Backtracking- NQueens problem

Problem definition:- The **N queens puzzle** is the problem of placing eight queens on an NxN cboard such that no two queens attack each other in the same row, column, or diagonal.

Explicit condition: Assuming the queens and the board both are numbered from 1 to N, and each i^{th} queen is placed in i^{th} row, the explicit condition needs column number where each queen is placed.

Thus, $S_i = \{x_1, x_2, x_3, \ldots\ldots\ldots x_{n-1}, x_n\}$

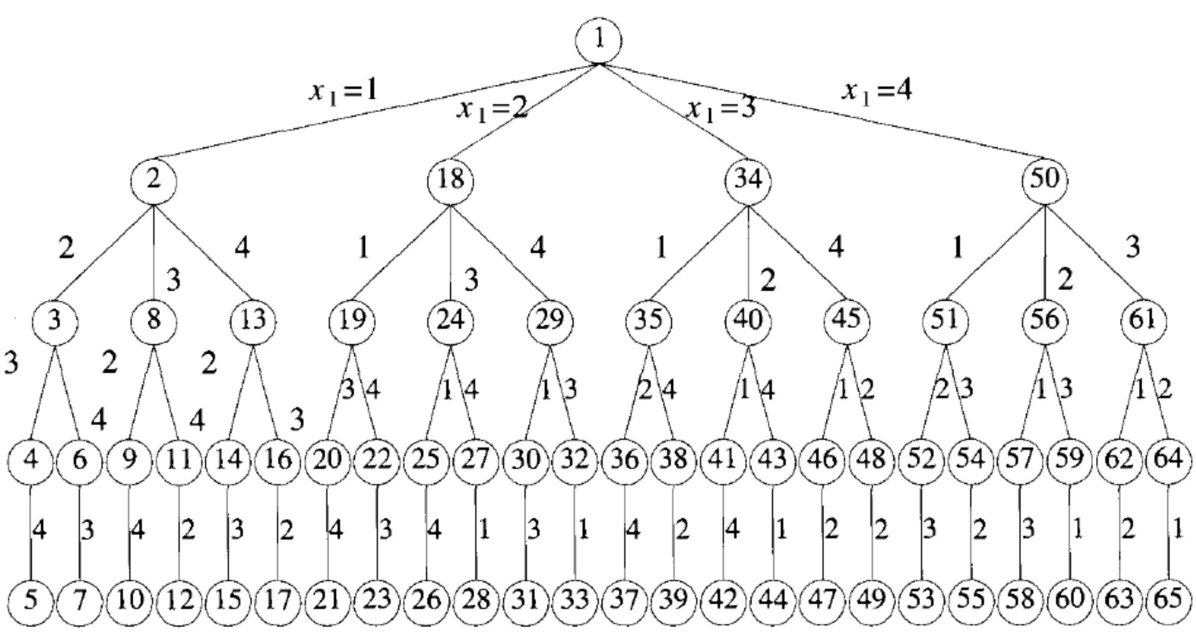
Implicit condition: No two queens attack each other in the same row, column, or diagonal

Backtracking condition-

If ((X[j]=i) OR(Abs(X[j] – i) == Abs(j-k))) // Two queens in same column or same diagonal

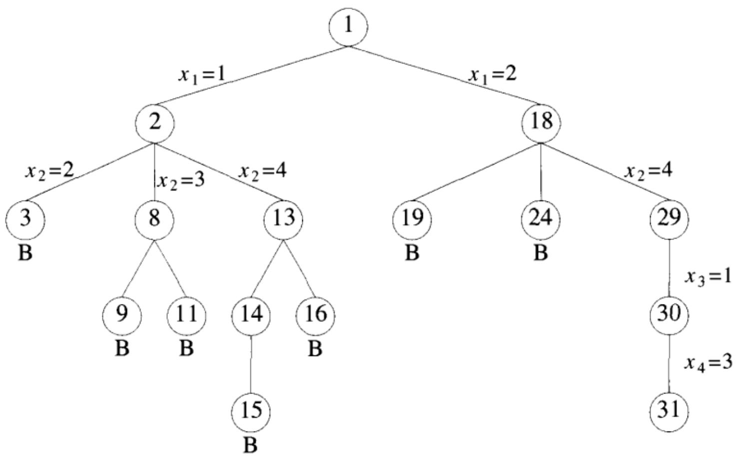
//then backtrack

State space/Solution space tree:



Backtracking tree(answer):

(Partial Backtracking tree showing the first solution)



Answer states:

For N=4,

S= {2,4,1,3}

S={3,1,4,2}

Complexity:

For n=8,

without constraints, T(n)= 64*63*62*……*57 = 1.8*10^{14}

with constraint every i^{th} queen in i^{th} row , T(n)= 8*8*….*8= 8^8

With added constraint of no column attack, T(n)= 8*7*6*…*1= (8!)

With one more added condition of no diagonal attacks, T(n)<= 8!=O(n!)

Backtracking-Graph coloring Problem

Definition: A coloring of a graph G=(V,E) is a mapping F:V → C where C is a finite set of colors such that if <v,w> is an element of E then F(v) is different from F(w); in other words, adjacent vertices are not assigned the same color.

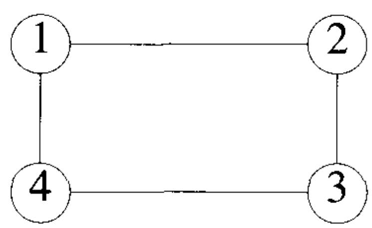
Conditions:

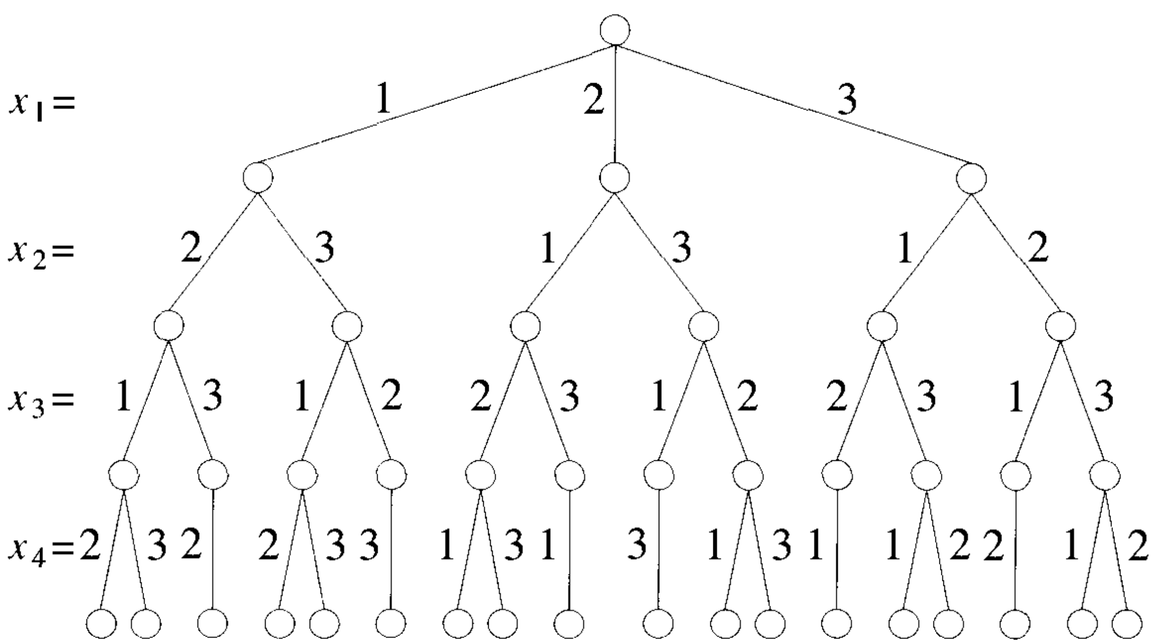
Explicit Condition: A vector X={x1,x2…Xn} for all n vertices for all possible combinations of colors

Implicit Condition: No two adjacent vertices or regions have the same color.

Backtracking Condition: If ((k,i) is an edge) and (Color[i]=Color[k]) then Backtrack!

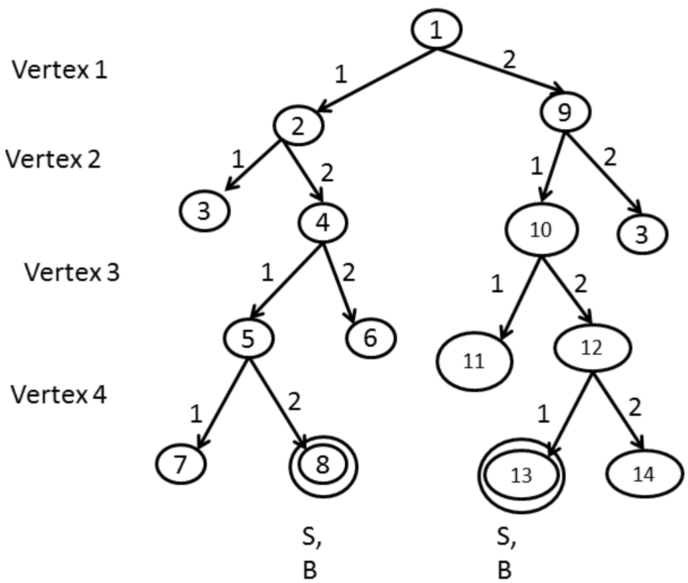
State space tree: (Graph & corresponding state-space tree)



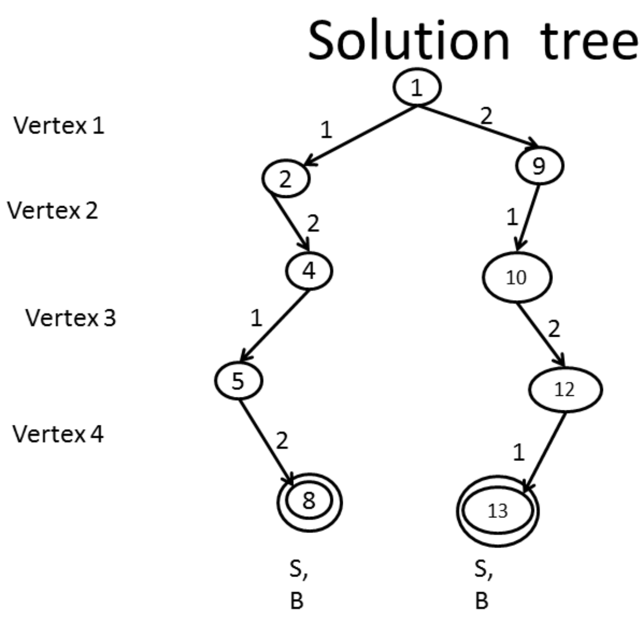


Backtracking tree:

For given graph, calculate minimum chromatic number using backtracking tree.



Solution Tree:



Complexity:

There will be O(V^m) configurations of colors where V= number of vertices & m= Chromatic number.

Backtracking-Sum Of Subsets

Definition: Given a set of non-negative integers, and a value *M*, determine all possible subsets of the given set whose summation sum equal to given *M*.

Conditions:

Explicit Condition: A vector X={x1,x2…Xn} for all n elements in the set where Xi=0 (element not added) or Xi=1 (element added in the solution tuple)

Implicit Condition: summation of the chosen numbers must be equal to given number M and one number can be used only once.

Backtracking Condition:

$$B_k(x_1, \ldots, x_k) = true \text{ iff } \sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i \geq m$$

$$\text{and} \quad \sum_{i=1}^{k} w_i x_i + w_{k+1} \leq m$$

State space tree: (Graph & corresponding state-space tree)

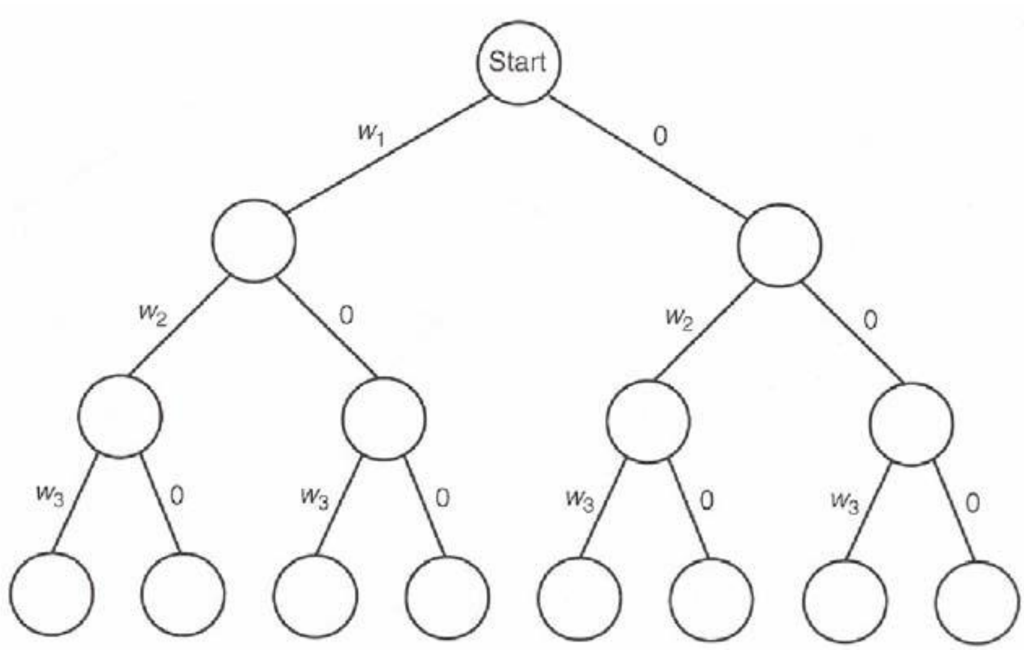


Figure 5.7 ● A state space tree for instances of the Sum-of-Subsets problem in which $n = 3$.

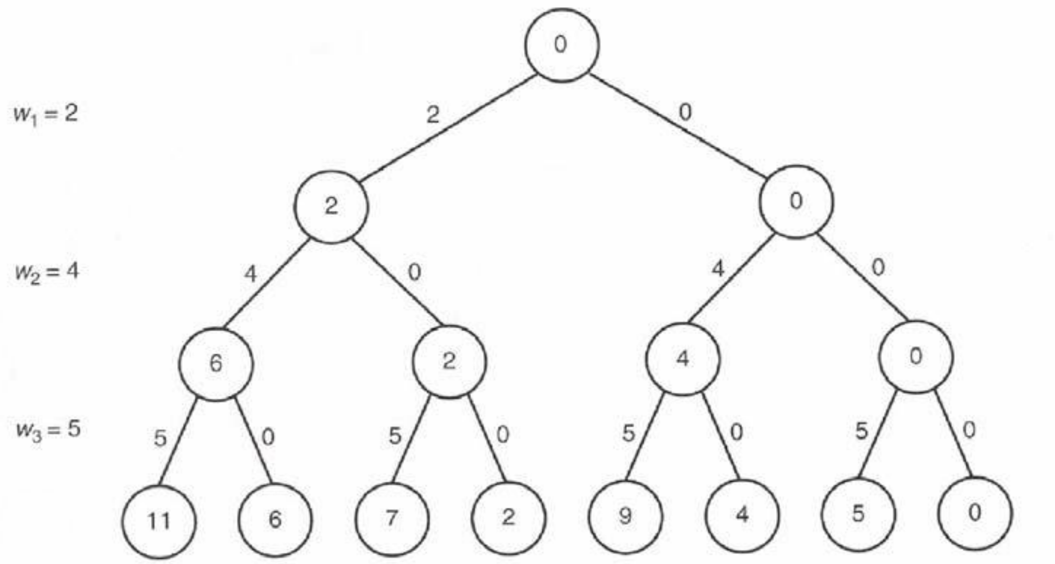


Figure 5.8 • A state space tree for the Sum-of-Subsets problem for the instance in Example 5.3. Stored at each node is the total weight included up to that node.

Backtracking tree:

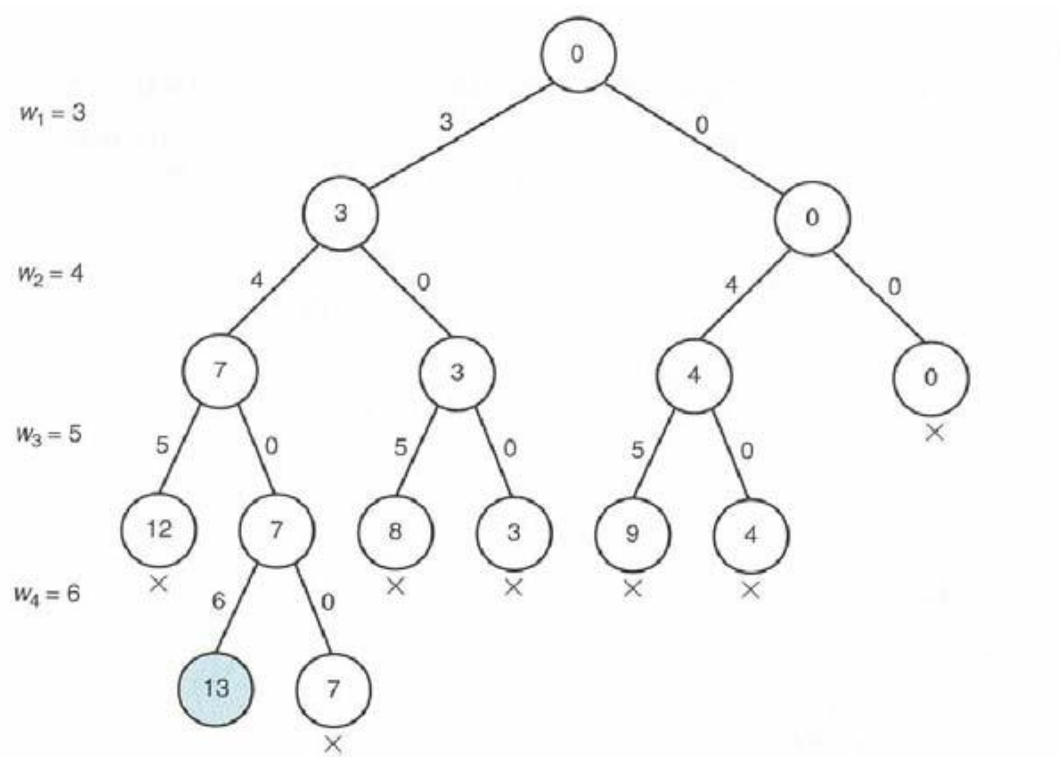


Figure 5.9 • The pruned state space tree produced using backtracking in Example 5.4. Stored at each node is the total weight included up to that node. The only solution is found at the shaded node. Each nonpromising node is marked with a cross.

Complexity:

Complexity of backtracking problems will be O(P(n) * n!) or O(P(n) * 2^n) where P(n) is polynomial in 'n' that depends on the way nodes are generated.