4 HEURISTIC SEARCH

Fig 4.1 First three levels of the tic-tac-toe state space reduced by symmetry

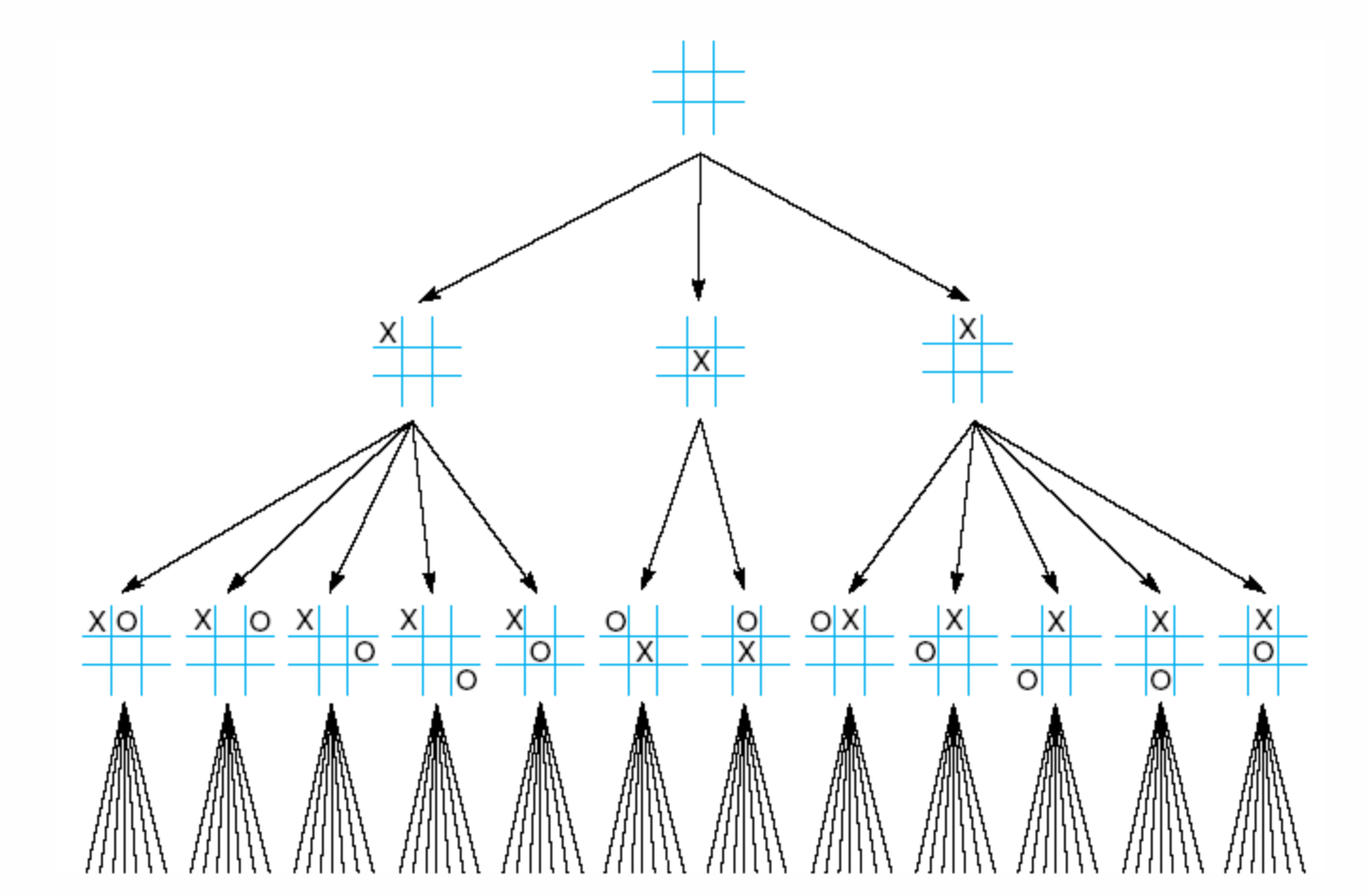


Fig 4.2 The “most wins” heuristic applied to the first children in tic-tac-toe.

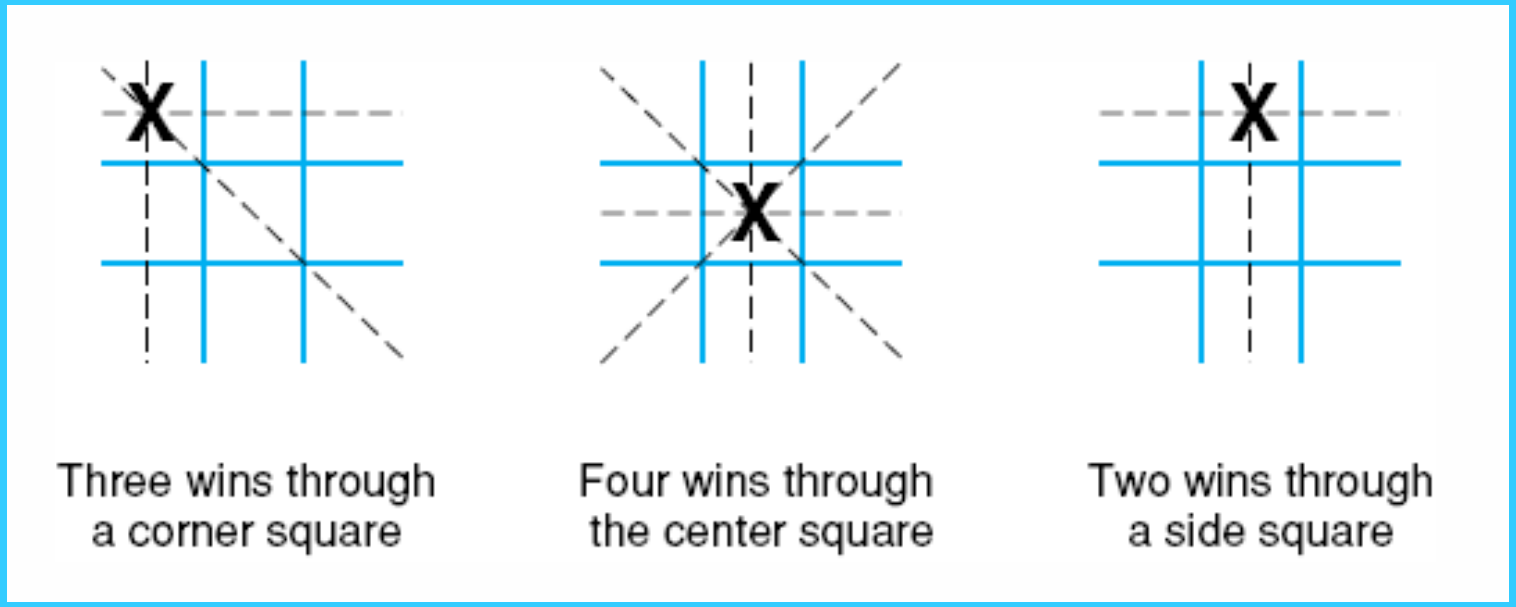
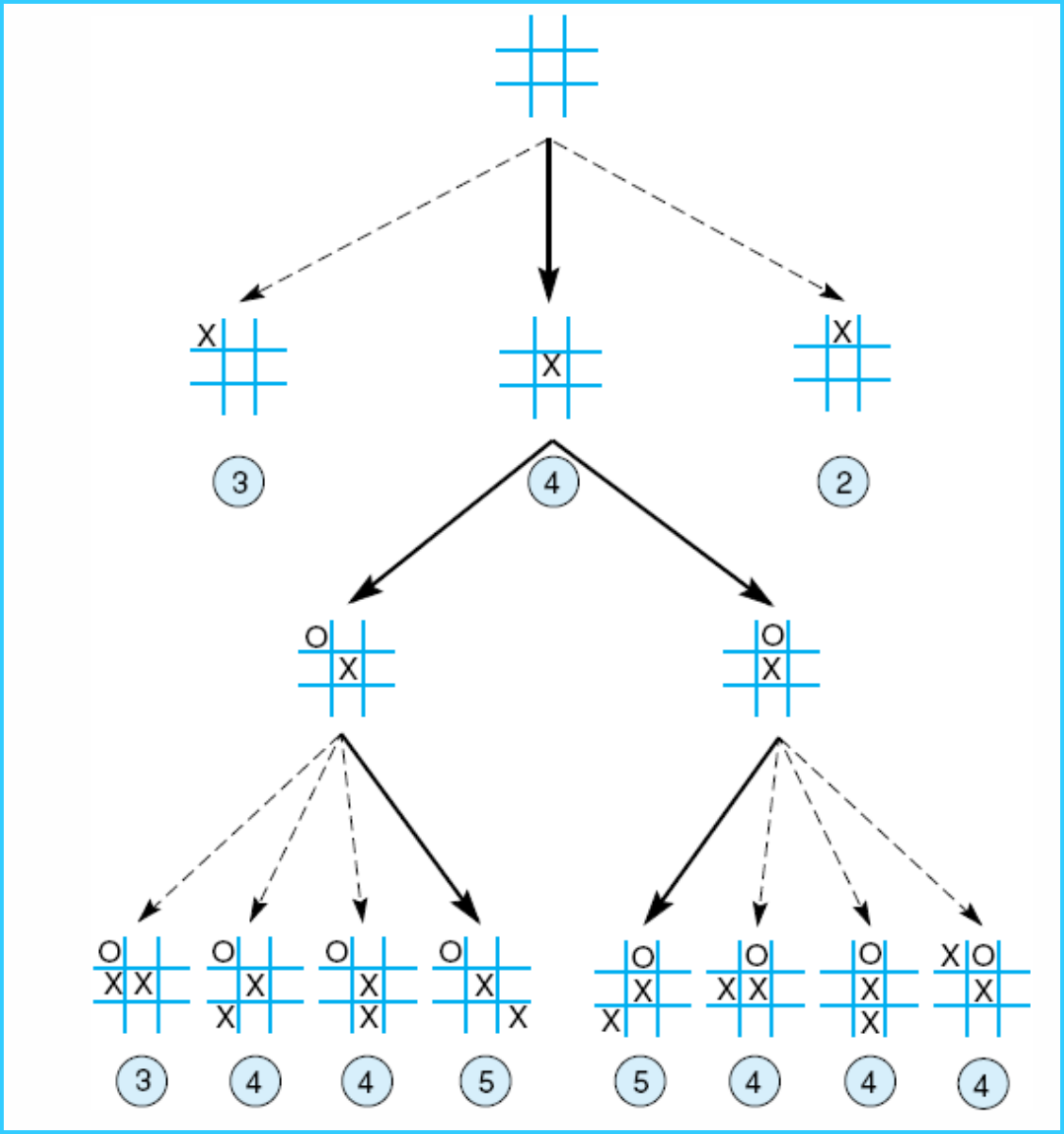
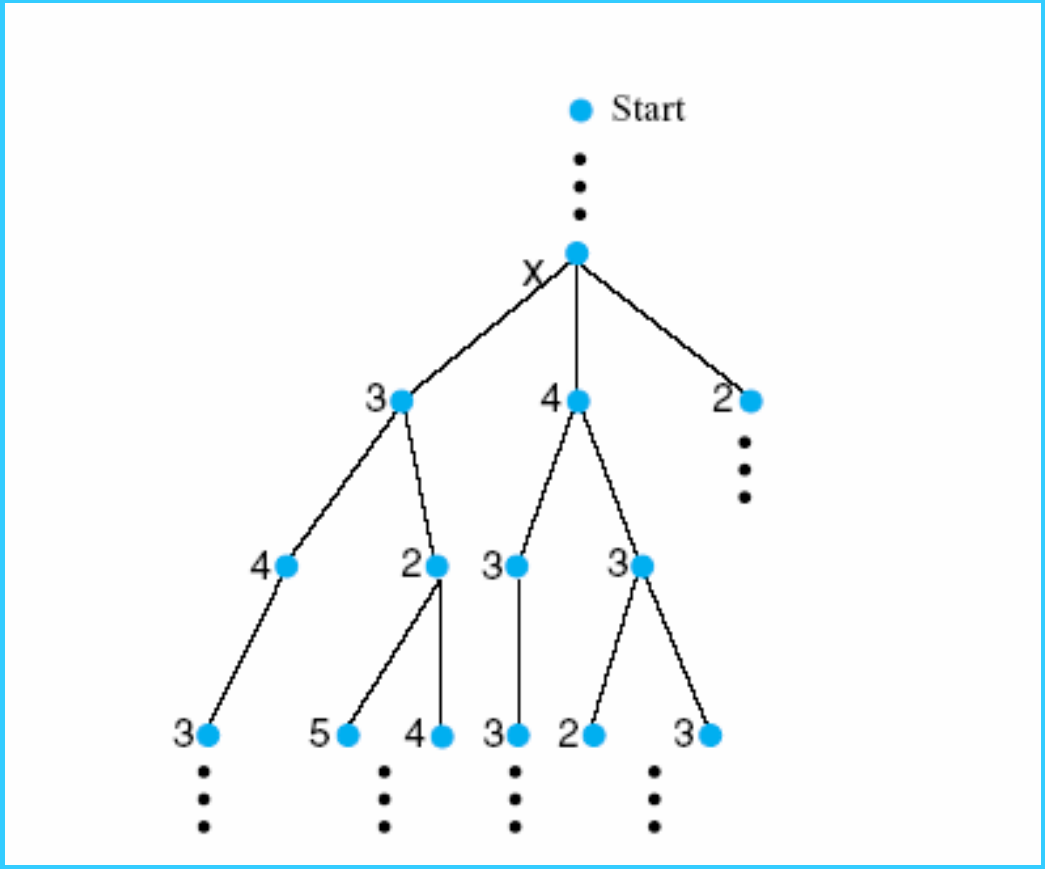


Fig 4.3 Heuristically reduced state space for tic-tac-toe.



4

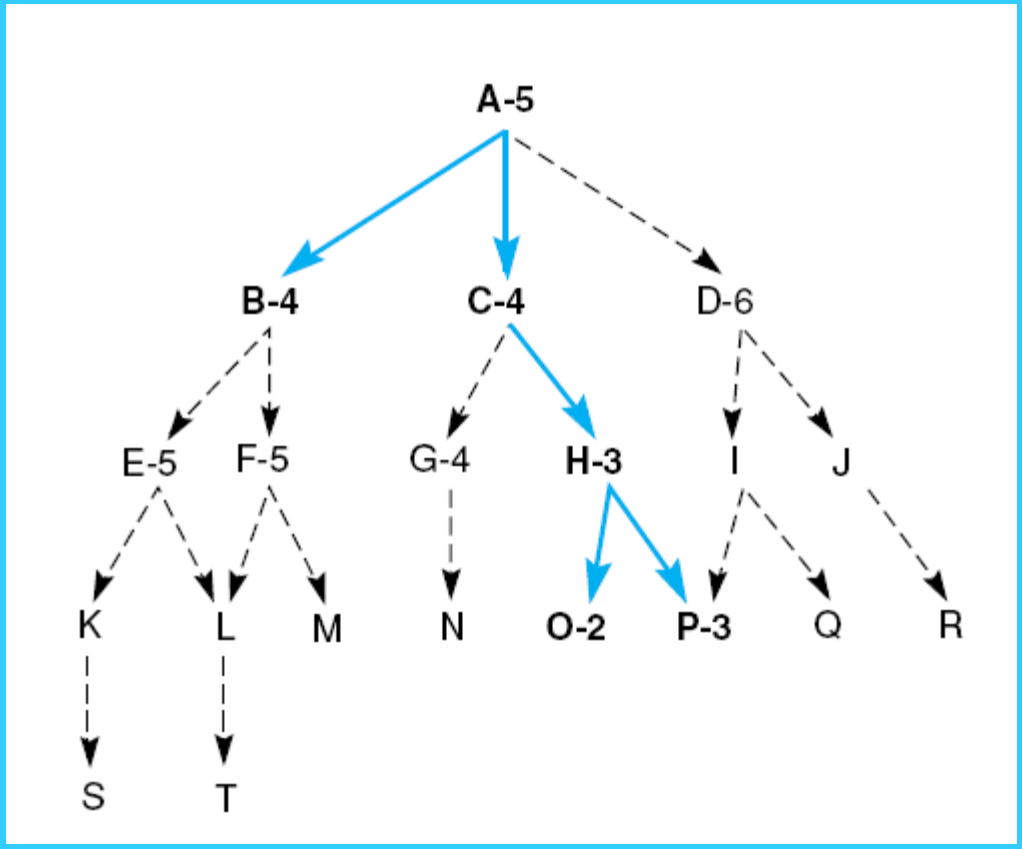
Fig 4.4 The local maximum problem for hill-climbing with 3-level look ahead



```
function best_first_search;

begin
   open := [Start];                                          % initialize
   closed := [ ];
   while open ≠ [ ] do                                       % states remain
      begin
         remove the leftmost state from open, call it X;
         if X = goal then return the path from Start to X
         else begin
                  generate children of X;
                  for each child of X do
                  case
                        the child is not on open or closed:
                           begin
                                 assign the child a heuristic value;
                                 add the child to open
                           end;
                        the child is already on open:
                           if the child was reached by a shorter path
                           then give the state on open the shorter path
                        the child is already on closed:
                           if the child was reached by a shorter path then
                              begin
                                 remove the state from closed;
                                 add the child to open
                              end;
                  end;                                       % case
                  put X on closed;
                  re-order states on open by heuristic merit (best leftmost)
               end;
return FAIL                                                  % open is empty
end.
```

Fig 4.10 Heuristic search of a hypothetical state space.



A trace of the execution of best_first_search for Figure 4.4

1. **open = [A5]; closed = [ ]**
2. **evaluate A5; open = [B4,C4,D6]; closed = [A5]**
3. **evaluate B4; open = [C4,E5,F5,D6]; closed = [B4,A5]**
4. **evaluate C4; open = [H3,G4,E5,F5,D6]; closed = [C4,B4,A5]**
5. **evaluate H3; open = [O2,P3,G4,E5,F5,D6]; closed = [H3,C4,B4,A5]**
6. **evaluate O2; open = [P3,G4,E5,F5,D6]; closed = [O2,H3,C4,B4,A5]**
7. **evaluate P3; the solution is found!**

Fig 4.11 Heuristic search of a hypothetical state space with open and closed states highlighted.

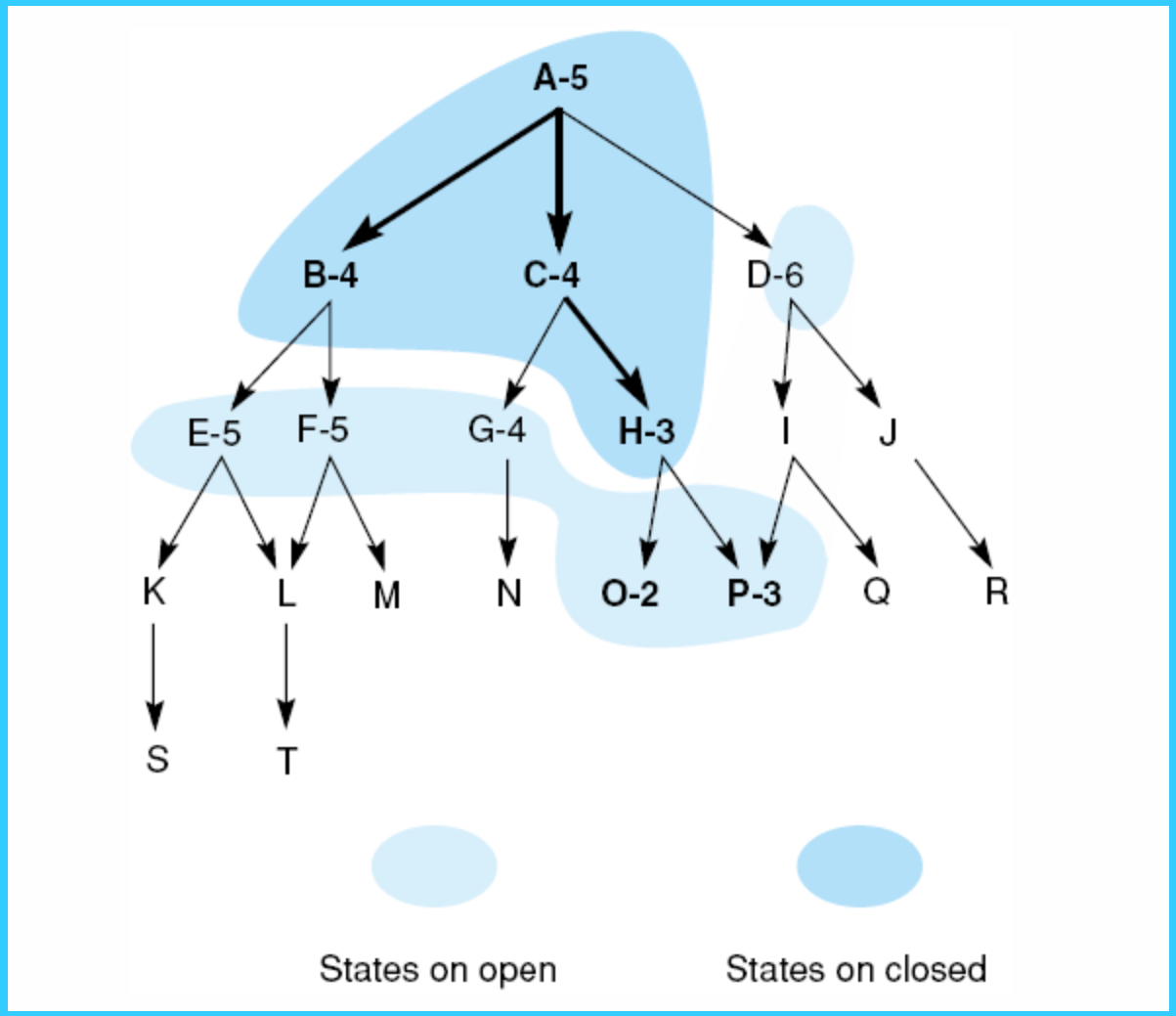


Fig 4.12 The start state, first moves, and goal state for an example-8 puzzle.

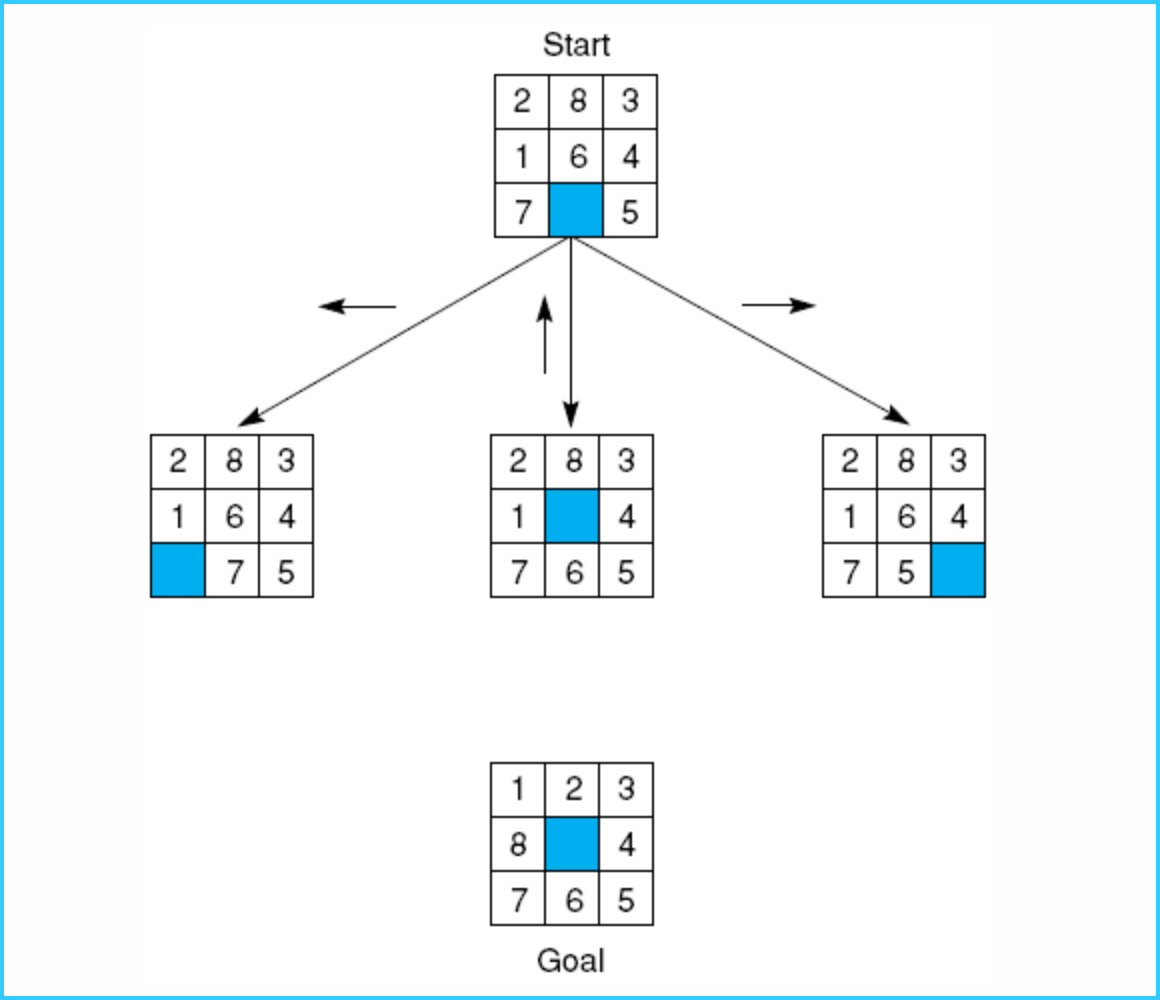


Fig 4.14 Three heuristics applied to states in the 8-puzzle.

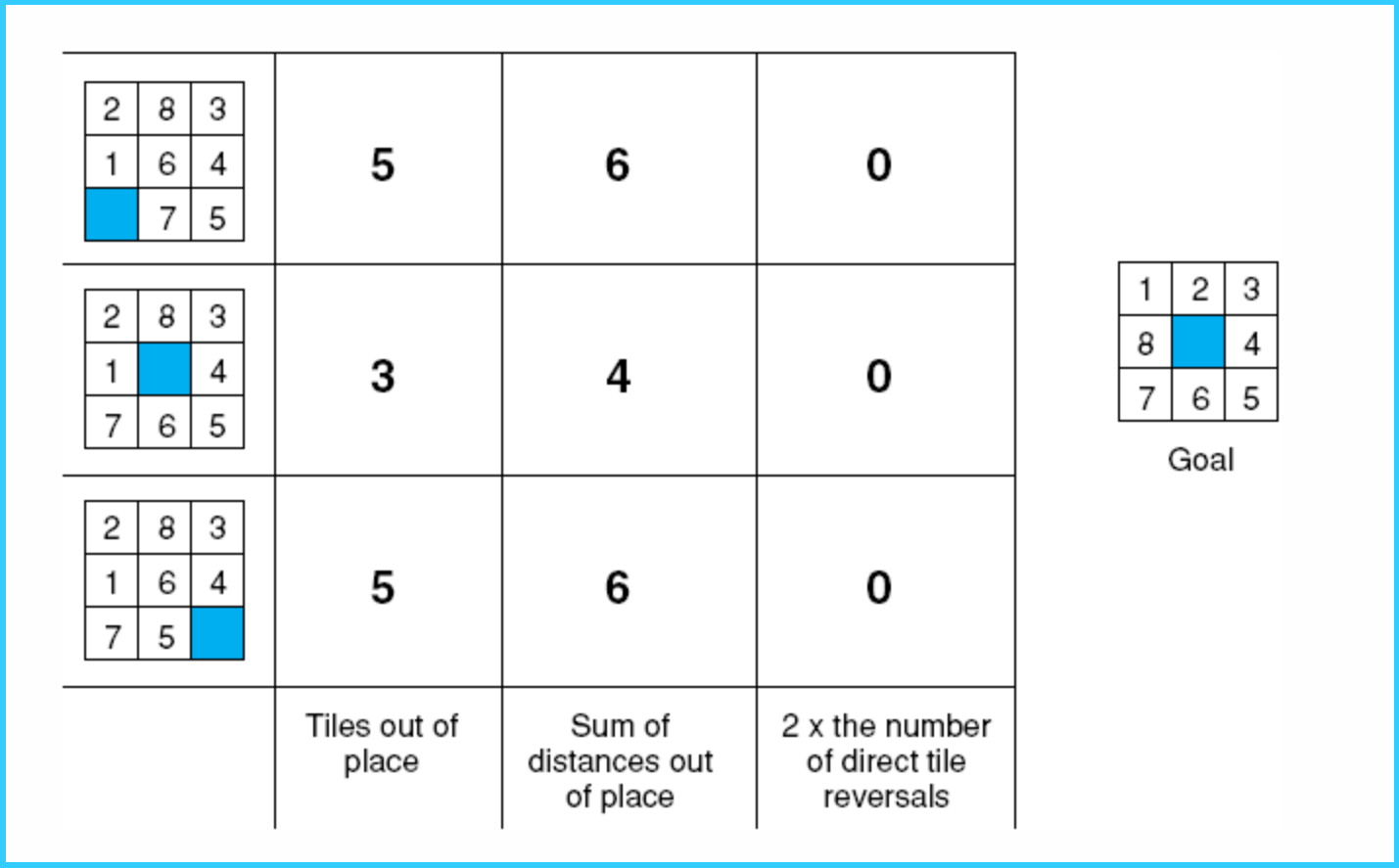


Fig 4.15 The heuristic f applied to states in the 8-puzzle.

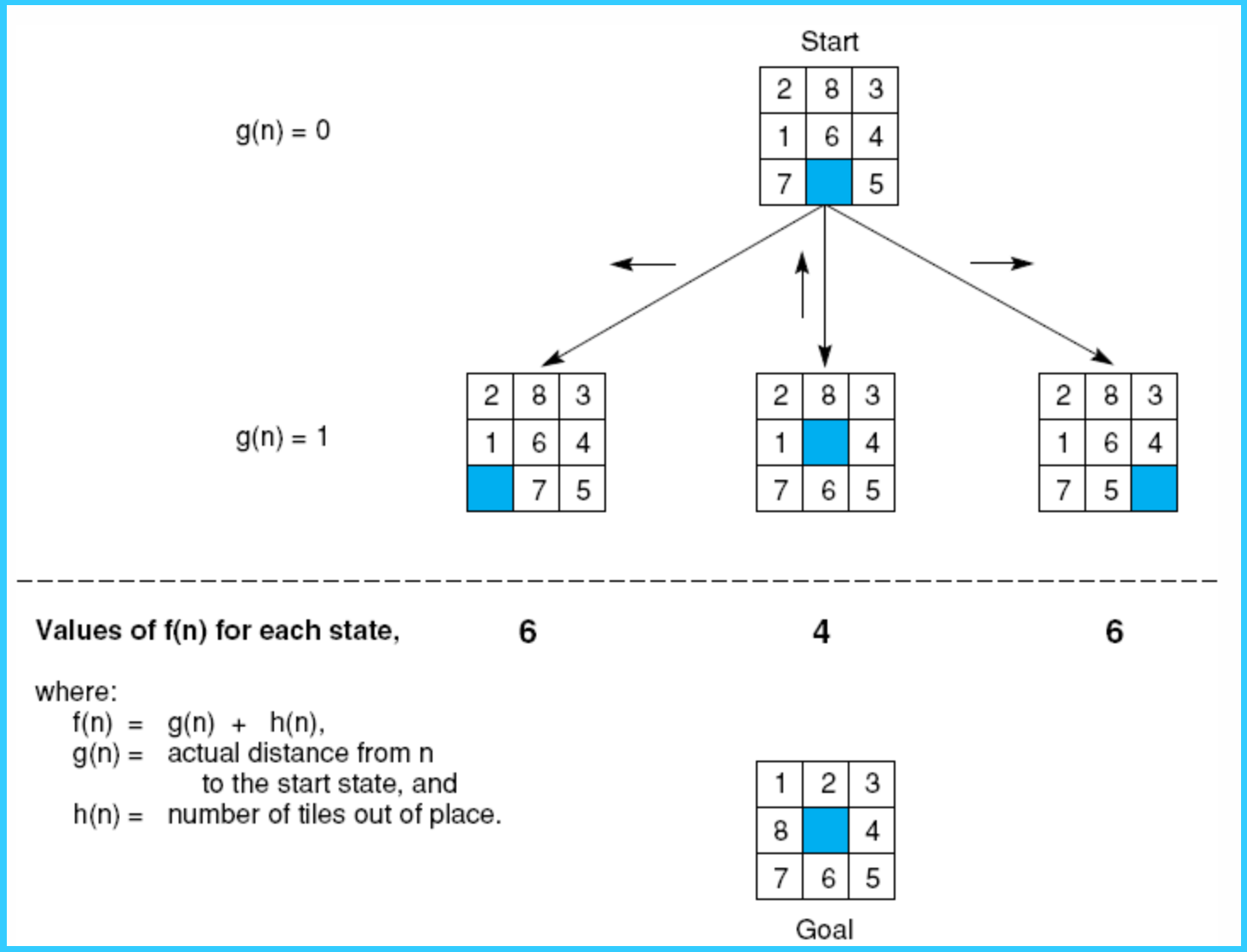


Fig 4.16 State space generated in heuristic search of the 8-puzzle graph.

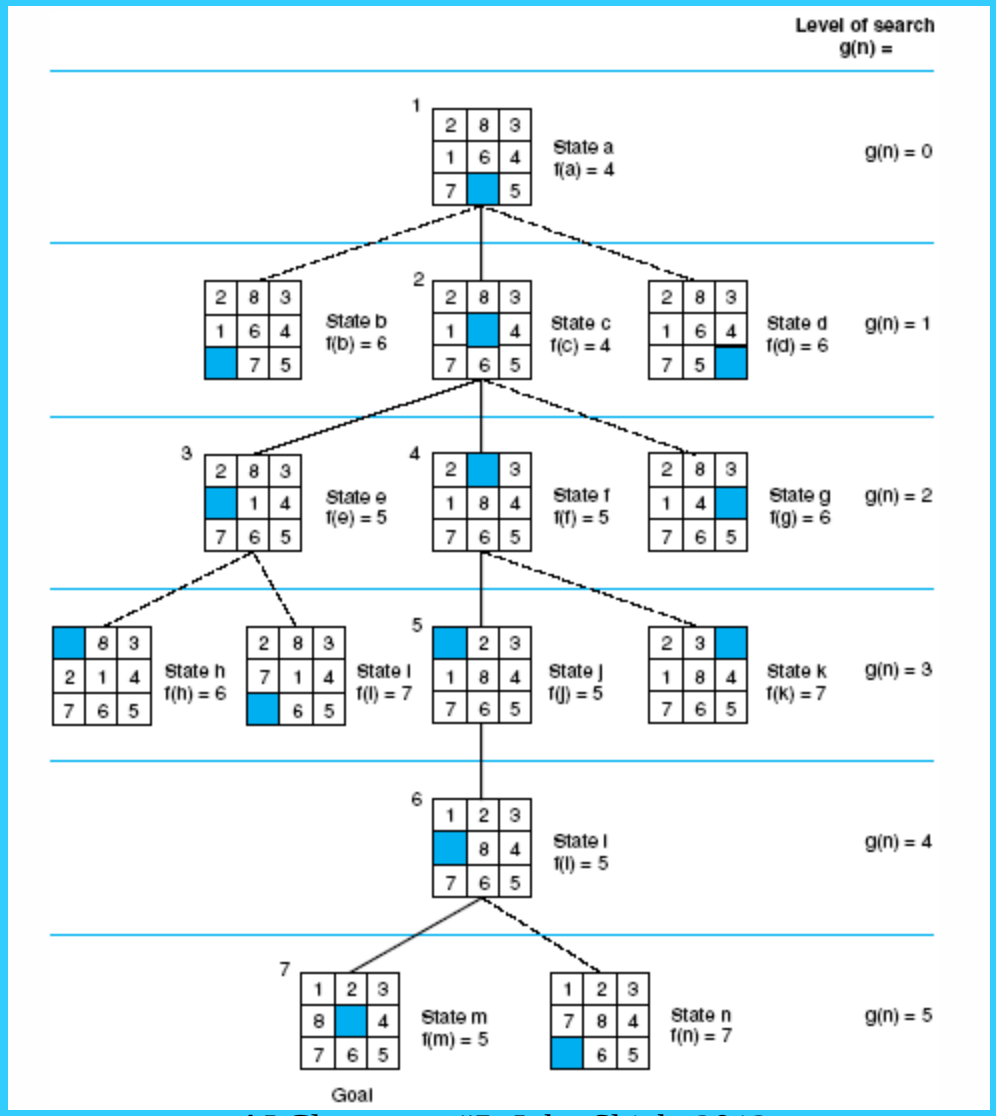
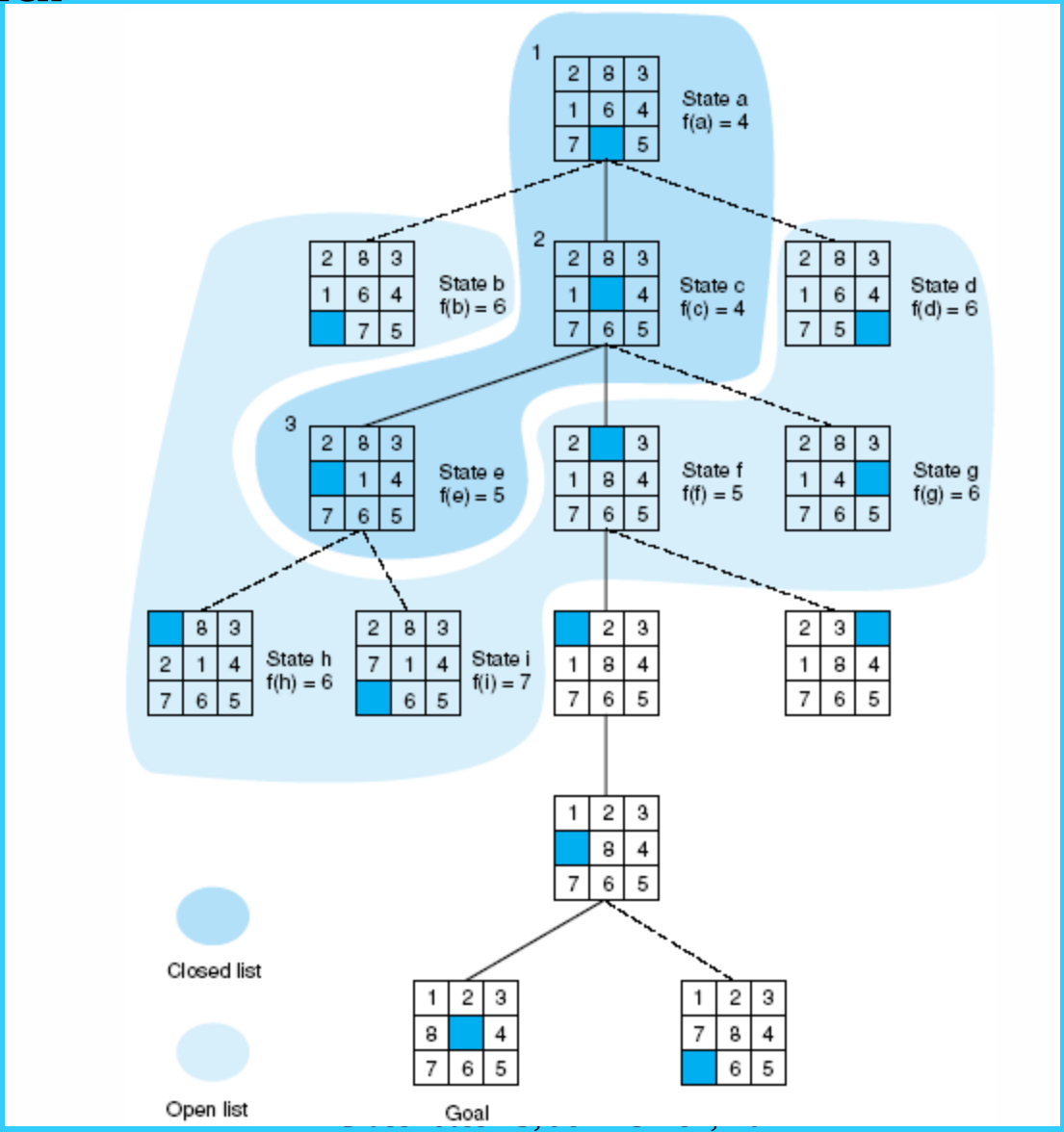


Fig 4.17 Open and closed as they appear after the 3rd iteration of heuristic search



DEFINITION

ALGORITHM A, ADMISSIBILITY, ALGORITHM A*

Consider the evaluation function f(n) = g(n) + h(n), where

n is any state encountered in the search.

g(n) is the cost of n from the start state.

h(n) is the heuristic estimate of the cost of going from n to a goal.

If this evaluation function is used with the best_first_search algorithm of Section 4.1, the result is called *algorithm A*.

A search algorithm is *admissible* if, for any graph, it always terminates in the optimal solution path whenever a path from the start to a goal state exists.

If algorithm A is used with an evaluation function in which h(n) is less than or equal to the cost of the minimal path from n to the goal, the resulting search algorithm is called *algorithm* A* (pronounced "A STAR").

It is now possible to state a property of A* algorithms:

All A* algorithms are admissible.

DEFINITION

MONOTONICITY

A heuristic function h is monotone if

1. For all states n_i and n_j, where n_j is a descendant of n_i,

   $h(n_i) - h(n_j) \leq cost(n_i,n_j)$,

   where $cost(n_i,n_j)$ is the actual cost (in number of moves) of going from state n_i to n_j.

2. The heuristic evaluation of the goal state is zero, or h(Goal) = 0.

DEFINITION

INFORMEDNESS

For two A* heuristics h_1 and h_2, if $h_1(n) \leq h_2(n)$, for all states n in the search space, heuristic h_2 is said to be *more informed* than h_1.

Fig 4.18 Comparison of state space searched using heuristic search with space searched by breadth-first search. The proportion of the graph searched heuristically is shaded. The optimal search selection is in bold. Heuristic used is f(n) = g(n) + h(n) where h(n) is tiles out of place.

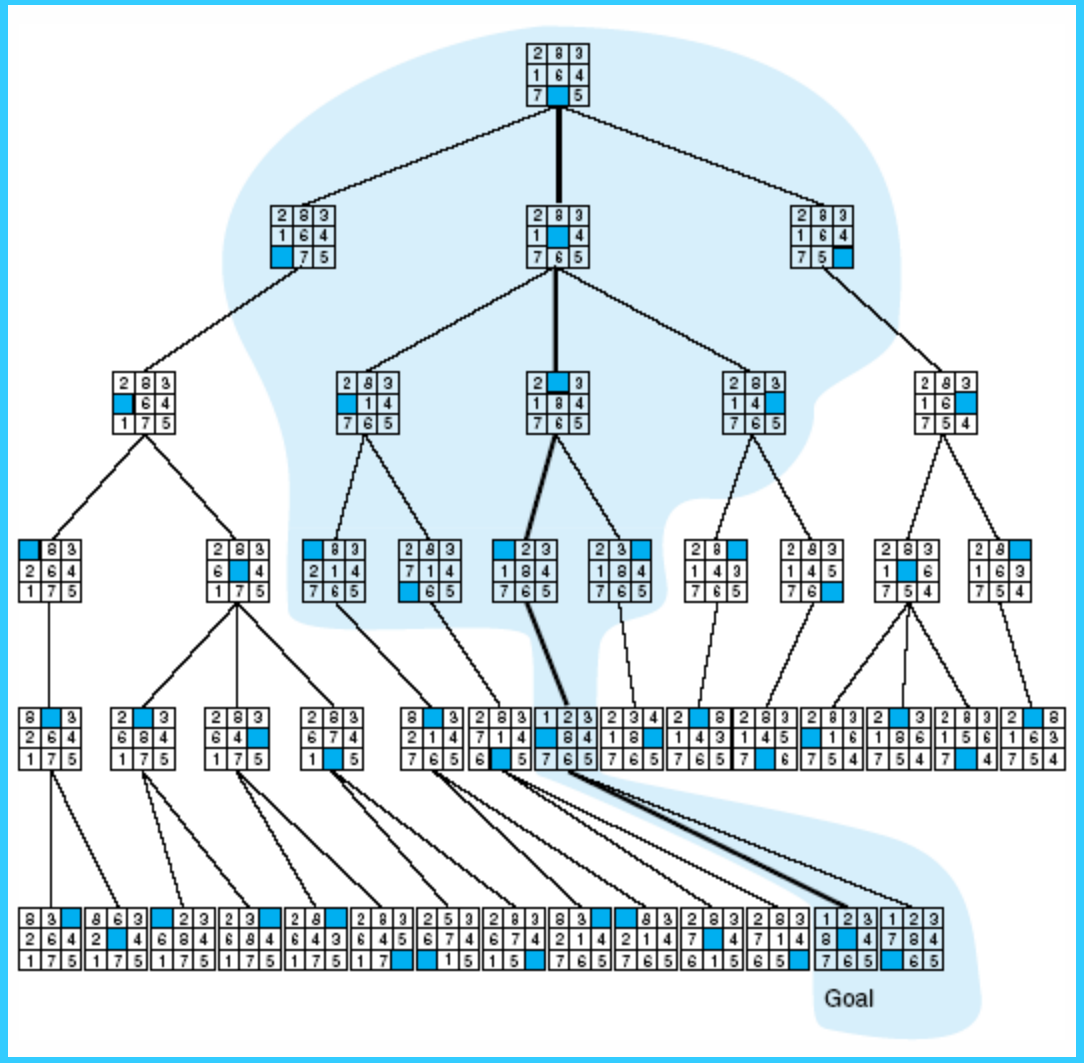


Fig 4.19 State space for a variant of nim. Each state partitions the seven matches into one or more piles.

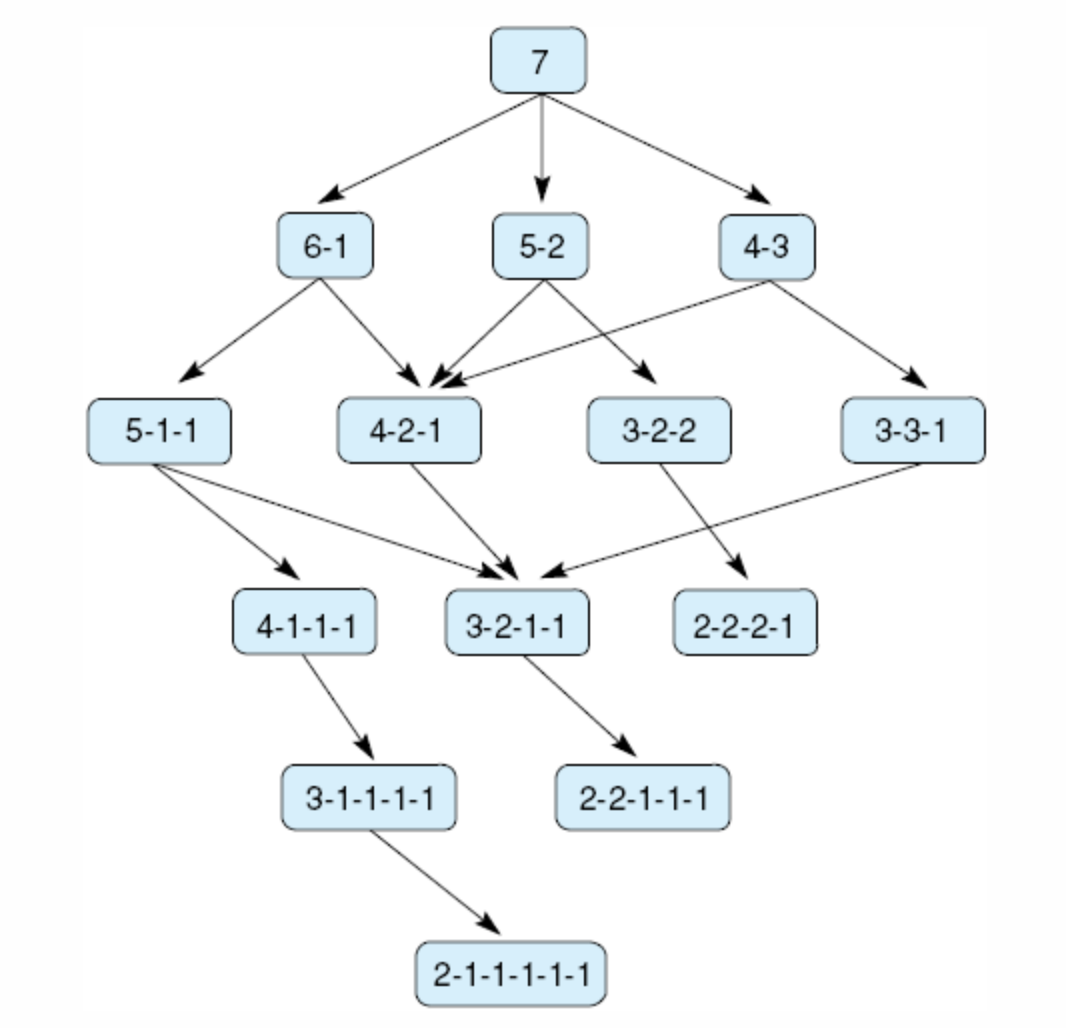


Fig 4.20 Exhaustive minimax for the game of nim. Bold lines indicate forced win for MAX. Each node is marked with its derived value (0 or 1) under minimax.

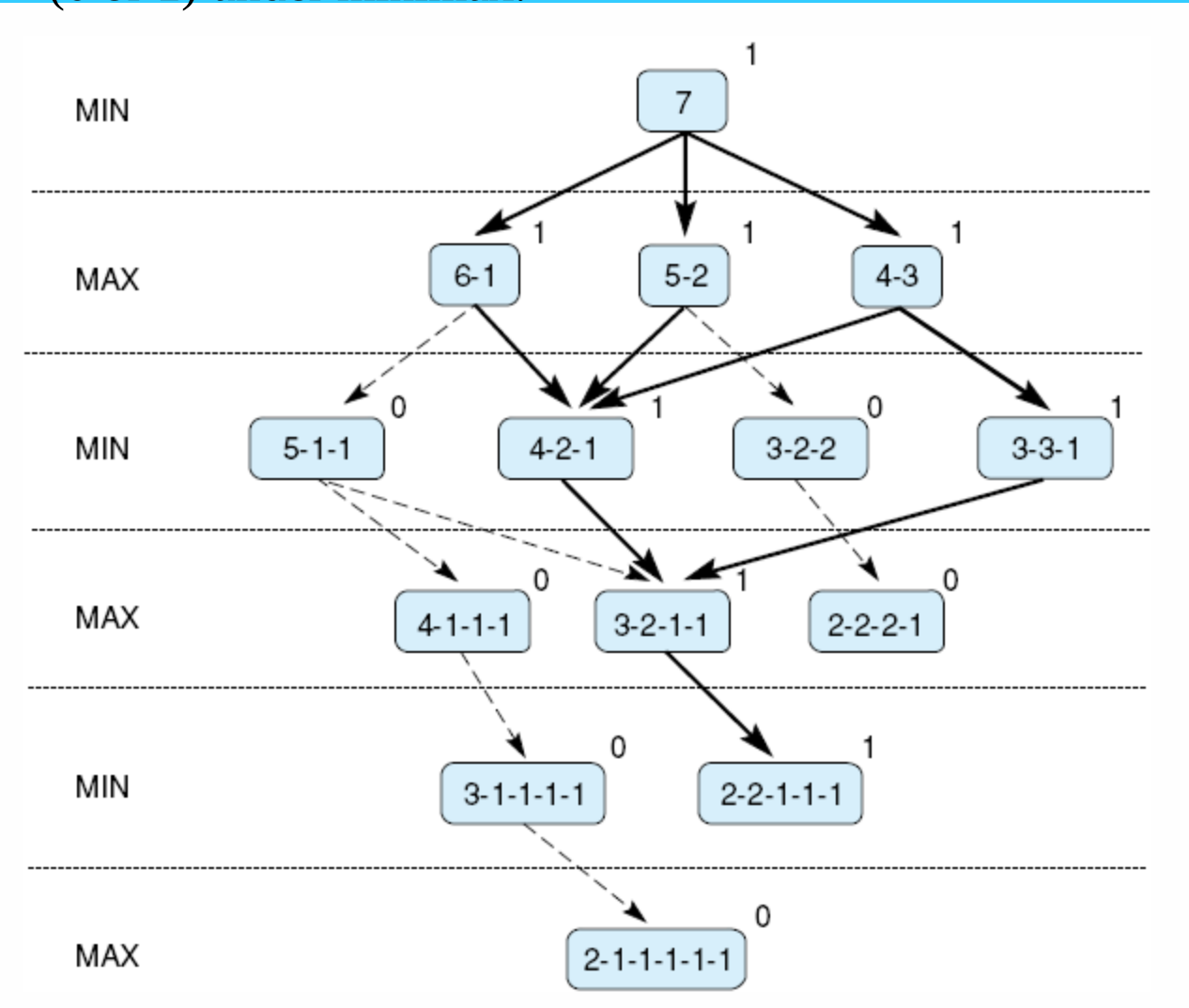


Fig 4.21 Minimax to a hypothetical state space. Leafstates show heuristic values; internal states show backed-up values.

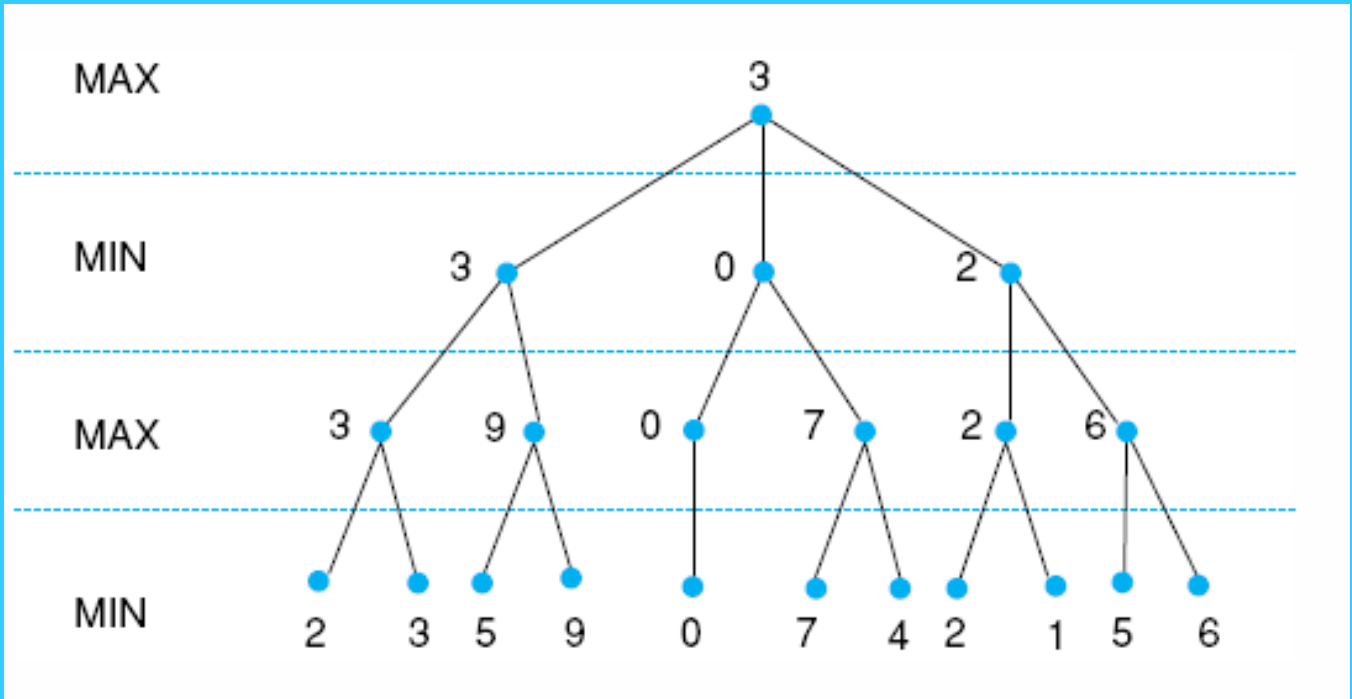
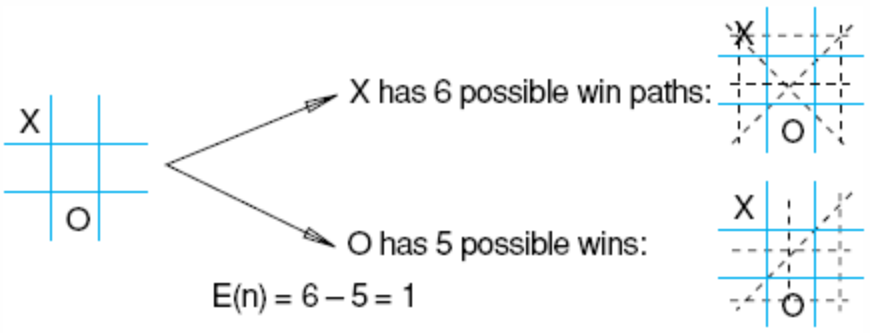
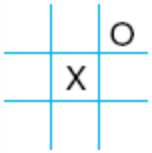


Fig 4.22 Heuristic measuring conflict applied to states of tic-tac-toe.



X has 4 possible win paths;
O has 6 possible wins

E(n) = 4 – 6 = –2



X has 5 possible win paths;
O has 4 possible wins

E(n) = 5 – 4 = 1

Heuristic is E(n) = M(n) – O(n)

where M(n) is the total of My possible winning lines

O(n) is total of Opponent's possible winning lines

E(n) is the total Evaluation for state n

Fig 4.23 Two-ply minimax applied to the opening move of tic-tac-toe, from Nilsson (1971).

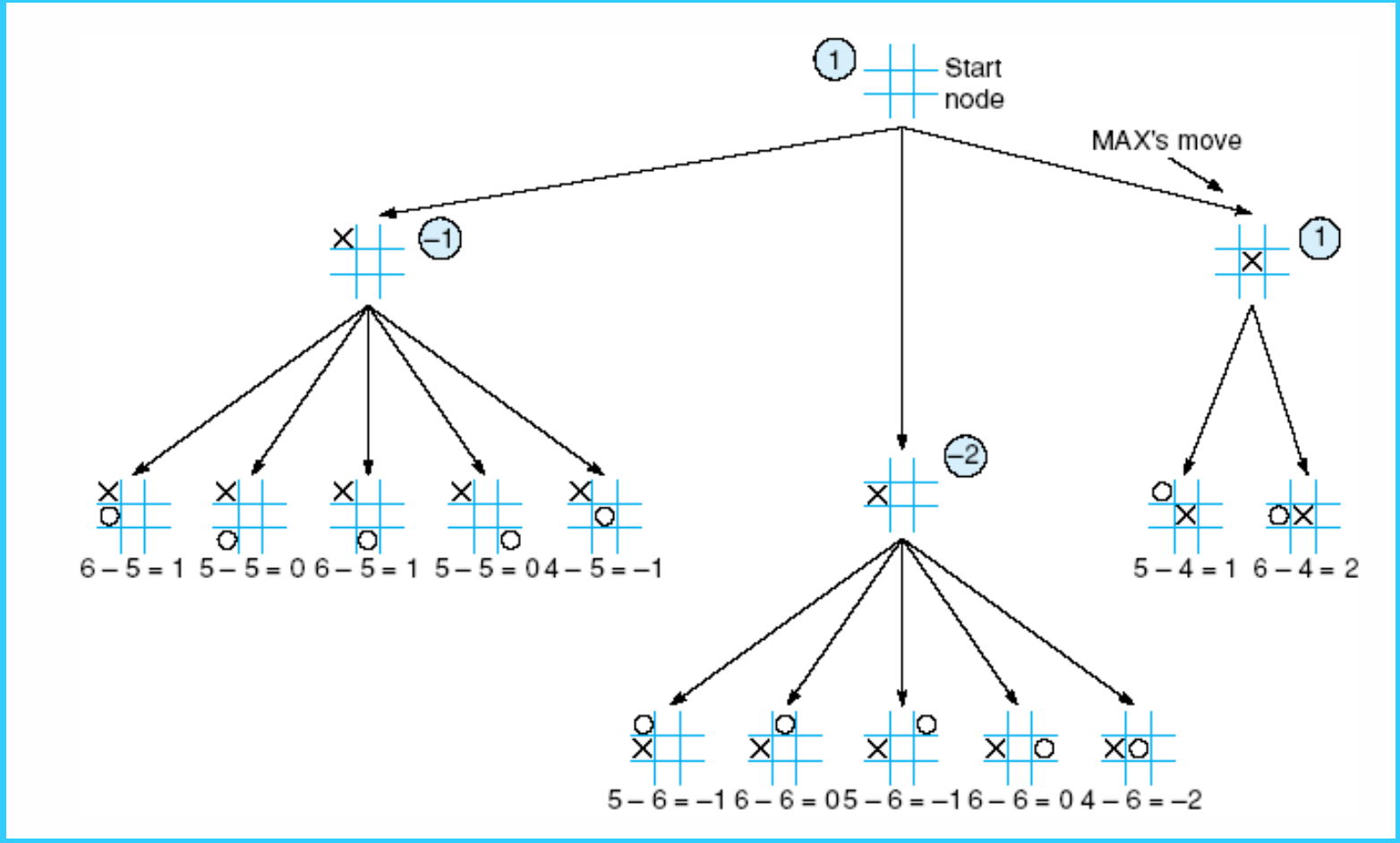


Fig 4.24 Two ply minimax, and one of two possible MAX second moves, from Nilsson (1971).

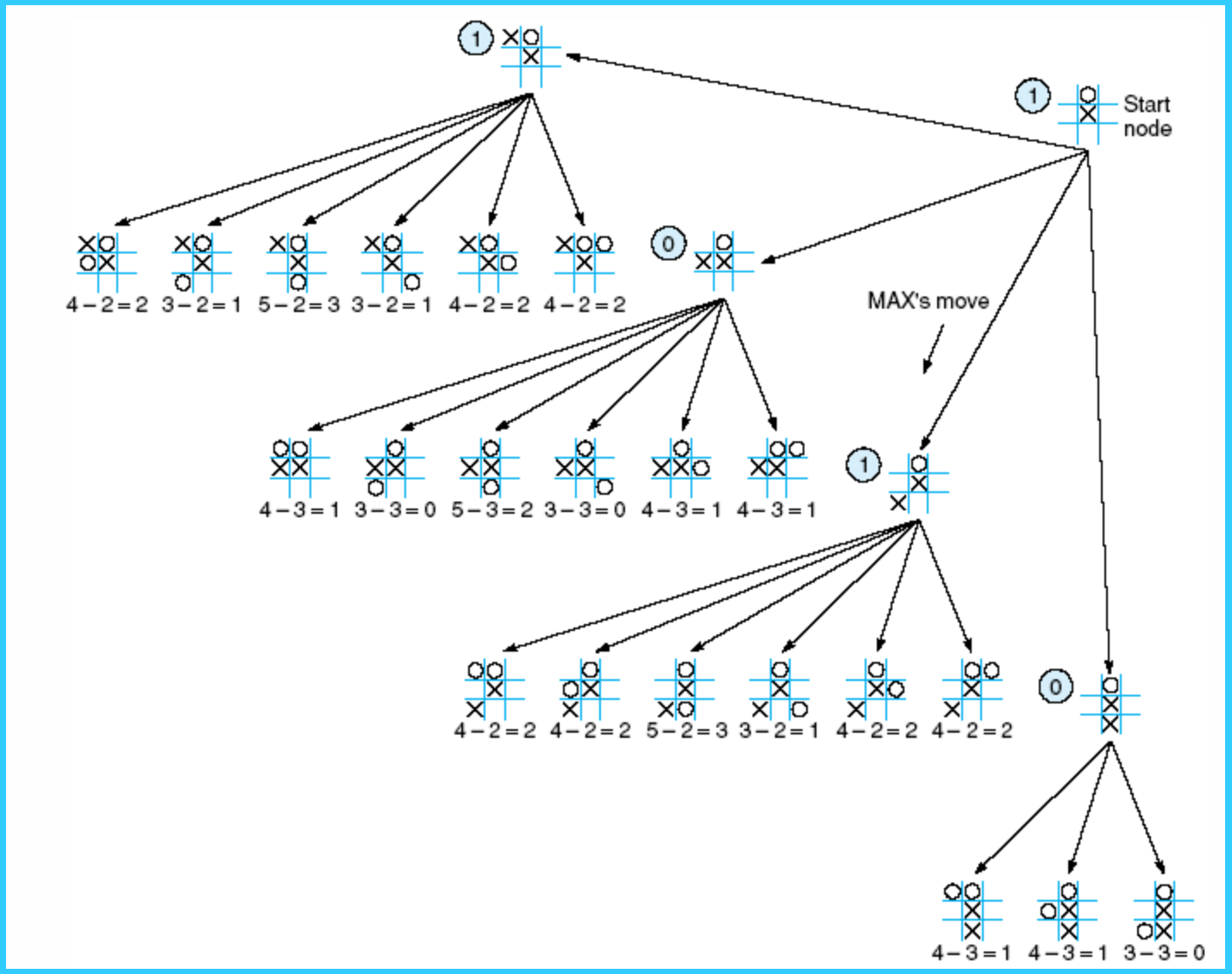


Fig 4.25 Two-ply minimax applied to X's move near the end of the game, from Nilsson (1971).

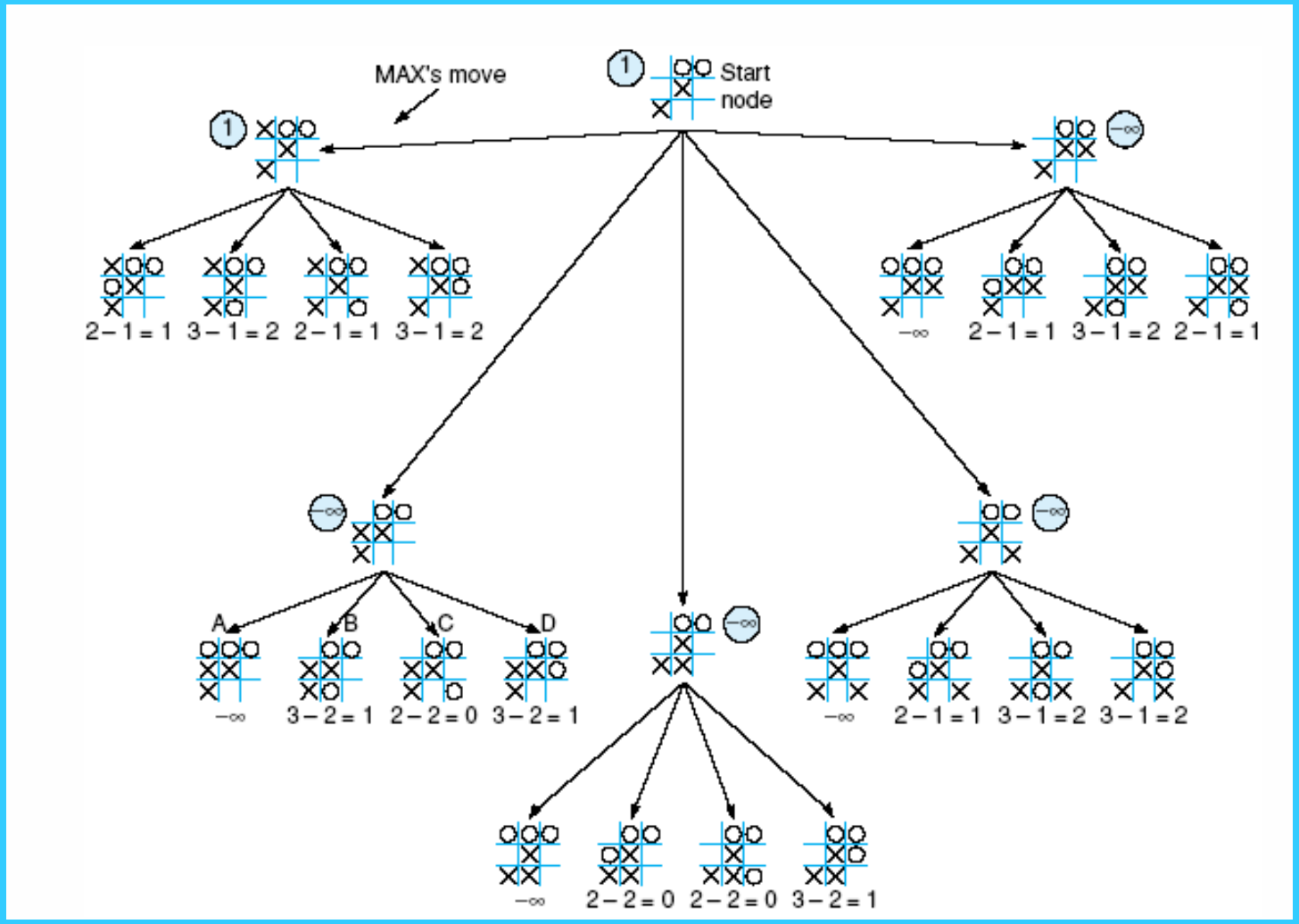


Fig 4.26 Alpha-beta pruning applied to state space of Fig 4.21. States without numbers are not evaluated.

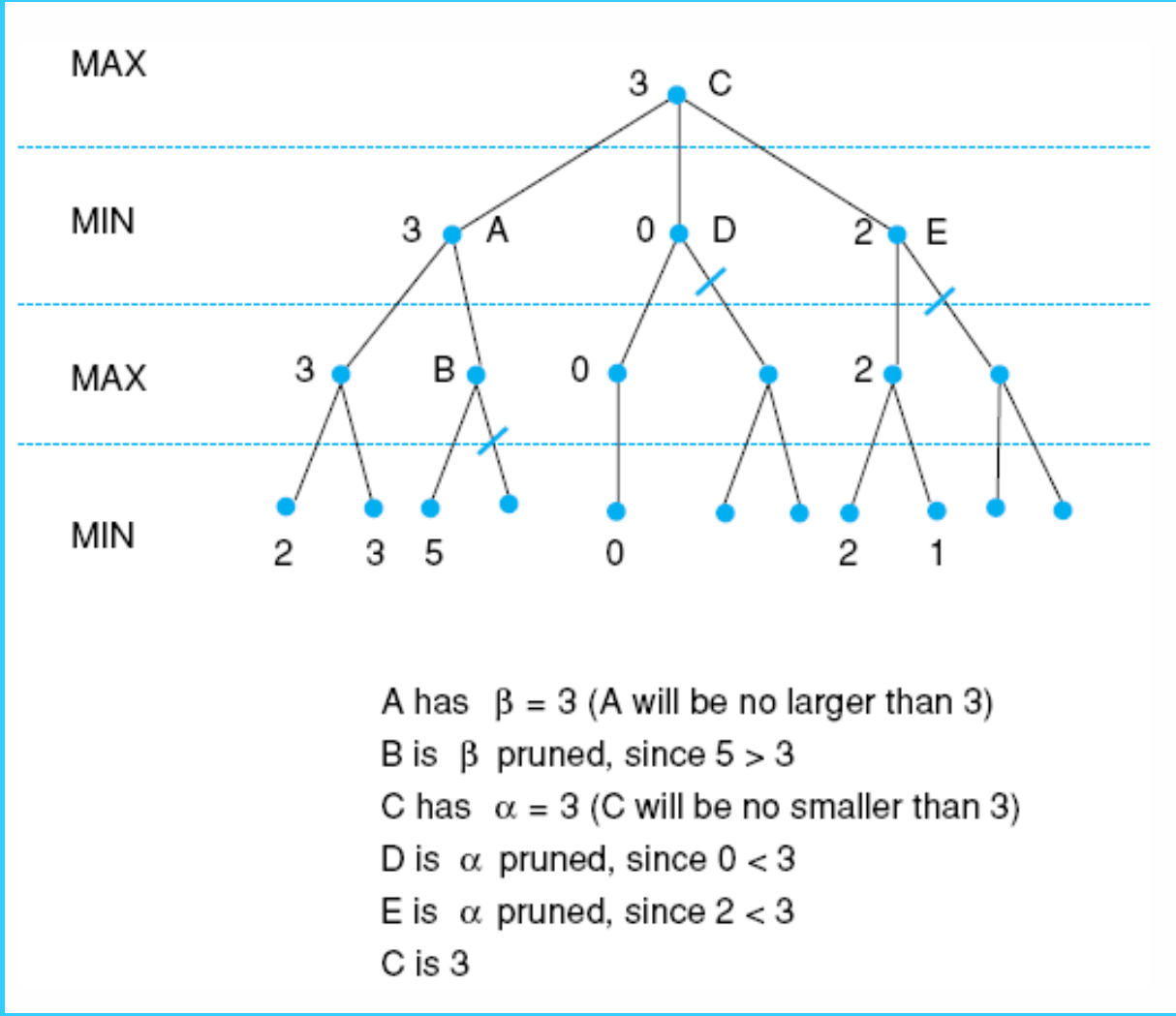


Fig 4.27 Number of nodes generated as a function of branching factor, B, for various lengths, L, of solution paths. The relating equation is $T = B(B^L - 1)/(B - 1)$, adapted from Nilsson (1980).

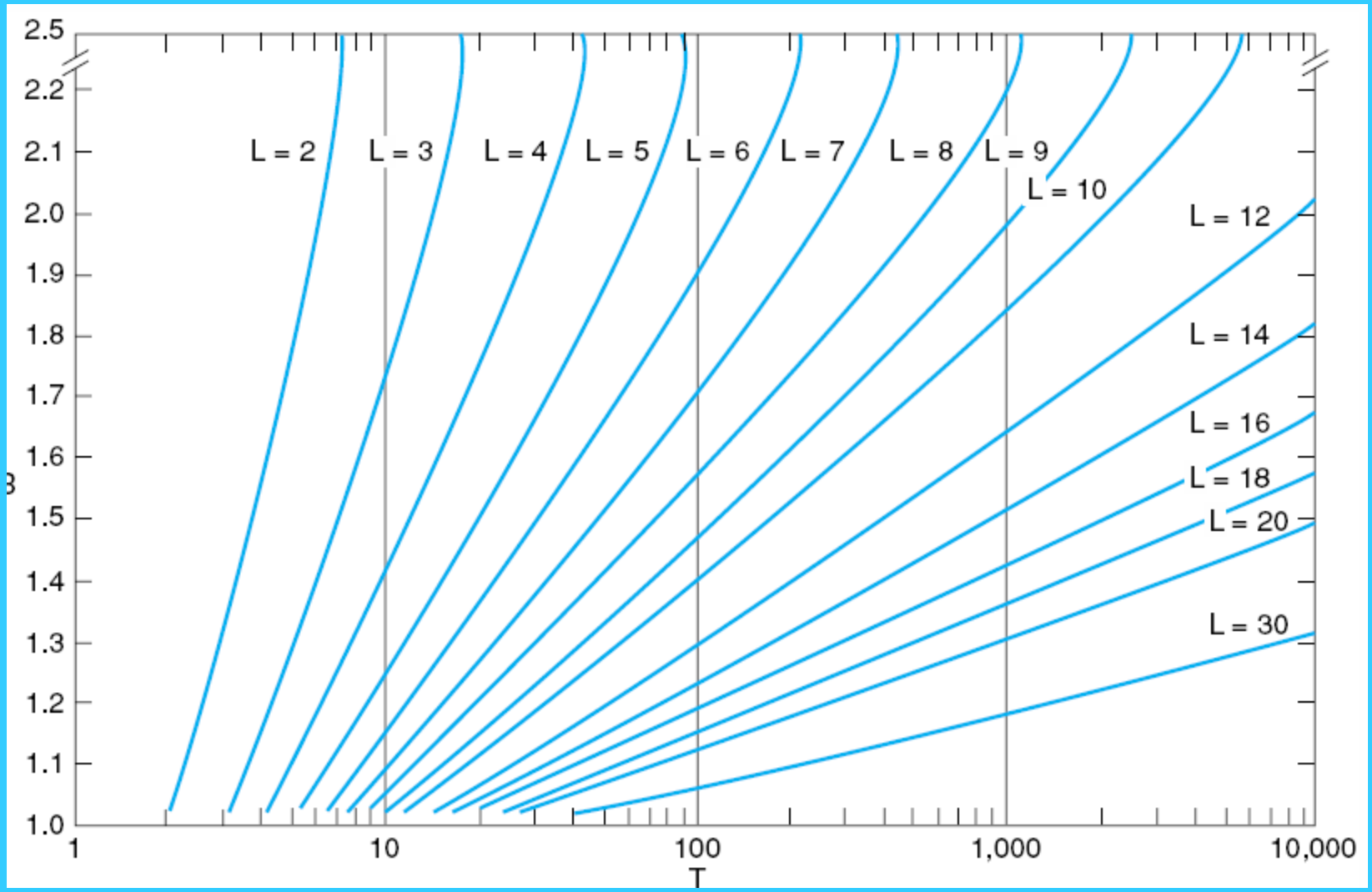


Fig 4.28 Informal plot of cost of searching and cost of computing heuristic evaluation against informedness of heuristic, adapted from Nilsson (1980).

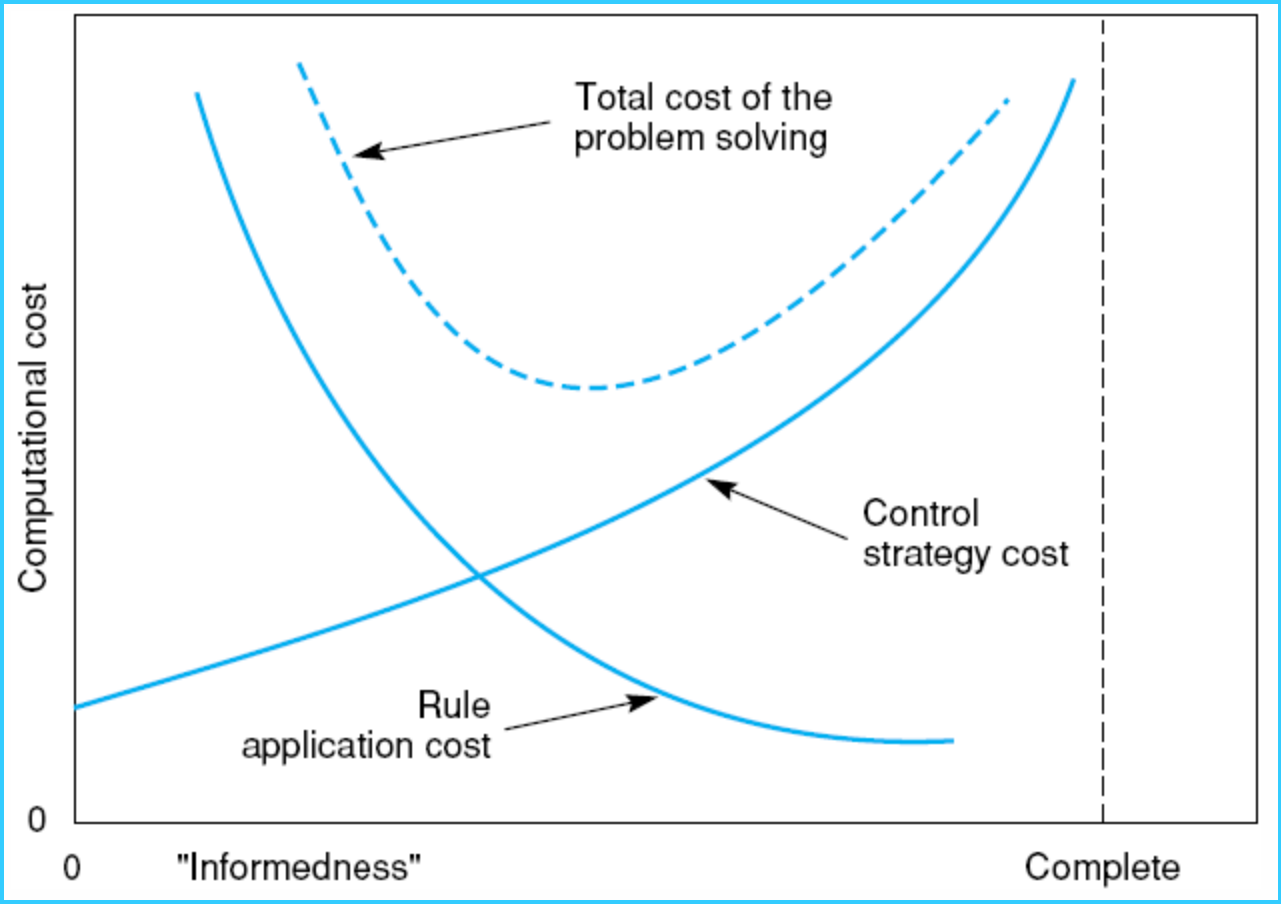


Fig 4.29 The sliding block puzzle.

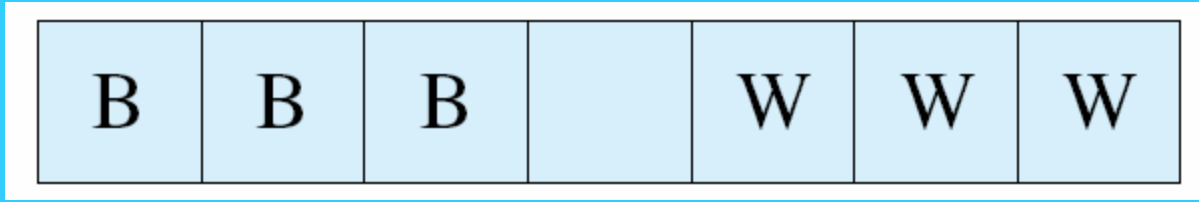


Fig 4.30.

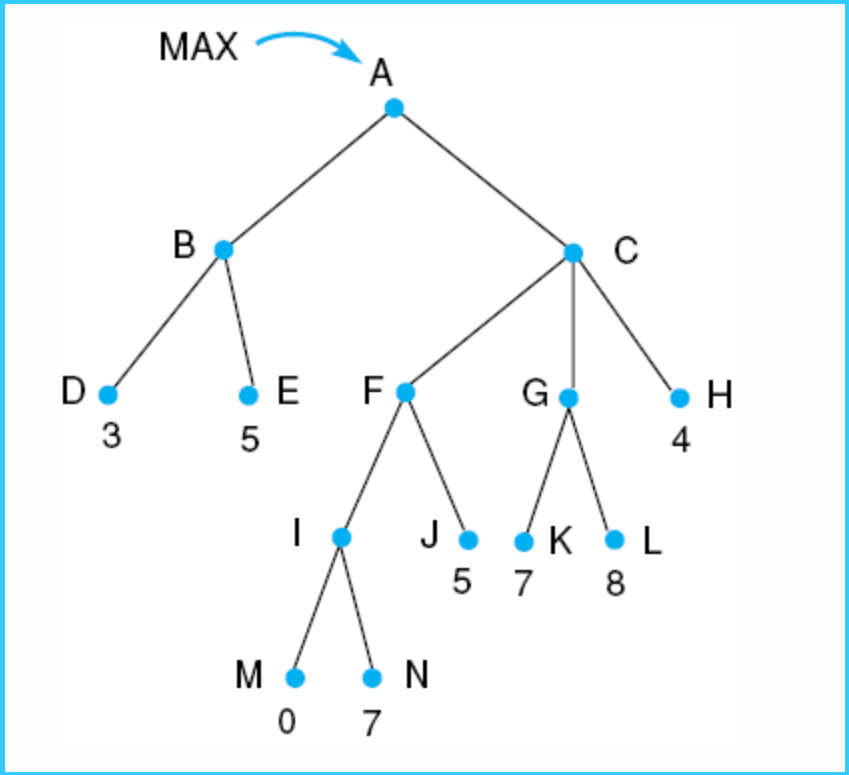


Fig 4.31.