Backtracking

Dr.Jayashree Katti

(Ref:Sahni)

Terminology

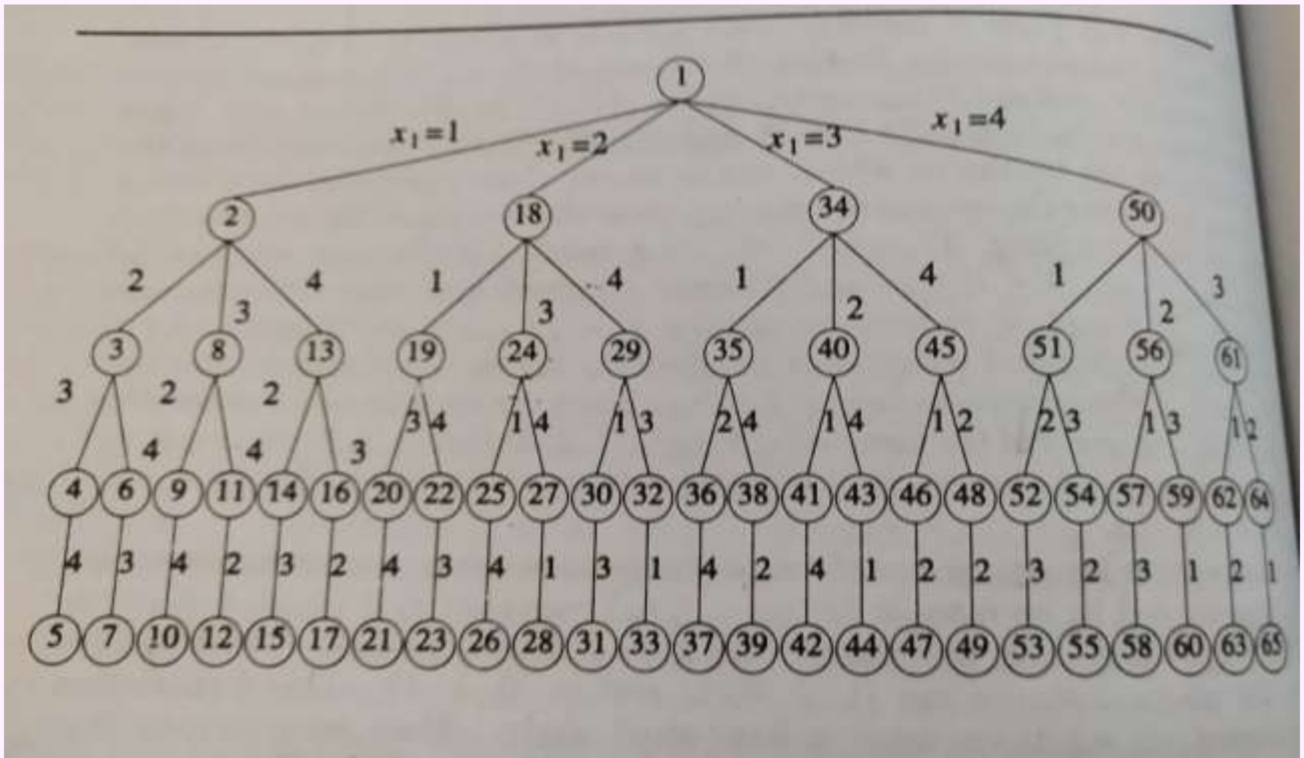
- Tree Organization
- Permutation Tree
- Problem State
- State Space
- Solution State
- Answer State
- State Space Tree

Terminology

- Implicit and explicit constraints
- Criterion Function
- Live Node
- E-node
- Dead Node
- Bounding Functions
- **Backtracking:**
  - Depth first node generation with bounding functions is called backtracking.

Tree Organization of 4-queens solution sapce

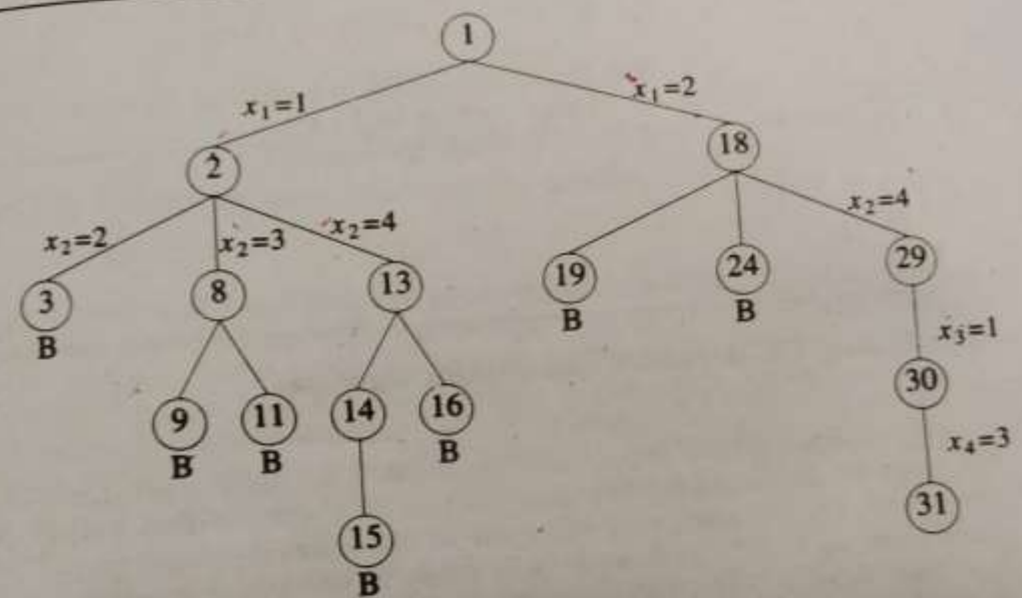
(Nodes Numbered in Depth First Search)



igure 7.2 Tree organization of the 4-queens solution space. Nodes are mbered as in depth first search.

Portion of the tree Generated during Backtracking

gure 7.5 Example of a backtrack solution to the 4-queens problem



igure 7.6 Portion of the tree of Figure 7.2 that is generated during back-
acking

Nqueens Algorithm
Nqueens(k,n)

```
Algorithm NQueens(k, n)
// Using backtracking, this procedure prints all
// possible placements of n queens on an n × n
// chessboard so that they are nonattacking.
{
    for i := 1 to n do
    {
        if Place(k, i) then
        {
            x[k] := i;
            if (k = n) then write (x[1 : n]);
            else NQueens(k + 1, n);
        }
    }
}
```

Algorithm 7.5 All solutions to the n-queens problem

Nqueens Algorithm
Place(k,i)

```
Algorithm Place(k, i)
// Returns true if a queen can be placed in kth row and
// ith column. Otherwise it returns false. x[ ] is a
// global array whose first (k − 1) values have been set.
// Abs(r) returns the absolute value of r.
{
    for j := 1 to k − 1 do
        if ((x[j] = i) // Two in the same column
            or (Abs(x[j] − i) = Abs(j − k)))
                // or in the same diagonal
            then return false;
    return true;
}
```

Algorithm 7.4 Can a new queen be placed?

Tracing

Nqueen(1,4)
i=1 Place (1,1) T
x[1]=1
Nqeen(2,4)

Nqueen(2,4)
i=1 Place (2,1) F
i=2 Place (2,2) F
i=3 Place (2,3) T
x[2]=3
Nqeen(3,4)

Nqueen(3,4)
i=1 Place (3,1) F
i=2 Place (3,2) F
i=3 Place (3,3) F
i=4 Place (3,4) F

q			
		q	

Tracing

Nqueen(1,4)
i=1 Place (1,1) T
x[1]=1
Nqeen(2,4)

Nqueen(2,4)
i=1 Place (2,1) F
i=2 Place (2,2) F
i=3 Place (2,3) F
i=4 Place (2,4) T
x[2]=4
Nqeen(3,4)

Nqueen(3,4)
i=1 Place (3,1) F
i=2 Place (3,2) F
x[3]=2
Nqeen(4,4)

Nqueen(4,4)
i=1 Place (4,1) F
i=2 Place (4,2) F
i=3 Place (4,3) F
i=4 Place (4,4) T

q			
			q
	q		

Tracing

Nqueen(1,4)
i=1 Place (1,1) T
x[1]=1
Nqeen(2,4)

Nqueen(2,4)
i=1 Place (2,1) F
i=2 Place (2,2) F
i=3 Place (2,3) F
i=4 Place (2,4) T
x[2]=4
Nqeen(3,4)

Nqueen(3,4)
i=1 Place (3,1) F
i=2 Place (3,2) F
i=3 Place (3,3) F
i=4 Place (3,3) F

q			
			q

Tracing

Nqueen(1,4)
i=1 Place (1,1) T
x[1]=1
Nqeen(2,4)

Nqueen(2,4)
i=1 Place (2,1) F
i=2 Place (2,2) F
i=3 Place (2,3) F
i=4 Place (2,4) T
x[2]=4
Nqeen(3,4)

Nqueen(3,4)
i=1 Place (3,1) F
i=2 Place (3,2) F
i=3 Place (3,3) F
i=4 Place (3,3) F

q			
			q

Tracing

Nqueen(1,4)
i=1 Place (1,1) T
x[1]=1
Nqeen(2,4)

Nqueen(2,4)
i=1 Place (2,1) F
i=2 Place (2,2) F
i=3 Place (2,3) F
i=4 Place (2,4) T
x[2]=4
Nqeen(3,4)

q			

Tracing

Nqueen(1,4)
i=1 Place (1,1) T
x[1]=2
Nqeen(2,4)

Nqueen(2,4)
i=1 Place (2,1) F
i=2 Place (2,2) F
i=3 Place (2,3) F
i=4 Place (2,4) T
x[2]=4
Nqeen(3,4)

Nqueen(3,4)
i=1 Place (3,1) F
x[3]=1
Nqeen(4,4)

Nqueen(4,4)
i=1 Place (4,1) F
i=2 Place (4,2) F
i=3 Place (4,3) F
x[4]=3

	q		
			q
q			
		q	

Recursive Backtracking Algorithm

```
Algorithm Backtrack(k)
// This schema describes the backtracking process using
// recursion. On entering, the first k - 1 values
// x[1], x[2], ..., x[k - 1] of the solution vector
// x[1 : n] have been assigned. x[ ] and n are global.
{
    for (each x[k] ∈ T(x[1], ..., x[k - 1]) do
    {
        if (B_k(x[1], x[2], ..., x[k]) ≠ 0) then
        {
            if (x[1], x[2], ..., x[k] is a path to an answer node)
                then write (x[1 : k]);
            if (k < n) then Backtrack(k + 1);
        }
    }
}
```

Algorithm 7.1 Recursive backtracking algorithm

Iterative Backtracking Algorithm

```
Algorithm IBacktrack(n)
// This schema describes the backtracking process.
// All solutions are generated in x[1 : n] and printed
// as soon as they are determined.
{
    k := 1;
    while (k ≠ 0) do
    {
        if (there remains an untried x[k] ∈ T(x[1], x[2], ...,
            x[k − 1]) and B_k(x[1], ..., x[k]) is true) then
        {
                if (x[1], ..., x[k] is a path to an answer node)
                    then write (x[1 : k]);
                k := k + 1; // Consider the next set.
        }
        else k := k − 1; // Backtrack to the previous set.
    }
}
```

Algorithm 7.2 General iterative backtracking method

The ... edges from level ... which is either zero or one. All paths from the root to a leaf node define the solution space. The left subtree of the root defines all subsets containing w_1, the right subtree defines all subsets not containing w_1, and so on. Now there are 2^4 leaf nodes which represent 16 possible tuples. □

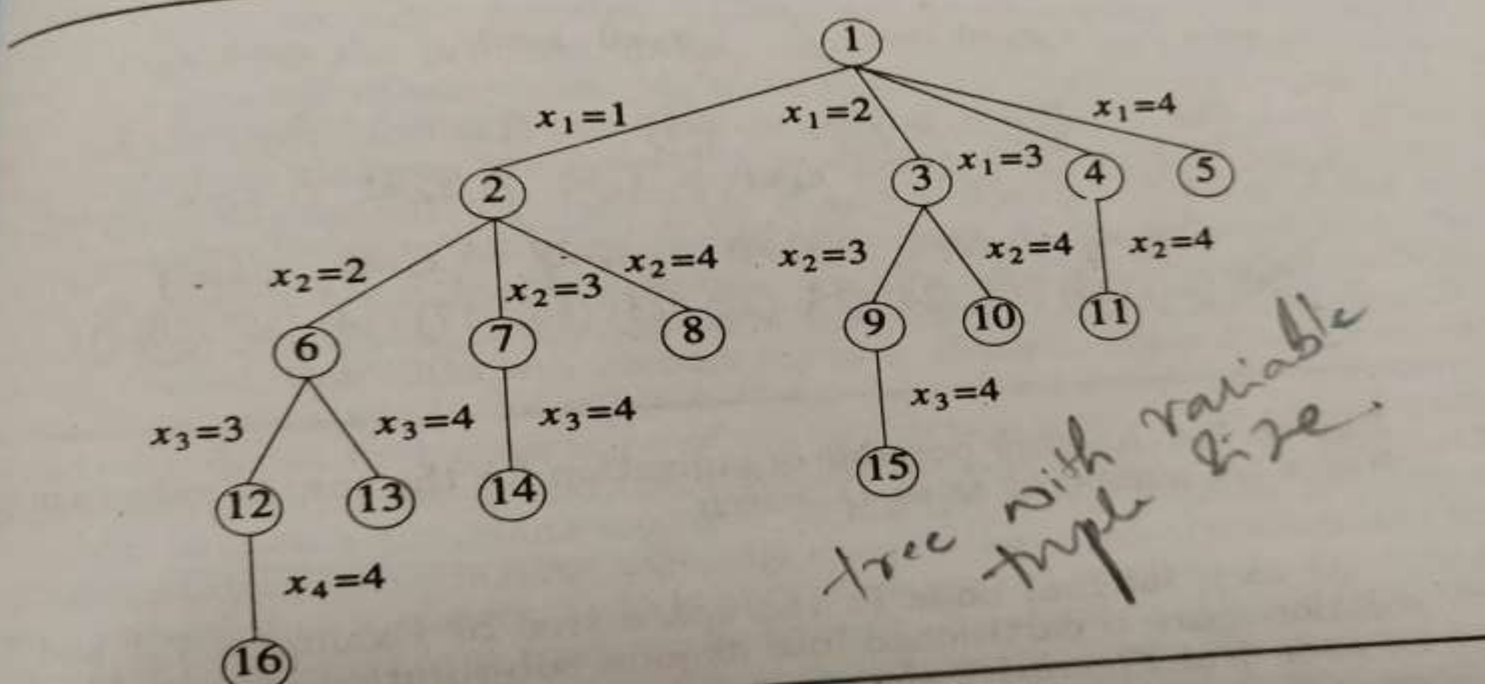


Figure 7.3 A possible solution space organization for the sum of subsets ... bered as in breadth-first search.

the solution space is referred to as the *state space tree.*

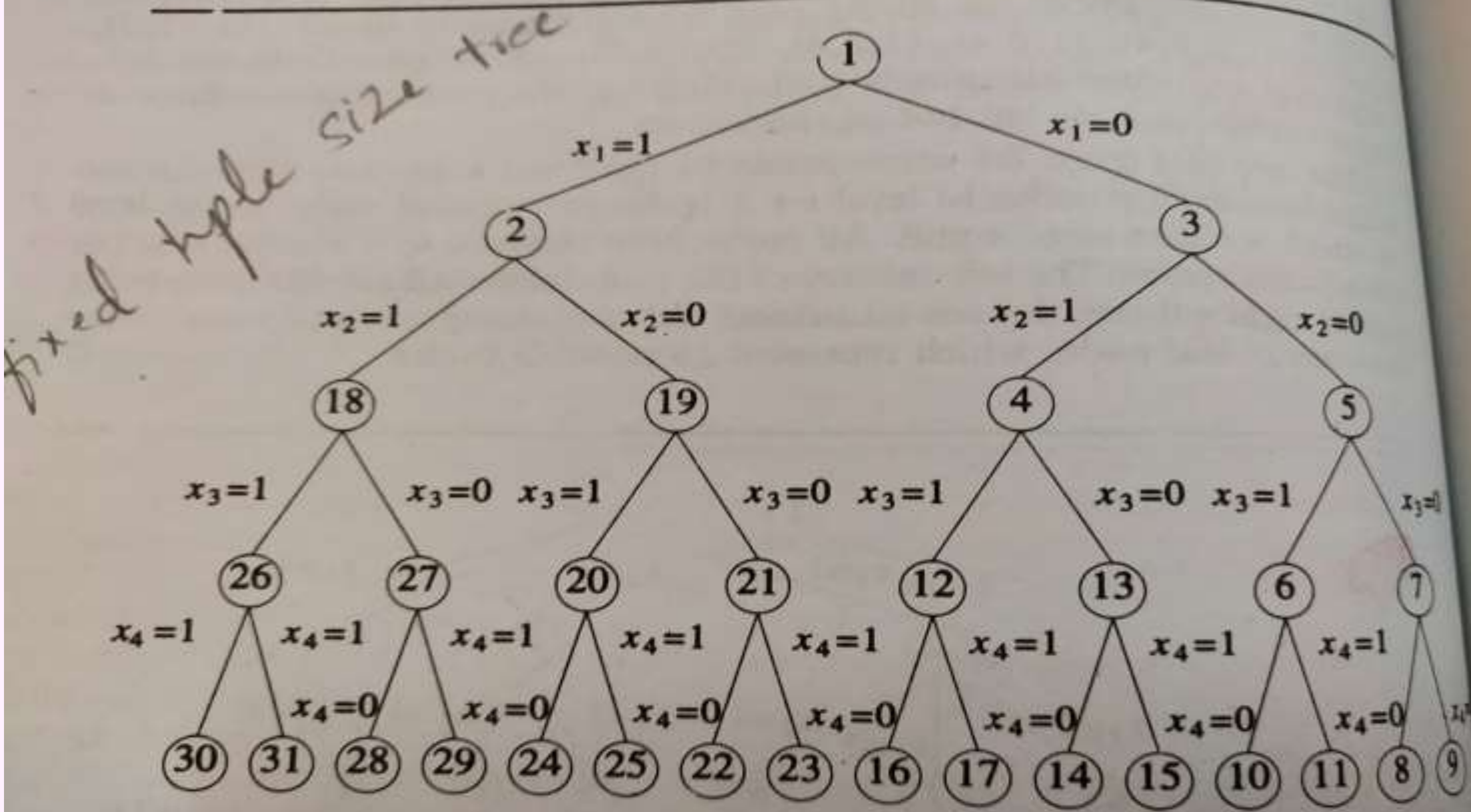


Figure 7.4 Another possible organization for the sum of subsets problems.

			Q1
		Q2	
Q3			
		Q4	

|1-2|------|4-3|