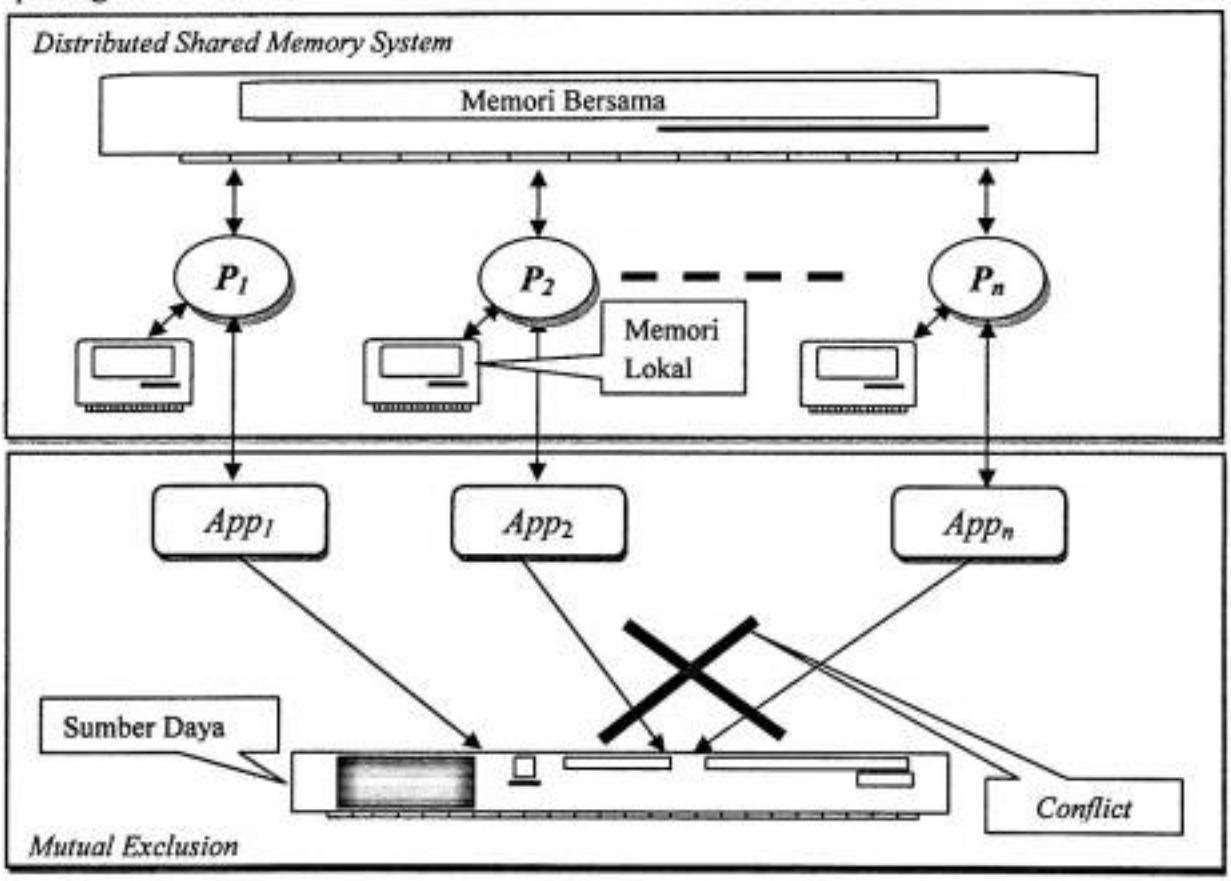
Syarat Mutual Exclusion :

1. *Safety* : paling banyak satu prosesor memiliki izin untuk mengaksesi sumber daya bersama.

2. *Livelock* : semua permintaan untuk mengakses sumber daya akhirnya akan dikabulkan.

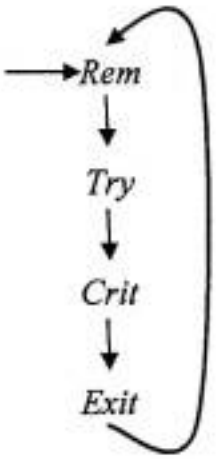
3. *Deadlock* : jika terdapat beberapa prosesor yang ingin mengakses sumber daya dan tidak ada prosesor lain yang mengakses sumber daya tersebut maka satu diantaranya boleh mengakses sumber daya pada waktu tersebut.

Masalah *mutual exclusion* pada sistem distribusi memori bersama terjadi ketika beberapa prosesor (P_i) yang menjalankan aplikasi (AP_i) berbeda, namun aplikasi-aplikasi tersebut mengakses sebuah sumber daya yang sama, seperti yang ditunjukkan pada gambar 2.2.1.



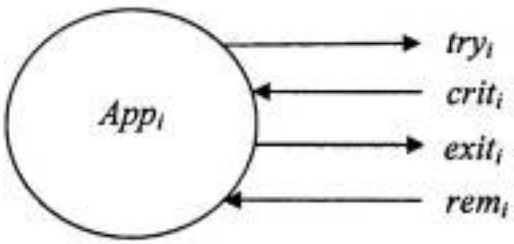
Gambar 2.2.1 Ilustrasi Masalah *Mutual Exclusion*

Algoritma *mutual exclusion* yang digunakan pada sebuah sistem terdistribusi memori bersama dirancang dengan beberapa *state* berikut. Proses yang tidak ingin menggunakan sumber daya bersama disebut *remainder region (R)*. Proses yang ingin menggunakan sumber daya bersama disebut *trying region (T)*. Proses yang menggunakan sumber daya bersama disebut *critical region (C)*. Setelah proses selesai menggunakan sumber daya disebut *exit section (E)*. Abstraksi algoritma *mutual exclusion* ini dapat dipandang sebagai sebuah sirkulasi *state* tanpa henti seperti yang diberikan pada gambar 2.2.2.



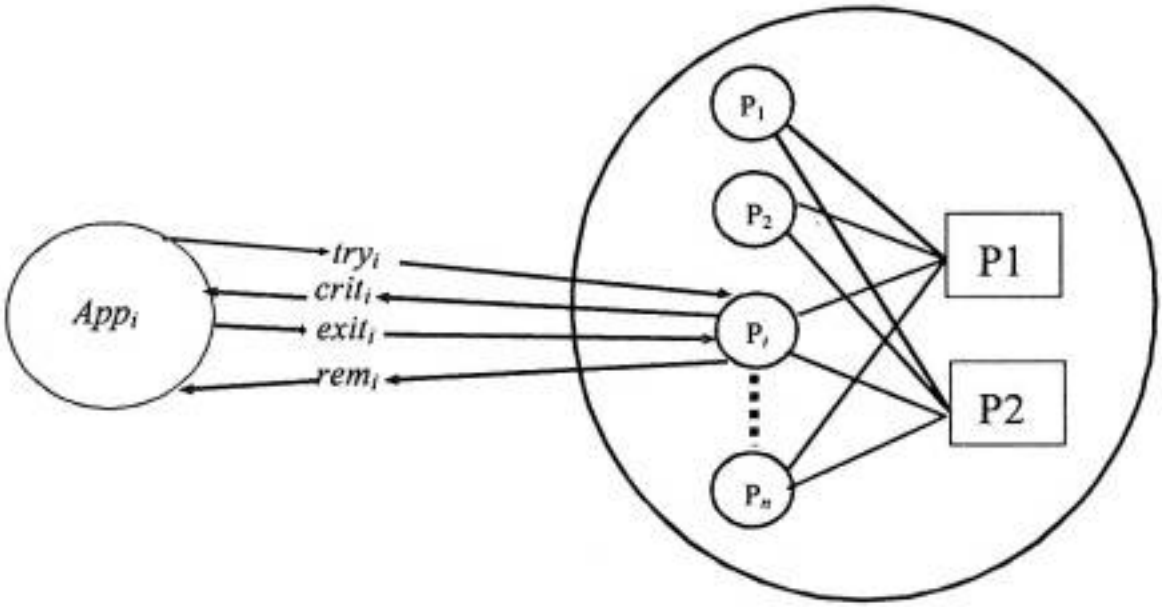
Gambar 2.2.2 Abstraksi Algoritma *Mutual Exclusion*

Sistem DSM terdiri dari n prosesor $\{1,2...n\}$ dan bebeapa n Aplikasi App_i, $1 \leq i \leq n$. Aplikasi App_i yang ingin mengkases sumber daya dapat di ilustrasikan seperti gambar 2.2.3.



Gambar 2.2.3 Skema App_i

Interaksi antara beberapa komponen dalam masalah *mutual exclusion* pada sistem DSM dimana App_i yang mengakses prosesor *Pi* dalam sistem DSM dapat dilihat pada gambar 2.2.4.



Gambar 2.2.4 Interakasi antar komponen dalam *mutual exclusion*

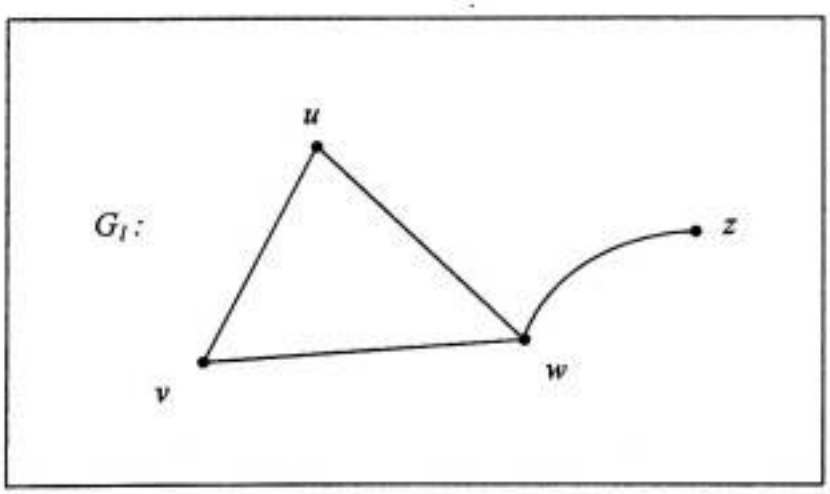
Dalam merancang sebuah algoritma untuk masalah *mutual exclusion* setiap proses memiliki sebuah segmen kode yang disebut *critical region*. *Critical region* inilah yang digunakan prosesor tersebut untuk dapat mengakses sumber daya. Masalah koordinasi eksekusi *critical region* oleh setiap prosesor diselesaikan dengan menyediakan akses secara *mutually exclusive* pada saat masuk ke dalam *critical region* (Lynch,1996).

II.3. POHON TURNAMEN

Graf adalah himpunan yang terdiri dari himpunan *V (Vertex)* yang elemennya disebut simpul (atau point atau node atau titik) dan himpunan *E (Edge)* yang merupakan pasangan tak urut dari simpul, anggotanya disebut ruas (rusuk atau sisi).

Definisi 2.3.1

Graf G adalah pasangan (V(G), X(G)), dimana V(G) adalah himpunan berhingga, yang elemen-elemennya disebut simpul (vertex), dan X(G) adalah himpunan pasangan-pasangan tak berurut dari elemen-elemen V(G) yang berbeda, yang disebut rusuk (edge).



Gambar 2.3.1 Graf G_1

Pada Gambar 2.3.1, $V(G) = \{u, v, w, z\}$ dan $X(G)$ terdiri dari pasangan-pasangan (u,v), (v,w), (u,w), *dan* (w,z), atau $X(G) = \{(u,v), (v,w), (u,w), (w,z)\}$. Sebelum memahami defenisi graf pohon, terlebih dahulu disajikan defenisi lintasan, terhubung, dan siklus.

Definisi 2.3.2

Lintasan (path) P dengan $n \geq 1$ titik adalah graf yang titik-titiknya dapat diurutkan dalam suatu barisan $u_1, u_2, \ldots, u_n$ sedemikian sehingga $E(P) = \{u_i u_{i+1} : i = 1, \ldots, n-1\}$.

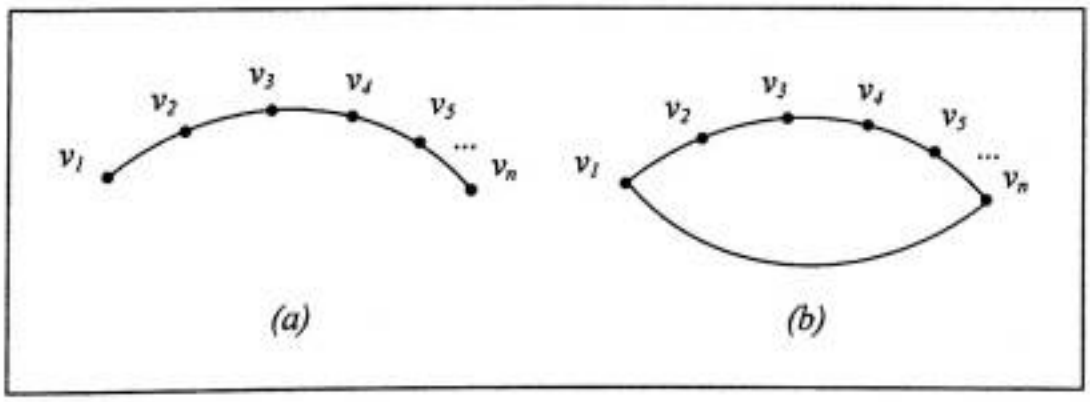
Definisi 2.3.3

Graf G dikatakan terhubung jika untuk setiap dua titik u dan v pada graf tersebut terdapat suatu lintasan yang memuat u dan v.

Definisi 2.3.4

Misalkan $P_n := v_1, v_2, \ldots, v_n$ adalah suatu lintasan berorde n dan $n \geq 3$, Graf $C_n := P_n + \{v_n v_1\}$ disebut siklus berorde n . Panjang suatu lintasan adalah banyaknya sisi pada lintasan tersebut.

Berdasarkan definisi diatas, lintasan P_n mempunyai panjang n-1, dan panjang siklus C_n adalah n.

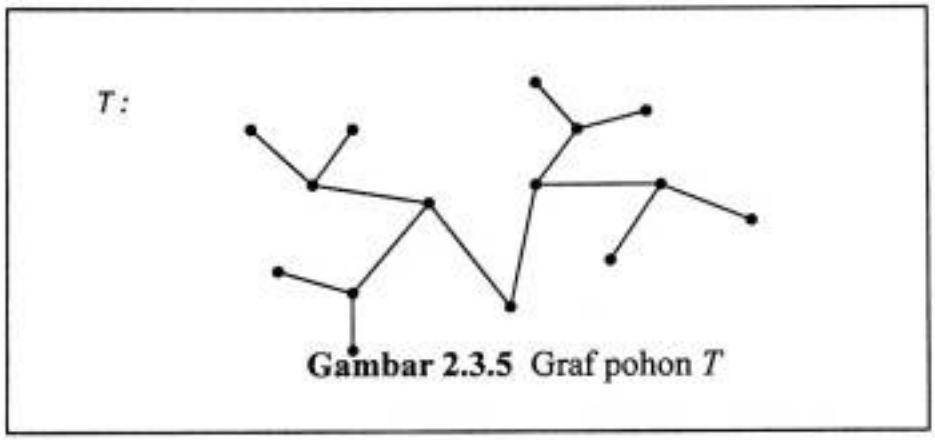


Gambar 2.3.4. (a) Lintasan P_n (b) Siklus C_n

Definisi 2.3.5

Misalkan T adalah graf terhubung. Graf T disebut pohon jika T tidak memuat siklus.

Contoh sebuah graf pohon T diberikan pada Gambar 2.3.5.



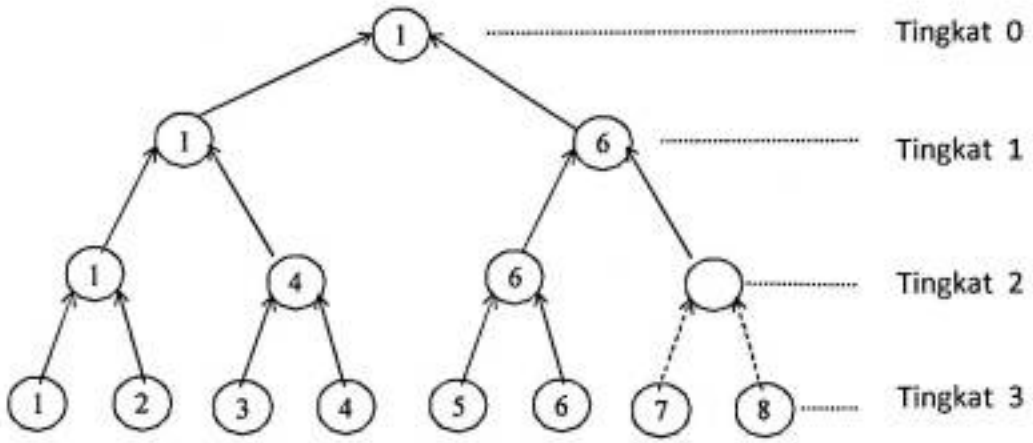
Gambar 2.3.5 Graf pohon T

Graf pohon didefinisikan sebagai suatu graf tak berarah yang tak terhubungkan *(connected undirected graph)* yang tidak mengandung rangkaian sederhana. Graf pohon adalah bentuk khusus dari suatu graf yang banyak diterapkan untuk berbagai keperluan. Pohon biner *(binary tree)* adalah sebuah pohon struktur data dimana setiap simpul memiliki paling banyak dua anak. Secara khusus anaknya dinamakan *kiri* dan *kanan* (Lawi,2008).

Pohon turnamen adalah pohon biner yang sangat efesien untuk menyeleksi proses-proses dengan basis array. Dalam pohon turnamen harus ada yang menang dan yang kalah dimana pada akhirnya akan tersisa satu pemenang Pohon turnamen biasa juga disebut pohon penyeleksian (*selection tree*).

Pohon turnamen yang dimaksud dalam tulisan ini adalah ilustrasi kompetesi berbentuk pohon dari sekumpulan prosesor yang bersaing untuk mengakses sumber daya, dimana pada akhirnya akan ada satu prosesor sebagai pemenang yang lebih dulu mengakses sumber daya tersebut sebelum prosesor-prosesor lainnya. Misalkan

terdapat delapan prosesor dalam DSM dan kompetisi prosesor dalam mengakses sumber daya diberikan pada gambar 2.3.6 (contoh pada gambar 2.3.6 prosesor satu sebagai pemenang sehingga berhak untuk mengakses sumber daya).



Gambar 2.3.6 Turnamen Pohon

Pada gambar 2.3.6 diandaikan terdapat delapan prosesor (1,2,3,4,5,6,7,8) yang ingin megakses sumber daya, Pada Tingkat 3, prosesor P_1 dan P_2 diandaikan P_1 memenangkan persaingan, P_3 dan P_4 diandaikan P_4 memenangkan persaingan, P_5 dan P_6 diandaikan P_6 yang memenangkan persaingan, P_7 dan P_8 (ditandai dengan garis putus-putus) sama-sama tidak mengakses sumber daya. Pada Tingkat 2, Prosesor P_1 dan P_4 diandaikan P_1 memenangakan persaingan, P_6 secara otomatis memenangkan persaingan. Pada Tingkat 1, prosesor P_1 dan P_6 diandaikan P_1 yang memenangkan persaingan. Pada Tingkat 0 tersisa satu prosesor yaitu P_1, berarti P_1 berhak mengakses sumber daya bersama terlebih dahulu.

II.4. KOMPLEKSITAS ALGORITMA

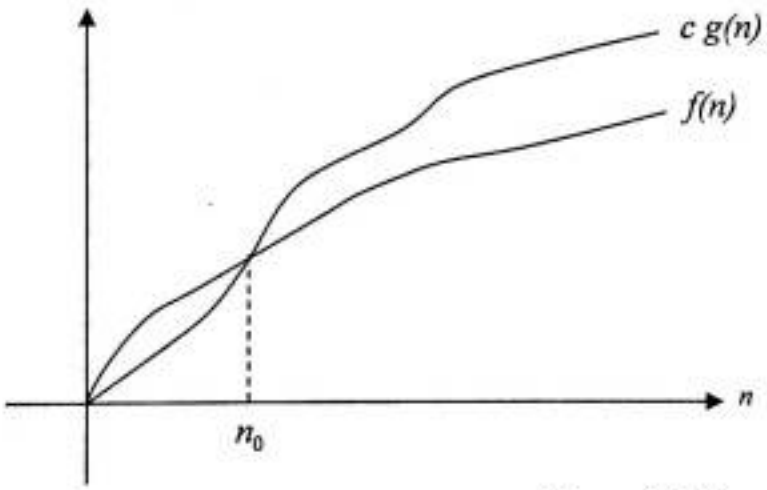
Kompleksitas algoritma adalah besaran yang dipakai untuk menerangkan model abstrak pengukuran waktu dan ruang pada sebuah algoritma. Ada dua macam kompleksitas algoritma, yaitu kompleksitas waktu dan kompleksitas ruang. Kompleksitas waktu, $T(n)$, diukur dari jumlah tahapan komputasi yang dibutuhkan untuk menjalankan algoritma sebagai fungsi dari ukuran masukan n. Kompleksitas ruang, $S(n)$, diukur dari memori yang digunakan oleh struktur data yang terdapat di dalam algoritma sebagai fungsi dari ukuran masukan n. Dengan menggunakan besaran kompleksitas waktu/ruang algoritma, kita dapat menentukan laju peningkatan waktu (ruang) yang diperlukan algoritma dengan meningkatnya ukuran masukan n.

Algoritma dapat dianalisis dari dua sisi yaitu masalah running time yang diperlukan serta besarnya storage/memori yang terlibat. Pada tulisan ini algoritma akan dianalisis dari sisi running time. Untuk menghitung kompleksitas algoritma terdapat beberapa notasi yang saling berhubungan yaitu, O, Ω, Θ. O (baca: *big-O*), waktu untuk kasus terburuk merupakan waktu maksimum yang diperlukan untuk mengeksekusi algoritma dengan input n. Ω (baca: *big - omega*), waktu untuk kasus terbaik merupakan waktu minimum yang diperlukan untuk mengeksekusi algoritma dengan input n. Θ (baca: *big - theta*), waktu untuk kasus rata-rata merupakan waktu rata-rata yang diperlukan untuk mengeksekusi algoritma dengan input n (Cormen, et. al, 2001).

Definisi 2.4.1

Misalkan f dan g adalah dua fungsi dengan domain bilangan asli. Kita tulis $f(n) = O(g(n))$ jika, dan hanya jika, $\exists\ c, n_0 > 0$ *sedemikian sehingga* $0 \leq f(n) \leq cg(n), \forall n \geq n_0$

Interpretasi $f(n) = O(g(n))$ ditunjukkan pada gambar berikut:



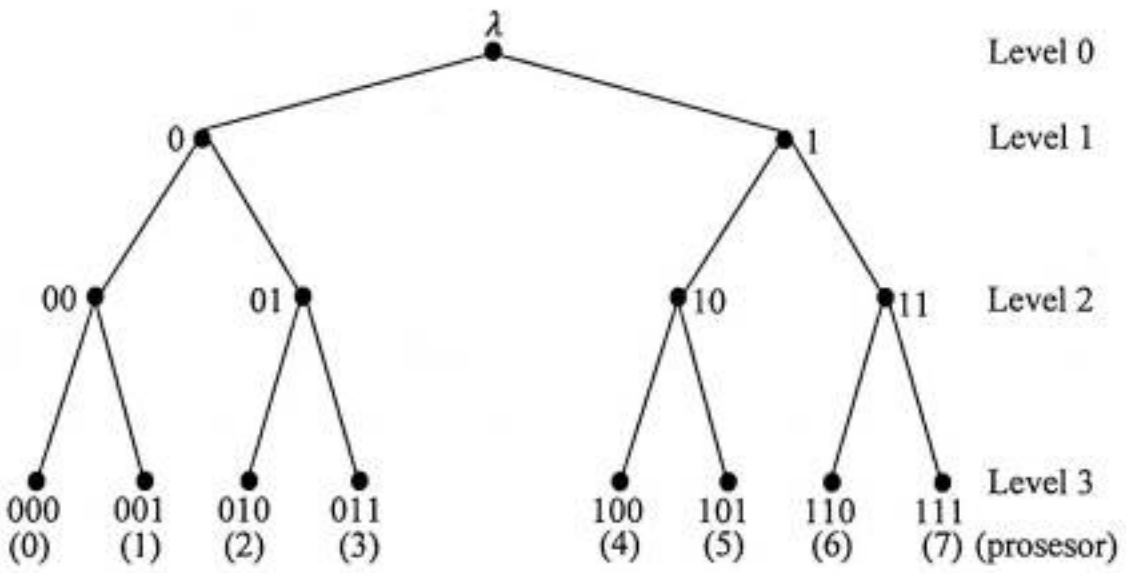
Gambar 2.4.1 Ilustrasi $f(n) = O(g(n))$

BAB III

ALGORITMA POHON TURNAMEN

III.1 PRESENTASE ALGORITMA

Algoritma pohon turnamen untuk masalah mutual exclusion pada sistem terdistribusi memori bersama dilatarbelakangi dari beberapa algoritma Dijkstra dan *n-process* Peterson (*Lynch*, 2003). Algoritma pohon turnamen adalah algoritma pohon biner lengkap (*complete binary tree*) dengan n-prosesor, dimana $n = 2^k$ untuk suatu bilangan asli k. Sebuah ilustrasi pohon turnamen diberikan pada gambar 3.1.1.



Gambar 3.1.1 Pohon Turnamen dengan 8 prosesor

Sebelum mendefenisikan presentase algoritma turnamen yang akan dibahas dalam skripsi ini akan di jelaskan beberapa notasi dan variabel bersama yang digunakan dalam algoritma pohon turnamen dengan n-prosesor.

- *comp(i, k)* : menyatakan moyang prosesor *i* (urutan bit ke-*k* dari kode biner *i*) pada level *k*. Contoh berikut adalah penggunaan notasi *comp* untuk pohon turnamen Gambar 3.1.1.

  Contoh : *comp* (6, 1) = 1

  comp (5, 1) = 1

  comp (2, 1) = 0

- *role* (*i, k*) : *bit* ke-(*k* + *1*) dari kode biner *i* pada level *k* (untuk mengecek apakah simpul daun *i* adalah *anak-kiri* atau *anak-kanan* dari *comp* (*i, k*)). Contoh berikut adalah penggunaan notasi *role* untuk pohon Turnamen Gambar 3.1.1.

  Contoh : *role* (6, 1) = 1

  role (5, 1) = 0

  role (2, 1) = 1

- *opponent* (*i, k*) : prosesor-prosesor yang menjadi lawan *i* pada tingkat kompetisi ke-*k*.. Contoh berikut adalah penggunaan notasi *opponent* untuk pohon Turnamen Gambar 3.1.1.

  Contoh : *opponent* (6, 1) = { 4, 5 } *opponent* (6, 2) = {7}

  opponent (5, 1) = { 6, 7 }

  opponent (2, 1) = { 0, 1 }

- *opposite* (*i, k*) : adalah anak dari *comp* (*i, k*) yang bukan orang tua langsung kode biner *i.*). Contoh berikut adalah penggunaan notasi *opposite* untuk pohon turnamen Gambar 3.1.1.

  Contoh : *opposite* (6, 1) = 10

  opposite (5, 1) = 11

  opposite (2, 1) = 00

III.2 VARIABEL BERSAMA (*SHARED VARIABLE*)

Beberapa variabel bersama (*shared variable*) yang digunakan dalam algoritma turnamen adalah sebagai berikut.

1. Variabel bersama *turn* (*x*) dengan panjang *array* log n – 1.

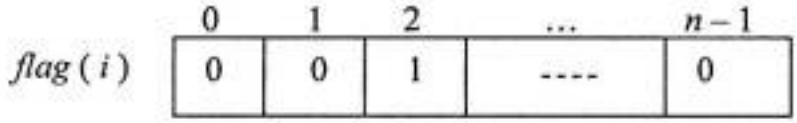
   Turn (*x*) dapat dibaca dan ditulis oleh semua proses. Nilai *turn* (*x*) adalah 0 atau 1, *turn* (*x*) ∈ { 0, 1 }, dimana *x* menyatakan indeks penulisan prosesor.

	0	1	2	...	*log n* – 1
turn (*x*)	1	0	1	----	0

Gambar 3.2.1 Interpretasi variabel bersama *turn* (*x*)

2. Variabel bersama *flag* (*i*) dengan panjang *array n*. Indeks dari *flag* (*i*) dimulai dari 0 sampai n – 1 dengan nilai *flag* (*i*) adalah 0, 1, ..., *log n* atau *flag* (*i*) ∈ { 0, 1, ...*log n* }.

   Variabel bersama *flag* (*i*) hanya dapat ditulis oleh prosesor *i* dan dapat dibaca oleh semua proses *j* (≠ *i*).



Gambar 3.2.2 Interpretasi variabel bersama *flag (i)*

III.3 PROSEDUR *N-TOURNAMENT*

Presentase algoritma turnamen dapat diberikan dengan prosedur berikut.

prosesor *i*:

input actions { *inputs* yang diterima prosesor *i* dari aplikasi App_i }: try_i ,rem_i ;
output actions { *output* yang diberikan prosesor *i* ke aplikasi *Appi* } :$crit_i$, rem_i ;
rem_i;
Remainder region
try_i ;
  for $k = \log n - 1$ **downto** 0
    begin
    flag (*i*) := *k*;
    turn (*comp* (*i* , *k*)) := *role* (*i*, *k*);
    waitfor [∀*j* ∈ *opponents* (*i*, *k*) : *flag* (*j*) > *k*]
          or [*turn* (*comp* (*i*, *k*)) ≠ *role* (*i*, *k*)]
   end ;
$crit_i$;
Critical region
$exit_i$;
  flag (*i*) := log *n*;

Gambar 3.3.1 Presentasi prosedur algorita turnamen

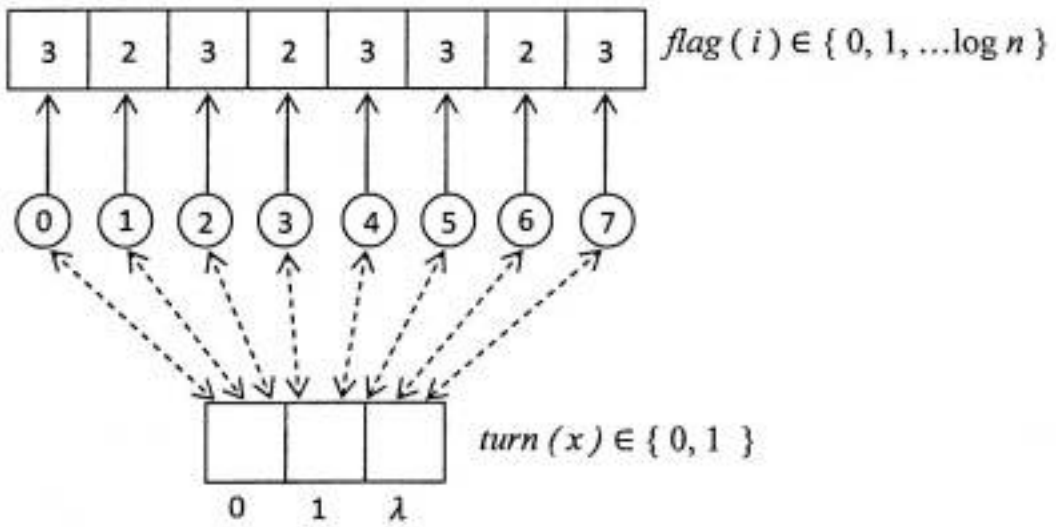
III.4 SKENARIO ALGORITMA

Pada sub bab ini akan diberikan sebuah skenario mengenai cara kerja algoritma turnamen. Misalkan jumlah prosesor $n = 8$ berarti *identifier* (penamaan) prosesor adalah 0, 1, 2, ...,7. Misalkan pada skenario ini prosesor 1,3 dan 6 masing-masing akan mengakses dan bersaing dalam menggunakan sumber daya secara bersamaan.

Skenario pada tiap level dapat diberikan sebagai berikut.

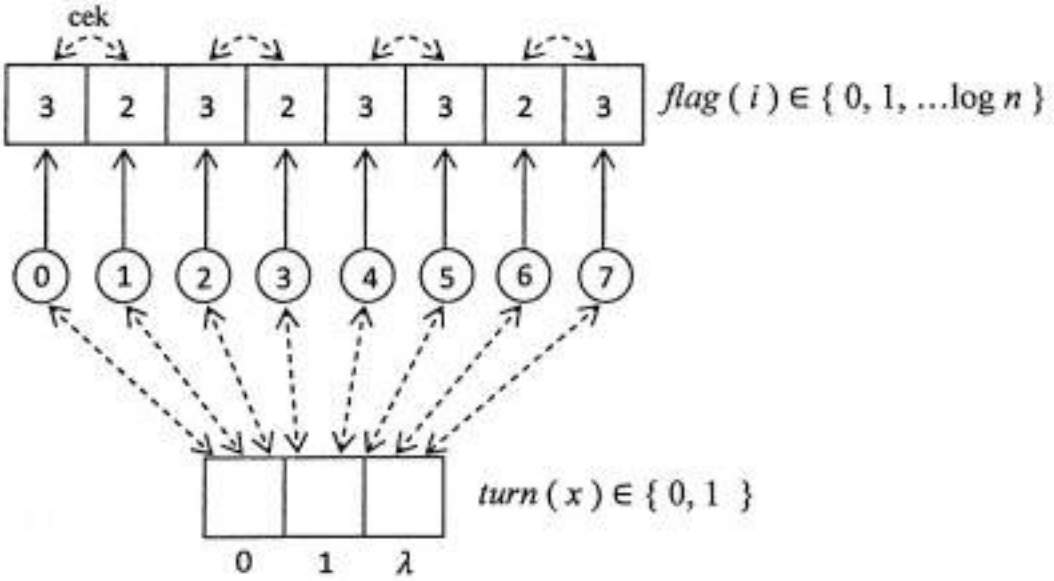
Level 3

1. Prosesor-prosesor menuliskan nilai *flag*nya, karena prosesor 1, 3, dan 6 ingin mengakses sumber daya maka nilai *flag*(1) = 2, *flag*(3) = 2, dan *flag*(6) = 2. Sedangakan nilai *flag* prosesor lainnya adalah 3.



Gambar 3.4.1 Skenario Tahap 1

2. Setelah masing-masing prosesor menuliskan nilai *flag*-nya maka prosesor akan mengecek nilai *flag* prosesor lawannya. Prosesor 1 akan mengecek nilai dari prosesor 0, prosesor 3 akan mengecek nilai dari prosesor 2, dan prosesor 6 akan mengecek nilai dari prosesor 7. Apakah nilai variable *flag* level 3 semua prosesor lawan lebih besar [$\forall j \in opponents\ (i, k) : flag\ (j) > k$] atau apakah indeks pada shared variabel *turn* level 3 tidak sama dengan indeks prosesor [$turn\ (comp\ (i, k)) \neq role\ (i, k)$]

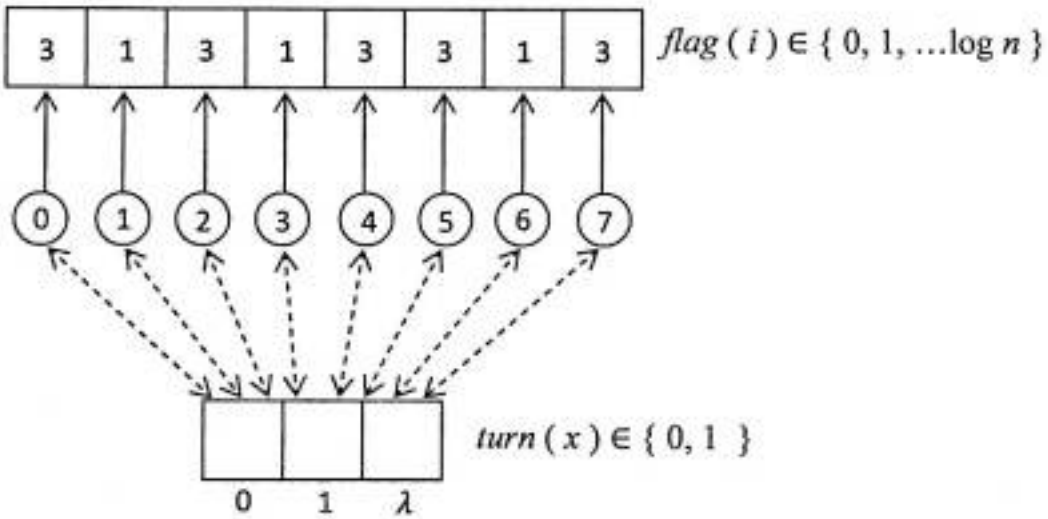


Gambar 3.4.2 Skenario Tahap 2

Karena nilai *flag*(0) > *flag*(1), *flag*(2) > *flag*(3), *flag*(6) < *flag*(7), maka prosesor 1,3, dan 6 memasuki level ke-2.

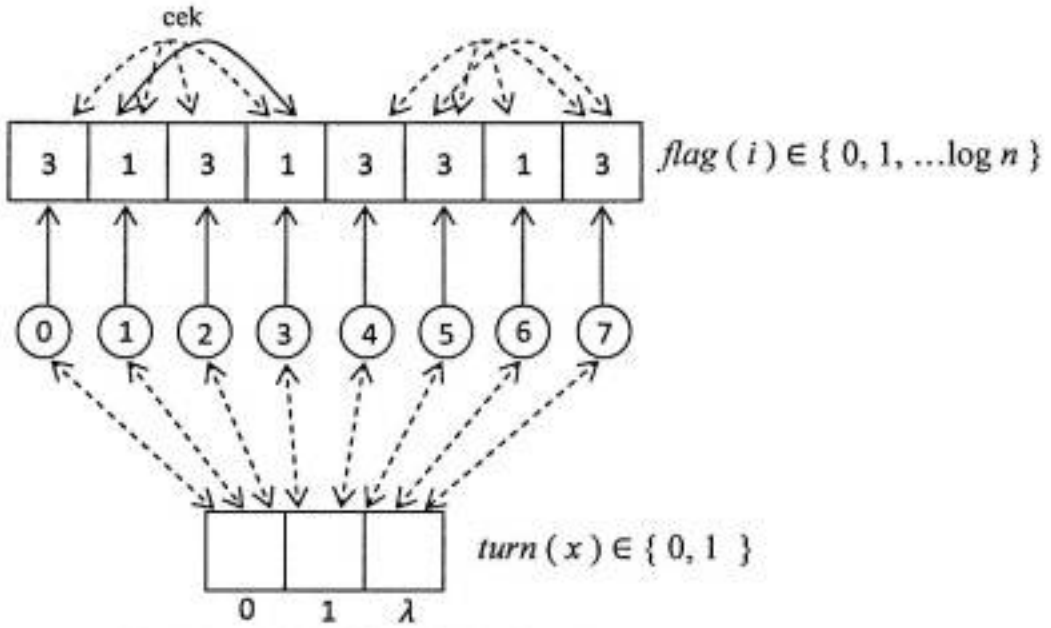
Level 2

3. Prosesor-prosesor menulis ulang nilai *flag*nya, karena prosesor 1, 3, dan 6 telah masuk pada level 2 maka nilai *flag*(1) = 1, *flag*(3) = 1, dan *flag*(6) = 1. Sedangakan nilai *flag* prosesor lainnya adalah 3.



Gambar 3.4.3 Skenario Tahap 3

4. Setelah masing-masing prosesor menuliskan nilai *flag*-nya maka prosesor akan mengecek nilai *flag* prosesor lawannya. Prosesor 1 akan mengecek nilai dari prosesor 0, 2 dan 3. Prosesor 3 akan mengecek nilai dari prosesor 2, 1 dan 0. Prosesor 6 akan mengecek nilai dari prosesor 7, 5 dan 4. Apakah nilai variable *flag* level 2 semua prosesor lawan lebih besar [$\forall j \in opponents\ (i, k) : flag\ (j) > k$] atau apakah indeks pada shared variabel *turn* level 2 tidak sama dengan indeks prosesor [$turn\ (comp\ (i, k)) \neq role\ (i, k)$]

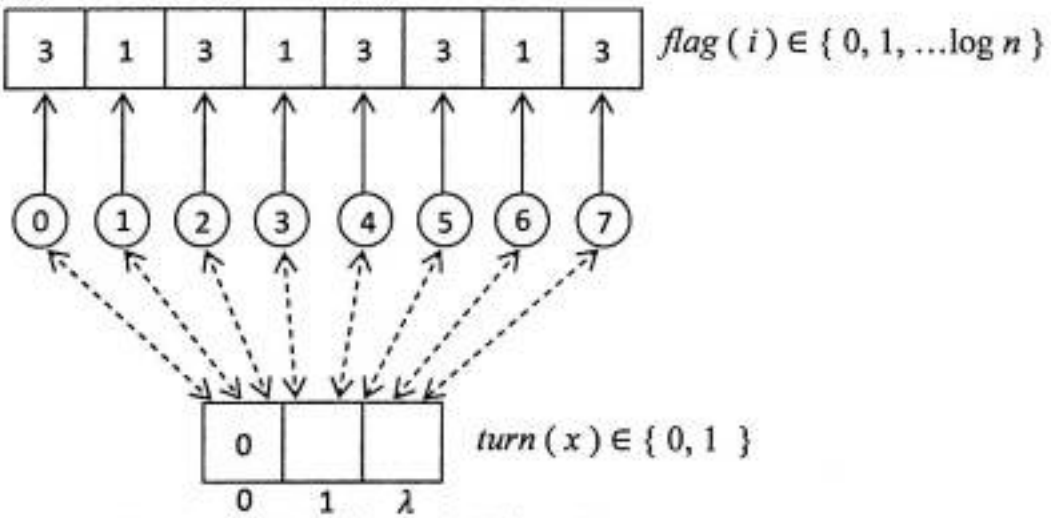


Gambar 3.4.4 Skenario Tahap 4

Nilai *flag*(0) > *flag*(3), *flag* (1) < *flag*(2), *flag*(1) = *flag*(3) karena nilai *flag*(1) = *flag*(3) maka pemenangnya akan ditentukan dari nilai variable *turn* pada level 2.

Nilai *flag*(6) < *flag*(4), dan *flag*(6) < *flag*(5), maka prosesor 6 memasuki level 1.

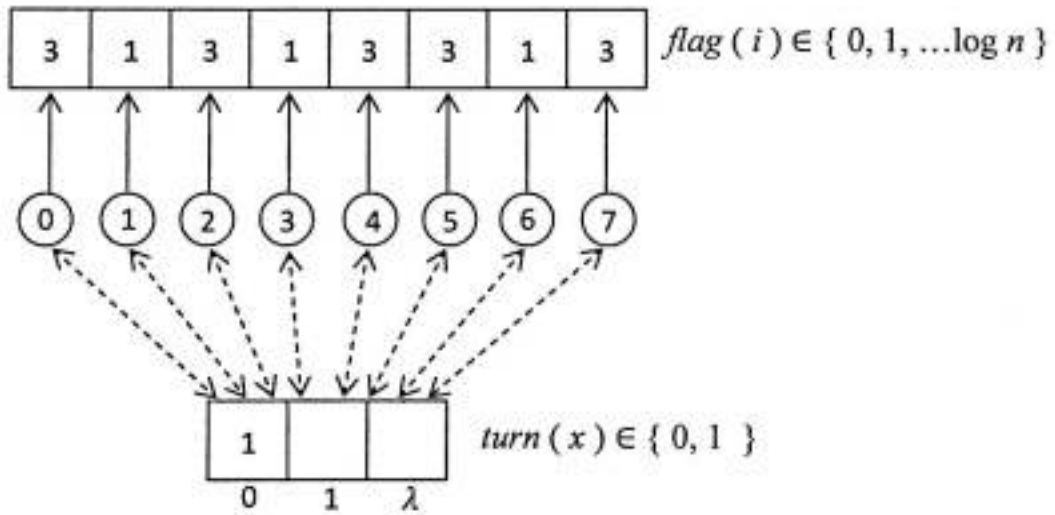
5. Prosesor 1 menuliskan indeksnya pada shared variabel *turn* level 2. Berarti *turn* ((*comp*(1,1)) = *role* (1,1) jadi *turn*(0) = 0



Gambar 3.4.5 Skenario Tahap 5

Prosesor 1 memeriksa ulang *flag* prosesor lawan, [$\forall j \in$ opponents (i,k): $flag(j)>k$] atau indeks pada shared variabel *turn* level 2 tidak sama dengan indeks prosesor 1 [$turn\ (comp(i,k)) \neq role\ (i,k)$]. Karena kedua syarat diatas tidak terpenuhi maka prosesor 1 menunggu.

6. Prosesor 3 menuliskan indeksnya pada shared variabel *turn* level 2. Berarti *turn* ($(comp(3,1)) = role\ (3,1)$ jadi $turn(0) = 1$.

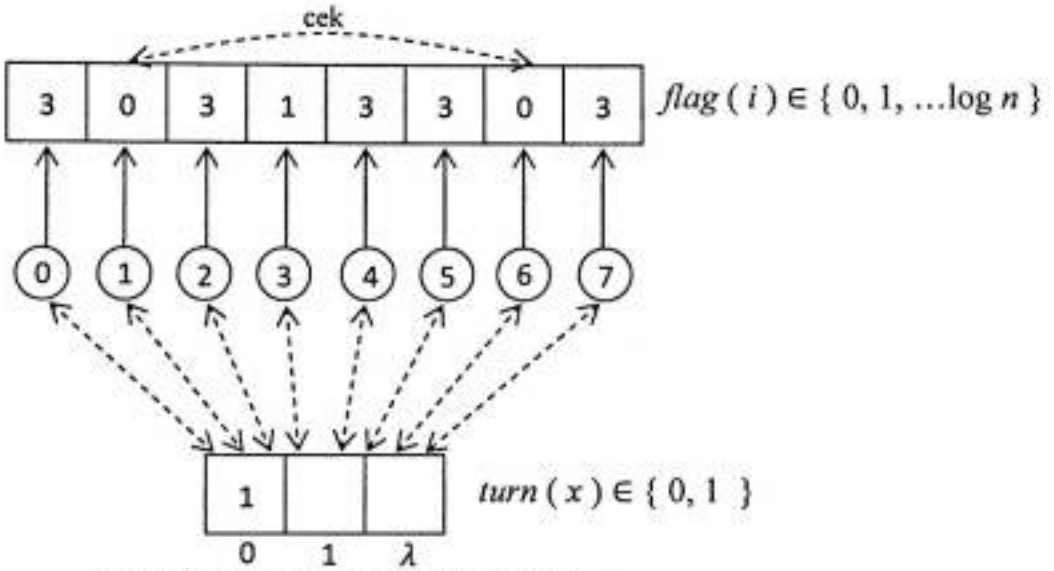


Gambar 3.4.6 Skenario Tahap 6

7. Karena prosesor 3 telah menuliskan indeksnya pada shared variabel *turn* level 2 maka syarat $turn(0) \neq 1$ terpenuhi untuk prosesor 1. Berarti prosesor 1 terdorong ke level 1.

Level 1

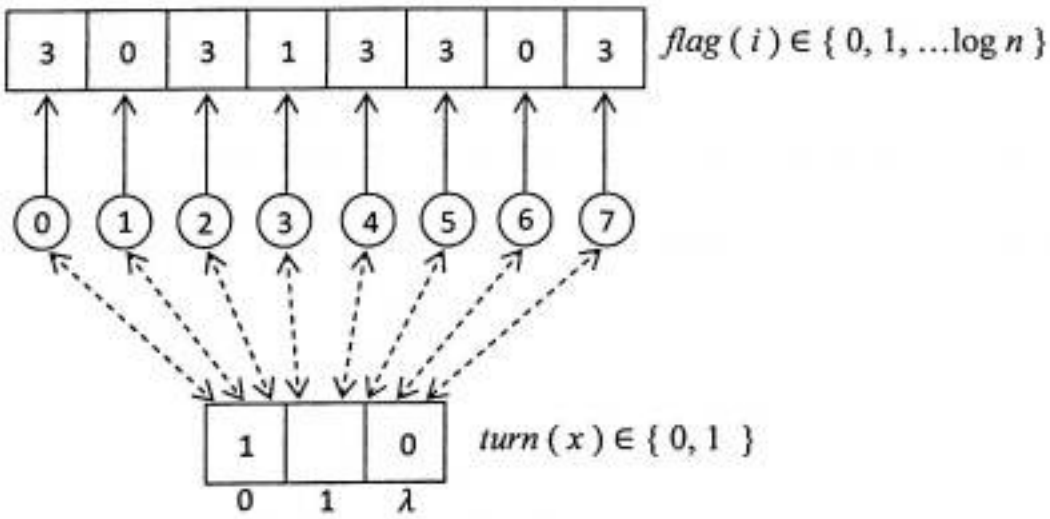
8. Prosesor-prosesor menulis ulang nilai *flag*nya, karena prosesor 1 dan 6 telah masuk pada level 1 maka nilai $flag(1) = 0$ dan $flag(6) = 0$.



Gambar 3.4.7 Skenario Tahap 8

Setelah masing-masing prosesor menuliskan nilai *flag*-nya maka prosesor akan mengecek nilai *flag* prosesor lawannya di level 1. Prosesor 1 akan mengecek nilai dari prosesor 6, begitupun sebaliknya. Karena nilai *flag*(1) = *flag* (6) maka akan ditinaju pada variabel *turn* level 1.

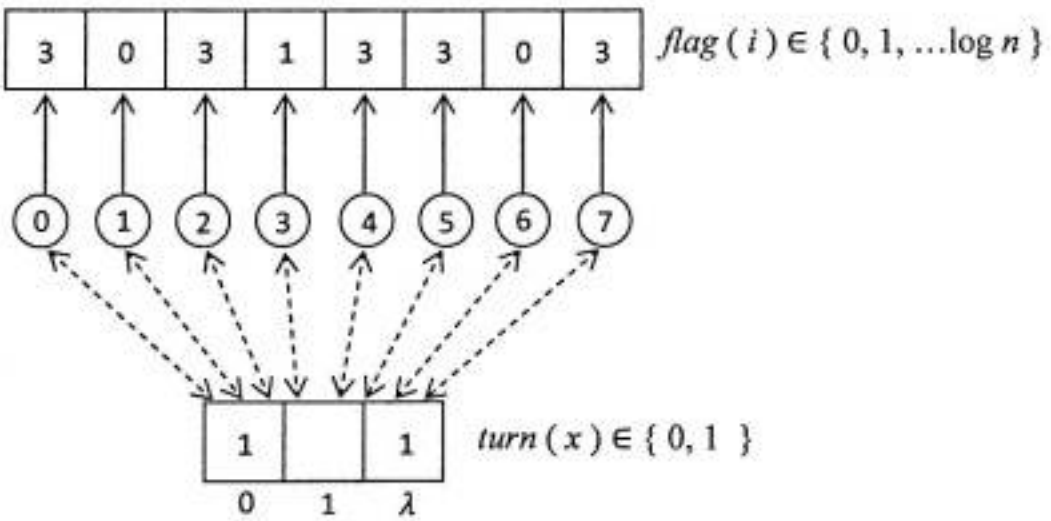
9. Prosesor 1 menuliskan indeksnya pada shared variabel *turn* level 1. Berarti *turn* ((*comp*(1,0)) = *role* (1,0) jadi *turn*(λ) = 0



Gambar 3.4.8 Skenario Tahap 9

Prosesor 1 memeriksa ulang *flag* prosesor lawan, [$\forall j \in opponents\ (i,k)$:*flag* $(j)>k$] atau indeks pada shared variabel *turn* level 1 tidak sama dengan indeks prosesor 1 [*turn* $(comp(i,k)) \neq role\ (i,k)$]. Karena kedua syarat diatas tidak terpenuhi maka prosesor 1 menunggu.

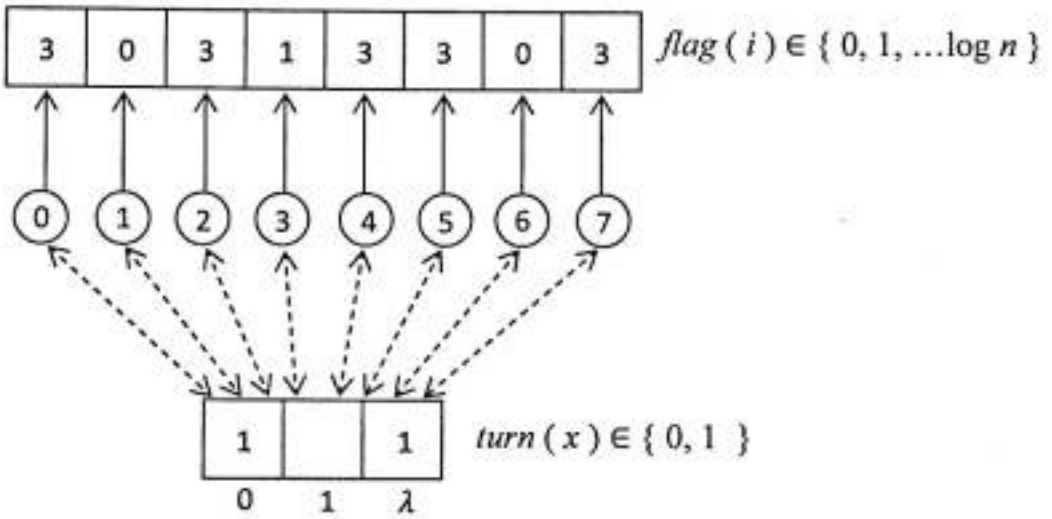
10. Prosesor 6 menuliskan indeksnya pada shared variabel *turn* level 1. Berarti *turn* $((comp(6,0)) = role\ (6,0)$ jadi $turn(\lambda) = 1$



Gambar 3.4.9 Skenario Tahap 10

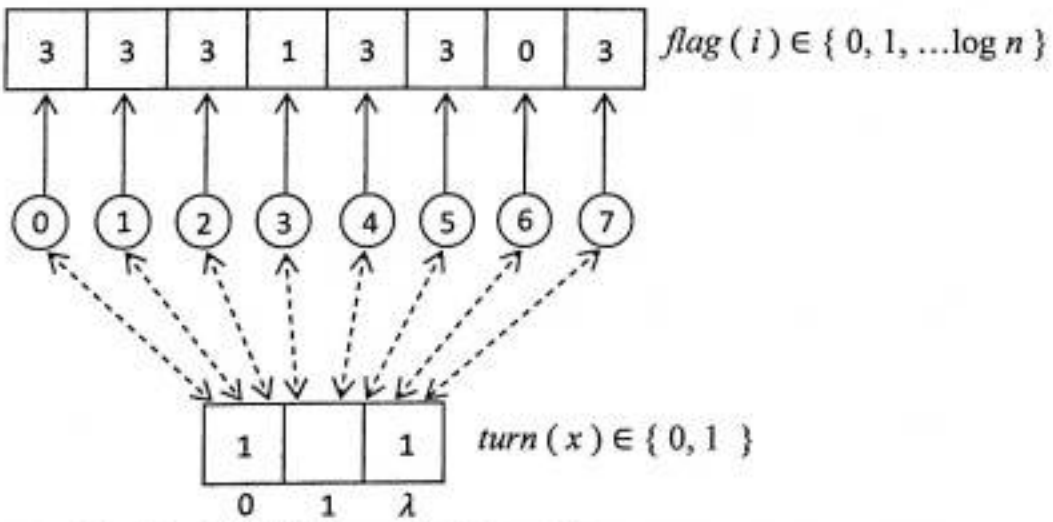
Level 0 (*Critical Section*)

11. Karena prosesor 6 menuliskan indeksnya pada shared variabel *turn* level 1 maka syarat $turn(\lambda) \neq 1$ terpenuhi untuk prosesor 1. Berarti prosesor 1 berhak mengakses *critical section* terlebih dahulu.



Gambar 3.4.10 Skenario Tahap 11

12. Setelah prosesor 1 selesai mengakses critical region maka prosesor 1 akan mengakses exit section dan menuliskan nilai *flag*-nya, *flag*(1)=3 untuk memberikan informasi pada prosesor lain bahwa telah selesai mengakses daerah kritisnya.



Gambar 3.4.11 Skenario Tahap 12

13. Prosesor 1 mengakses *remainder region*

BAB IV

ANALISIS RUNNING TIME ALGORITMA

IV.1 BUKTI KEBENARAN ALGORITMA POHON TURNAMEN

Sebelum memberikan teorema utama untuk bukti kebenaran algoritma turnamen berikut diberikan dua *lemma* yang menjadi landasan bukti untuk teorema utama.

Lemma IV.1.1 : *Dalam sistem algoritma pohon turnamen, terdapat beberapa level-k, $1 \leq k \leq \log n$, dimana paling banyak satu prosesor yang memenangkan kompetisi dari tiap sub-pohon (subtree) di level-k.* ♦

Lemma IV.1.2 : *Jika prosesor i adalah pemenang pada level k dan jika setiap prosesor lain adalah pesaing pada level k, maka* $[turn(comp(i,k)) \neq role\ (i,k)\ ]$. ♦

Bukti dari kedua lemma ini dapat di pahami langsung dari skenario yang di berikan pada Bab III.

Dari Lemma IV.1.1 dan Lemma IV.1.2 dapat diberikan teorema berikut.

Teorema IV.1.3 : *Algoritma turnamen memenuhi syarat Mutual Exclusion*

1. *Safety* : paling banyak satu prosesor yang mengakses *critical section.*
2. *Livelock* : semua permintaan untuk mengakses *critical section* akhirnya akan dikabulkan.

3. *Deadlock* : jika terdapat beberapa prosesor yang ingin mengakses *critical section* dan tidak ada prosesor lain yang mengakses *critical section* maka satu diantaranya boleh mengakses *critical section* pada waktu tersebut.

Bukti.

Dari hasil lemma IV.1.1 dan lemma IV.1.2.◆

IV.2 KOMPLEKSITAS RUNNING TIME

Teorema IV.2.1 :

Pada algoritma turnamen, waktu ketika prosesor i memasuki trying protocol sampai memasuki daerah kritis adalah paling banyak $(n-1)c + O(n^2\ell)$.

Bukti.

Misalkan $T(0)$ adalah waktu maksimum sejak prosesor memasuki trying protocol hingga memasuki daerah kritis. Misalkan $T(k)$ adalah waktu maksimum sejak prosesor menjadi pemenang pada level k sampai memasuki daerah kritis, untuk $0 \le k \le \log n - 1$, yang akan dicari adalah batas atas $T(0)$.

Berdasarkan prosedur algoritma diketahui $T(\log n) \le \ell$, karena hanya dibutuhkan satu langkah untuk memasuki daerah kritis setelah memenangkan

kompetisi akhir. Dalam membatasi $T(0)$, atur keadaan rekurensi $T(k)$ dalam ekspresi $T(k+1)$, dimana $0 \le k \le \log n - 1$.

Andaikan prosesor i baru saja sebagai pemenang pada level k jika $k \ge 1$, atau baru saja memasuki *trying protocol* jika $k = 0$ .Misalkan $x = comp(i, k+1)$. Kemudian dengan waktu 2ℓ, prosesor i mengatur variablel $turn(x)$, dan mengatur $role(i, k+1)$. Misalkan π menyatakan kejadian $set - turn(x)$. Akan dipertimbangkan dua kasus berikut:

Pertama, jika $turn(x)$ diatur ke beberapa nilai selain i dengan waktu $T(k+1)+c+(2^{k+1}+4)\ell$ setelah π, maka i adalah pemenang pada level $k+1$ dengan waktu tambahan$(2^k + 1)\ell$. Kemudian dengan waktu tambahan $T(k+1)$, i memasuki daerah kritis. Dalam kasus ini, total waktu dari π sampai i memasuki daerah kritis adalah paling banyak $2T(k+1)+c+(2^{k+1}+2^k+5)\ell$.

Kedua, pada kasus lain, asumsikan bahwa $turn(x)$ tidak diatur ke suatu nilai selain i dengan waktu $T(k+1)+c+(2^{k+1}+4)\ell$ setelah π. Maka tidak ada prosesor yang bisa mengatur *flag*nya ke $k+1$ dengan waktu $T(k+1)+c+(2^{k+1}+3)\ell$ setelah π. Misal prosesor j adalah saingan prosesor i pada level $k+1$ dengan $flag(j) \ge k+1$ jika π terjadi. Maka dalam waktu $(2k+1)\ell + T(k+1)+c+\ell = T(k+1)+c+(2^k+2)\ell$ setelah π, semua prosesor j mengatur nilai *flag*nya manjadi 0.

Jadi, dengan waktu $T(k+1)+c+(2^k+2)\ell$ setelah π, semua prosesor j (saingan dari prosesor i) pada level $k+1$ dimana $flag(j) \geq k+1$ jika π terjadi, mengatur nilai *flag*nya menjadi 0. Seperti yang telah diasumsikan sebelumnya, dengan tambahan waktu $(2^k+1)\ell$ setelah itu tidak ada prosesor yang mengatur nilai *flag*nya menjadi $k+1$. Dalam hal ini waktu yang cukup bagi prosesor i untuk mendeteksi bahwa semua nilai variabel *flag* adalah lebih kecil dari $k+1$ dan demikian juga prosesor i untuk menjadi pemenang pada level $k+1$. Dalam kasus ini, prosesor i sebagai pemenang pada level $k+1$ dengan waktu $T(k+1)+c+(2^{k+1}+3)\ell$ setelah π. Kemudian lagi, dengan waktu lainnya $T(k+1)$, prosesor i memasuki daerah kritis. Dalam kasus ini, waktu total sejak π sampai prosesor i memasuki daerah kritis adalah paling banyak $2T(k+1)+c+(2^{k+1}+3)\ell$.

Waktu kasus terburuk adalah paling banyak 2ℓ ditambah maksimum waktu pada dua kasus diatas , yaitu $2T(k+1)+c+(2^{k+1}+2^k+7)\ell$. Jadi, diperoleh fungsi rekursif berikut:

$$T(k) \leq 2T(k+1)+c+(2^{k+1}+2^k+7)\ell, \text{untuk } 0 \leq k \leq \log n-1$$

$$T(n-1) \leq \ell$$

Berikut ini penyelesaian fungsi rekursif diatas

Untuk beberapa nilai konstan a, sedemikian sehingga $(2^{k+1}+2^k+7) \leq a.2^k$

Maka

$$T(0) \le 2T(1) + 2^0 c + a2^0 \ell$$

$$\le 2^2 T(2) + (2^0 + 2^1)c + a(2^0 + 2^2)\ell$$

$$\le 2^3 T(3) + (2^0 + 2^1 + 2^2)c + a(2^0 + 2^2 + 2^4)\ell$$

$$\vdots$$

$$\le 2^k T(k) + (2^0 + 2^2 + \ldots + 2^{k-1})c + a(2^0 + 2^2 + \ldots + 2^{2k-2})\ell$$

$$\vdots$$

$$\le 2^k T(k) + (2^0 + 2^2 + \ldots + 2^{k-1})c + a(2^0 + 2^2 + \ldots + 2^{2k-2})\ell$$

$$\le 2^{\log n} T(\log n) + (2^0 + 2^2 + \ldots + 2^{\log n - 1})c + a(2^0 + 2^2 + \ldots + 2^{(\log n - 1)})\ell$$

$$\le (n-1)c + n\ell + O(n^2\ell)$$

$$= (n-1)c + O(n^2\ell) \blacklozenge$$

BAB V

PENUTUP

V.1 KESIMPULAN

Berdasarkan hasil analisis kami pada Bab IV, mengenai algoritma turnamen untuk masalah mutual exclusion pada sistem terdistribusi memori bersama dapat disimpulkan bahwa

- Algoritma turnamen menyelesaikan masalah *mutual exclusion* untuk n proses.
- Kompleksitas running time algoritma turnamen adalah $(n-1)c + O(n^2\ell)$.

V.2 SARAN

Untuk penelitian lebih lanjut, kajian mengenai modifikasi algoritma turnamen adalah menarik untuk dilakukan.

DAFTAR PUSTAKA

- Lester, Bruce P., 1993. *The Art of Parallel Programming.* NJ, Prentice-Hall.
- Lynch, Nancy A., 1996. *Distributed Algorthms.* Morgan Kaufman Publishers,Inc. San Fransisco, California.
- Razak, Maryuni., *Analisa Algoritma Mutual Exclusion Peterson pada Distributed Shared Memory System.* Universitas Hasanuddin, Makassar.
- Lawi, A., 2008. *Catatan Mata Kuliah Matematika Diskrit.* Universitas Hasanuddin, Makassar.
- Cormen, Thomas H., Charles E., 2001. *Introduction to Algorithms , 2nd edition.* McGraw-Hill, New York.