TARC
TUNKU ABDUL RAHMAN UNIVERSITY COLLEGE

# Artificial Intelligence

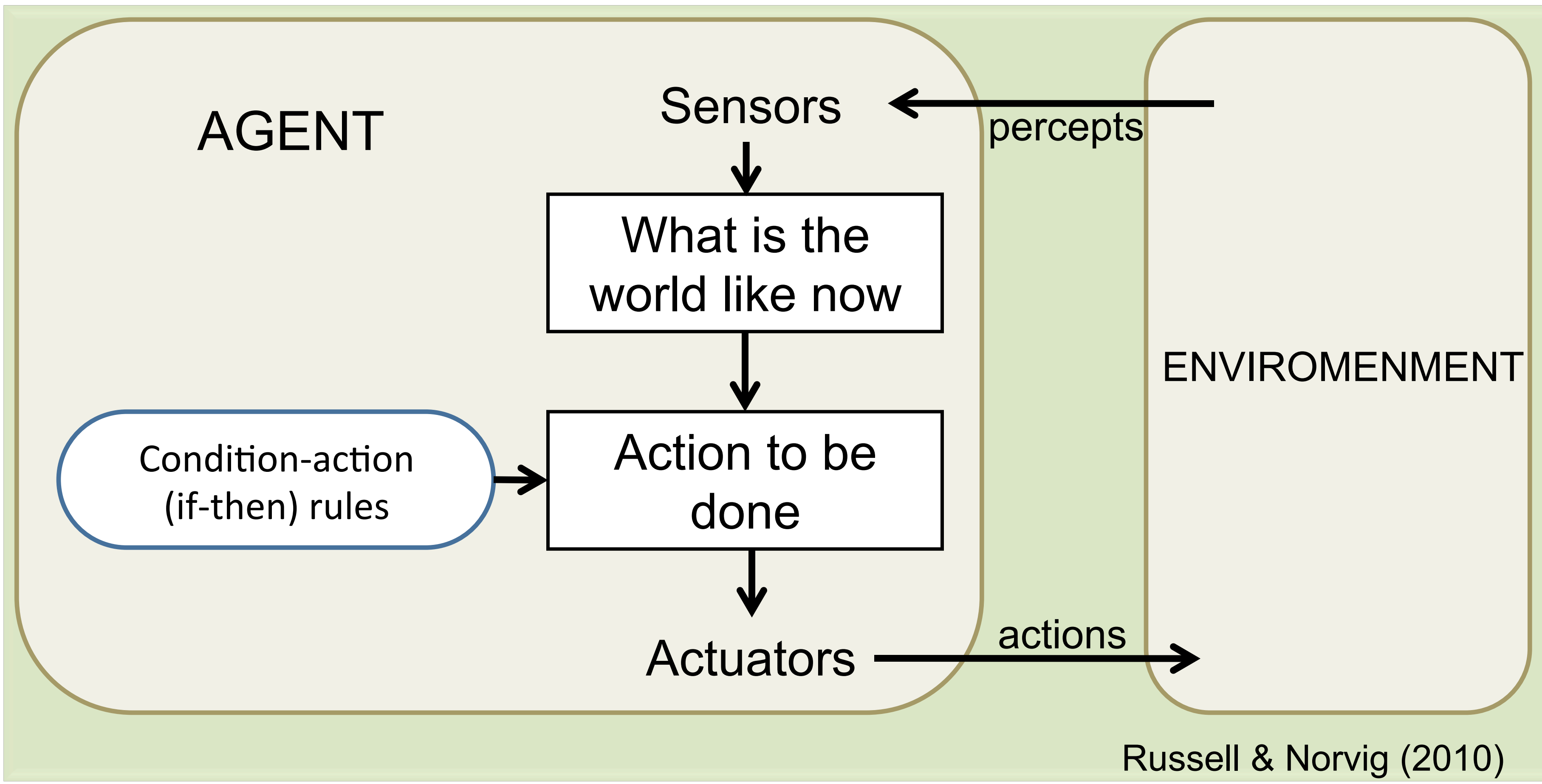
## Chapter 2 Problem-Definition & Problem Solving

# Chapter Outline

- Problem-solving concept
- Measuring problem-solving performance

# Problem-solving agents

- They decide what to do by finding sequences of actions that lead to desirable states.

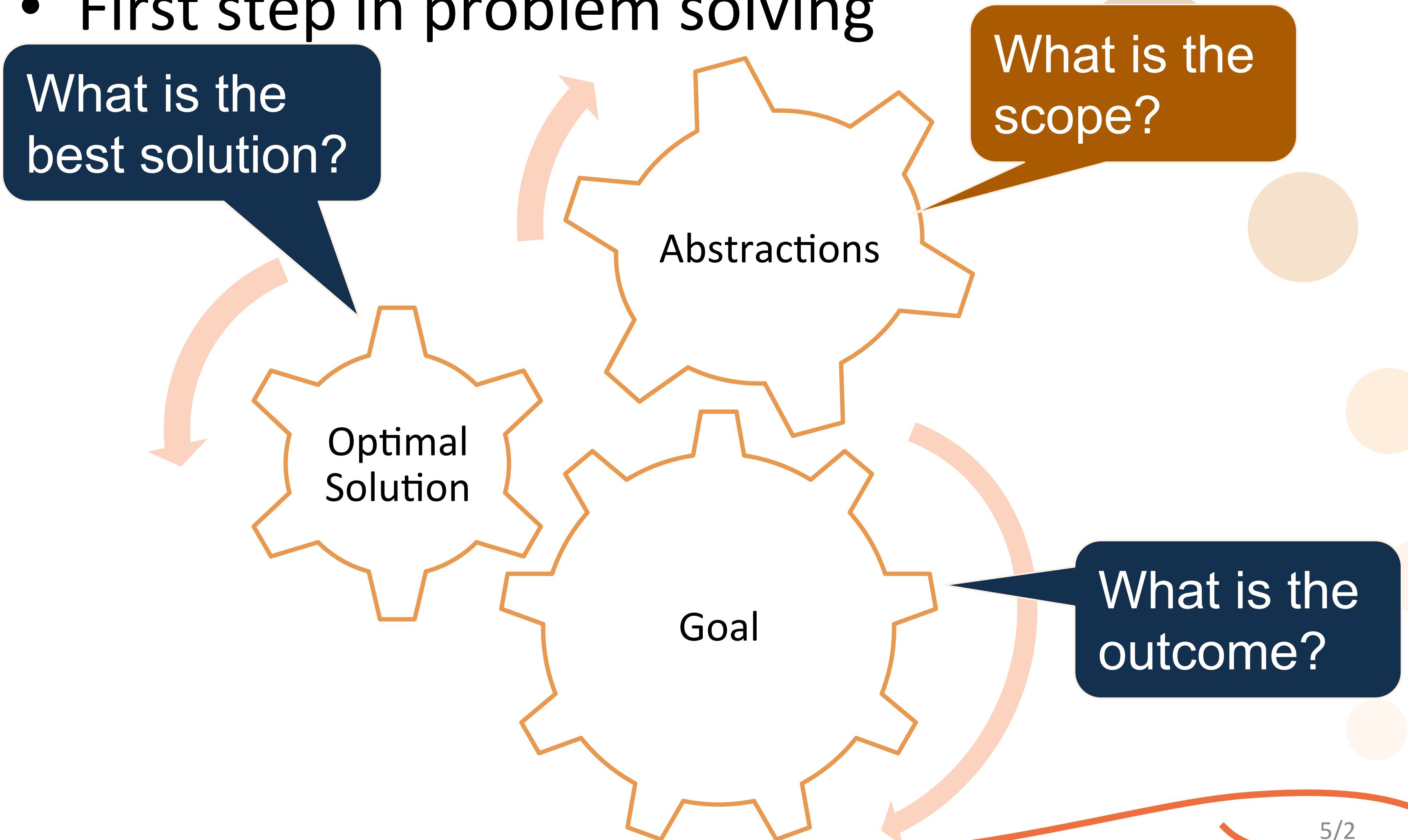


# Problem-solving concept

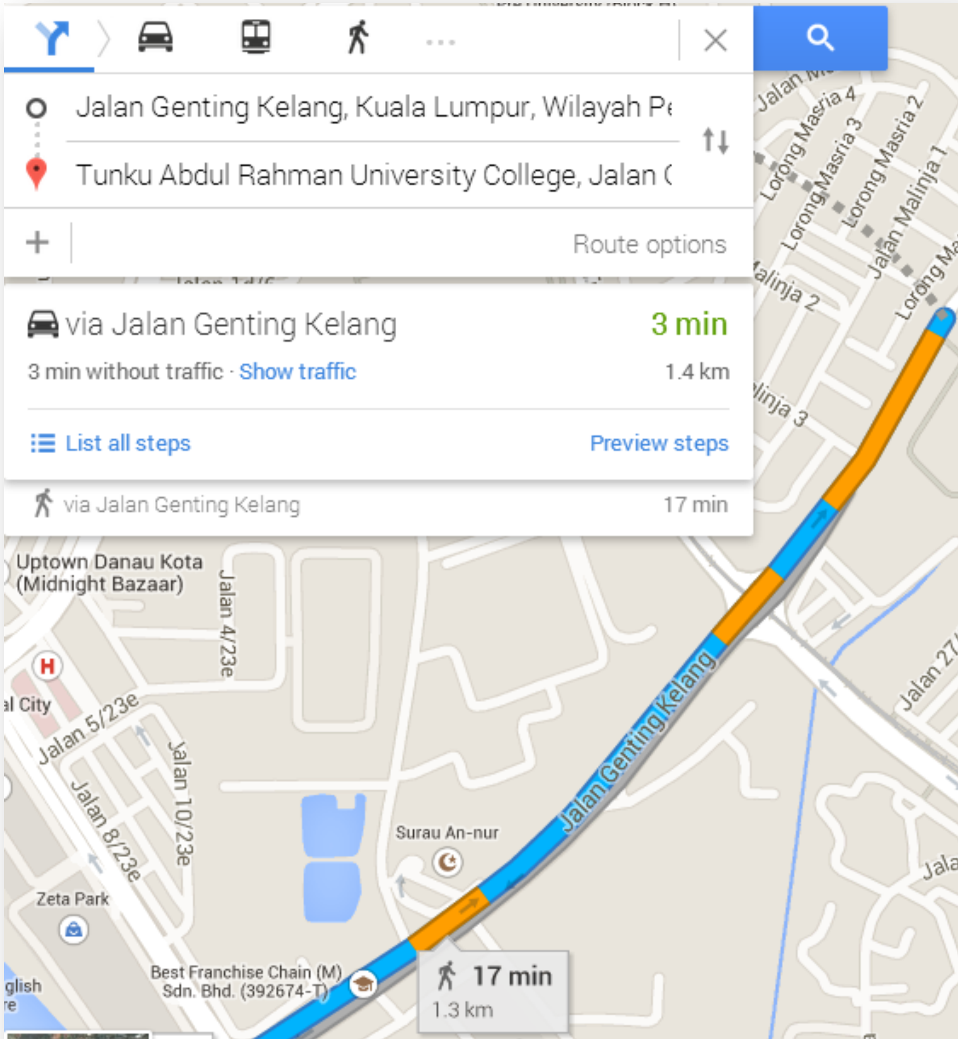
- Goal formulation
- Problem formulation / definition
- Search-solution-execution

# Goal formulation

- First step in problem solving



# Example: Map




Goal?

Optimal Solution?

Abstractions?

# Problem definition

A problem can be defined by 4 components:

1. Initial State — How it starts?
2. Successor Function, State Space, Path — How is it solved?
3. Goal Test — Is it the desired outcome?
4. Step Cost, Path Cost — What is the cost?

# Step 1: Formulating Goal

## Goal

- To reach KLCC

## Optimal Solution

- To get the shortest path to KLCC

## Abstraction (Scope and remove unwanted details)

- E.g. We do not care about the time used to reach KLCC, which can be caused by unexpected factors, such as traffic jam

# Step 2: Formulating Problem (1)

1 **Initial state** (the state that the agent starts in)

– e.g. Go( _ , In(TBR), 0)

# Step 2: Formulating Problem (1)

2 **Successor function** (the possible Actions available to the agent, that will change the state)

For example, a function can be given as:

- *Go(To(child), In(Parent), State_Cost)*

E.g.:

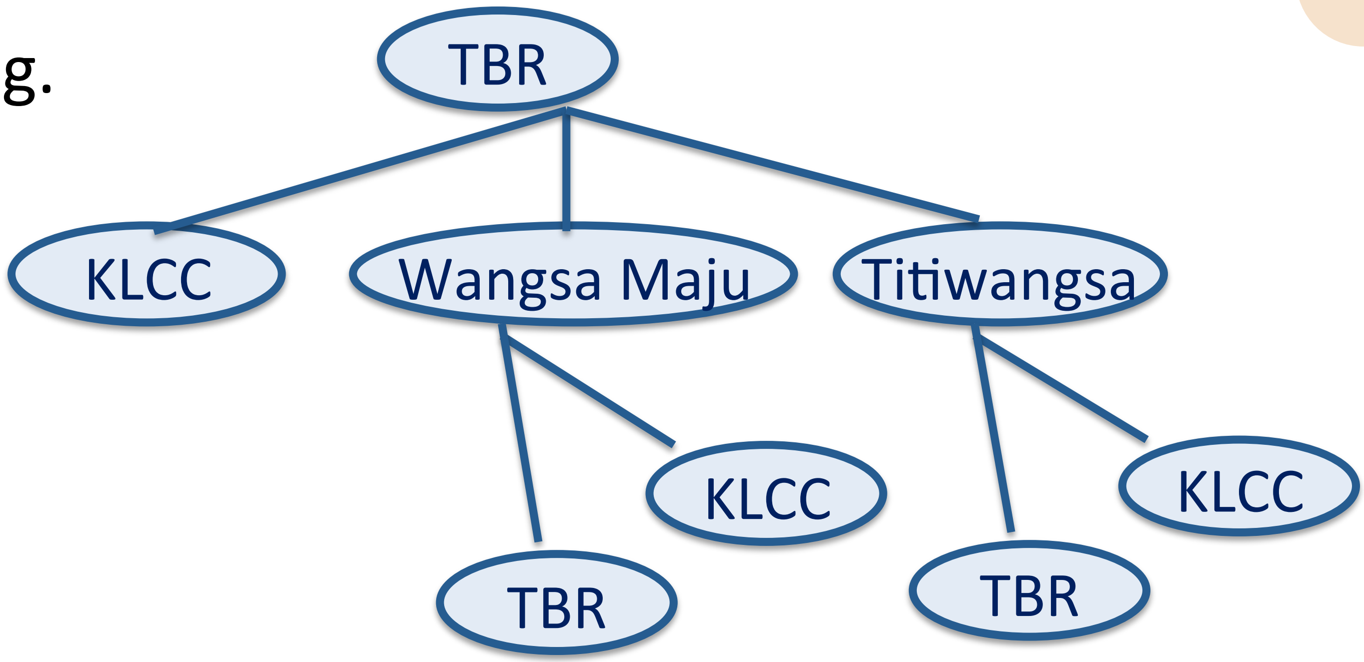
- *Go(To(Wangsa Maju, In(TBR), 3)*
- *Go(To(KLCC), In(Wangsa Maju), 13),*

# Step 2: Formulating Problem (1)

2 **State space** (the set of all states reachable from the initial state)

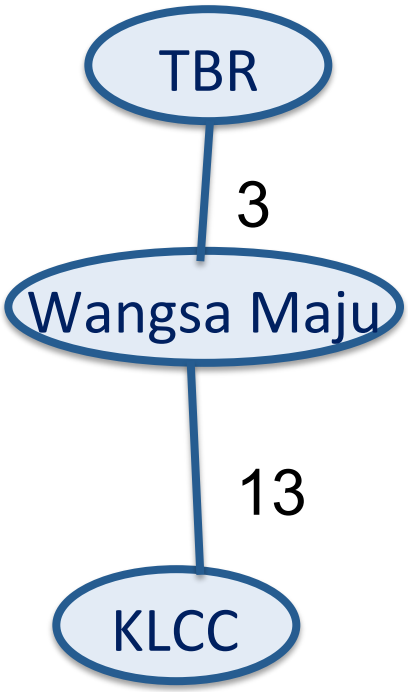
– Usually represented by a graph or a tree

– E.g.



# Step 2: Formulating Problem (1)

2 **Path** (Sequence of states connected by a sequence of actions)




From the trace of the program:

```
>>Go( _ , In(TBR), 0)
>>Go(To(Wangsa Maju, In(TBR), 3)
>>Go(To(KLCC), In(Wangsa Maju), 13)
>>Go( _ , In(KLCC), _)
```

# Step 2: Formulating Problem (1)

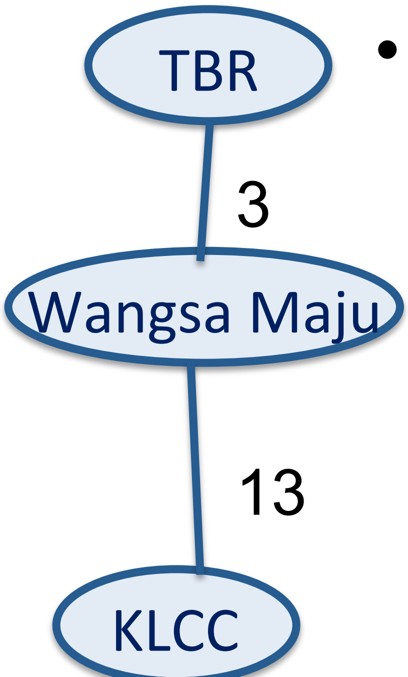
3 **Goal Test** (To determine whether a given state is a goal state)

– Sample of Goal Test Algorithm

```
If
     Go(To(KLCC), In( _ ), _ )
Then
     Go( _ , In(KLCC), _),
    show pathcost,
    return true
```

# Step 2: Formulating Problem (1)

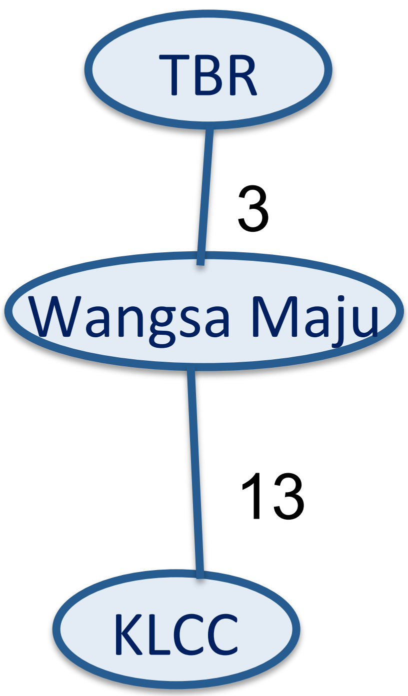
4 **Step cost** (route distance, from one state to another state)




- E.g. the step cost from TBR to Wangsa Maju = 3km

# Step 2: Formulating Problem (1)

4 **Path cost (**Sum of the costs of the individual actions along the path)




From the trace of the program:

>>Go( _ , In(TBR), 0)

pathcost = 0

>>Go(To(Wangsa Maju, In(TBR), 3)

pathcost = 3 + 0

>>Go(To(KLCC), In(Wangsa Maju), 13)

pathcost = 13 + 3 + 0

>>Go( _ , In(KLCC), _)

pathcost = 16

# Problem (2) Missionaries & Cannibals

The goal state (all on the right)


Boat can hold 1 or 2 people

missionaries >= cannibals

missionary

cannibal

# Mission

- The goal is given,
  1. What is the optimal solution?
  2. Suggest an abstraction.

# Problem Formulation

- **Initial State**
  - characterize the state:
    - the no. of missionaries on the left bank, ML
    - the number of cannibals on the left bank, CL
    - the side the boat is on, B.
  - Representation in code:
    - [ML, CL, B]
    - **start([3, 3, left]).**
    - **goal([0, 0, right]).**

# Successor function

- There are 8 possible moves:
  1. Move 1 missionary from the left bank
     - **move**([ML,CL,left],[MLNew,CL, right]).
     - E.g. **move**([3, 3, left], [2, 3, right]).
  2. Move 1 cannibal from the left bank
     - **move**([ML, CL, left], [ML, CLNew, right]).
     - E.g. **move**([3, 3, left], [3, 2, right]).
  3. And so on…

- Alternatively, we can represent in such a way
  - State([3,3, Left, 0,0])

# State space

- The set of all states reachable from the initial state

state([3, 3,left,0,0]

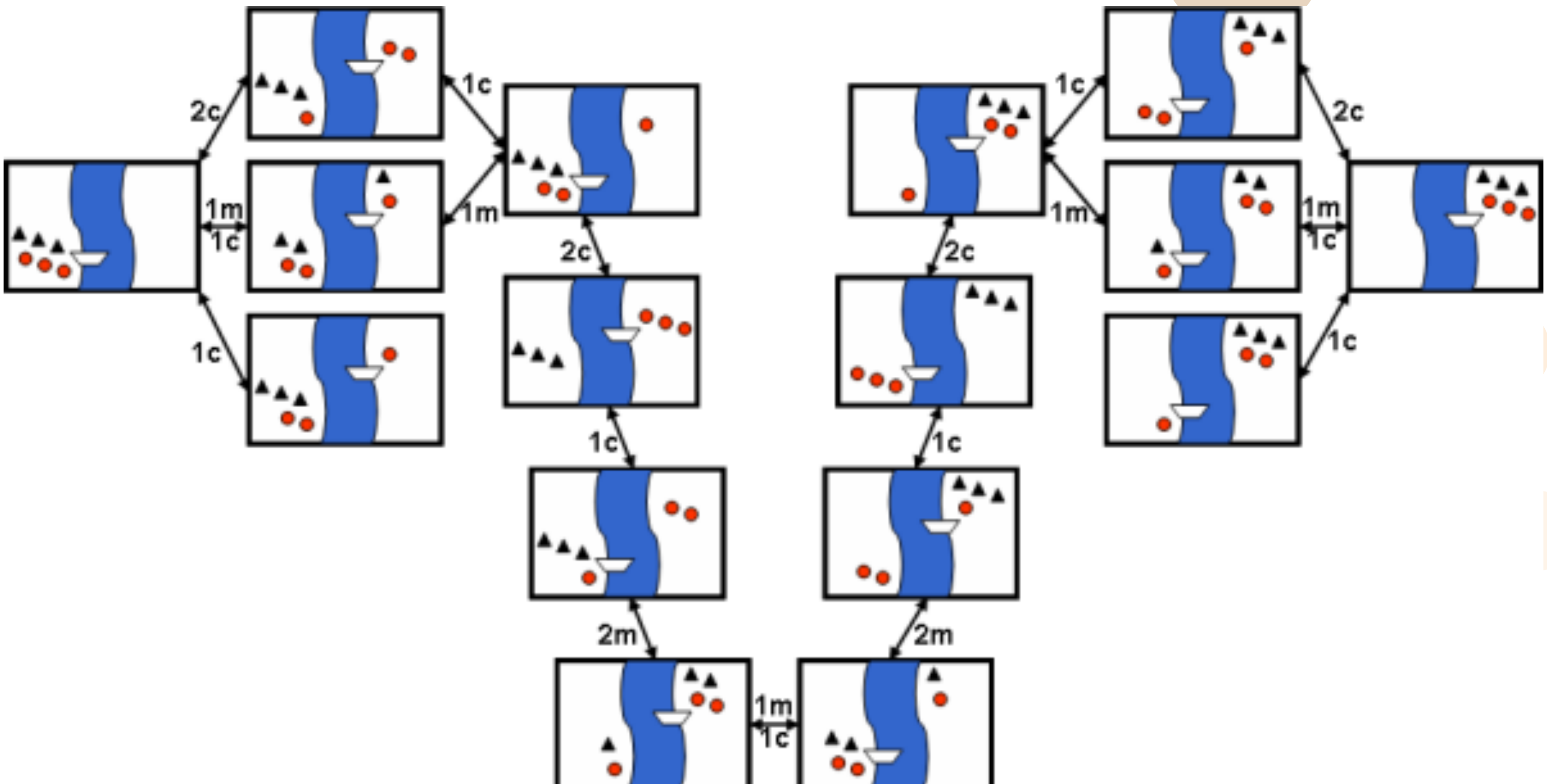
state([3, 1,right,0,2]) state([2, 2,right,1,1])

...

state([3, 2,left,0,1]) state([2, 3,left,1,0])

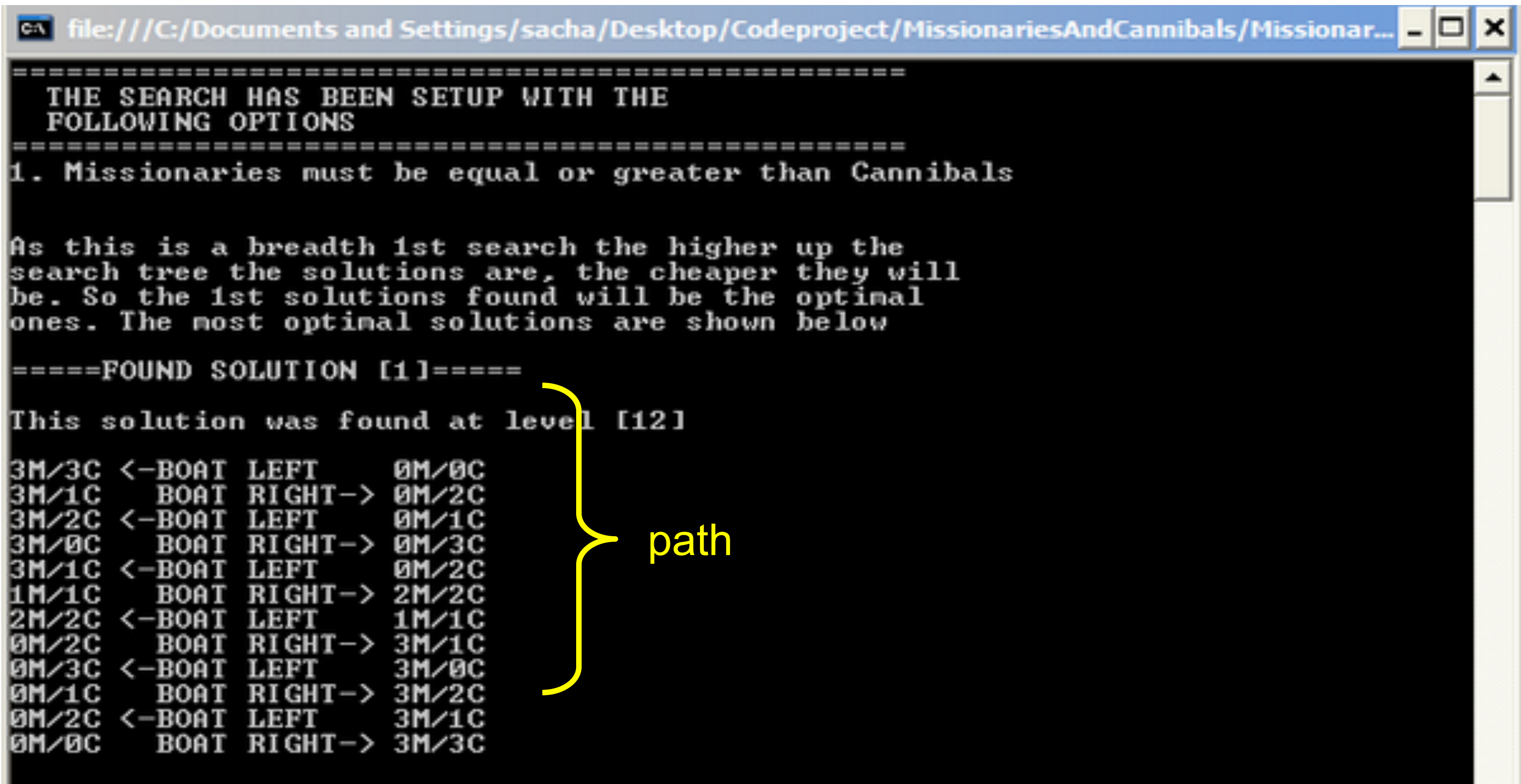
... ... ...

# Path



# Path

- Sequence of states connected by a sequence of actions



# Goal Test

- It is a function to check whether everyone is carried to the left side of the river bank, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place.

```
goal([0, 0, right]).
search([ML, CL, B], Solution):-
     goal(Solution).
```

# Cost

- **Step Cost**
  - Whenever the parent expands the child, the state level will increase by one, **so the step cost is???**
- **Path Cost**
  - The path cost is the number of steps in the path
  - What do you think is the optimal path cost?

# Search-solution-execution

## Search

- the process of looking for the best sequence of path

## Solution

- A search algorithm takes a problem as input and returns a solution in the form of an action sequence

## Execution

- Once a solution is found, the actions it recommends can be carried out.

# Searching for Solutions

- Search tree
- Search graph
- Expanding/generating states
- Search strategy

# Search Tree/Graph

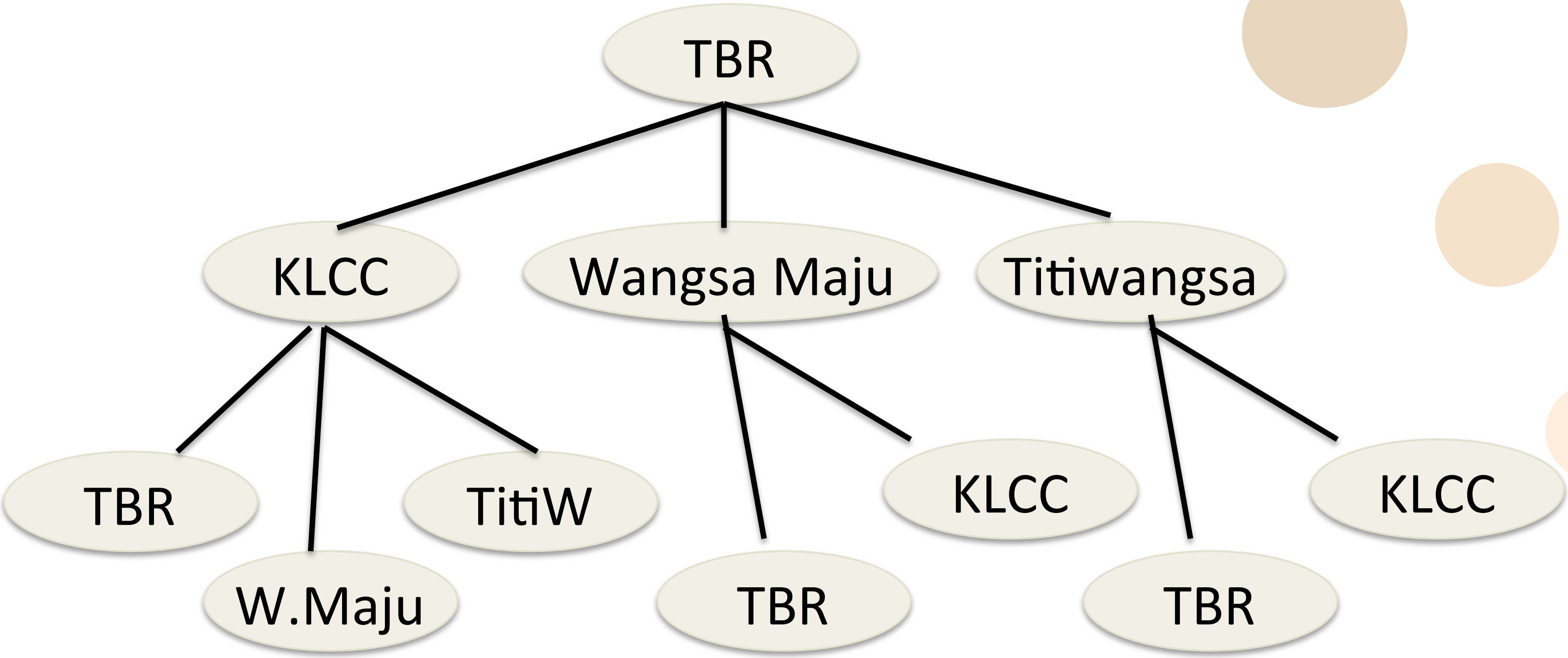
**SEARCH TREE**

- Generated by the initial state and the successor function that together define state space.

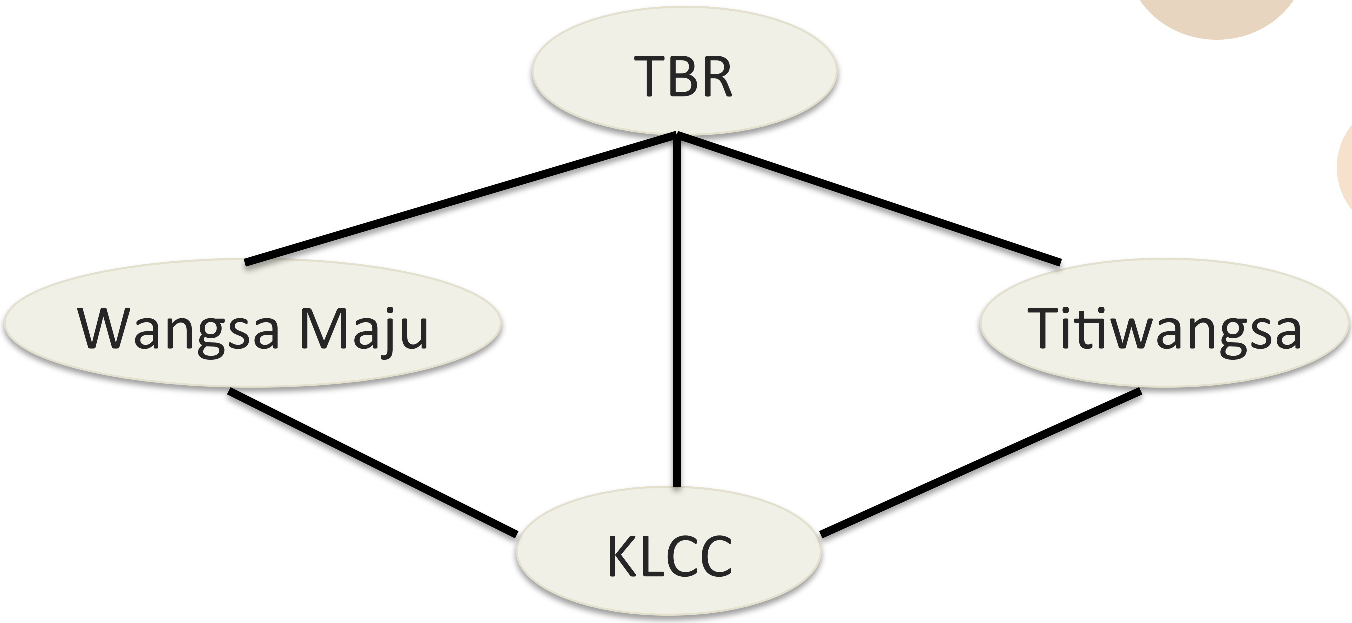
**SEARCH GRAPH**

- The same state can be reached from multiple paths.

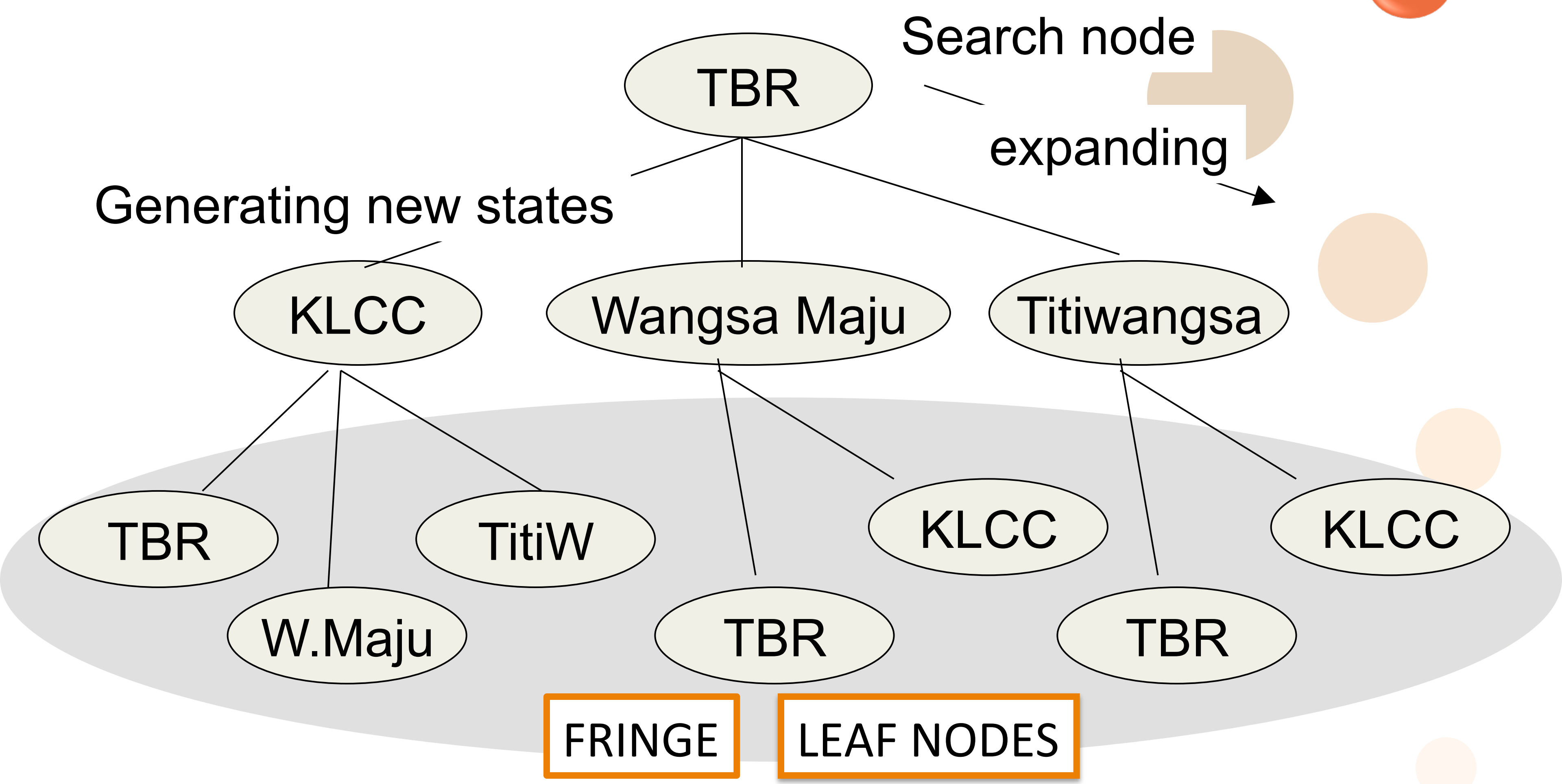
# Search Tree


TBR
KLCC
Wangsa Maju
Titiwangsa
TBR
W.Maju
TitiW
TBR
KLCC
TBR
KLCC

# Search Graph



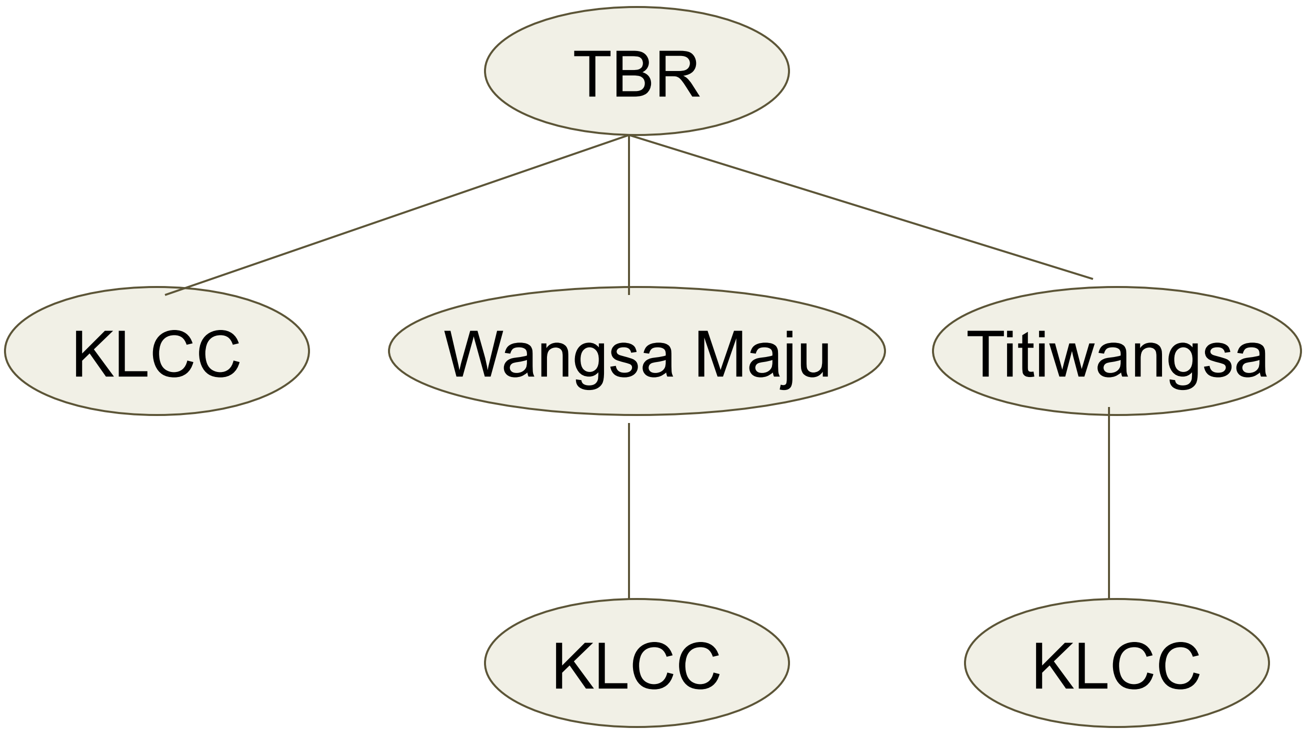
# Theories of Search Tree



# Avoiding Repeated States

- Wasting time
- Can cause a solvable problem become unsolvable if the algorithm does not detect them
- How to solve?

# Search Tree without Repeated State




Depth, d = 0

d = 1

d = 2

32/2

# Measuring problem-solving performance

## Completeness

- Is the algorithm guaranteed to find a solution when there is one?

## Optimality

- Does the strategy find the optimal solution?

## Time complexity

- How long does it take to find a solution?

## Space Complexity

- How much memory is needed to perform the search?

# Example of searching solution

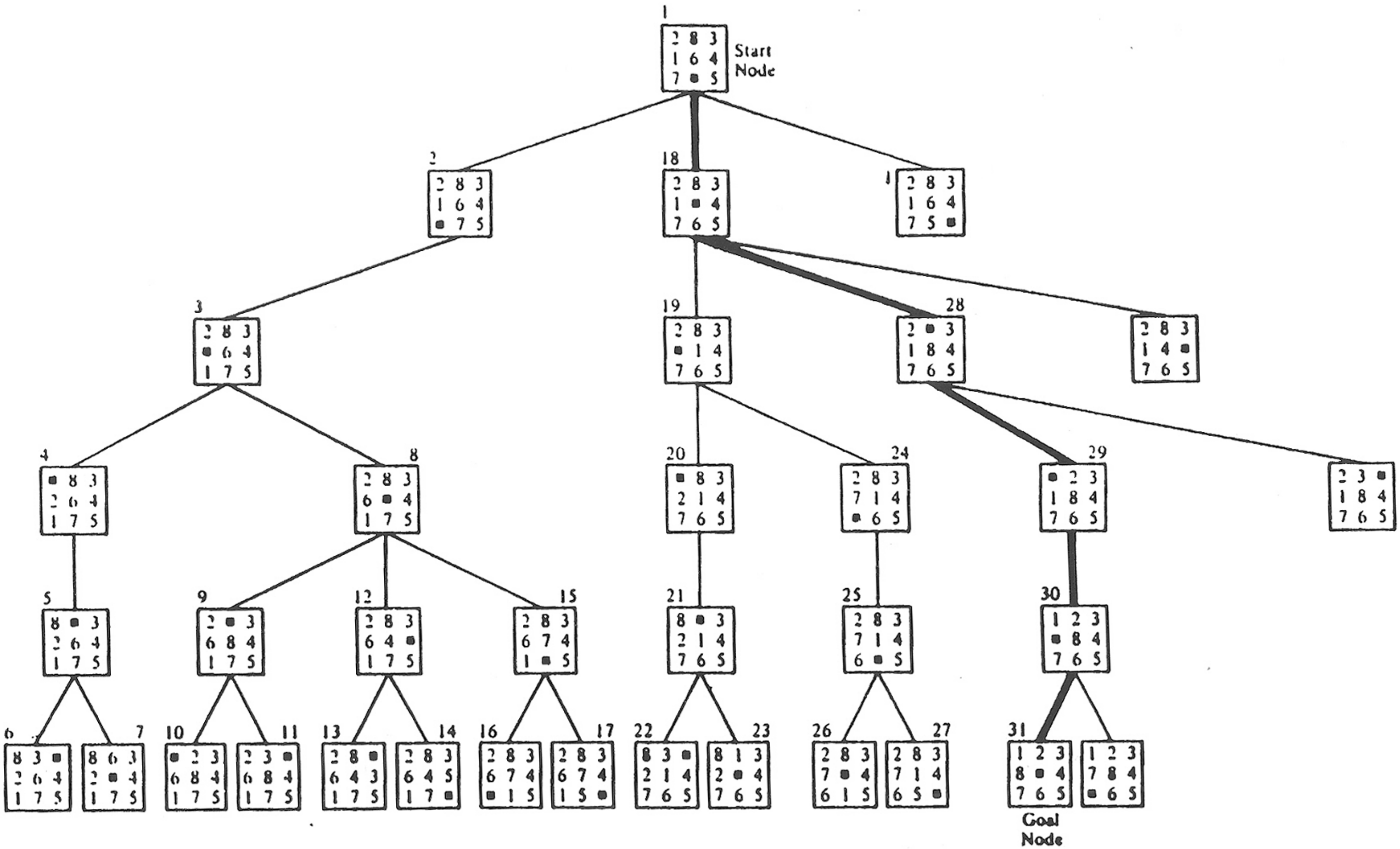



*Fig. 2.6 A search tree produced by a depth-first search.*

# Uninformed Search

Next Lecture