Artificial Intelligence

AI2002

Rushda Muneer

Spring 2026

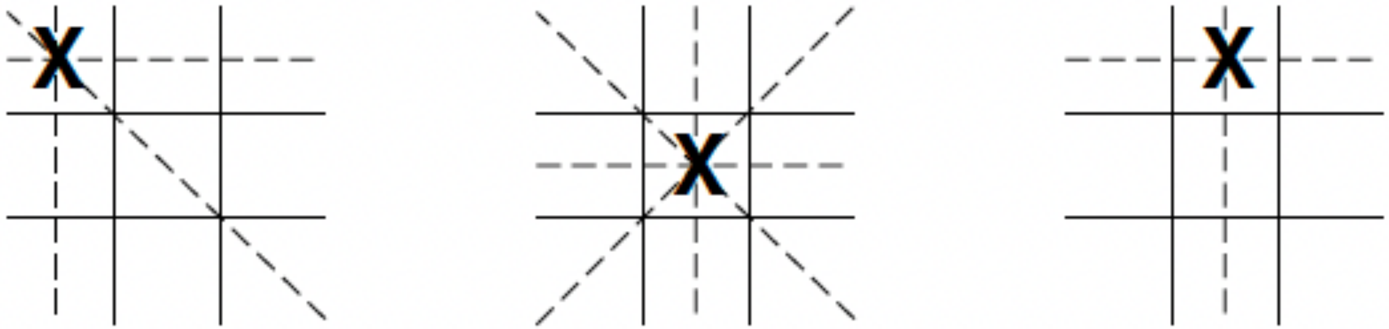
Heuristic Search Methods

- A heuristic search make use of the available information in making search more efficient.
- It uses heuristics or rules of thumb to decide which parts of the tree to examine.
- A heuristic is a rule or method that always improves the decision process
- Medical Diagnosis
  - A given set of symptoms may have several possible causes;
  - Doctors choose the most likely diagnosis and plan of treatment

When Heuristic Search Methods are Used?

- A problem may have an exact solution but the computational cost of finding it may be prohibitive.
- The search space growth is explosive (increasing Exponentially or Factorially with the depth of the search)
- Exhaustive (brute force) techniques may fail to find a solution in a practical length of time
- Heuristics counter this problem by searching along the most promising path
- It eliminates states and their descendants from consideration defeating the explosive growth of the state space and finding an acceptable answer

A heuristic for Tic-Tac-Toe



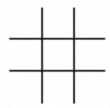
Three wins through a corner square

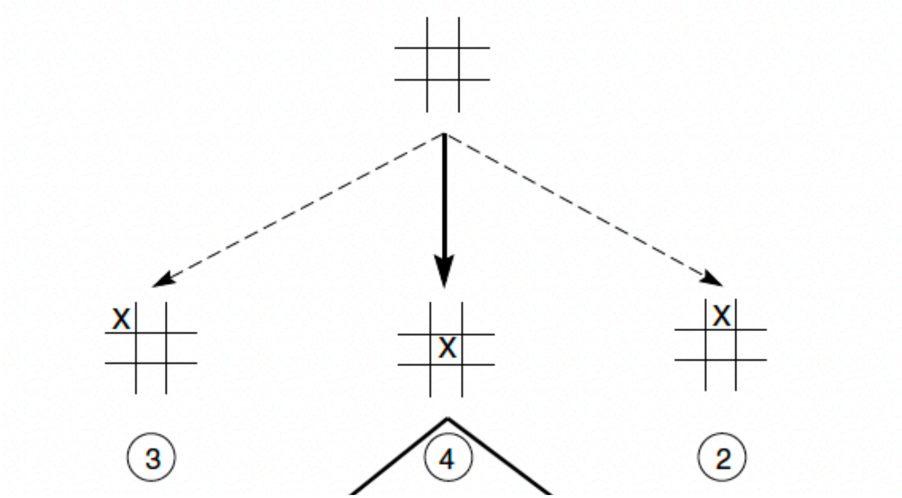
Four wins through the center square

Two wins through a side square

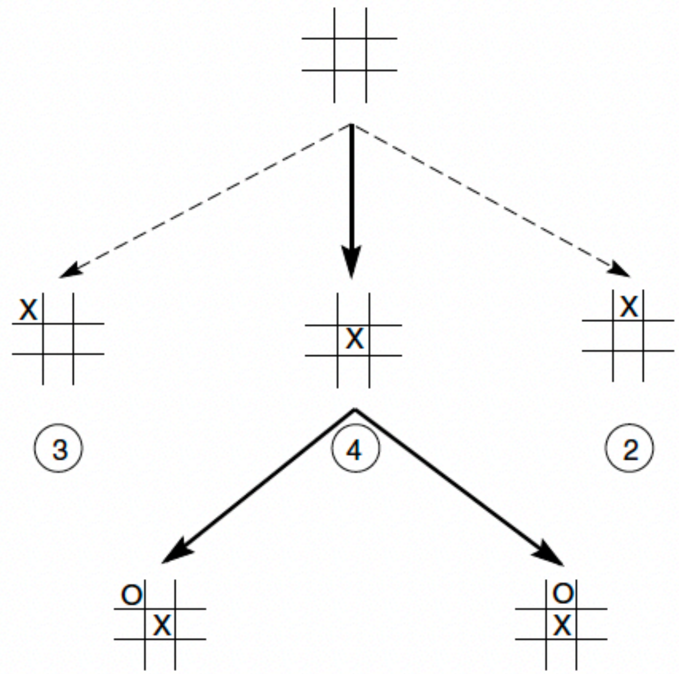
Figure 4.2 The most wins heuristic applied to the first children in tic-tac-toe.

A simple heuristic for a game of tic tac toe could be the possible winning patterns attainable from the current move





X
X
X
3
4
2


X
X
X
3
4
2
O
X
O
X

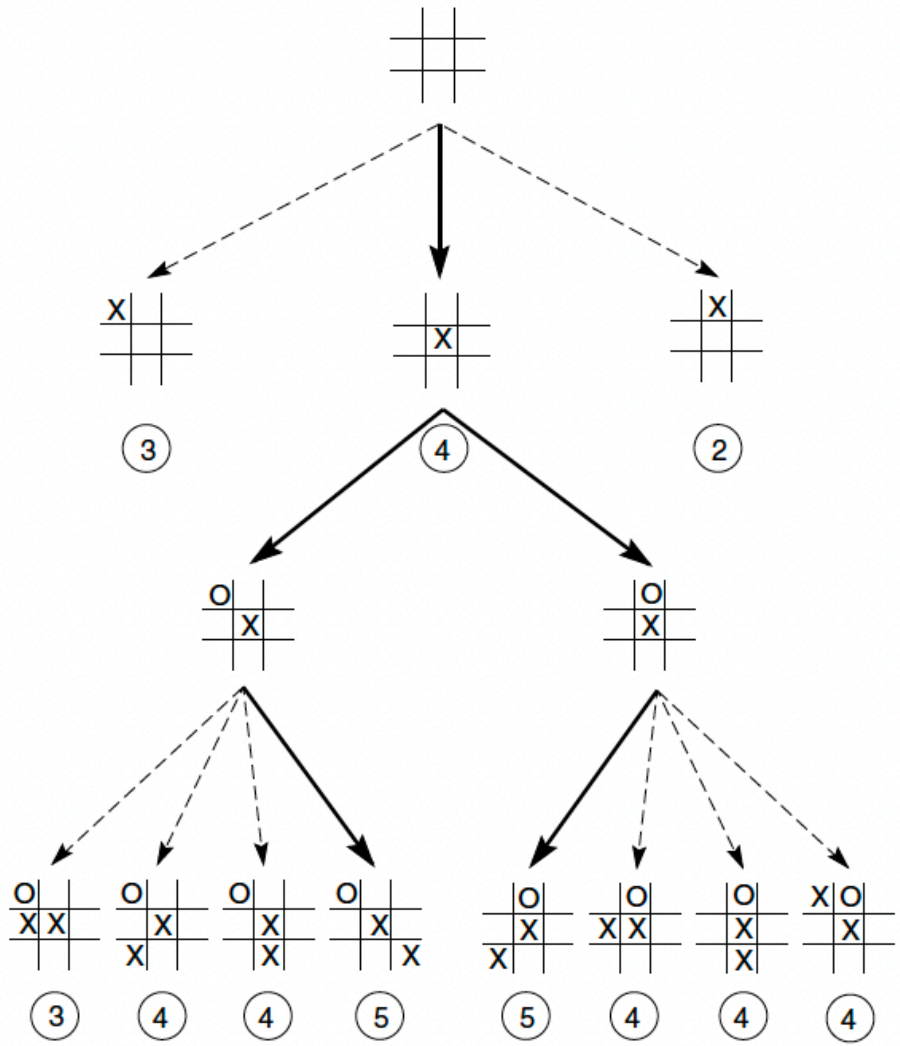


Figure 4.3 Heuristically reduced state space for tic-tac-toe.

Best-First Search Algorithm

- The general approach we consider is called best-first search. Best-first search is an instance of the general TREE-SEARCH or GRAPH-SEARCH algorithm in which a node is selected for expansion based on an evaluation function, f(n).
- The evaluation function is construed as a cost estimate, so the node with the lowest evaluation is expanded first.
- Most best-first algorithms include as a component of f a heuristic function, denoted h(n):
- h(n) = estimated cost of the cheapest path from the state at node n to a goal state.

Best-First Search Algorithm

- Like the depth-first and breadth-first search algorithms, best-first search uses lists to maintain states:
- open to keep track of the current fringe of the search and
- closed to record states already visited.
- An added step in the algorithm orders the states on open according to some heuristic estimate of their “closeness” to a goal.
- Thus, each iteration of the loop considers the most “promising” state on the open list.

- begin
  - open := [Start];
  - closed := [ ];
  - while open ≠ [ ] do
  - 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;
      - put X on closed;
      - re-order states on open by heuristic merit (best leftmost)
    - end;
  - return FAIL
- end.

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!

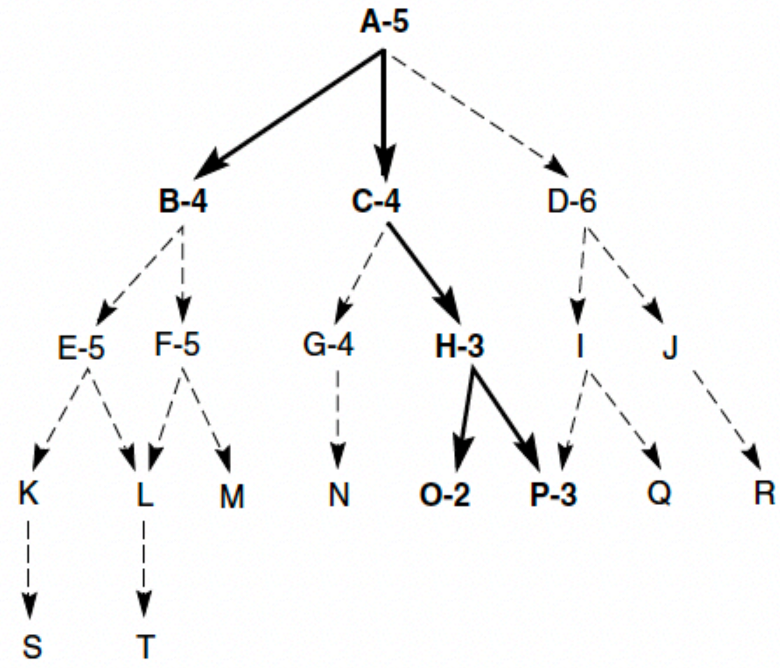
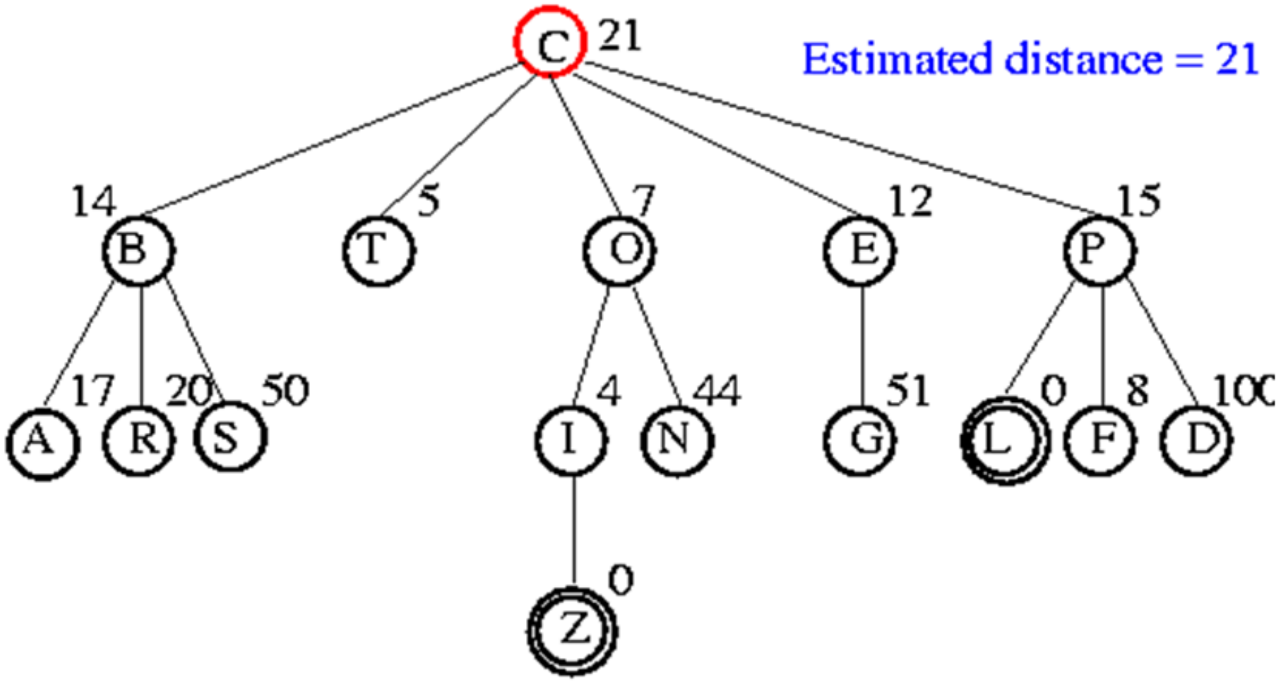
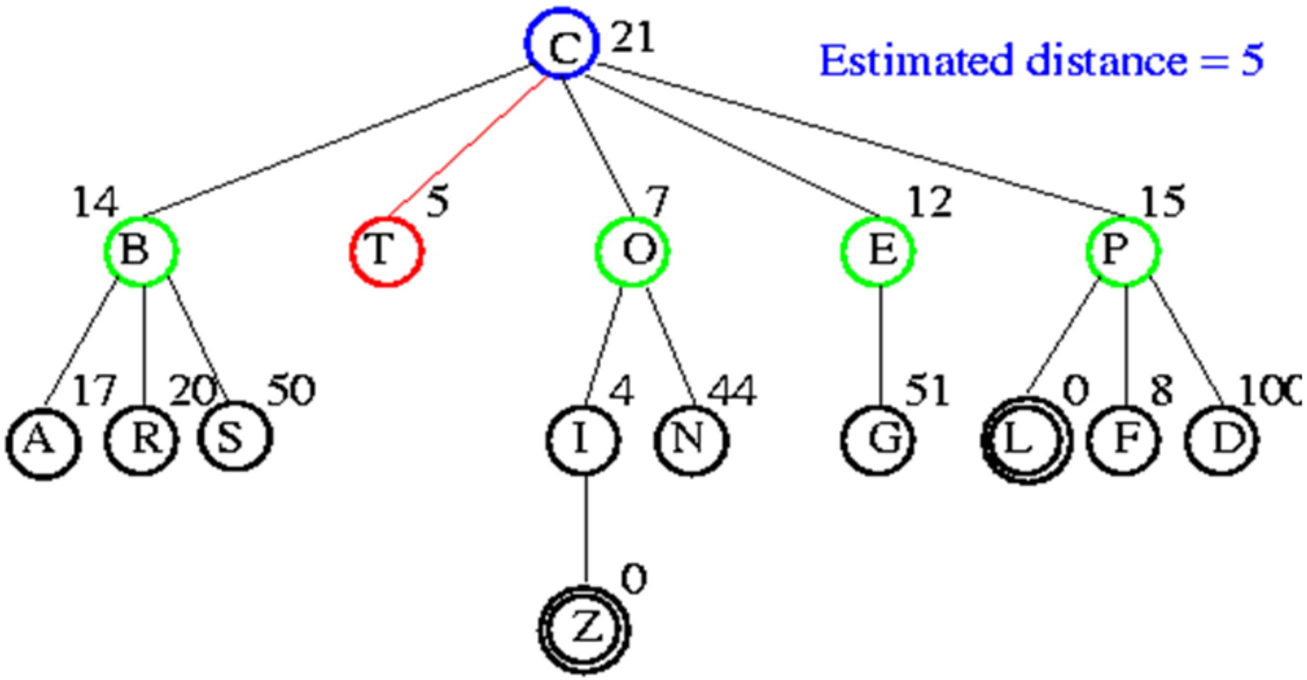


Figure 4.10 Heuristic search of a hypothetical state space.

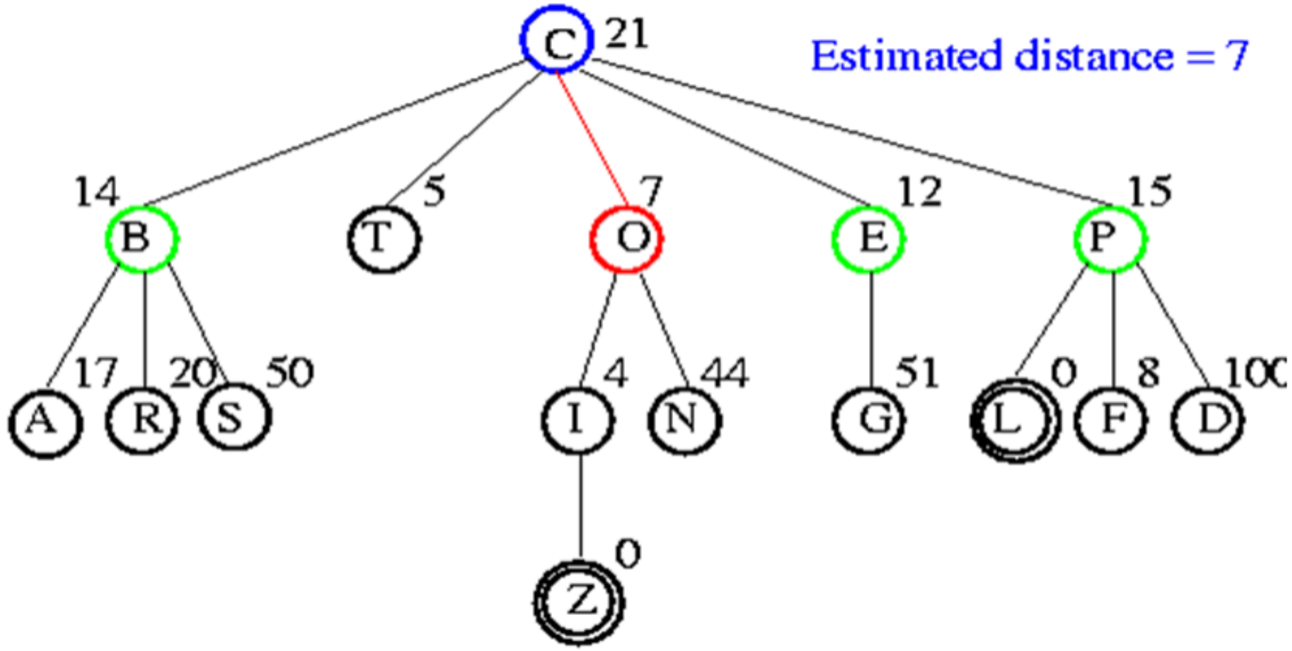
1. open = [C21];
   closed = [ ]



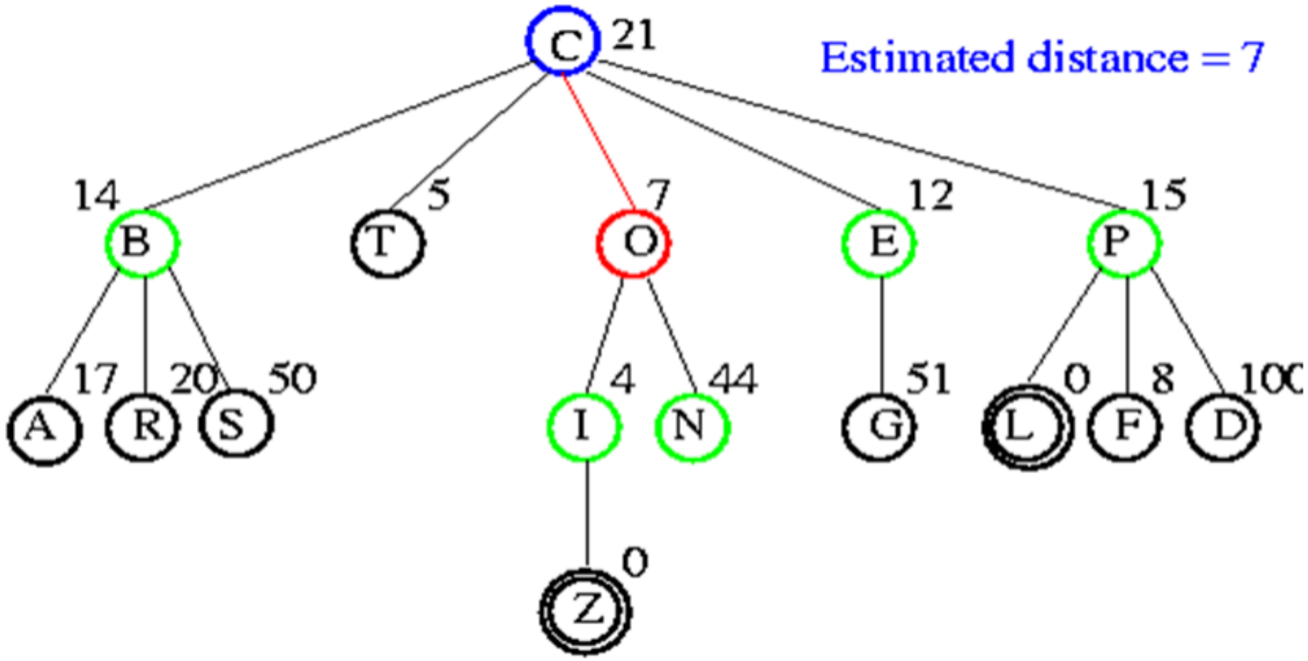
1. open = [C21];
   closed = [ ]
2. open = [T5, O7, E12, B14, P15];
   closed = [C21]



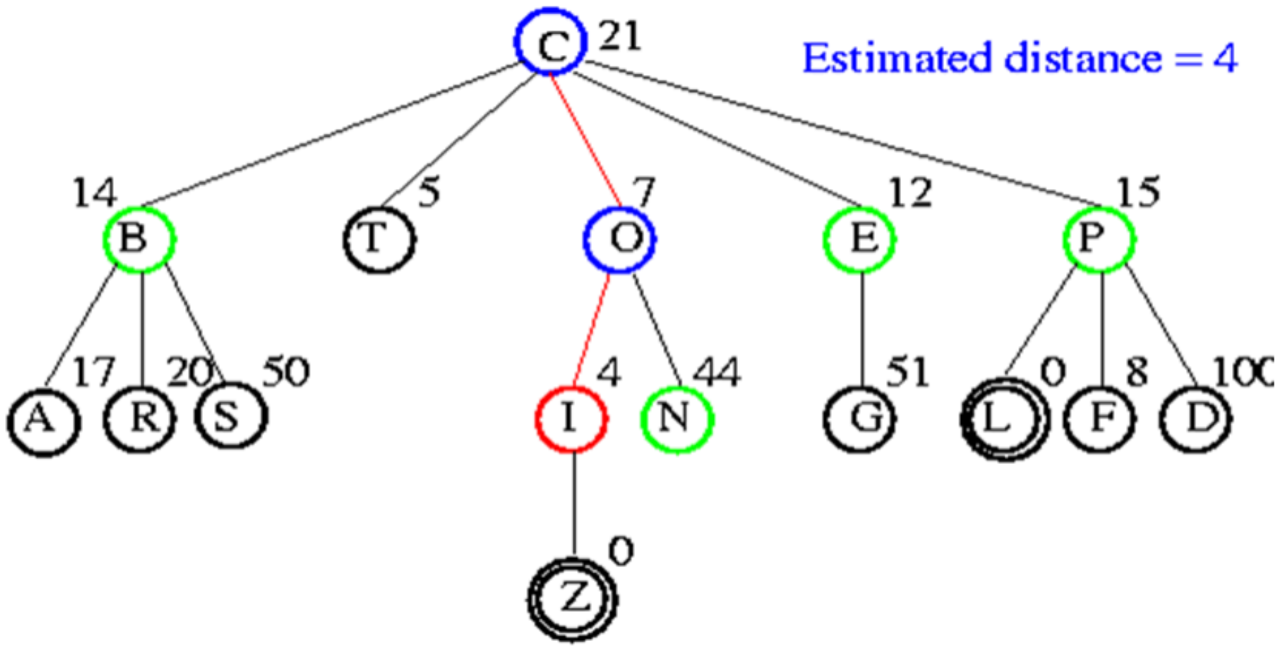
1. open = [C21]; closed = [ ]
2. open = [T5, O7, E12, B14, P15]; closed = [C21]
3. open = [O7, E12, B14, P15]; closed = [T5, C21]



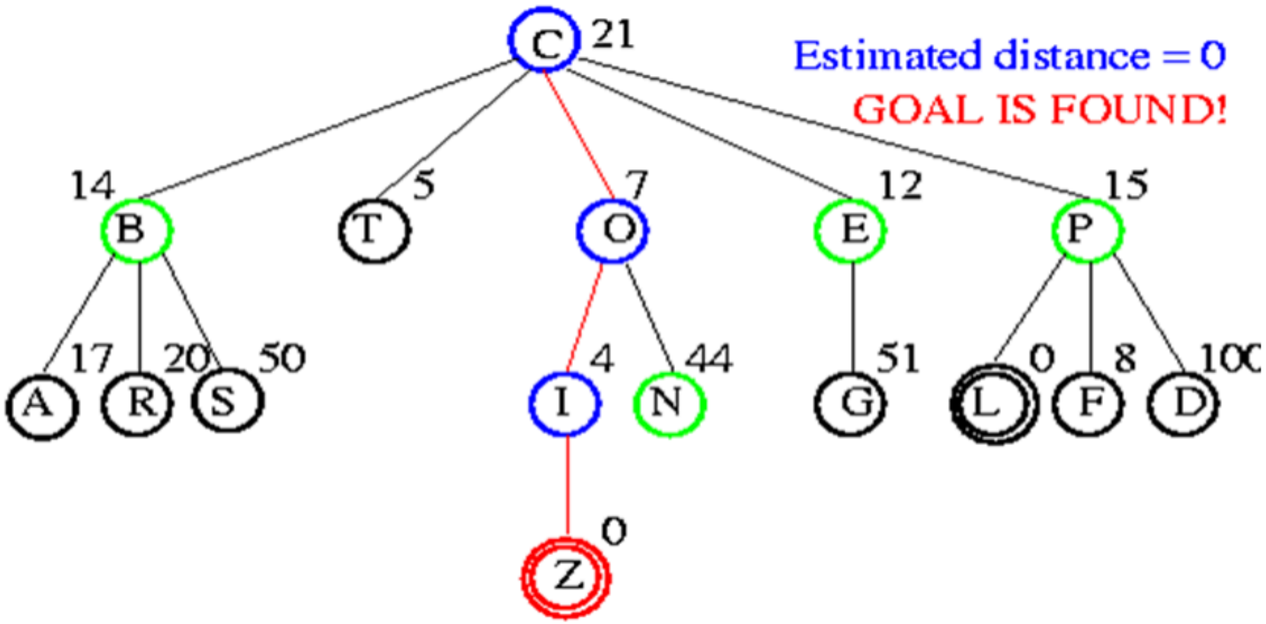
1. open = [C21]; closed = [ ]
2. open = [T5, O7, E12, B14, P15]; closed = [C21]
3. open = [O7, E12, B14, P15]; closed = [T5, C21]
4. open = [I4, E12, B14, P15, N44]; closed = [O7, T5, C21]



1. open = [C21]; closed = [ ]
2. open = [T5, O7, E12, B14, P15]; closed = [C21]
3. open = [O7, E12, B14, P15]; closed = [T5, C21]
4. open = [I4, E12, B14, P15, N44]; closed = [O7, T5, C21]
5. open = [Z0, E12, B14, P15, N44]; closed = [I4, O7, T5, C21]



1. open = [C21]; closed = [ ]
2. open = [T5, O7, E12, B14, P15]; closed = [C21]
3. open = [O7, E12, B14, P15]; closed = [T5, C21]
4. open = [I4, E12, B14, P15, N44]; closed = [O7, T5, C21]
5. open = [Z0, E12, B14, P15, N44]; closed = [I4, O7, T5, C21]



8-puzzle problem

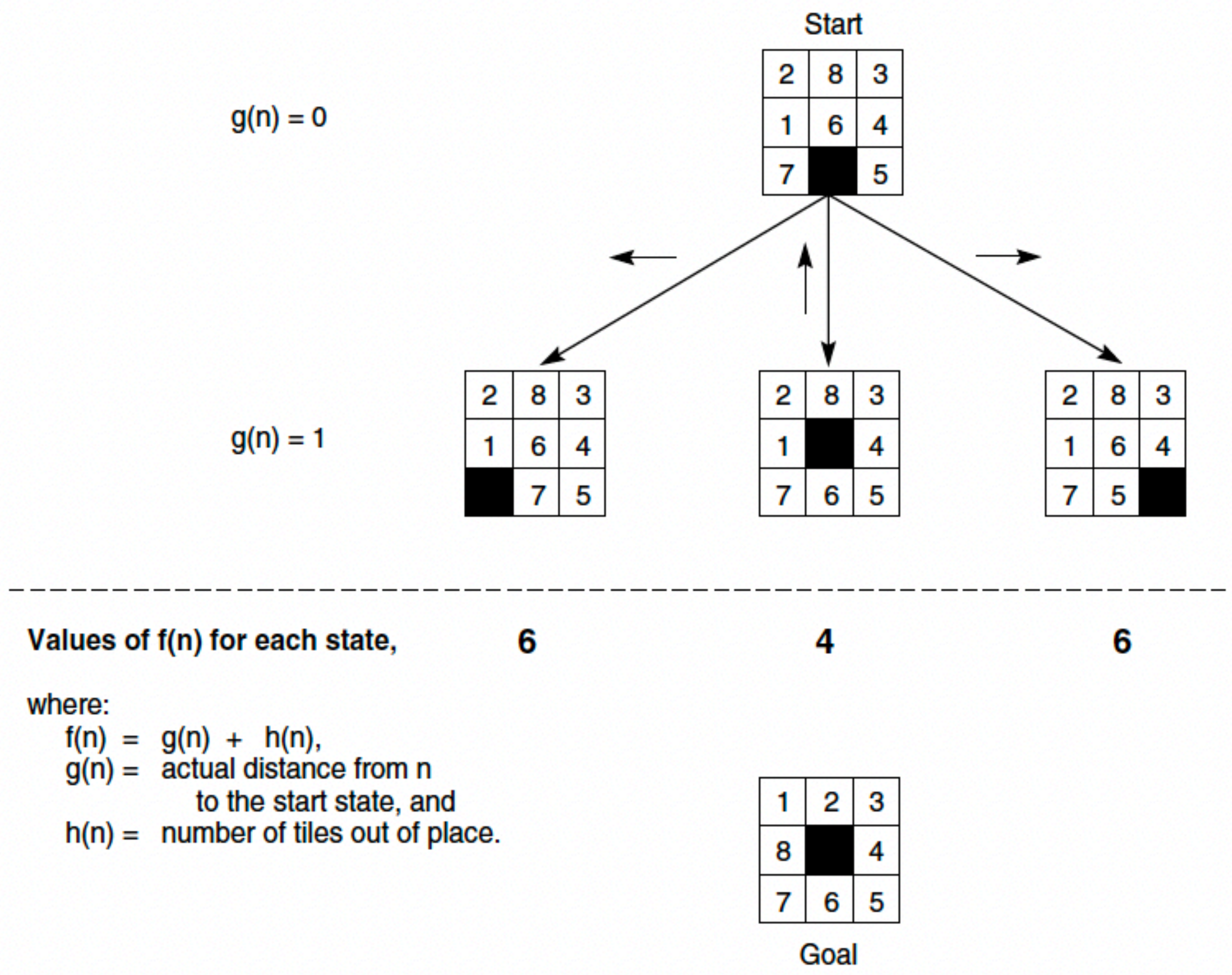
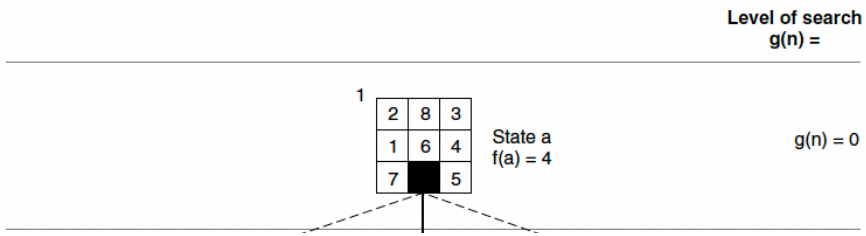
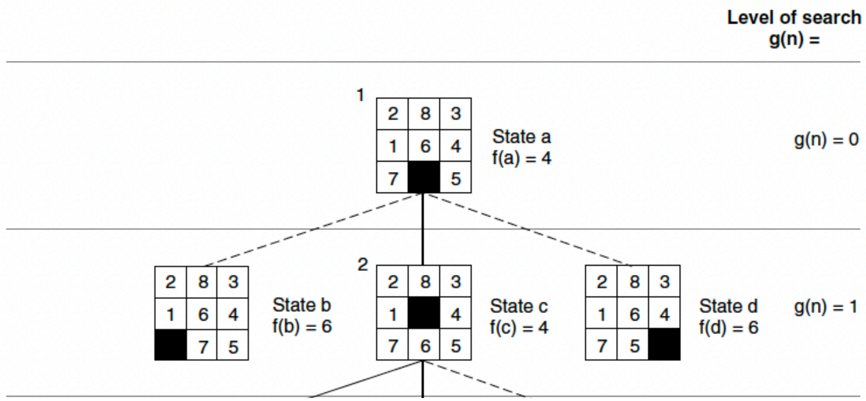
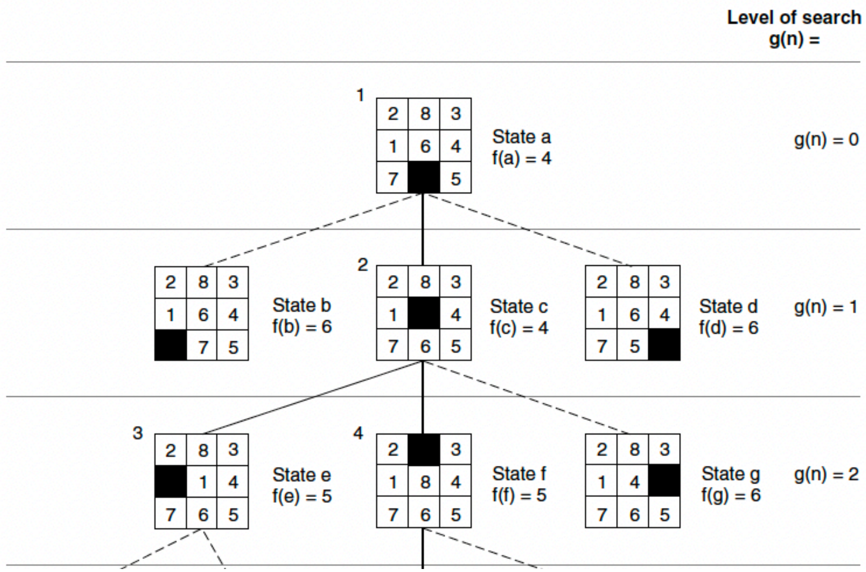
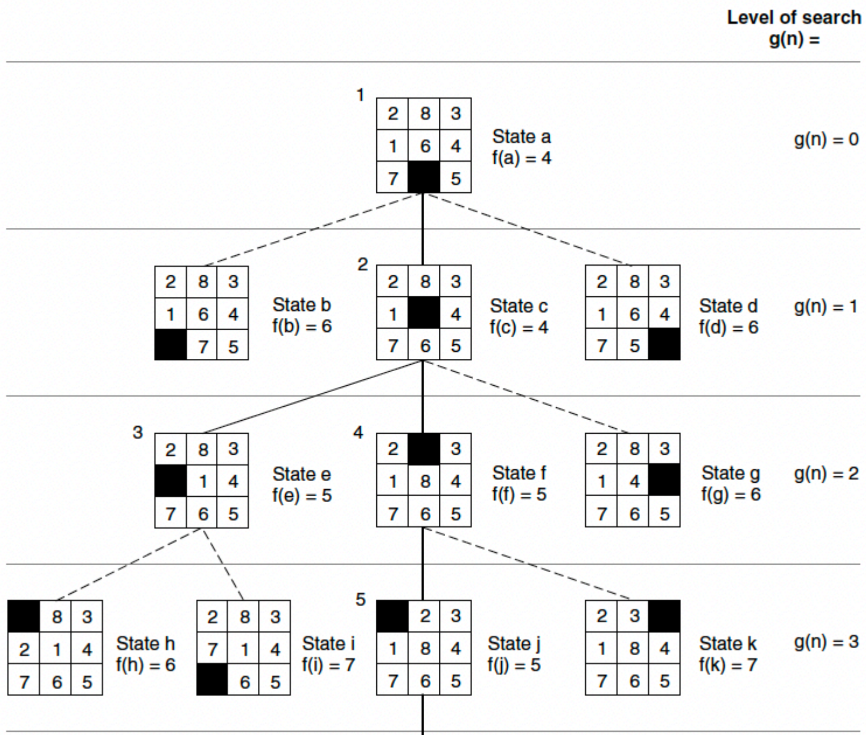


Figure 4.15 The heuristic f applied to states in the 8-puzzle.

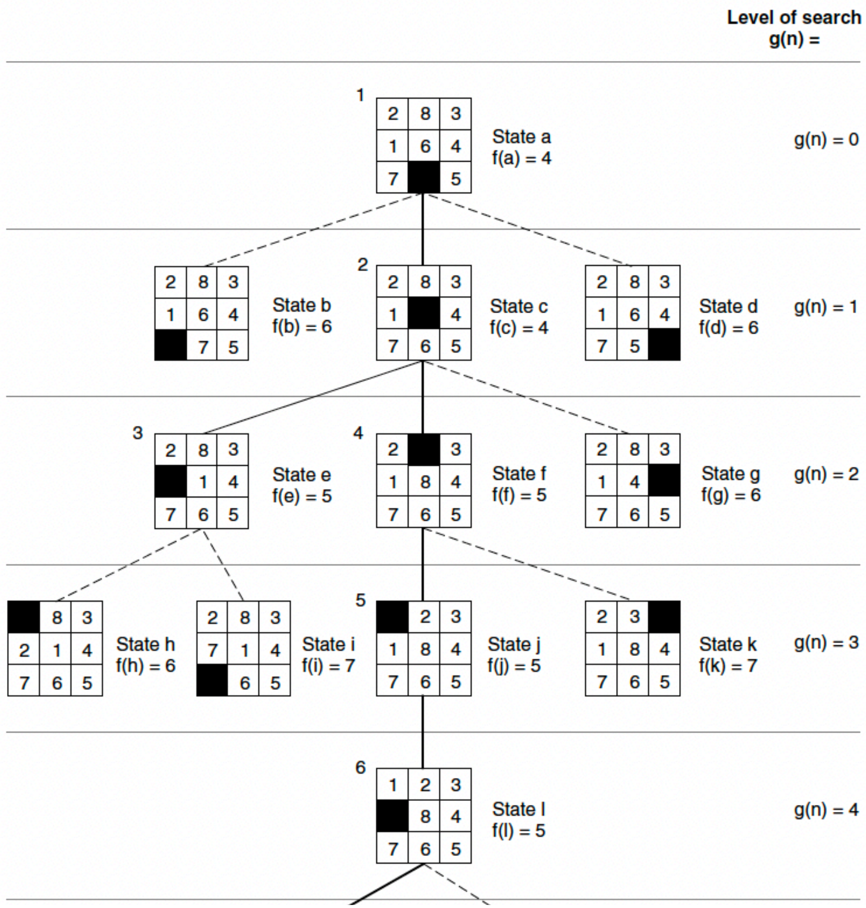

Level of search
g(n) =
1
2 8 3
1 6 4
7 5
State a
f(a) = 4
g(n) = 0


Level of search
g(n) =
1
2 8 3
1 6 4
7 5
State a
f(a) = 4
g(n) = 0
2
2 8 3
1 6 4
7 5
State b
f(b) = 6
2 8 3
1 4
7 6 5
State c
f(c) = 4
2 8 3
1 6 4
7 5
State d
f(d) = 6
g(n) = 1


Level of search
g(n) =
1
2 8 3
1 6 4
7 5
State a
f(a) = 4
g(n) = 0
2
State b
f(b) = 6
State c
f(c) = 4
State d
f(d) = 6
g(n) = 1
3
4
State e
f(e) = 5
State f
f(f) = 5
State g
f(g) = 6
g(n) = 2



Level of search
g(n) =
1
State a
f(a) = 4
g(n) = 0
2
State b
f(b) = 6
State c
f(c) = 4
State d
f(d) = 6
g(n) = 1
3
4
State e
f(e) = 5
State f
f(f) = 5
State g
f(g) = 6
g(n) = 2
5
State h
f(h) = 6
State i
f(i) = 7
State j
f(j) = 5
State k
f(k) = 7
g(n) = 3



Level of search
g(n) =
1
State a
f(a) = 4
g(n) = 0
2
State b
f(b) = 6
State c
f(c) = 4
State d
f(d) = 6
g(n) = 1
3
4
State e
f(e) = 5
State f
f(f) = 5
State g
f(g) = 6
g(n) = 2
5
State h
f(h) = 6
State i
f(i) = 7
State j
f(j) = 5
State k
f(k) = 7
g(n) = 3
6
State l
f(l) = 5
g(n) = 4

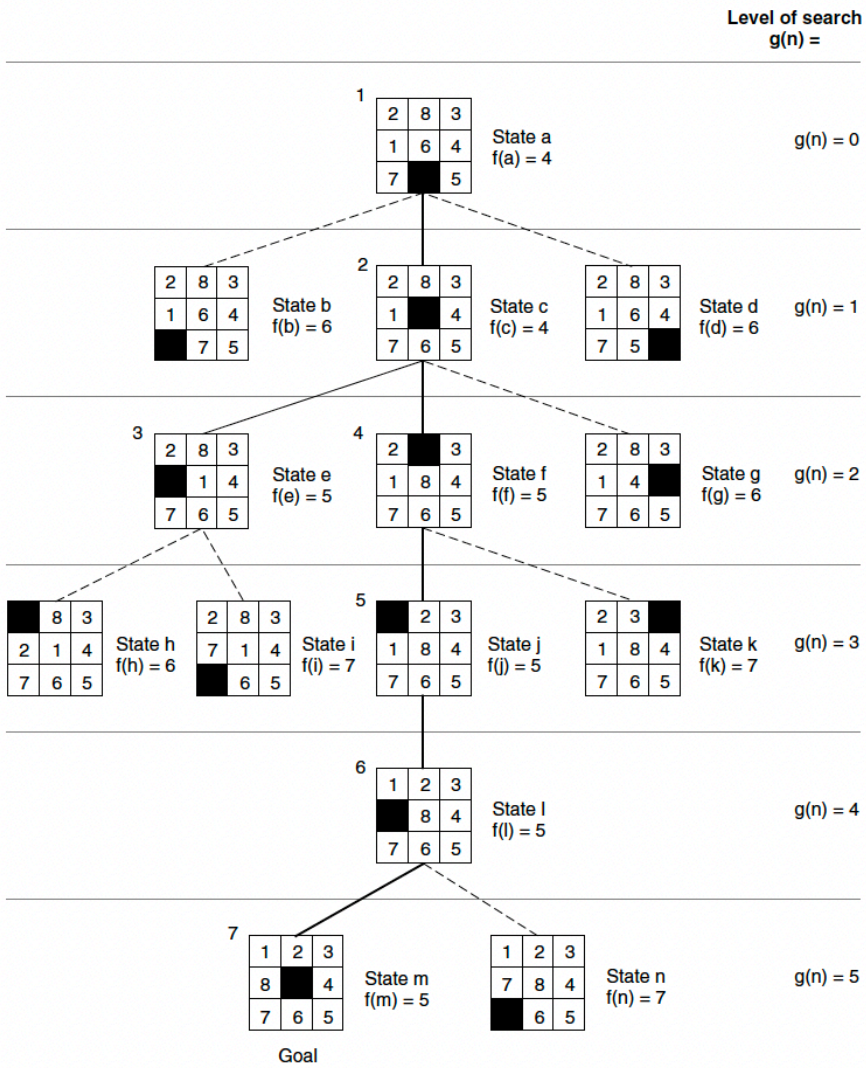


Figure 4.16 State space generated in heuristic search of the 8-puzzle graph.

Homework Readings

- Modern Approach (3.5.1)
- G. F. Lugar (4.0, 4.2)