5.1 – The Backtracking Technique

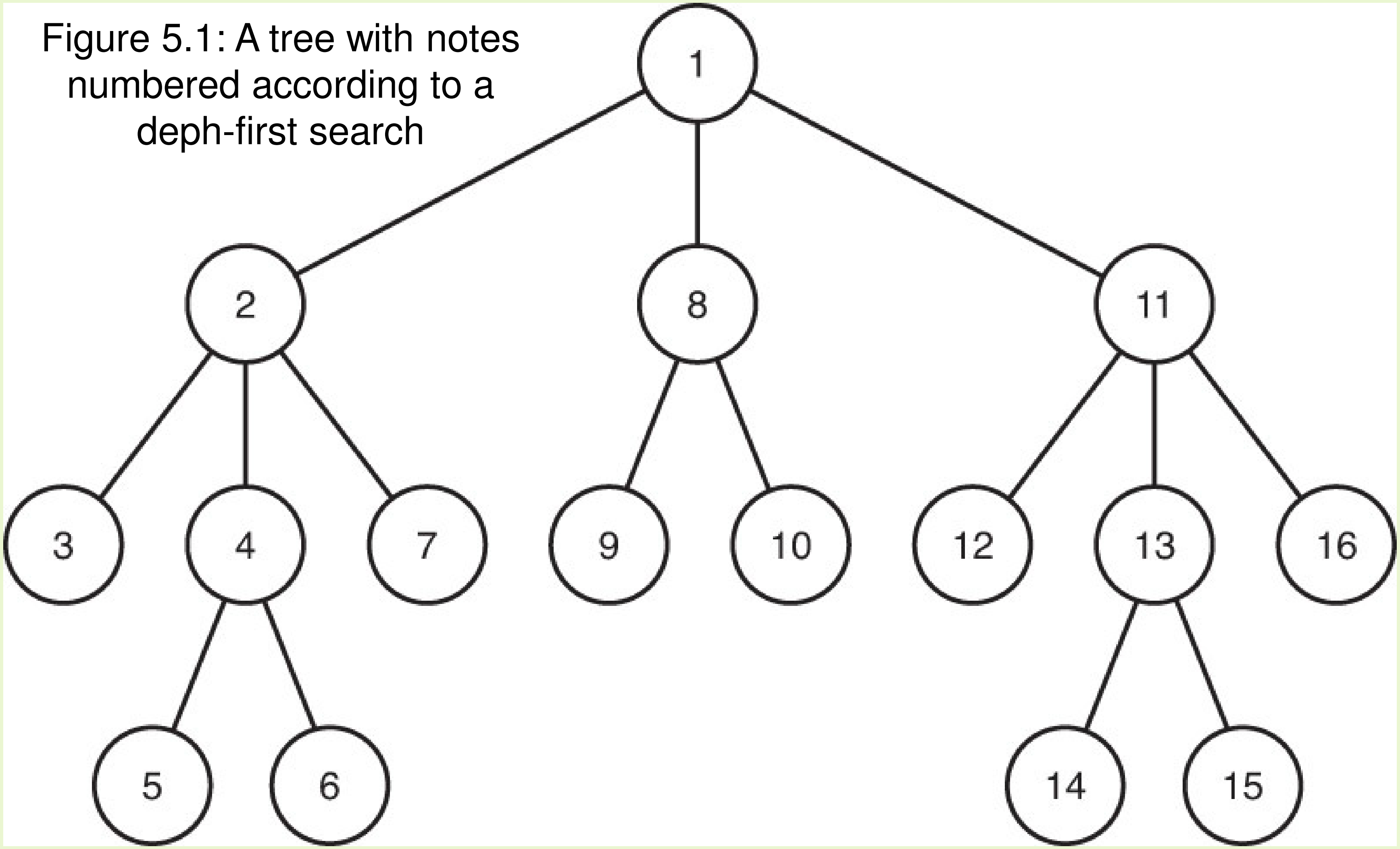
- Used to solve problems in which a ***sequence*** of objects is chosen from a set, so that the sequence satisfies a criterion
- A sequence of choices are made at various decision points while trying to find a solution
- If a dead end is reached, the algorithm has to backtrack to a previous decision point and try a different path
- In the worst case, performance is exponential, but better results often achieved in practice

Depth-First Search

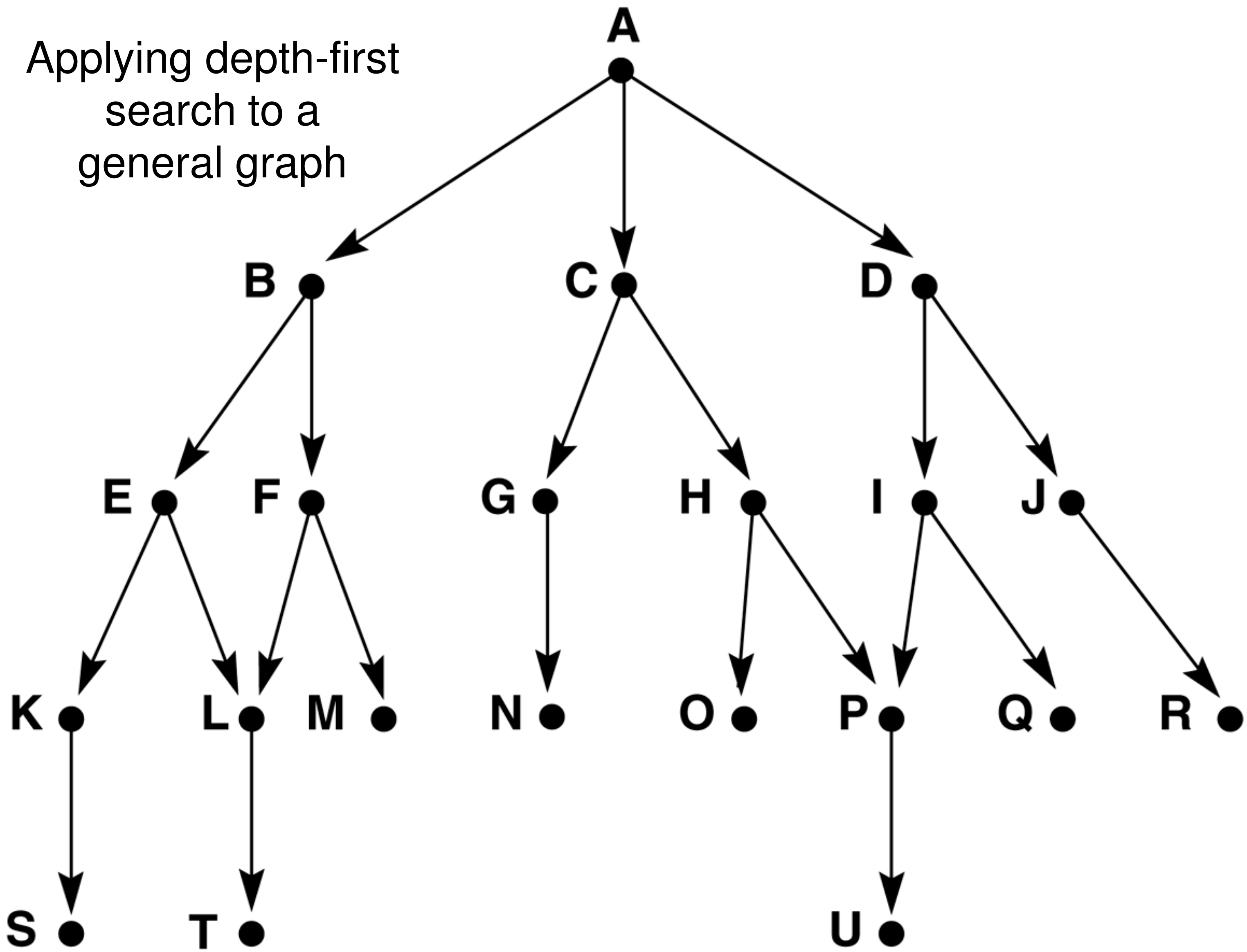
- A backtracking algorithm is essentially a modified depth-first search
- DFS is a generalized preorder traversal, in which a node is visited first, followed by a recursive traversal of its children.

```
void depth_first_tree_search (node v)
{
  node u;
  visit v;
  for (each child u of v)
      depth_first_tree_search(u);
}
```

Figure 5.1: A tree with notes numbered according to a deph-first search

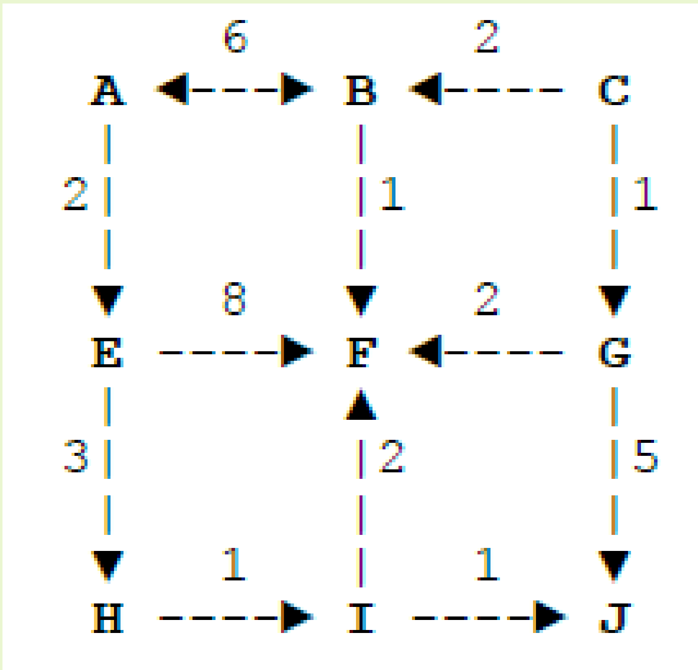


Applying depth-first search to a general graph



Applying depth-first search to a general graph

(Assume that adjacent vertices are visited in alphabetical order)



The n-Queens Problem

- The n-queens problem asks how n queens can be placed on an n x n chess board so that no two queens attack each other
- Instead of looking at all combinations, we notice two queens cannot be in the same row.

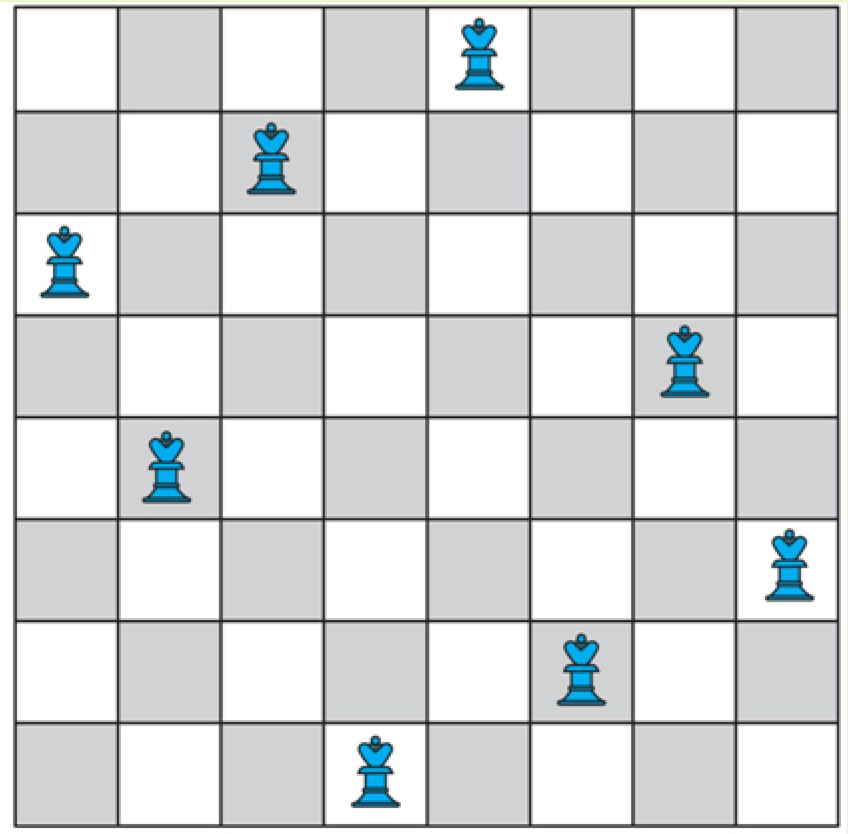
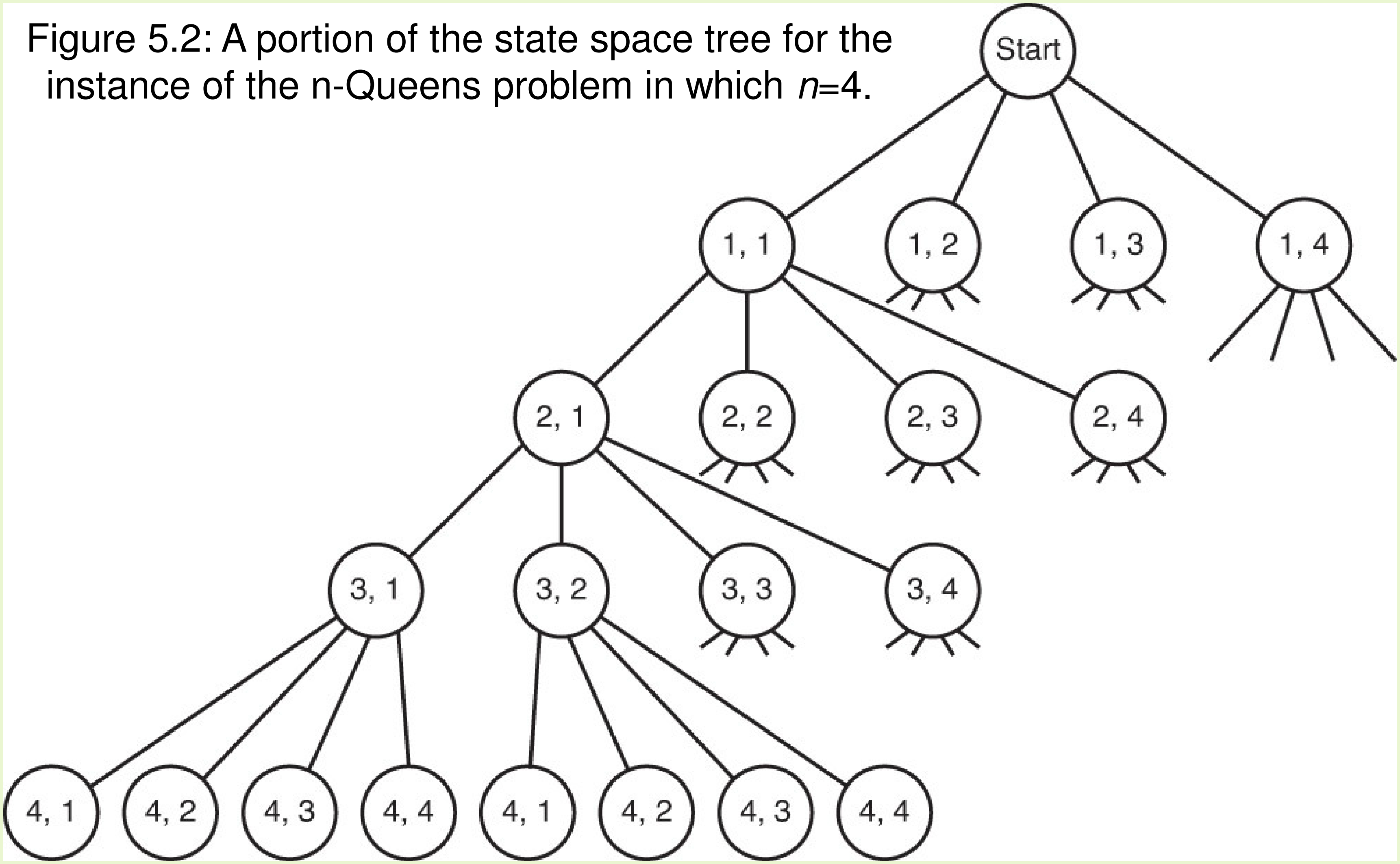


Figure 5.2: A portion of the state space tree for the instance of the n-Queens problem in which n=4.



State Space Trees

- Figure 5.2 is an example of a state space tree, in which each node represents a possible partial solution –or state- to a problem.
- A simple depth-first search would generate and check every path from the root to a leaf until a solution is found ($4^4 = 256$ candidate solutions!)

$$[< 1,1 >, < 2,1 >, < 3,1 >, < 4,1 >]$$
$$[< 1,1 >, < 2,1 >, < 3,1 >, < 4,2 >]$$
$$[< 1,1 >, < 2,1 >, < 3,1 >, < 4,3 >]$$
$$[< 1,1 >, < 2,1 >, < 3,1 >, < 4,4 >]$$
$$[< 1,1 >, < 2,1 >, < 3,2 >, < 4,1 >]$$

The Backtracking Advantage

- In backtracking, we take advantage of the fact that some paths cannot possibly lead to a solution:

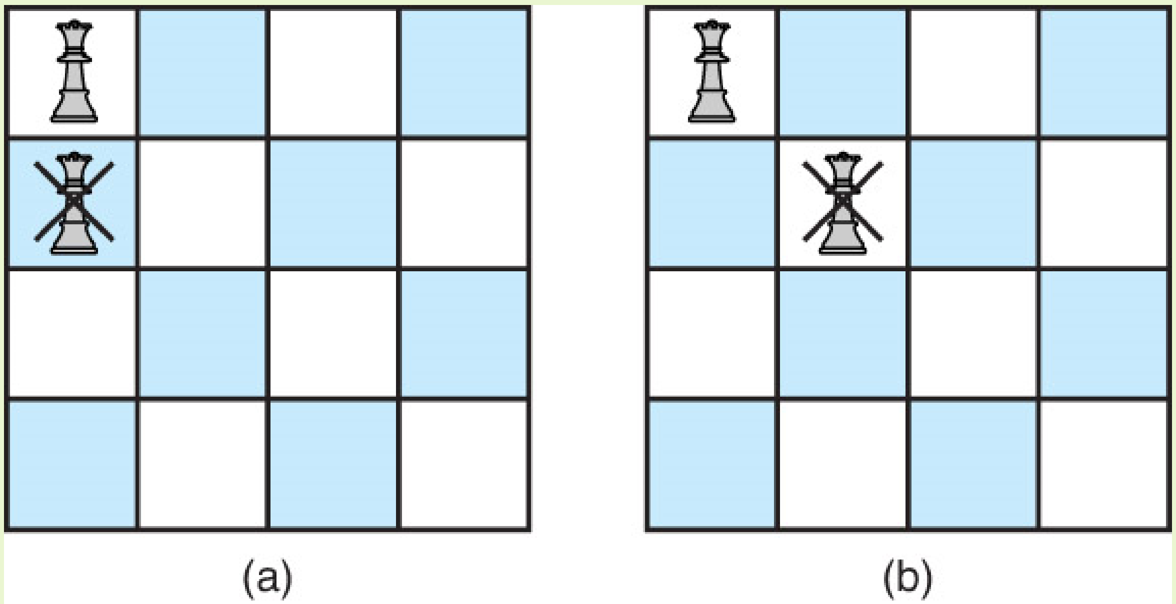


Figure 5.3: Diagram showing that if the first queen is placed in column 1, the second queen cannot be placed in column 1 (a) or column 2 (b).

The Backtracking Algorithm

In backtracking, each node visited via DFS is examined and found to be either

- Nonpromising: cannot lead to a solution, OR
- Promising: needs to be explored further.

Paths involving a nonpromising node are pruned from further consideration, resulting in a pruned state space tree.

```
void checknode (node v)
{
  node u;

  if (promising(v))
     if (there is a solution at v)
        write the solution;
     else
        for (each child u of v)
           checknode(u);
}
```

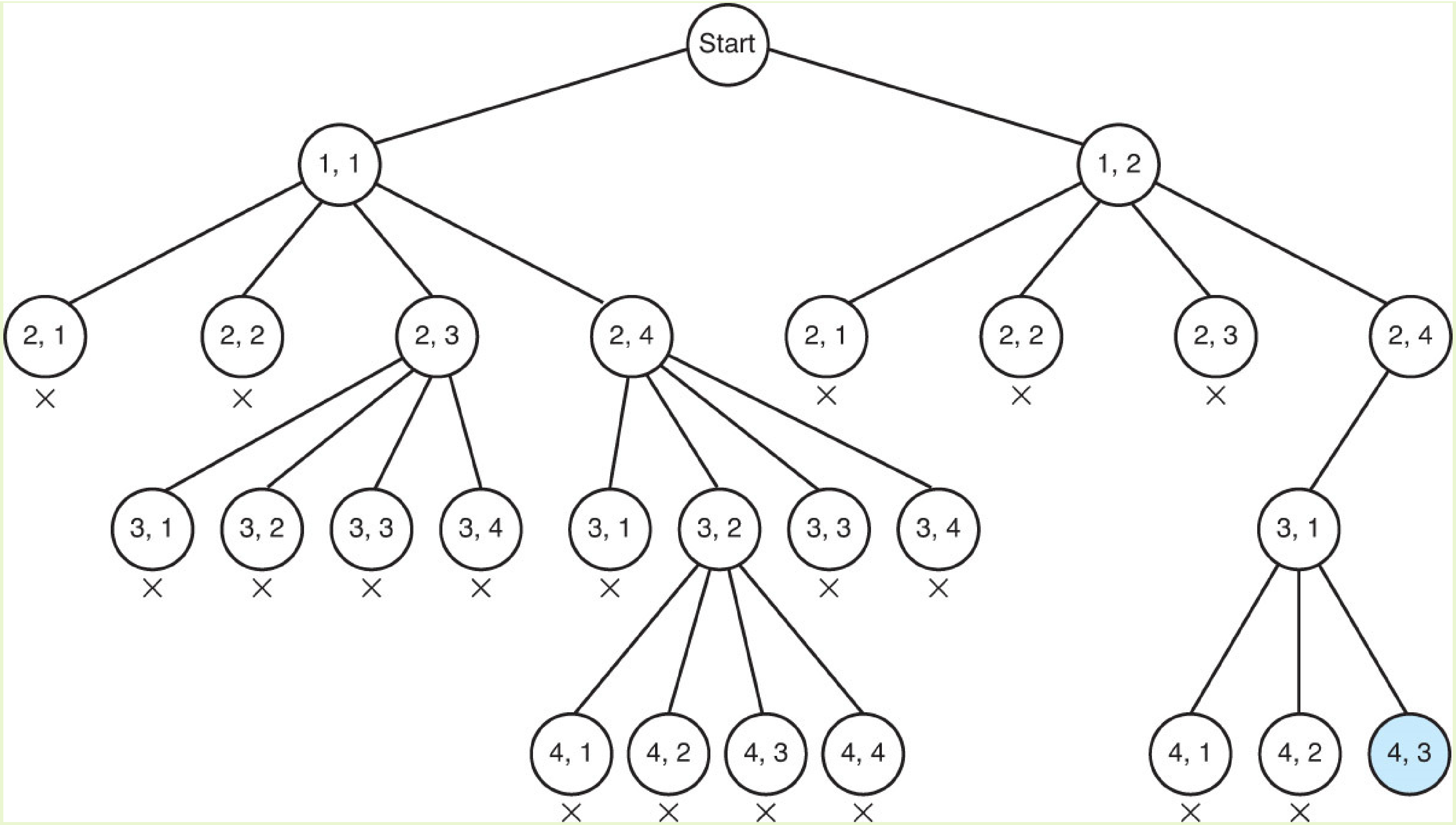


Figure 5.4: A portion of the pruned state space tree produced when backtracking is used to solve the instance of the *n*-Queens problems in which *n*=4.

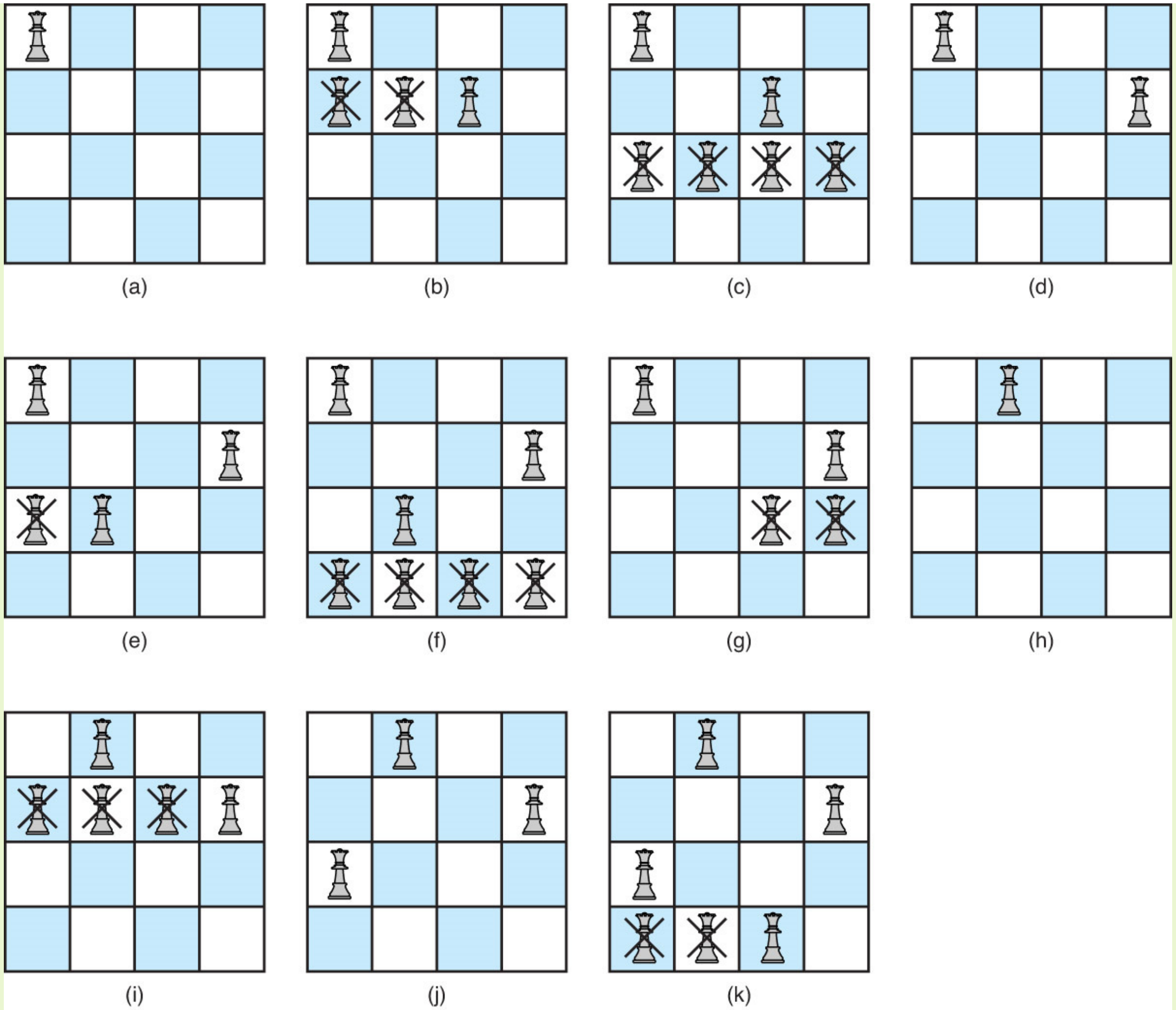


Figure 5.5: The actual chessboard positions that are tried when backtracking is used to solve the instance of the *n*-Queens problem in which *n*=4. Each non promising position is marked with a cross.