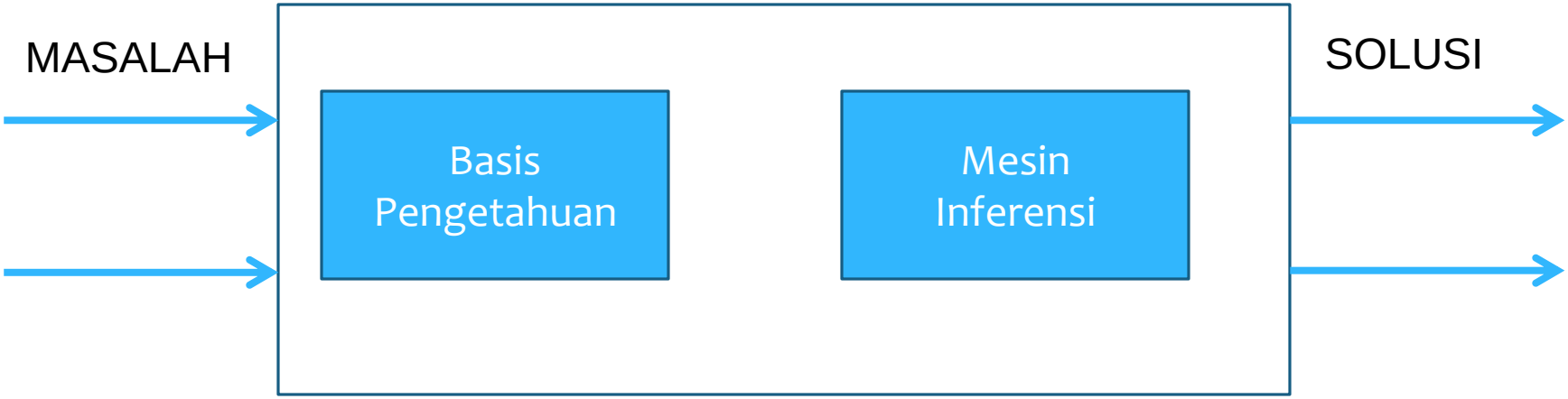
Kecerdasan Buatan

Pertemuan 2: Problem Solving

Sistem yang menggunakan AI



Sistem yang menggunakan AI (2)

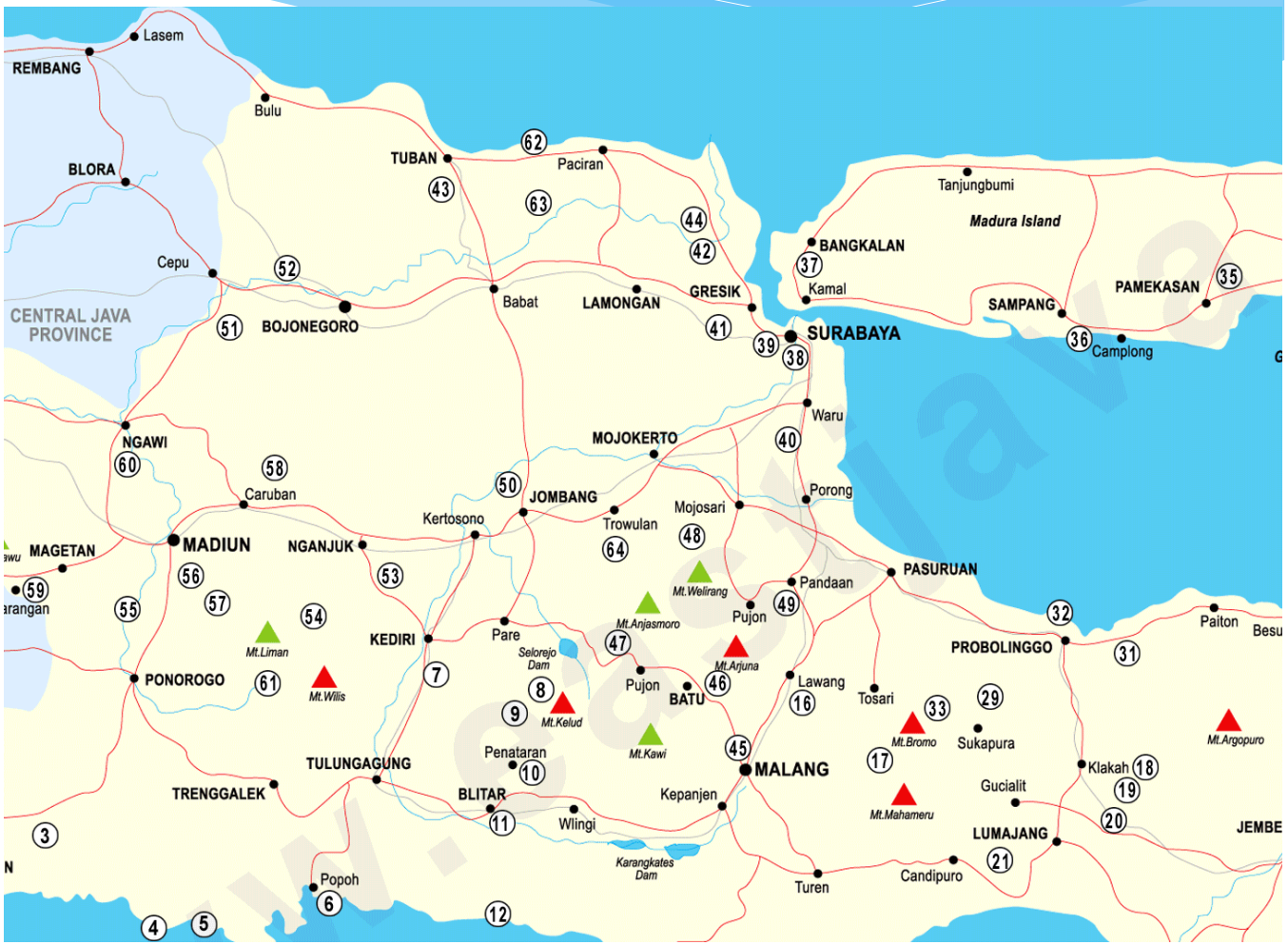
* Empat hal yang perlu dipertimbangkan untuk membangun sistem yang mampu menyelesaikan masalah:
  * Mendefinisikan masalah dengan tepat → spesifikasi tentang keadaan awal (initial state) dan solusi yang diharapkan
  * Menganalisis masalah dan mencari teknik penyelesaian masalah yang sesuai
  * Merepresentasikan pengetahuan
  * Memilih teknik penyelesaian masalah yang terbaik

Penyelesaian Masalah (Problem Solving)

* Beberapa istilah/terminologi:
  * **State** (Keadaan): Deskripsi dari dunia sekitar
  * **Goal** (Tujuan): Sekumpulan state yang memiliki sifat/properti yang diinginkan
  * **Action** (Tindakan): Menyebabkan perubahan dari sebuah state ke state lain
* Mencari **action mana yang menuju ke state goal**: ini adalah langkah untuk penyelesaian masalah:
  * Pendefinisian masalah (problem formulation)
    * State awal (*initial state*), *action* dan *state* apa yang perlu dipertimbangkan (disebut *successor function*), *goal test*, dan *biaya (path costs)*
  * Menjalankan algoritma pencarian (*searching algorithm*)
    * Memeriksa beberapa urutan *action* yang dapat dijalankan
    * Mengembalikan sebuah solusi dalam bentuk urutan *action*

Contoh: Liburan di Jawa Timur (1)

Posisi di kota Trenggalek. Esok harus di Surabaya untuk naik pesawat ke Singapura



Liburan di Jawa Timur (2)

* Formulasi tujuan/pendefinisian masalah:
  * Berada di Surabaya
* Formulasi permasalahan:
  * State (kata benda) : kota-kota di Jawa Timur
  * Action (kata kerja): Berkendara melalui kota-kota
* Hasil pencarian final:
  * Urutan kota yang dilalui, misal: Trenggalek, Tulungagung, Kediri, Kertosono, Jombang, Mojokerto, Surabaya

Pendefinisian Masalah

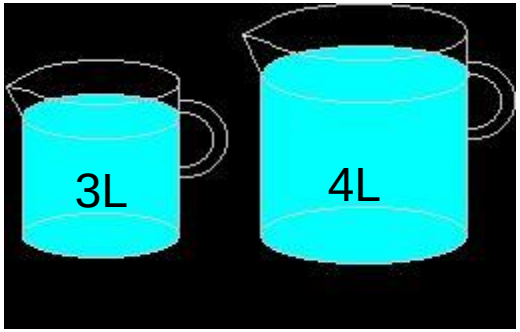
* Permasalahan didefinisikan dalam 4 hal:
  * **State awal**
  * **Fungsi successor** (action yang akan dilakukan dan state berikutnya yang akan dituju)
    * Diberikan sebuah state s, *A(s)* adalah fungsi yang mengembalikan state-state lain yang dapat dicapai dari state s dengan 1 kali action.
  * **Goal test** (menentukan apakah state sekarang adalah state tujuan)
  * **Path cost** (jumlah dari seluruh biaya yang ditimbulkan oleh action yang telah dilakukan)

Contoh Pendefinisian Masalah pada studi kasus Liburan di Jawa Timur

* State
  * Kota di Jawa Timur: {“Trenggalek”, “Surabaya”, “Jember”, “Malang”, … }
* State Awal
  * “Trenggalek”
* Action dan state yang dipertimbangkan, sesuai fungsi successor
  * Misal: A(Trenggalek) = {<Trenggalek → T.Agung, T.Agung >, … }
* Goal Test
  * Misal: x = “Surabaya”
* Path Cost
  * Misal: jumlah jarak tempuh, jumlah kota yang dilewati, dll.

Permasalahan Teko Air

* Terdapat dua teko air tanpa label yang masing-masing berukuran 3L dan 4L.
* Tersedia pasokan air yang tak terbatas.
* Teko air tersebut dapat diisi air, dikosongkan, atau isinya dituang ke teko yang lainnya.
* Tujuan akhirnya adalah mendapatkan 2L air.



Permasalahan Teko Air (2)

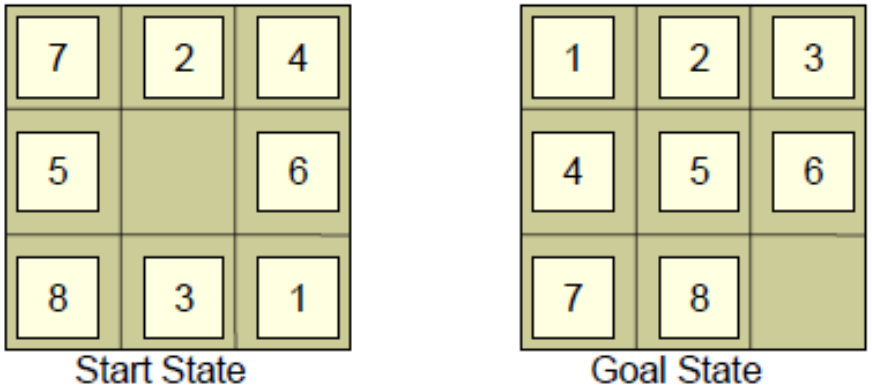
* Representasi: (x,y) = (vol air di teko 3L, vol air di teko 4L), $0 \leq x \leq 3, 0 \leq y \leq 4$
* State awal: (0, 0)
* Goal state: (_, 2) atau (2, _)
* Tuliskan urutan langkah-langkahnya untuk setiap statenya sehingga tercapai goal state.

Action Teko Air

(x,y) = (vol air di teko 3L, vol air di teko 4L)

Action	Batasan	Keterangan
A1(x,y) → (3,y)		Isi penuh teko 3L
A2(x,y) → (x,4)		Isi penuh teko 4L
A3(x,y) → (0,y)		Kosongkan teko 3L
A4(x,y) → (x,0)		Kosongkan teko 4L
A5(x,y) → (0,x+y)	$[0 \le x+y \le 4]$	Tuang semua isi teko 3L ke teko 4L
A6(x,y) → (x+y,0)	$[0 \le x+y \le 3]$	Tuang semua isi teko 4L ke teko 3L
A7(x,y) →(x+y-4,4)	$[x+y>4]$	Penuhi teko 4L dengan isi teko 3L
A8(x,y) →(3,x+y-3)	$[x+y>3]$	Penuhi teko 3L dengan isi teko 4L

Contoh: Permasalahan 8-Puzzle



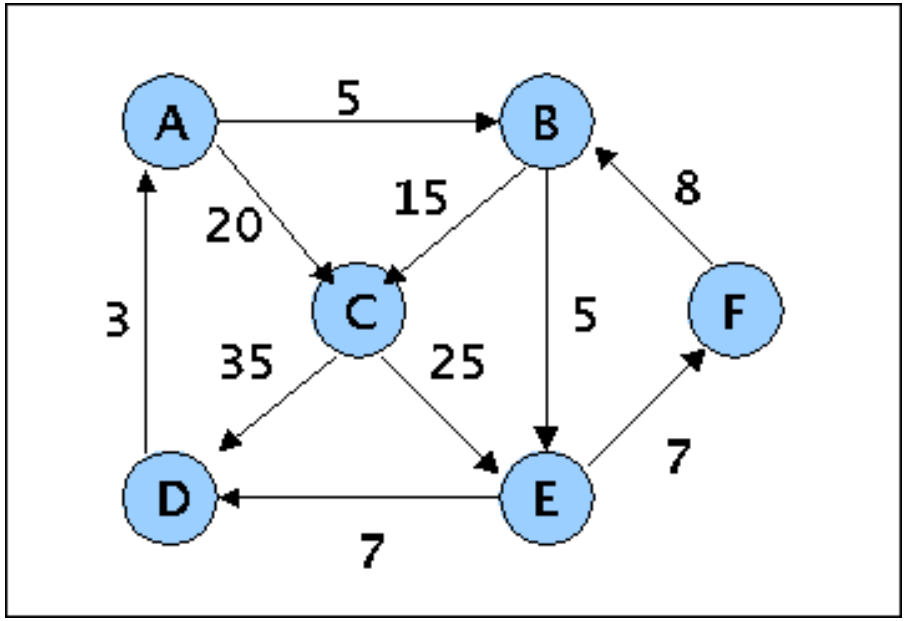
* **State:** posisi dari setiap 8 keping dan keping kosong
* **State awal:** posisi sembarang
* **Fungsi successor:** State legal/sah yang dihasilkan dari action (pergerakan keping kosong ke *Kiri, Kanan, Atas,* dan *Bawah*)
* **Goal state:** State sekarang sama dengan state di sebelah kanan
* **Path cost:** Setiap langkah bernilai 1, path cost adalah jumlah seluruh langkah yang ditempuh

Ruang Keadaan (State Space)

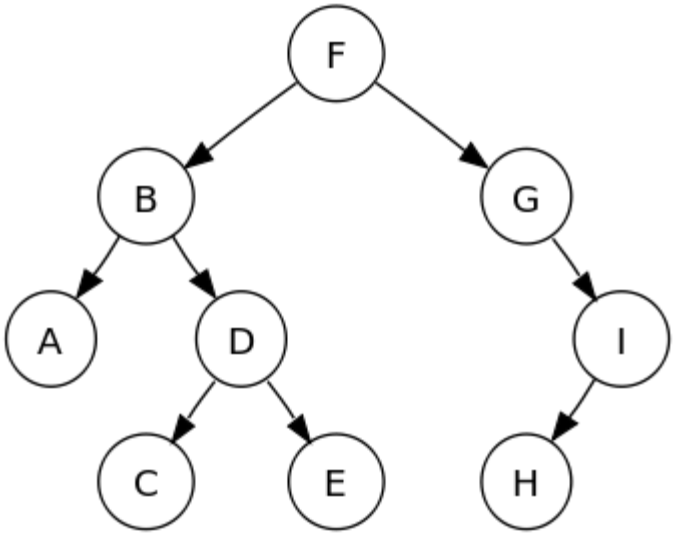
* Contoh Liburan di Jawa Timur dan Permasalahan Teko Air merepresentasikan masalah dalam **Ruang Keadaan,** yakni ruang yang berisi semua keadaan yang mungkin.
* Beberapa cara untuk merepresentasikan Ruang Keadaan antara lain:
  * Graph Keadaan
  * Pohon Pelacakan

Graph Keadaan

* Terdiri dari node-node yang menunjukkan keadaan awal dan keadaan baru yang akan dicapai dengan menggunakan operator
* Sangat memungkinkan adanya siklus dalam graph, sehingga akan menyulitkan proses pelacakan



Pohon Pelacakan

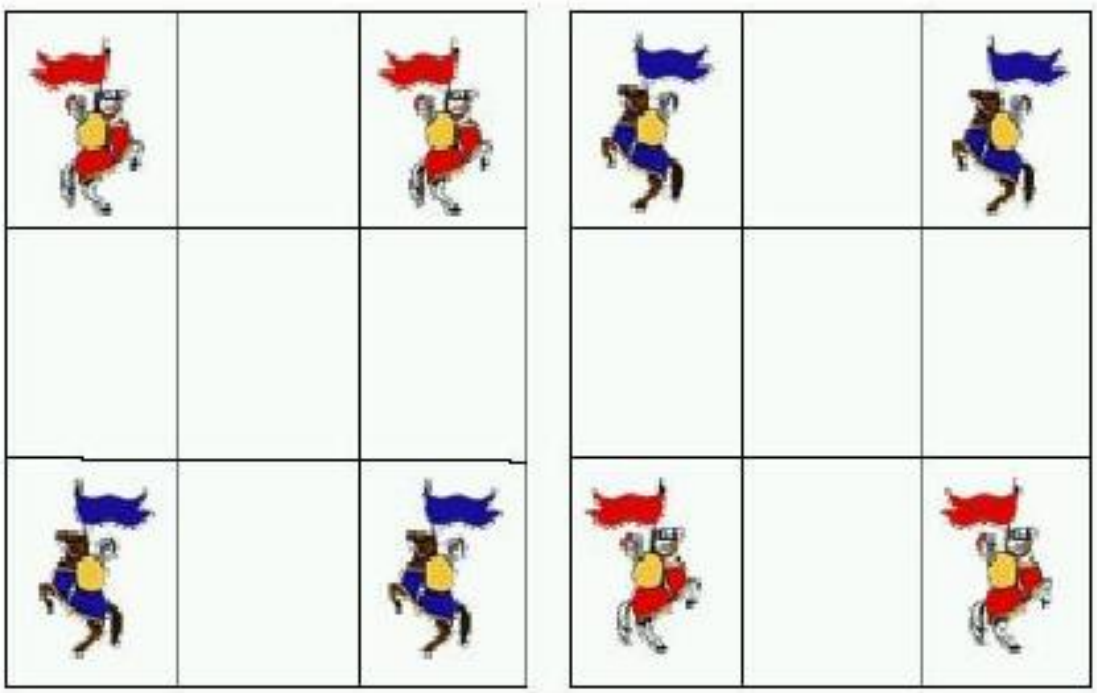


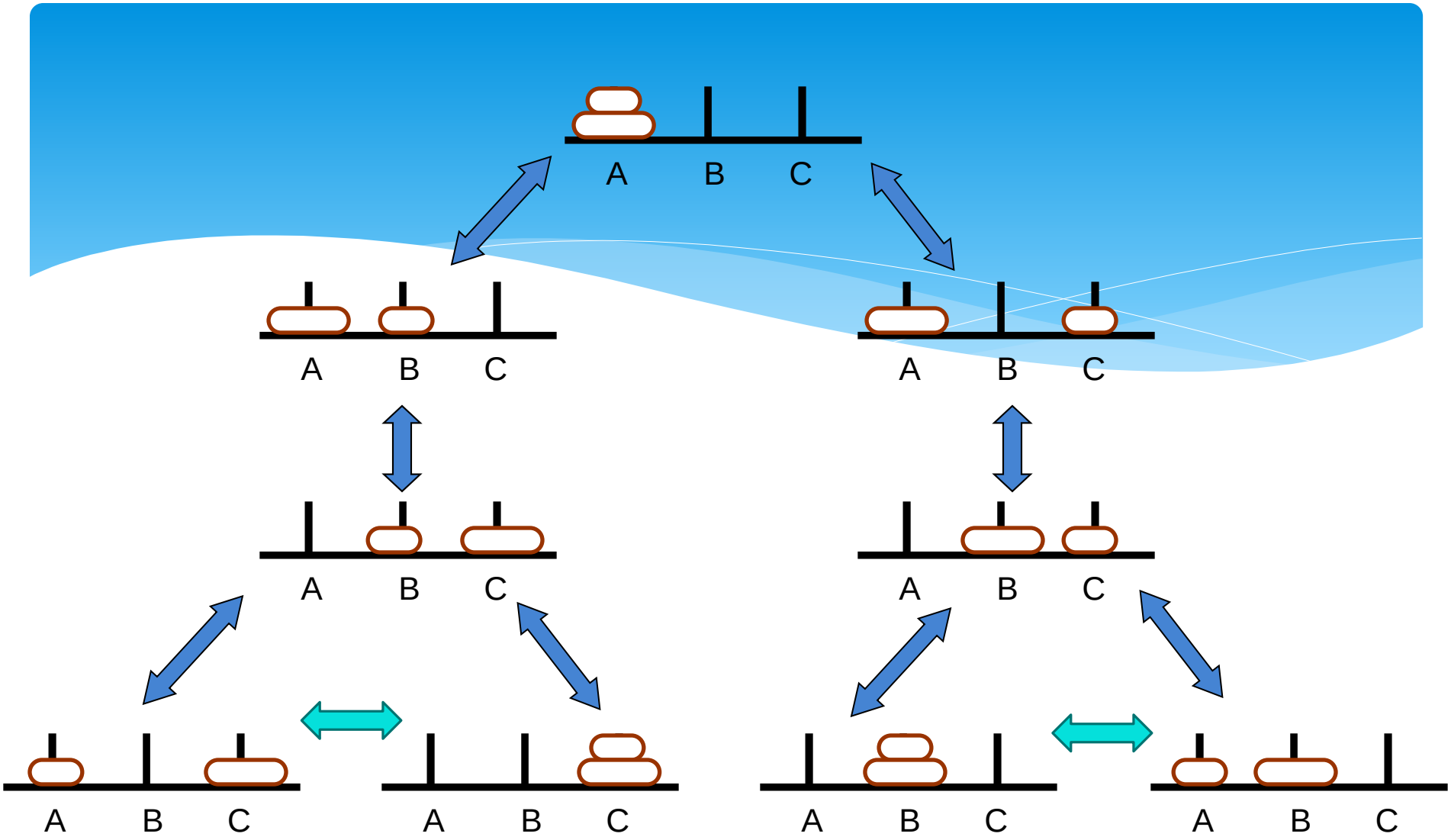
* Digunakan untuk menggambarkan keadaan secara hirarkis.
* Node akar menunjukkan keadaan awal, sedangkan node daun menunjukkan keadaan akhir (goal state) atau dapat juga keadaan buntu (dead-end state)
* Struktur ini menghilangkan kemungkinan adanya siklus (loop)

Permasalahan Ruang Keadaan (*State Space*) yang lainnya

* Penukaran bidak Kuda pada permainan catur
* Menara Hanoi
* Misionaris dan Kanibal

Penukaran Bidak Kuda





Permasalahan Menara Hanoi

Missionaris and Canibal (1)

* Ada 3 missionaris dan 3 kanibal ingin menyeberangi sungai dengan sebuah sampan.
* Sampan mampu menampung maksimum 2 orang.
* Jumlah missionaris tidak boleh lebih sedikit daripada jumlah kanibal pada suatu sisi.
* Missionaris dan kanibal bisa menggunakan sampan

Missionaris and Canibal (2)

State : missionaris,canibal,boat

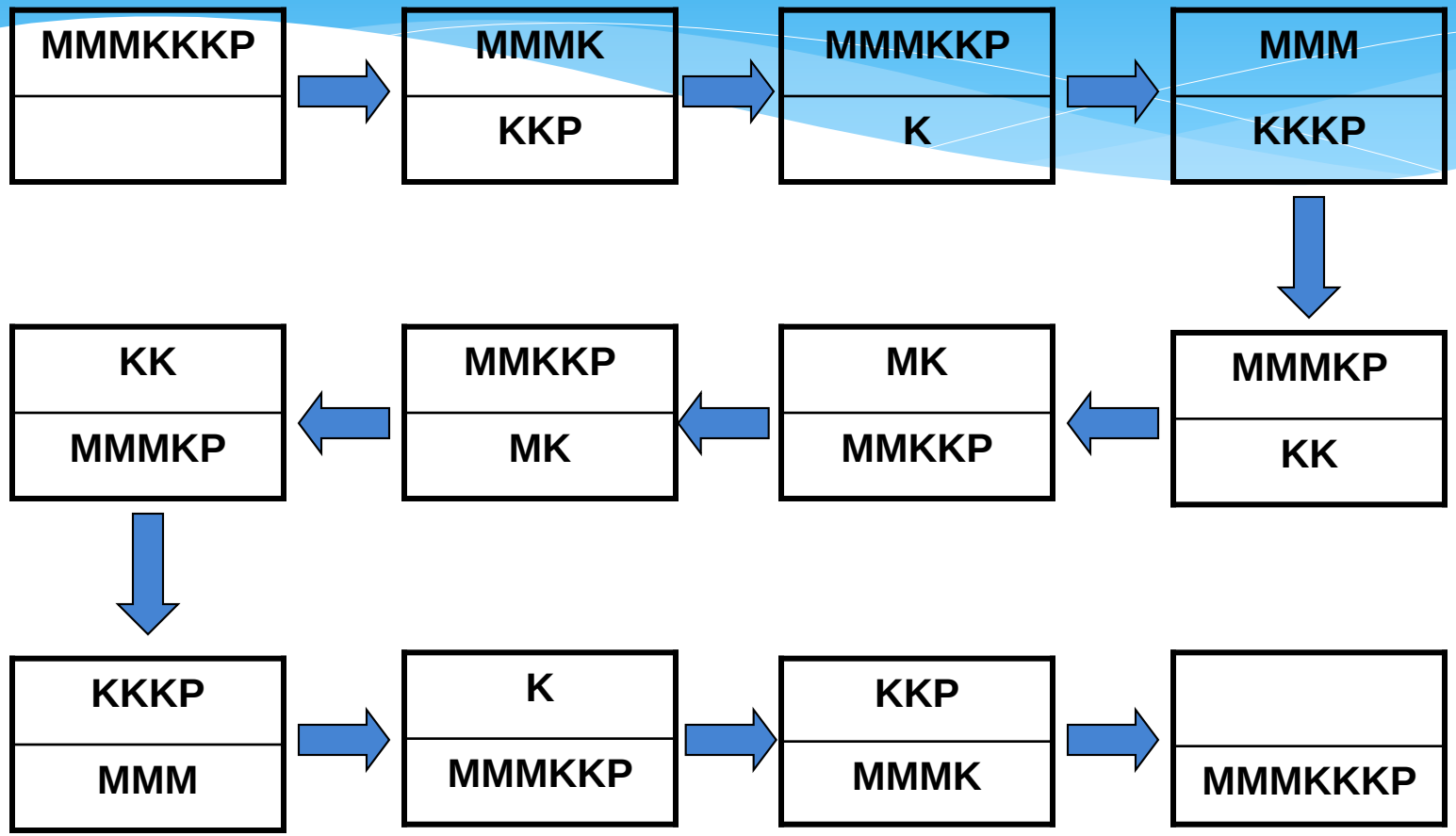
Start state : 3,3,1

Kemungkinan state :

3,3,1	3,2,1	3,1,1	3,0,1
3,3,0	3,2,0	3,1,0	3,0,0
2,3,1	2,2,1	2,1,1	2,0,1
2,3,0	2,2,0	2,1,0	2,0,0
1,3,1	1,2,1	1,1,1	1,0,1
1,3,0	1,2,0	1,1,0	1,0,0
0,3,1	0,2,1	0,1,1	0,0,1
0,3,0	0,2,0	0,1,0	0,0,0

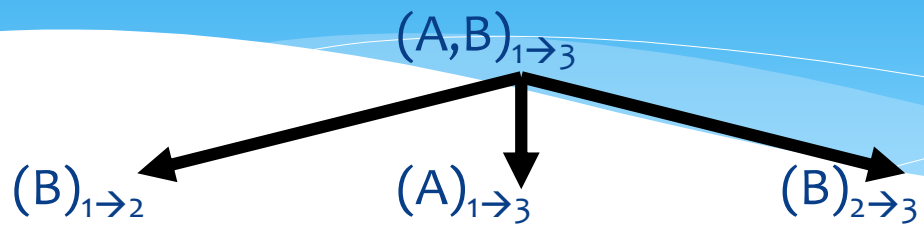
End state : 0,0,0

Biru : start dan end state
Hitam : aman
Merah : jumlah kanibal > missionaris pada kedua sisi sungai
Hijau : kondisi yang tidak mungkin tercapai

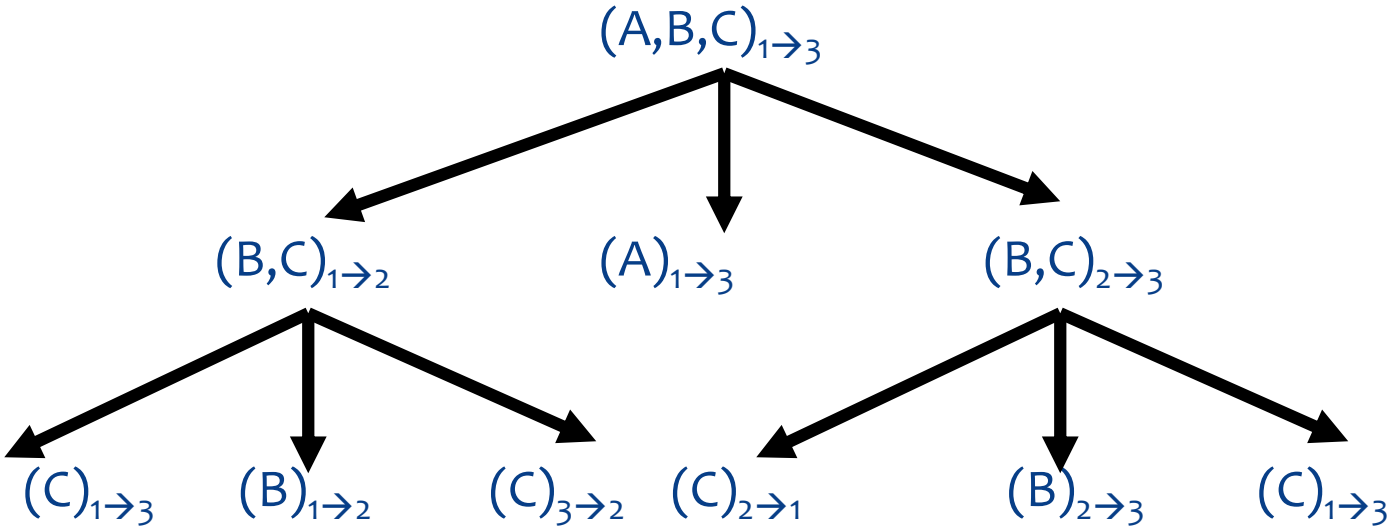

MMMKKKP
MMMK
KKP
MMMKKP
K
MMM
KKKP
MMMKP
KK
MK
MMKKP
MMKKP
MK
KK
MMMKP
KKKP
MMM
K
MMMKKP
KKP
MMMK
MMMKKKP

Reduksi Masalah

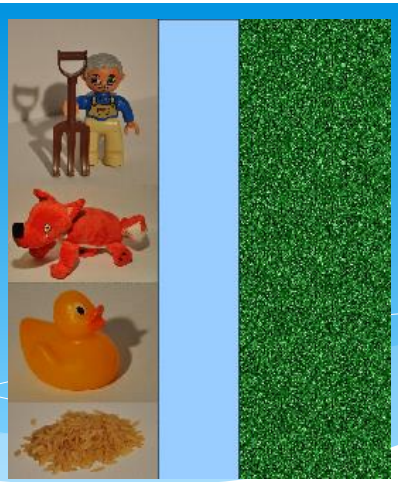
- Tower of Hanoi 2 piringan



- 3 piringan



Latihan : State Space



Permasalahan Petani (p), Serigala (s), Angsa (a) dan Padi (pa).
Petani ingin menyeberangi sungai menggunakan perahu kecil dengan semua barang bawaannya (serigala, angsa, padi). Syarat :

- hanya 1 jenis barang bawaan yang dapat diangkut setiap satu kali menyeberang
- serigala, angsa dan padi tidak boleh di satu sisi sungai yang sama tanpa adanya petani di sisi tersebut

p >< s, a, pa

- angsa dan padi tidak boleh di satu sisi sungai yang sama tanpa adanya petani di sisi tersebut

p, s >< a, pa

- serigala, angsa tidak boleh di satu sisi sungai yang sama tanpa adanya petani di sisi tersebut

p, pa >< s, a

Selesaikan permasalahan ini dengan metode state space (seperti contoh)

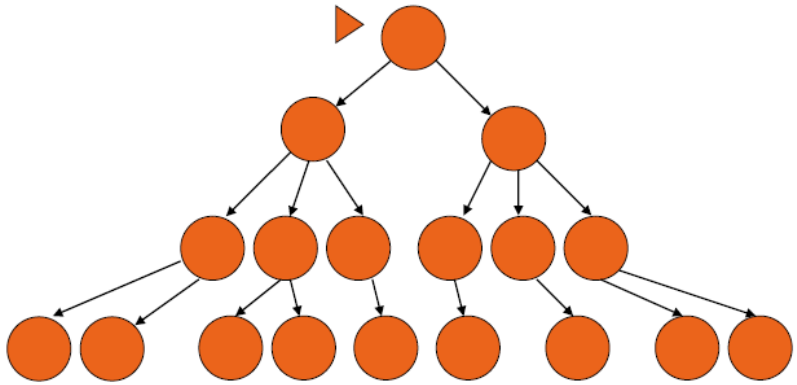
1. Buat dulu state spacenya
2. Gambarkan/tuliskan urutan pencarian solusinya

Metode Pencarian dan Pelacakan

* Keberhasilan sistem cerdas dalam mendapatkan sebuah solusi ditentukan oleh kesuksesannya dalam pencarian dan pencocokan.
* Ada dua teknik pencarian dan pencocokan:
  * Pencarian Buta (Blind Search/Uninformed Search)
  * Pencarian Terbimbing (Informed Search)

Uninformed Search (Blind Search/Pencarian Buta)

* Ide Dasarnya adalah mengeksplorasi ruang keadaan dengan menghasilkan *successor* dari state yang diperluas
* Pencarian buta hanya menggunakan informasi yang tertulis pada definisi permasalahan
* Contoh:
  * Breadth-first search
  * Depth-first search
  * Depth-limited search
  * Iterative deepening search



Prosedur Breadth-first Search

1. Tentukan *L* sebagai daftar node awal pada permasalahan
2. Misal *n* adalah node pertama pada *L*. Jika *L* kosong, maka pencarian gagal.
3. Jika *n* adalah state tujuan, berhenti, dan kembalikan node n beserta jalur (path) dari node awal menuju *n*.
4. Jika *n* bukan state tujuan, hapus *n* dari *L* dan tambahkan seluruh anak dari *n* ke bagian **akhir** dari *L*. Beri label jalur yang ditempuh dari node awal menuju tiap-tiap anak. Kembali ke langkah 2.

Prosedur Breadth-first Search (2)

* Ciri dari BFS:
  * Tree dieksplorasi dari atas ke bawah
  * Setiap node pada kedalaman d dikunjungi sebelum node pada kedalaman d+1

Contoh Penyelesaian Permasalahan Teko Air dengan BFS

* Isi L dengan node awal → L = {<nil: (0,0)>}
* n = <nil: (0,0)>
* Goal test – state *n* bukan state tujuan

  Bagaimana mencari seluruh anak dari n?

  Gunakan operasi valid pada state n

  Misal: a1(isi teko 3L), a2(isi teko 4L)
* Hapus *n* dari L dan tambahkan anak-anak n ke bagian akhir dari L → L = {<a1: (3,0)>, <a2: (0,4)>}

Contoh Penyelesaian Permasalahan Teko Air dengan BFS (2)

* Hapus *n* dari L dan tambahkan anak-anak n ke bagian akhir dari L →

  L = {<a1: (3,0)>, <a2: (0,4)>}

* n = <a1: (3,0)>
* Goal test – state *n* bukan state tujuan

  Bagaimana mencari seluruh anak dari n?

  Gunakan operasi valid pada state n

  Misal: a2(isi teko 4L), a3(kosongkan teko 3L), a4(pindahkan seluruh isi teko 3L ke teko 4L

* Hapus *n* dari L dan tambahkan anak-anak n ke bagian akhir dari L

  → L = {<a2: (0,4)>, <a2: (3,4)>, <a3: (0,0)>, <a4: (0,3)>}

Contoh Penyelesaian Permasalahan Teko Air dengan BFS (3)

* Hapus *n* dari L dan tambahkan anak-anak n ke bagian akhir dari L

  → L = {<a2: (0,4)>, <a2: (3,4)>, <a3: (0,0)>, <a4: (0,3)>}

* n = <a2: (0,4)>

* Goal test – state *n* bukan state tujuan

  Bagaimana mencari seluruh anak dari n?

  Gunakan operasi valid pada state n

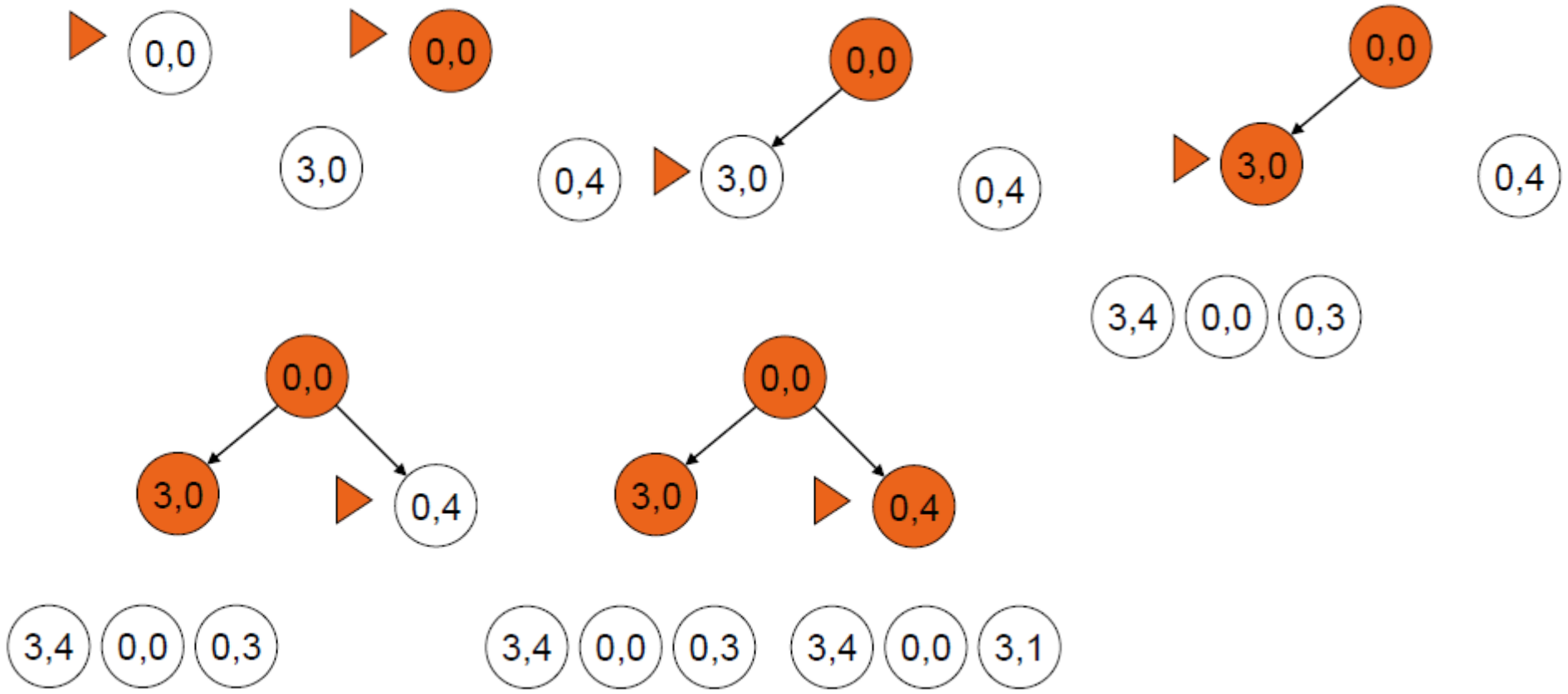
  Misal: a1(isi teko 3L), a4(kosongkan teko 4L), a8(tuang isi teko 4L ke teko 3L sampai teko 3L penuh)

* Hapus *n* dari L dan tambahkan anak-anak n ke bagian akhir dari L

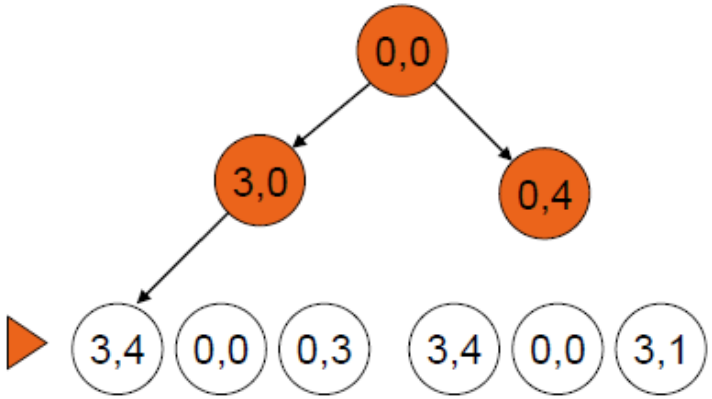
  → L = {<a2: (3,4)>, <a3: (0,0)>, <a4: (0,3)>, <a1: (3,4)>, <a4: (0,0), <a8: (3,1)}

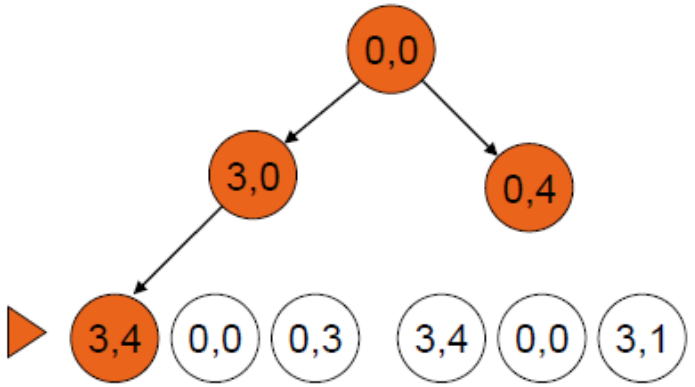
Dan seterusnya …..

Representasi Tree untuk BFS

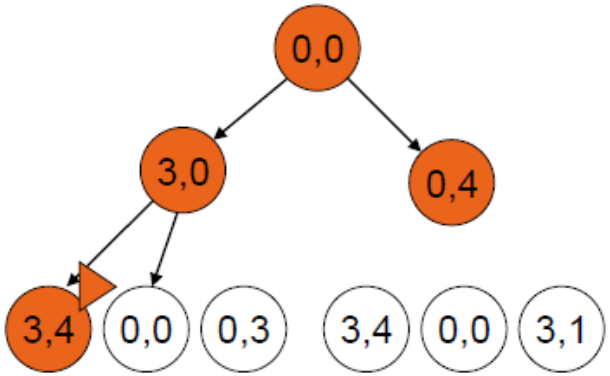


Representasi Tree untuk BFS (2)

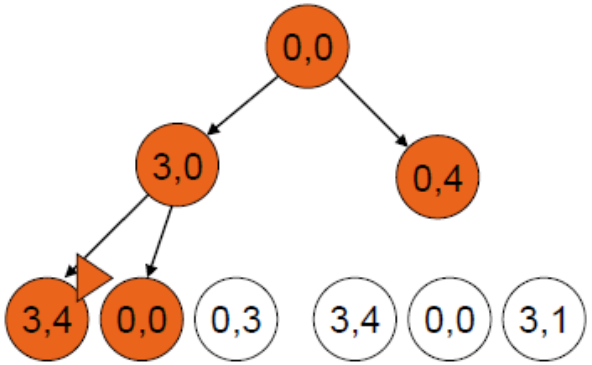




0,4 3,0

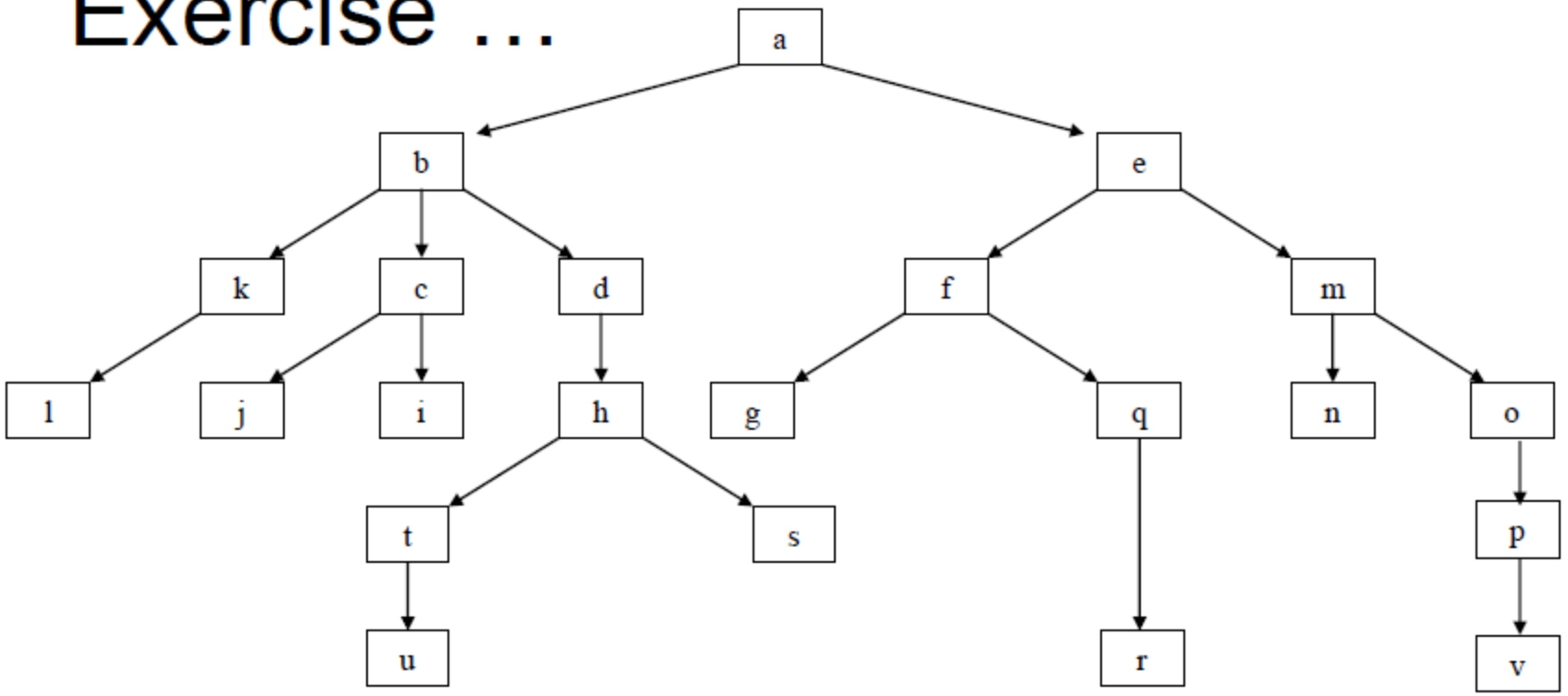


0,4 3,0



0,4 3,0 3,0 0,4

Exercise …



Tuliskan urutan node yang dikunjungi jika menggunakan Breadth-first Search

Prosedur Depth-first Search

1. Tentukan L sebagai daftar node awal dari permasalahan
2. Misal n adalah node pertama pada L. Jika L kosong, pencarian gagal.
3. Jika n adalah state tujuan, berhenti dan kembalikan node tersebut beserta jalur/path dari node awal menuju n.
4. Jika n bukan state tujuan, hapus n dari L dan tambahkan seluruh anak dari n di bagian **depan** dari L. Beri label jalur yang ditempuh dari node awal menuju tiap-tiap anak. Kembali ke langkah 2.

Prosedur Depth-first Search (2)

* Ciri dari DFS:
  * Mengekspansi pohon pencarian sedemikian rupa sehingga node ditelusuri dari kiri ke kanan
  * Kita selalu mengeksplorasi anak dari node yang baru diekspansi terlebih dahulu
  * Jika node ini tidak memiliki anak, lakukan prosedur mundur (backtrack) 1 langkah sebelum memilih node lain untuk diekspansi

Penyelesaian permasalahan teko air dengan DFS

* Isi L dengan node awal → → L = {<nil: (0,0)>}
* n = <nil: (0,0)>
* Goal test – state *n* bukan state tujuan

  Bagaimana mencari seluruh anak dari n?

  Gunakan operasi valid pada state n

  Misal: a1(isi teko 3L), a2(isi teko 4L)
* Hapus *n* dari L dan tambahkan anak-anak n ke bagian depan dari L → L = {<a1: (3,0)>, <a2: (0,4)>}

Penyelesaian permasalahan teko air dengan DFS (2)

* Hapus *n* dari L dan tambahkan anak-anak n ke bagian awal dari L →

  L = {<a1: (3,0)>, <a2: (0,4)>}
* n = <a1: (3,0)>
* Goal test – state *n* bukan state tujuan

  Bagaimana mencari seluruh anak dari n?

  Gunakan operasi valid pada state n

  Misal: a2(isi teko 4L), a3(kosongkan teko 3L), a4(pindahkan seluruh isi teko 3L ke teko 4L
* Hapus *n* dari L dan tambahkan anak-anak n ke bagian awal dari L

  → L = {<a2: (3,4)>, <a3: (0,0)>, <a4: (0,3)>, <a2: (0,4)>}

Penyelesaian permasalahan teko air dengan DFS (3)

* Hapus *n* dari L dan tambahkan anak-anak n ke bagian awal dari L

  → L = {<a2: (3,4)>, <a3: (0,0)>, <a4: (0,3)>, <a2: (0,4)>}
* n = <a2: (3,4)>
* Goal test – state *n* bukan state tujuan

  Bagaimana mencari seluruh anak dari n?

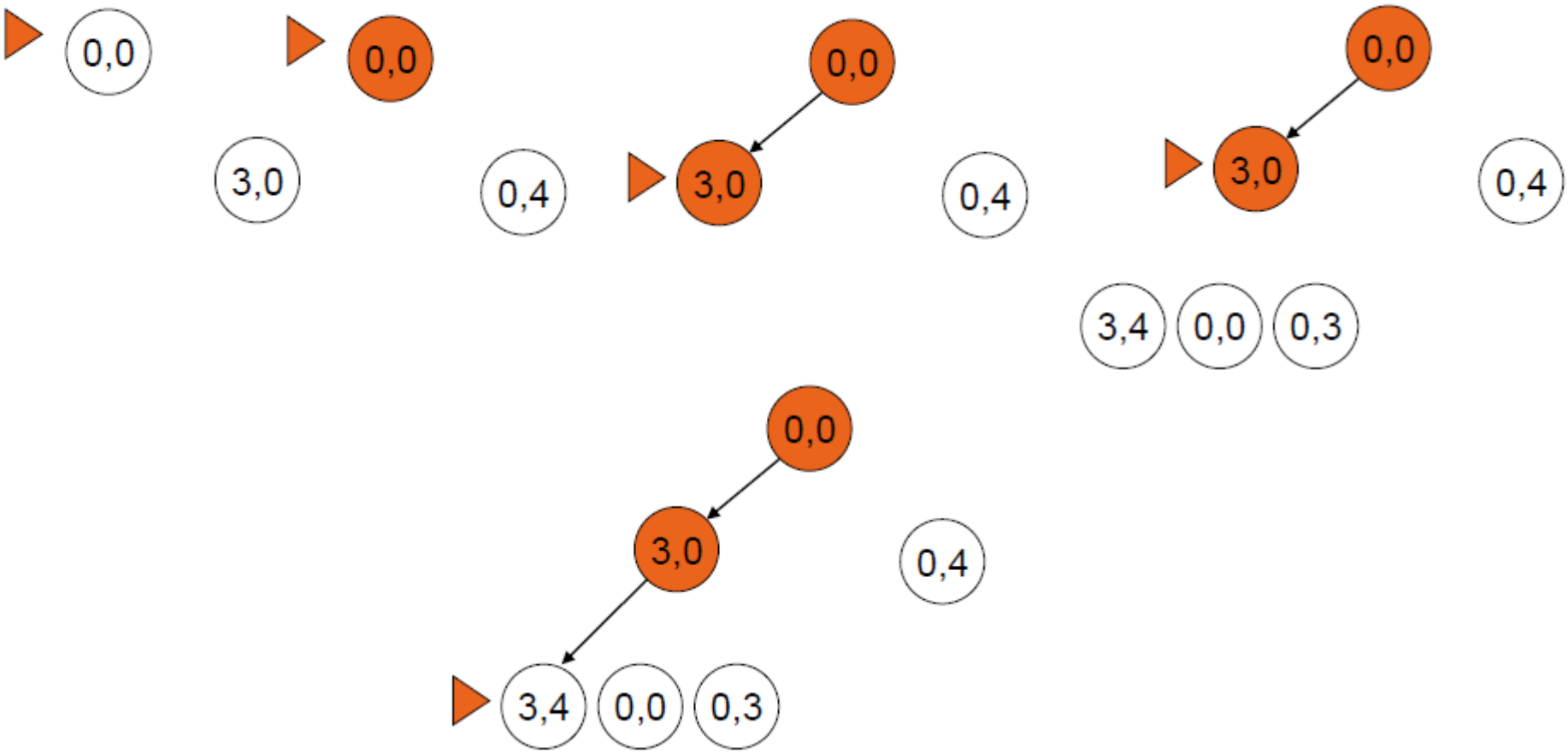
  Gunakan operasi valid pada state n

  Misal: a3(kosongkan teko 3L), a4(kosongkan teko 4L)
* Hapus *n* dari L dan tambahkan anak-anak n ke bagian awal dari L

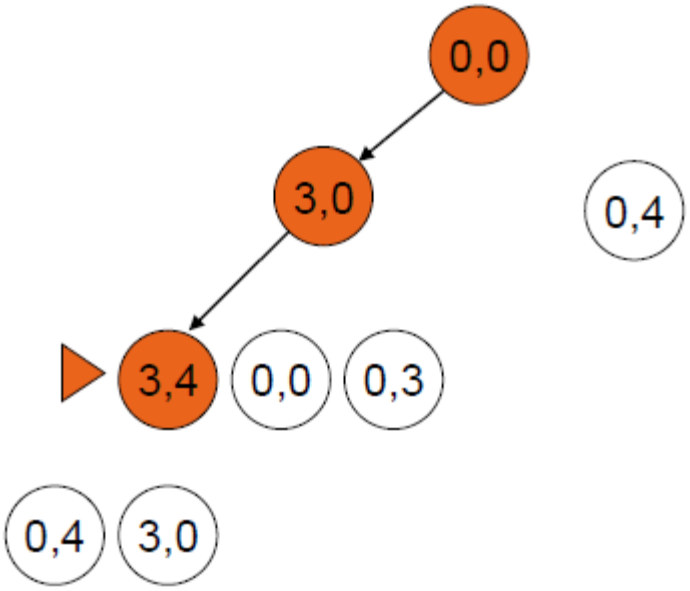
  → L = {<a3: (0,4)>, <a4: (3,0)>, <a3: (0,0)>, <a4: (0,3)>, <a2: (0,4)>}

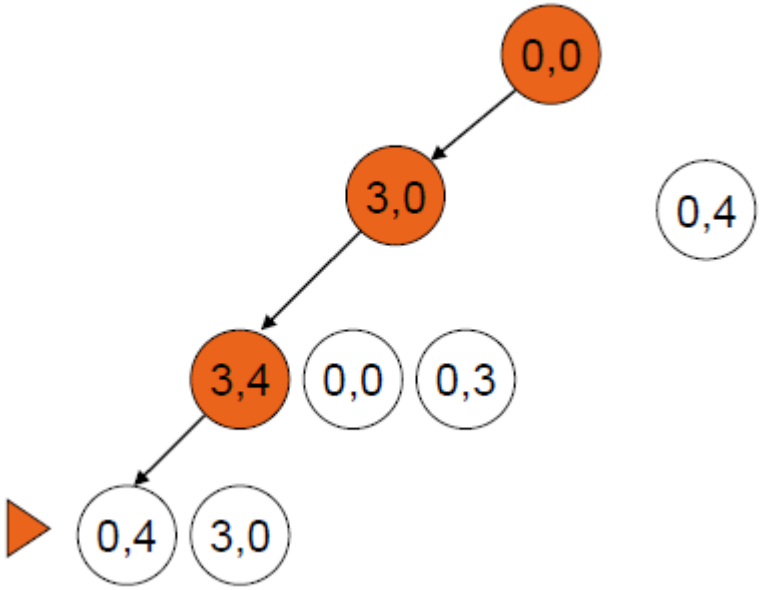
Dan seterusnya …..

Representasi Tree untuk DFS

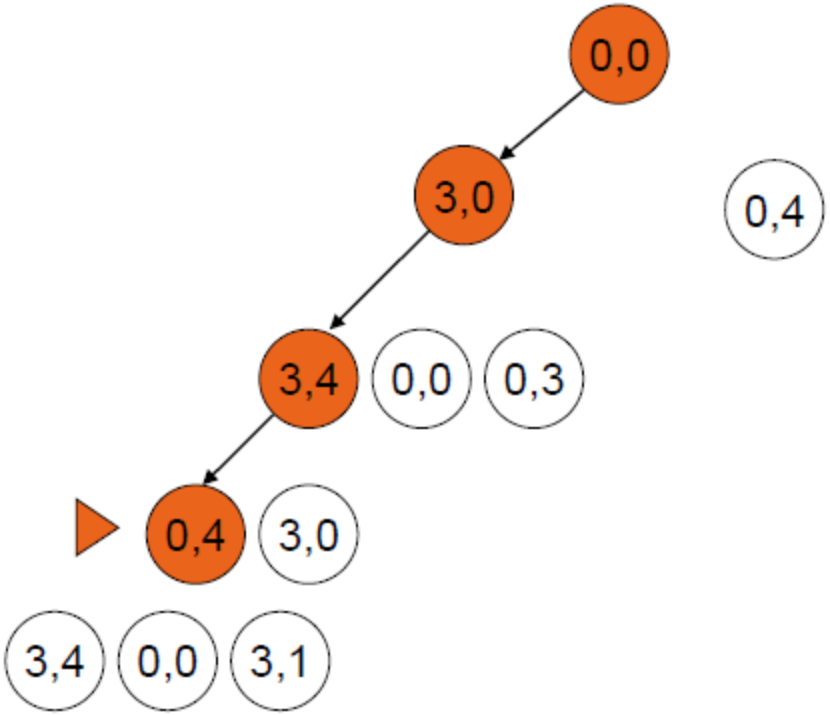


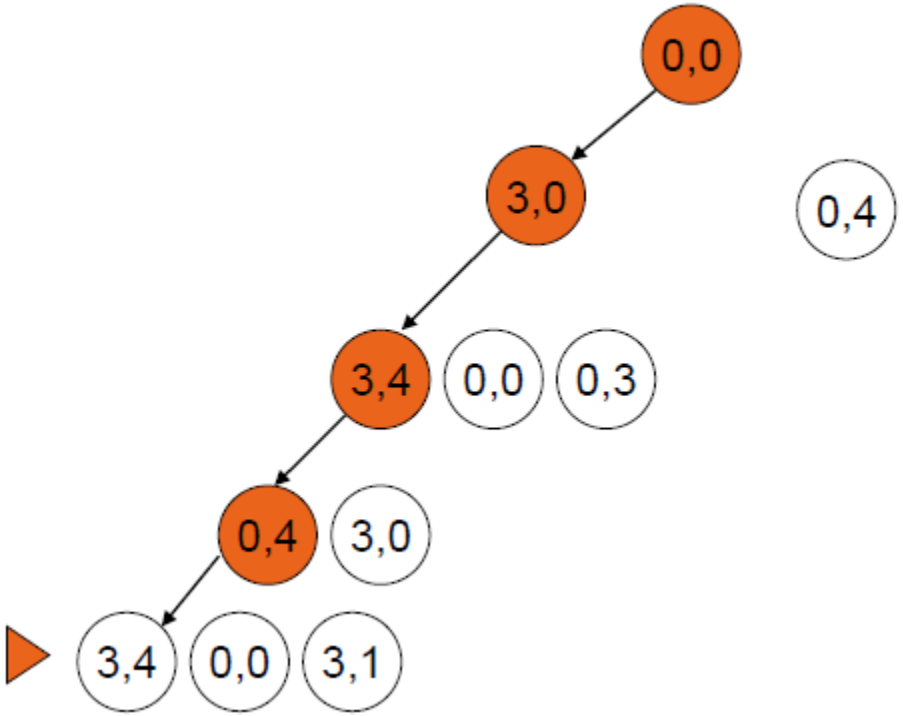
Representasi Tree untuk DFS (2)



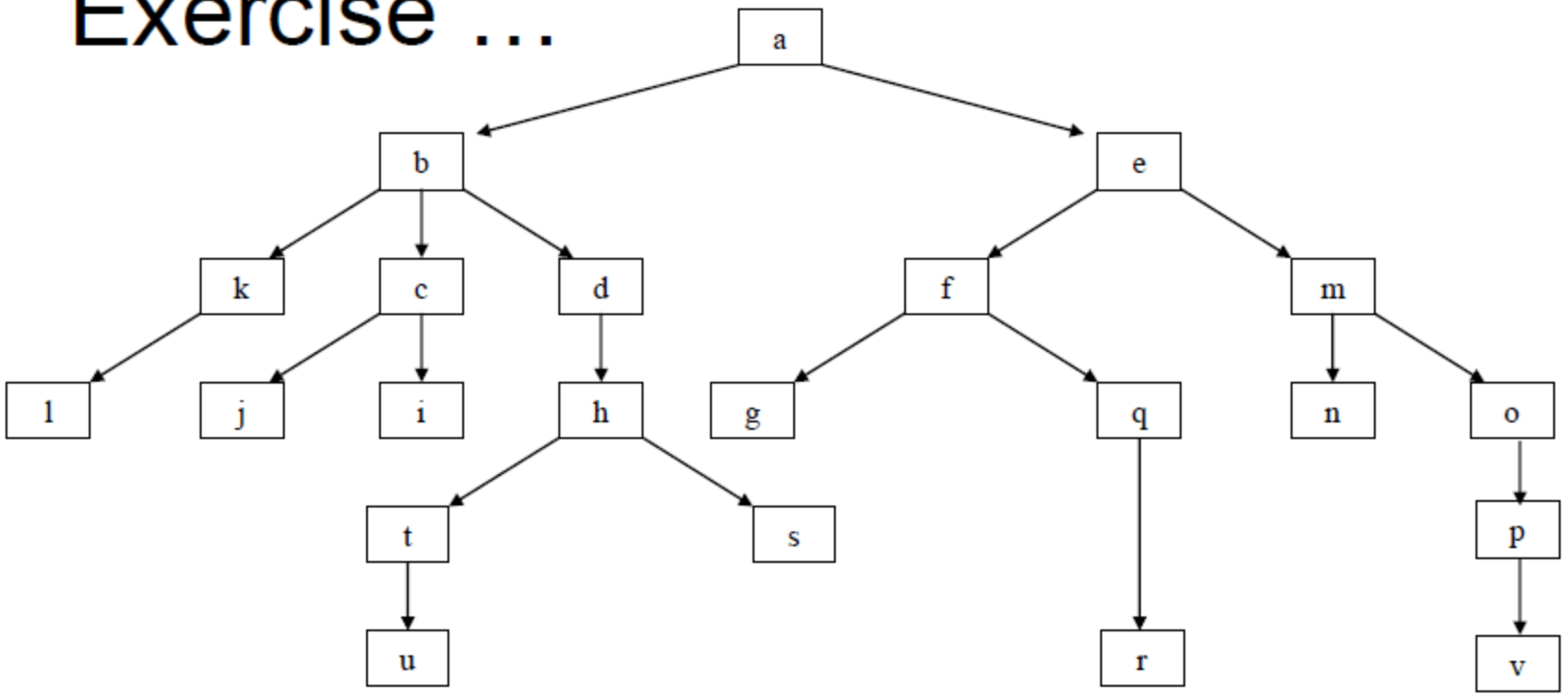


Representasi Tree untuk DFS (3)

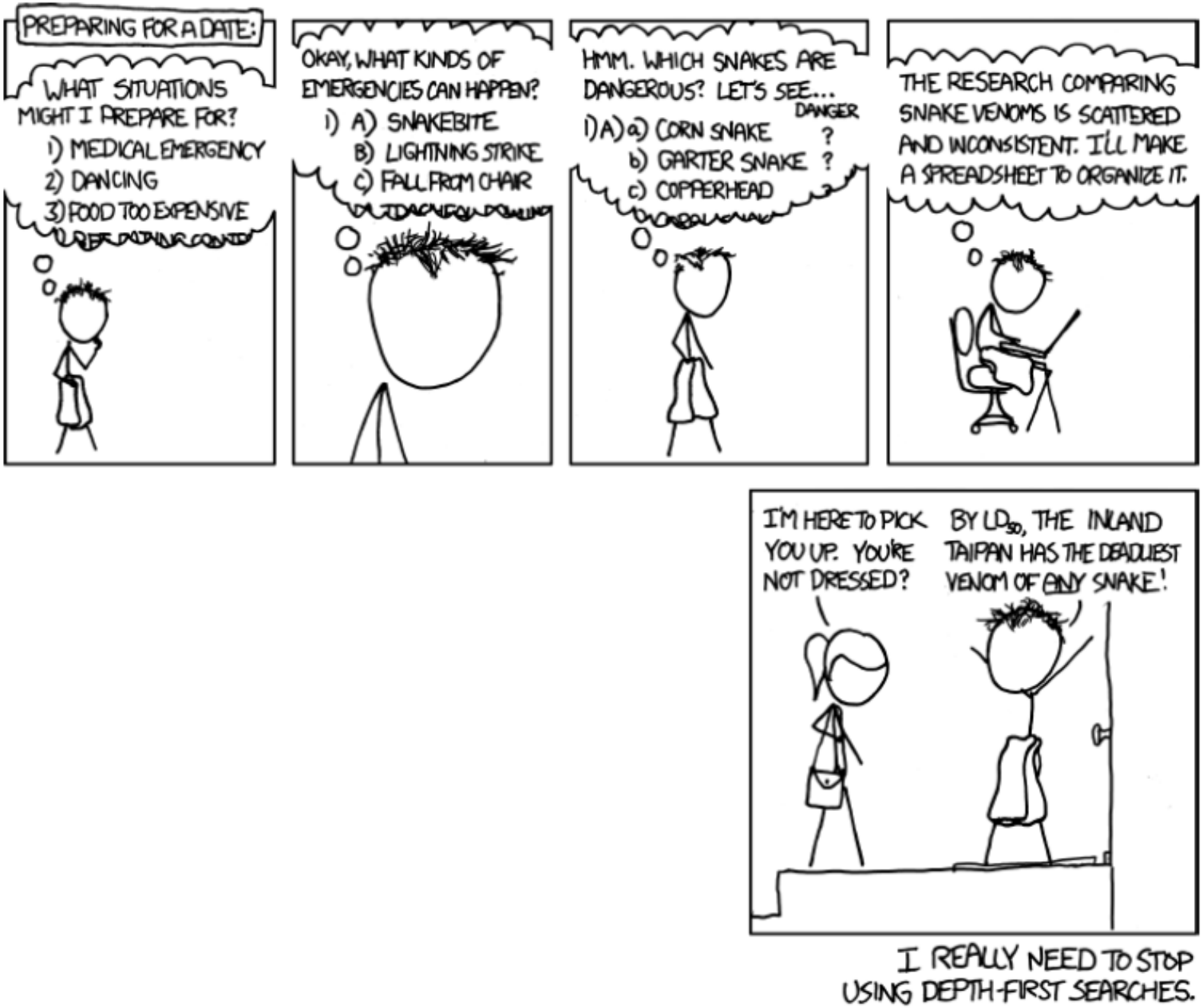




Exercise …



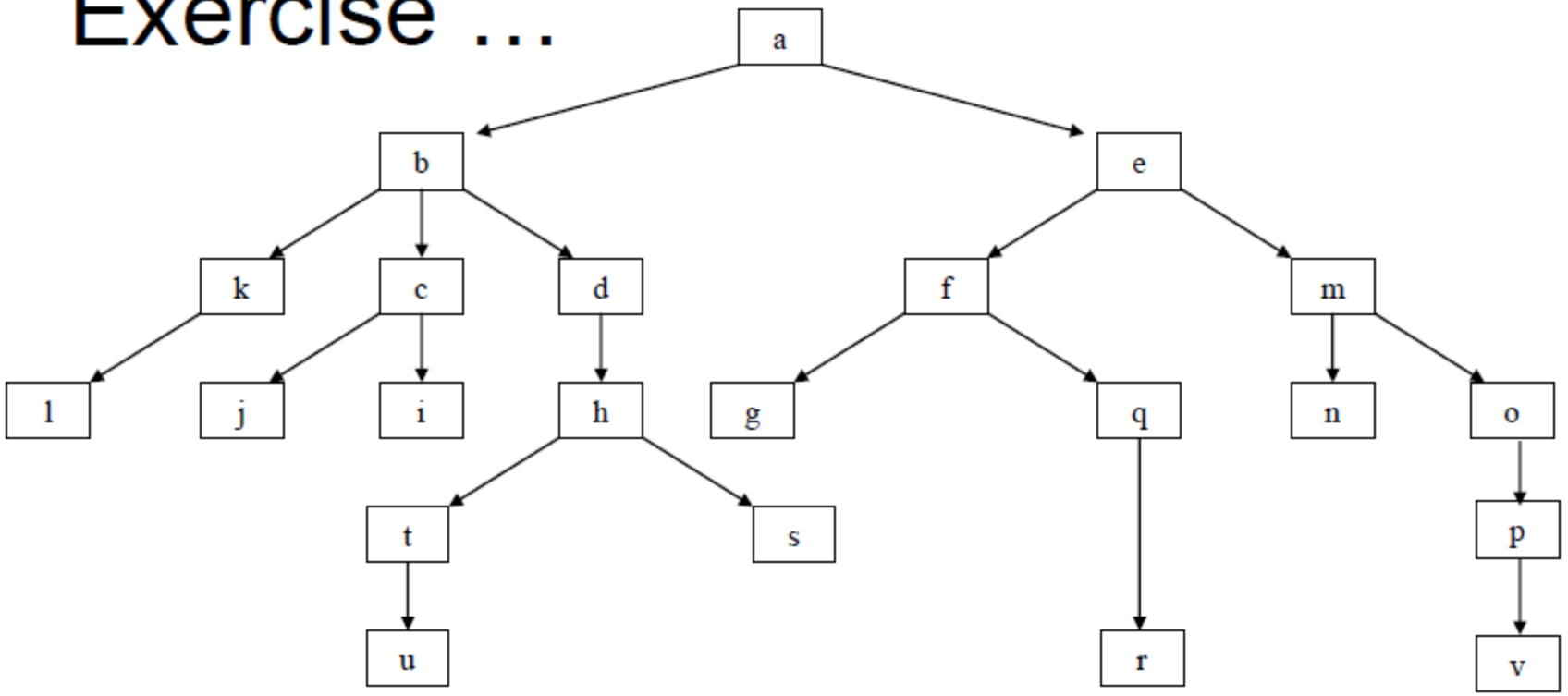
Tuliskan urutan node yang dikunjungi jika menggunakan Depth-first Search


PREPARING FOR A DATE:
WHAT SITUATIONS MIGHT I PREPARE FOR?
1) MEDICAL EMERGENCY
2) DANCING
3) FOOD TOO EXPENSIVE
OKAY, WHAT KINDS OF EMERGENCIES CAN HAPPEN?
1) A) SNAKEBITE
B) LIGHTNING STRIKE
C) FALL FROM CHAIR
HMM. WHICH SNAKES ARE DANGEROUS? LET'S SEE...
1) A) a) CORN SNAKE ?
b) GARTER SNAKE ?
c) COPPERHEAD
DANGER
THE RESEARCH COMPARING SNAKE VENOMS IS SCATTERED AND INCONSISTENT. I'LL MAKE A SPREADSHEET TO ORGANIZE IT.
I'M HERE TO PICK YOU UP. YOU'RE NOT DRESSED?
BY LD_{50}, THE INLAND TAIPAN HAS THE DEADLIEST VENOM OF ANY SNAKE!
I REALLY NEED TO STOP USING DEPTH-FIRST SEARCHES.

Depth-limited Search

* Sama dengan Depth-first Search, hanya menetapkan suatu batasan kedalaman maksimum (l) yang akan ditelusuri
* Root memiliki kedalaman 0

Exercise …



Tuliskan urutan node yang dikunjungi jika menggunakan Depth-limited Search dengan maksimum kedalaman adalah 3

Iterative-deepening Search

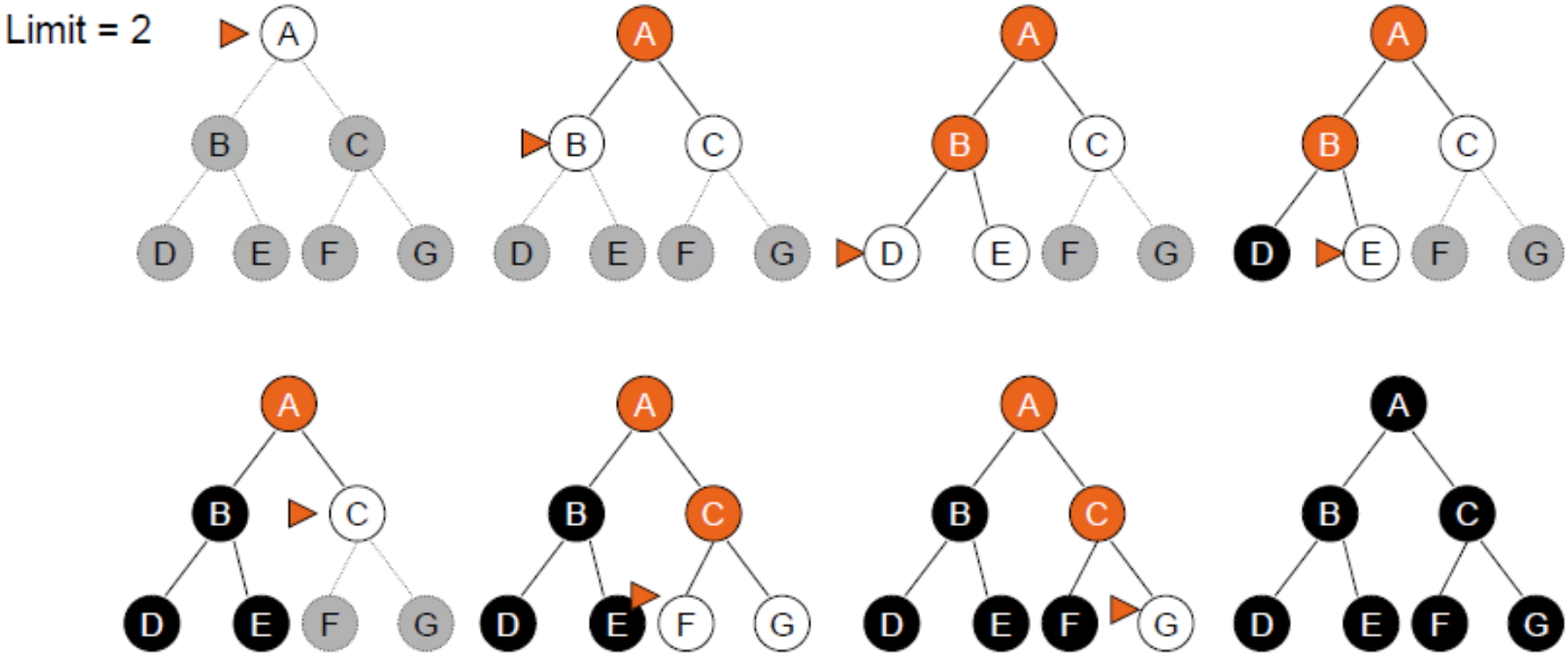
* Mirip dengan Depth-limited Search, hanya secara bertahap meningkatkan kedalaman node yang ditelusuri

Iterative-deepening Search

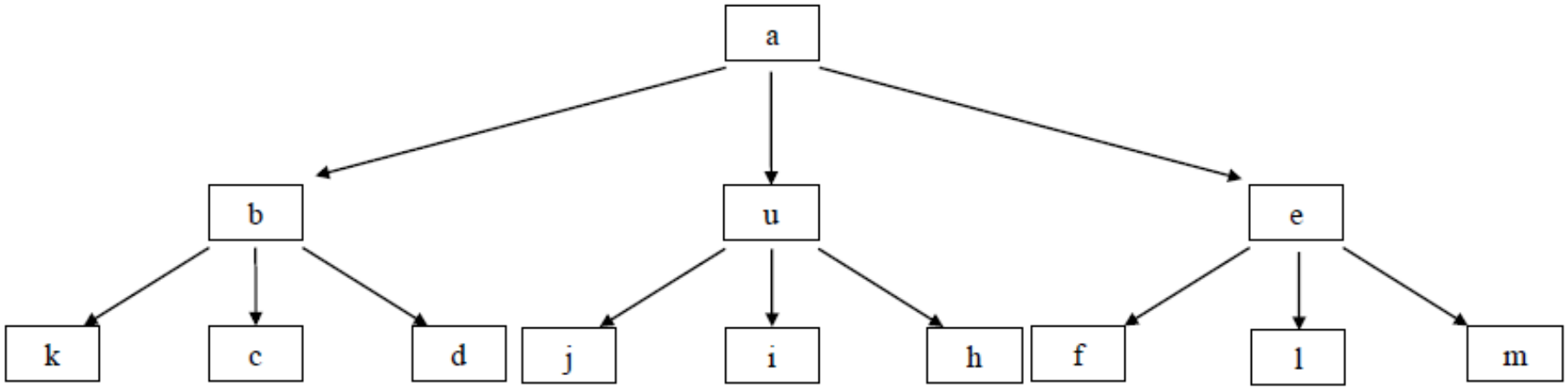
Limit = 0

Limit = 1

Iterative-deepening Search



Latihan



Tuliskan urutan penelusuran node jika menggunakan Iterative-deepening Search

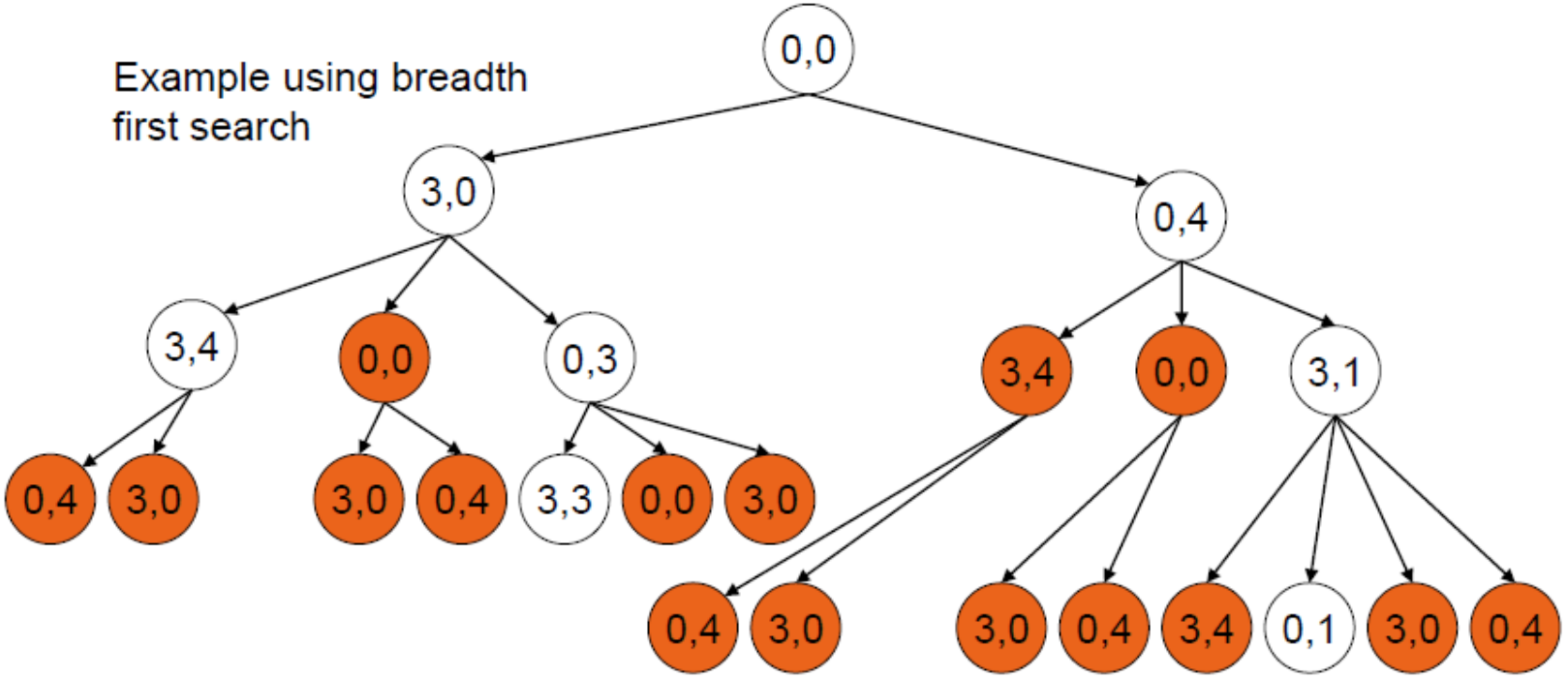
Perbandingan

	Breadth-First	Depth-First	Depth-Limited	Iterative Deepening
Complete?	• Yes	• No	• No	• Yes
Time	• $O(b^{d+1})$	• $O(b^m)$	• $O(b^l)$	• $O(b^d)$
Space	• $O(b^{d+1})$	• $O(bm)$	• $O(bl)$	• $O(bd)$
Optimal?	• Yes	• No	• No	• Yes

b = jumlah node
m = kedalaman pada tree untuk Depth-First Search
l = kedalaman pada tree untuk Depth-limited Search
d = kedalaman pada tree untuk Breadth-First Search dan Iterative-deepening Search

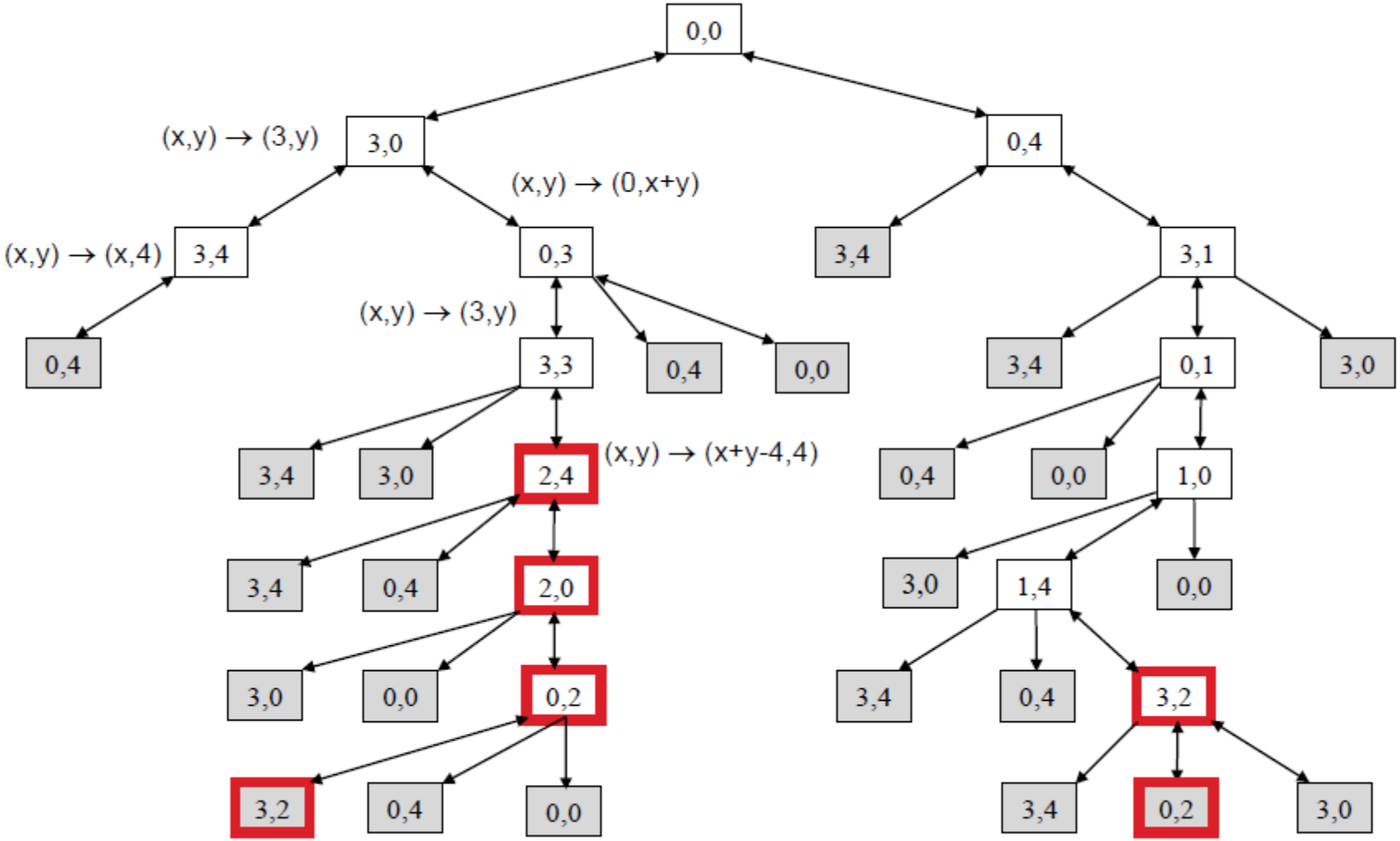
Penanganan State Berulang

* HATI-HATI! Kegagalan dalam mendeteksi state yang berulang akan mengakibatkan permasalahan linier menjadi permasalahan eksponensial



Tips Penanganan State Berulang

* Jangan kembali pada state yang telah dikunjungi
* Jangan membuat jalur/path yang mengandung siklus/cycle di dalamnya
* Jangan men-generate state yang sudah di-generate sebelumnya



Contoh penanganan state berulang

Tugas

* Selesaikan permasalahan Petani-Beras-Angsa-Serigala dengan menggunakan salah satu dari metode penelusuran:
  * DFS
  * BFS
  * Depth Limited Search (limit kedalaman: 7)
  * Iterative Deepening Search
* Terapkan penanganan state berulang pada penyelesaian permasalahan yang Anda buat

Ringkasan

* Pendefinisian masalah
* Ruang keadaan
* Pencarian Buta/Blind Search/Uninformed Search
  * Breadth-first Search
  * Depth-first Search
  * Depth-limited Search
  * Iterative-deepening Search
* Penanganan State Berulang