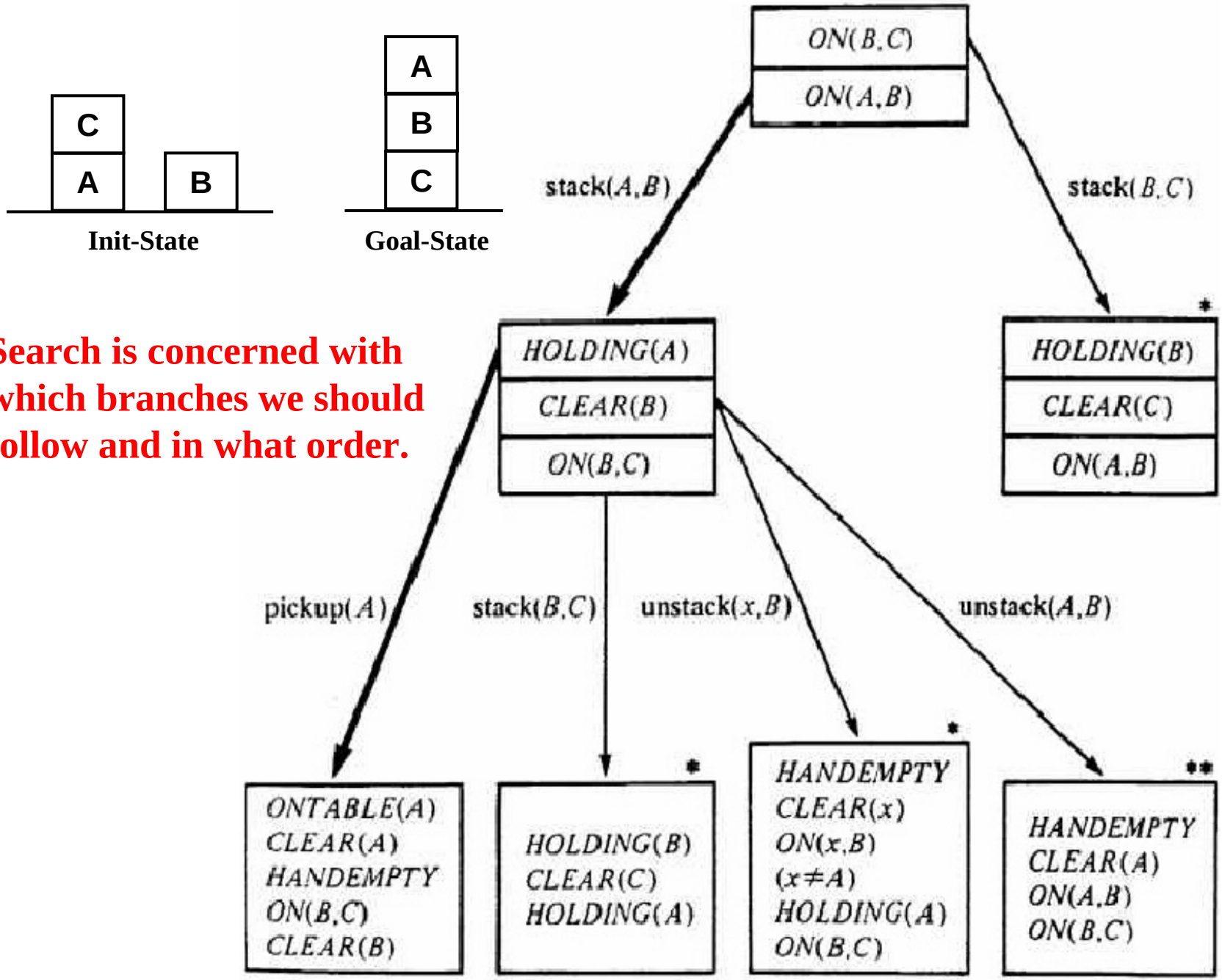

C
A
B
Init-State
A
B
C
Goal-State
Search is concerned with which branches we should follow and in what order.
ON(B,C)
ON(A,B)
stack(A,B)
stack(B,C)
HOLDING(A)
CLEAR(B)
ON(B,C)
*
HOLDING(B)
CLEAR(C)
ON(A,B)
pickup(A)
stack(B,C)
unstack(x,B)
unstack(A,B)
ONTABLE(A)
CLEAR(A)
HANDEMPTY
ON(B,C)
CLEAR(B)
*
HOLDING(B)
CLEAR(C)
HOLDING(A)
*
HANDEMPTY
CLEAR(x)
ON(x,B)
(x≠A)
HOLDING(A)
ON(B,C)
**
HANDEMPTY
CLEAR(A)
ON(A,B)
ON(B,C)

General Problem Solving

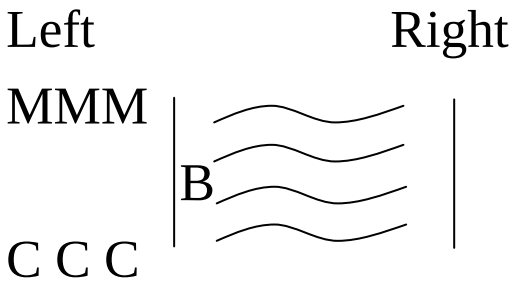
Many kinds of problems can be formulated as search problems in terms of three key ingredients

- A Starting State
- A Termination Test
- A set of operations that can be applied to change the current state of the problem

This is called state-space search.

Consider the following problem. (Missionaries and Cannibals)

There are three missionaries, three cannibals, a boat and a river.



We want to transport everyone to the right side of the river, but the boat can take only two people at a time (at least one person must bring the boat back). In addition, if cannibals outnumber missionaries at either side, then they kill the missionaries.

How can a computer solve the problem?

Choose a state representation:

L	R

Initial State:

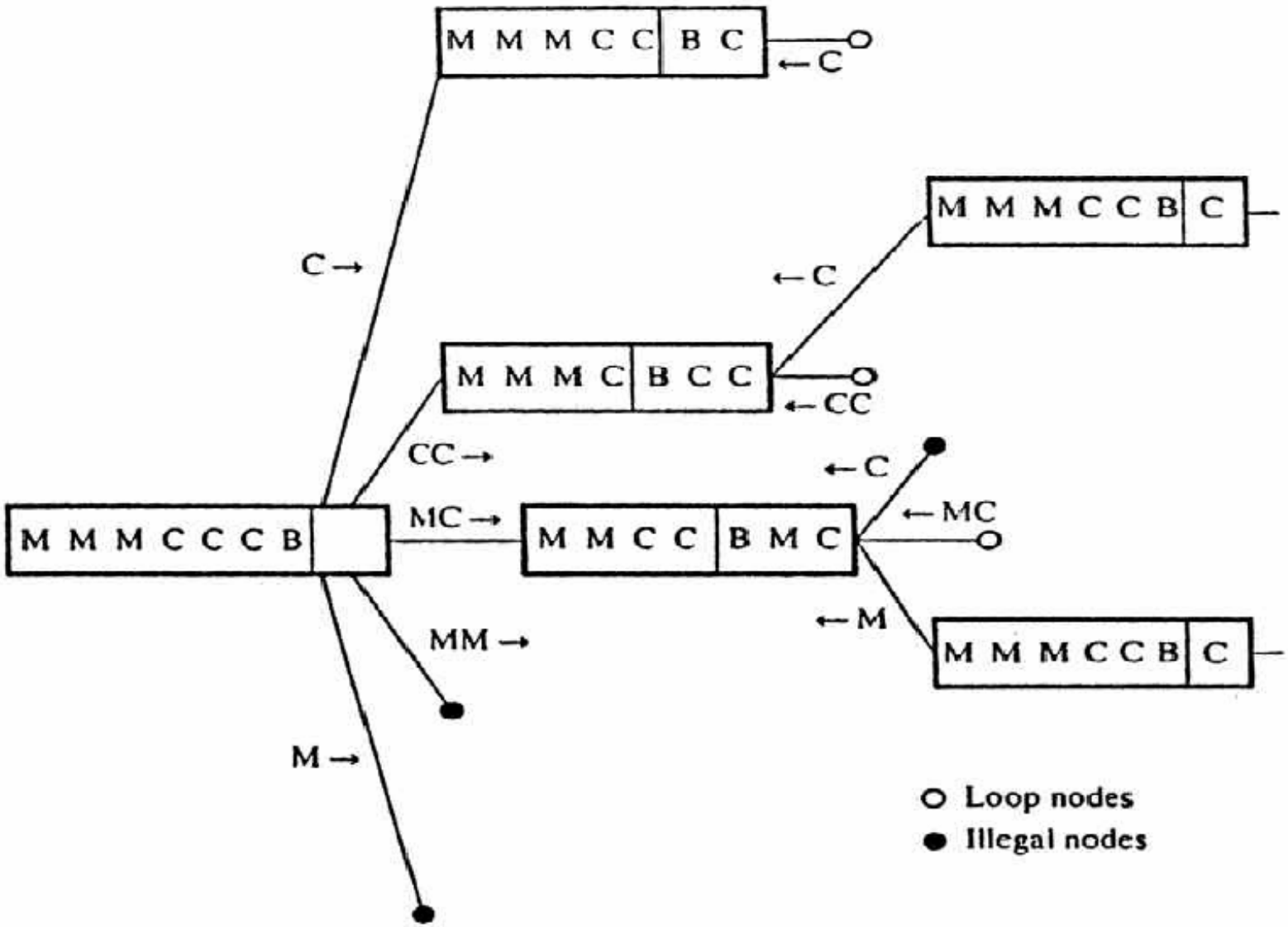
MMMCCCB	

Termination Criteria:

	BMMMCCC

Legal Operations:

C →	← C
CC →	← CC
MM →	← MM
M →	← M
MC →	← MC



Developing the missionaries and cannibals search space.

Think of state space search as a graph in which the states are nodes and the operations are arcs. The key is that the space is generated as you go, not pre-enumerated.

Question:

What is the state-space for a chess game?

What is the first node?

What are all the arcs connecting to this node?

Solutions for State-Space Search

I. Generate-and-Test:

The simplest form of state-space search is generate-and-test.

The following algorithm summarizes the method:

1. Generate a possible solution, in the form of a state.
   Ex. A new board position in chess

2. Test the success condition to see if this is a solution.

3. If the current state is a solution, then quit, else go back to step 1.

II. Depth-First Search:

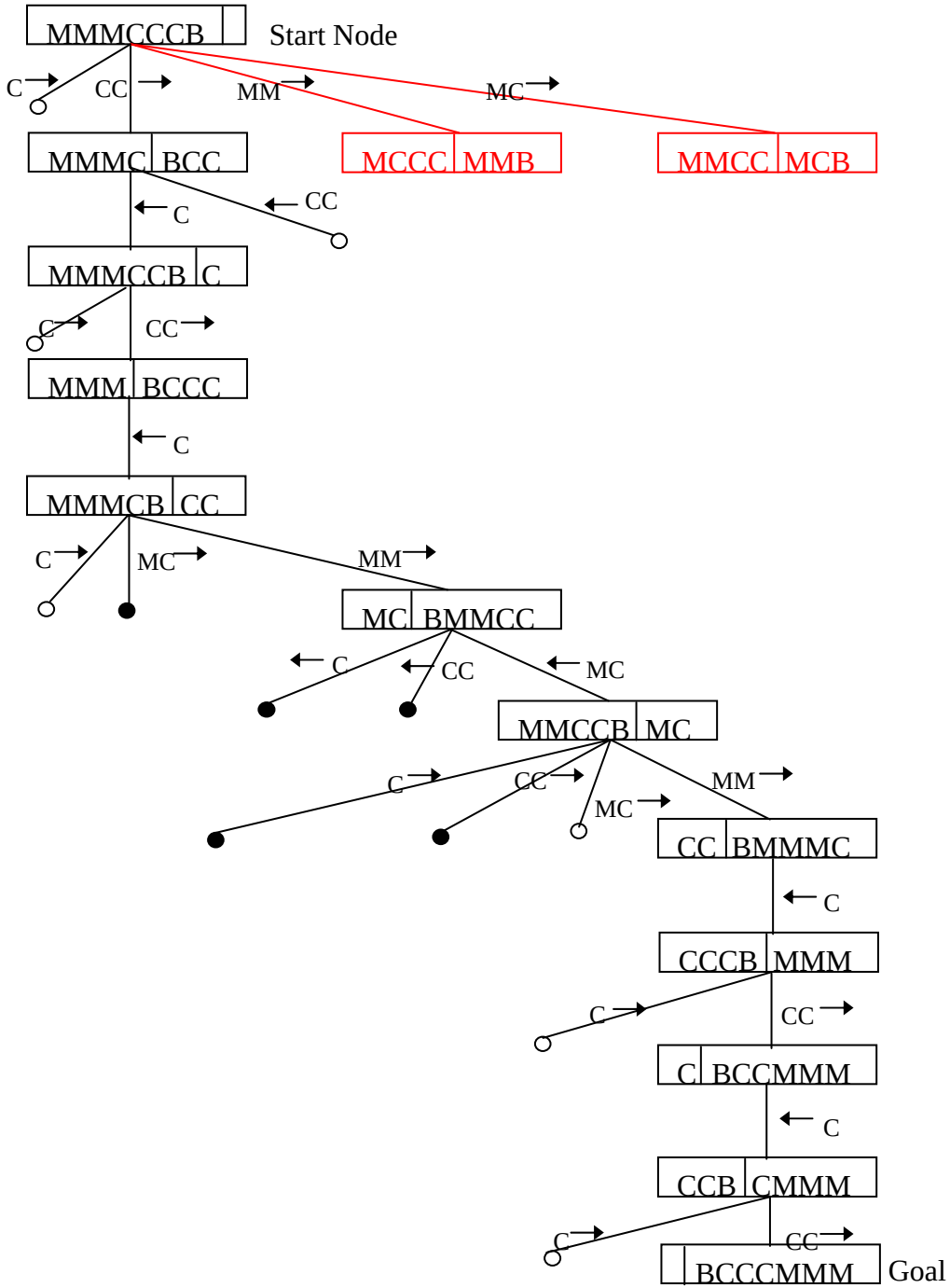
At any given node N, consider the children of N before considering its siblings.

Algorithm:

```
Depth_FS(success( ), current, pending)
    if success(current) = true, then done
    else pending = expand(current) +
                                append old pending to end of pending
    if pending = ( )  them failed
    else Depth_FS(success( ), first(pending), rest(pending))
```

Question:

How does depth-first search work for the missionaries and the cannibals?


MMMCCCB
Start Node
C
CC
MM
MC
MMMC BCC
MCCC MMB
MMCC MCB
C
CC
MMMCCB C
C
CC
MMM BCCC
C
MMMCB CC
C
MC
MM
MC BMMCC
C
CC
MC
MMCCB MC
C
CC
MC
MM
CC BMMMC
C
CCCB MMM
C
CC
C BCCMMM
C
CCB CMMM
C
CC
BCCCMMM
Goal

III. Breadth-First search:

At any node N, consider N's siblings before considering its children. Breadth-first search goes through the state space layer by layer.

Question:

Write the general algorithm for breadth-first search.

Algorithm for Breadth-First search:

```
Breadth_FS(success( ), current, pending)
    if success(current) = true, then done
    else pending = pending + expand(current) and
                                append to end of pending
    if pending = ( ) then failed
    else Breadth_FS(success( ), first(pending), rest(pending))
```


Admissibility:

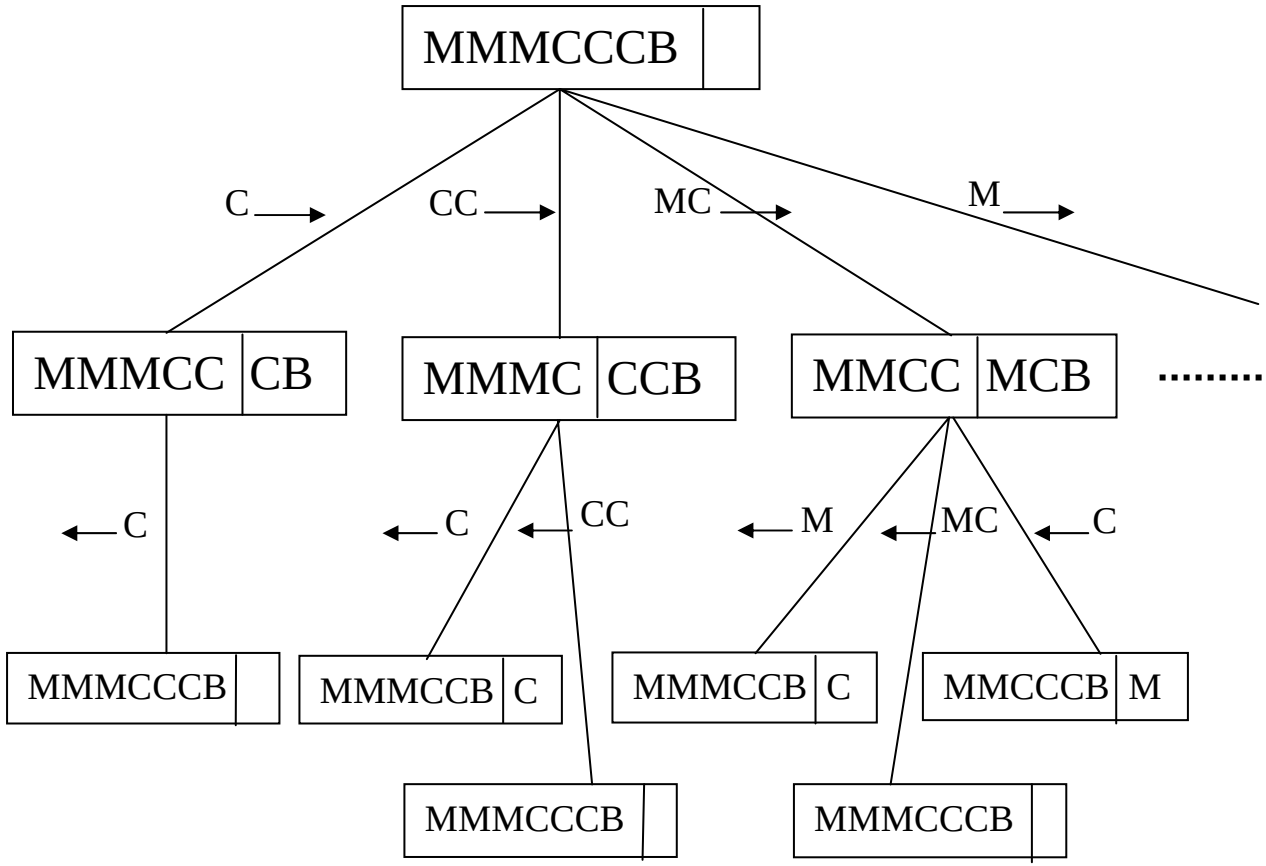
A method which finds the shortest (least cost) solution, if one exists, is called admissible.

Question:

Which one of Generate-and-Test, depth-first or breadth-first search methods is admissible?

Question:

How does the search space (breadth-first) look like for the missionaries and the cannibals?



Breadth First Search

The main problem with the above exhaustive search methods:

Combinatorial Explosion.

(The number of nodes grows exponentially.)

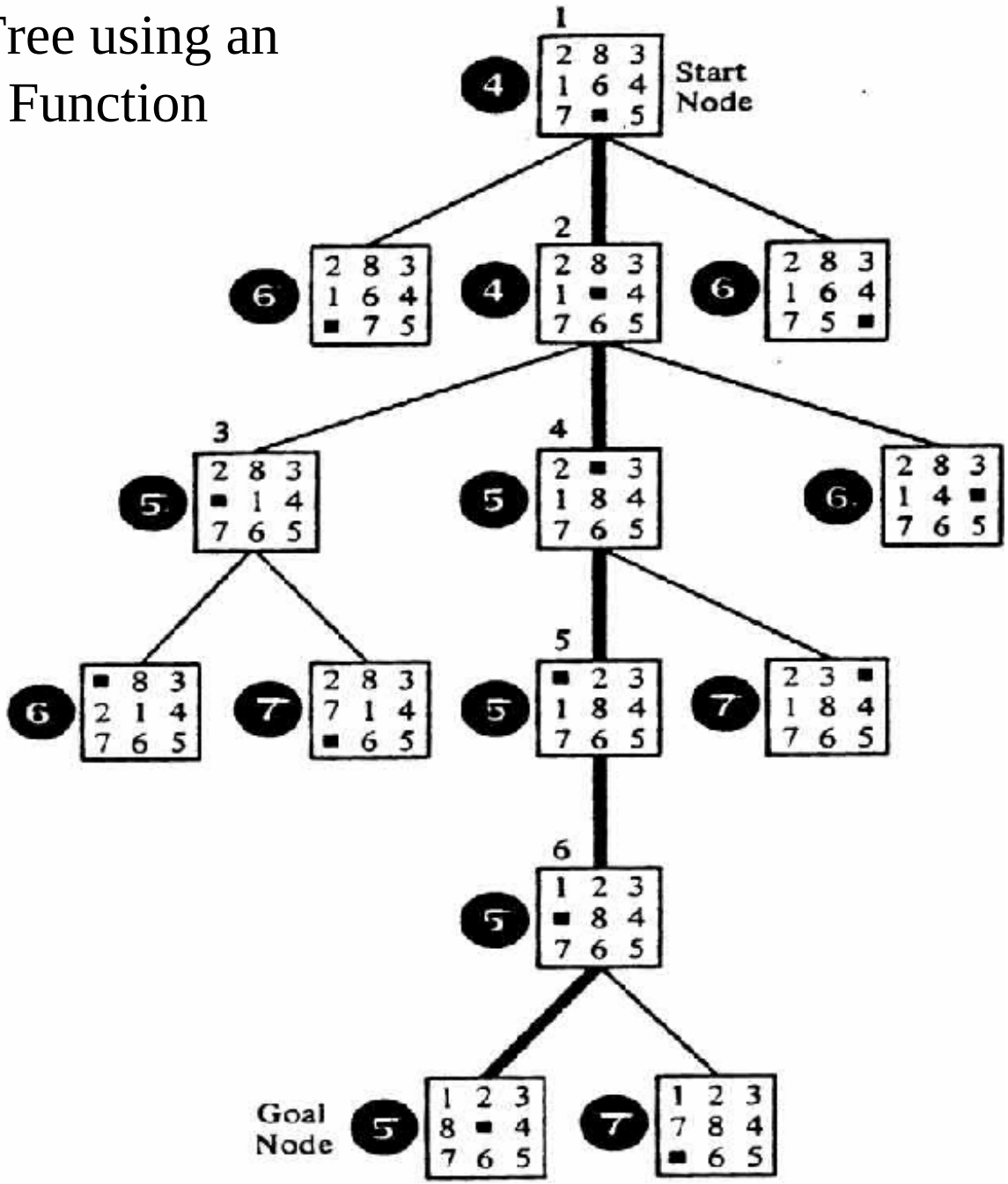
Hill Climbing

This involves giving the program an evaluation function which it can apply to the current state of the problem to obtain a rough estimate of how well things are going.

An algorithm for hill-climbing:

1. Generate a possible solution (same as step 1 of Generate-and-test).
2. Apply possible operation to this point in state-space that generates a new set of possible solutions.
3. If any state is a solution (in this set), then quit, else take the best state from the set, and make it the current state.

A Search Tree using an Evaluation Function



Problems with Hill-Climbing:

1. Your evaluation function may not be a faithful estimate of the goodness of the current state of the problem.

   Ex: In chess, I may have more pieces than you may have, but you may have a better board position. Thus evaluation function based on pieces may not work well.

2. Local Minima: The evaluation function may take us to a local minima, while the solution may require us to go down, and find the goal on a lower point.