Backtracking and Branch & Bound

Sonali Gogate

Sum of subsets (1/3)

Given positive numbers wi, (1 <= i <= n) and a value m – the problem calls for finding all subsets of wi whose sum is m.

Suppose

n = 4 & (w1, w2, w3, w4) = (11, 13 ,24, 7).   m = 31,

Which subsets will be the solution subsets?

1. (11, 13, 7)
2. (24, 7)

If we describe the solution by indices of the values, the solution sets are (1, 2, 4) and (3, 4).

The solution will have a k-tuple (x1, … xk) with 1 <= k <= n.

Sum of subsets (2/3)

Explicit constraint –

Xi = { i | i is an integer and 1 <= i <= n}

Implicit constraint –

No 2 Xi be same (no number to be taken more than once in the sum).

As (1, 2, 4) subset is same as (1 4, 2)

Another implicit constraint –

Xi < X(i+1) for 1 <= i <= k

Sum of subsets (3/3)

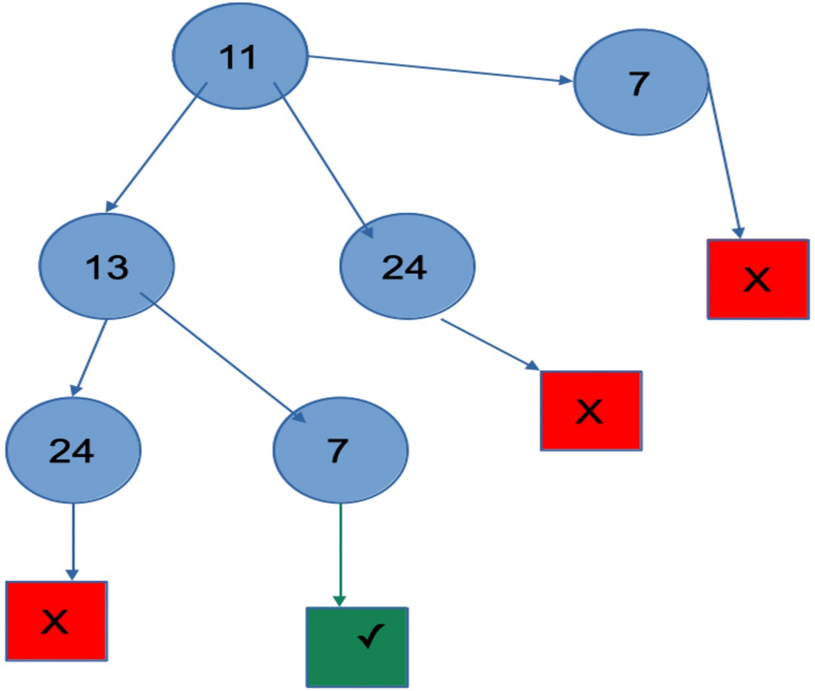
Another way to express the solution to the sum of subsets problem:

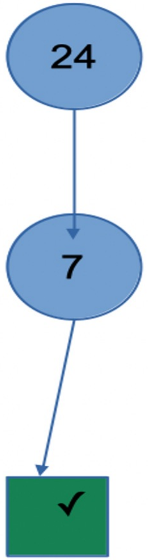
Solution set is always an n-tuple (x1… xn) such that each xi takes value from {0, 1}, 1 <= i <= n.

So it is an n-tuple of 0s and 1s.

Solutions in this format for the above example are (1, 1, 0, 1) and (0, 0, 1, 1)

How can we arrive at the solution set?





(W1,w2,w3,w4)

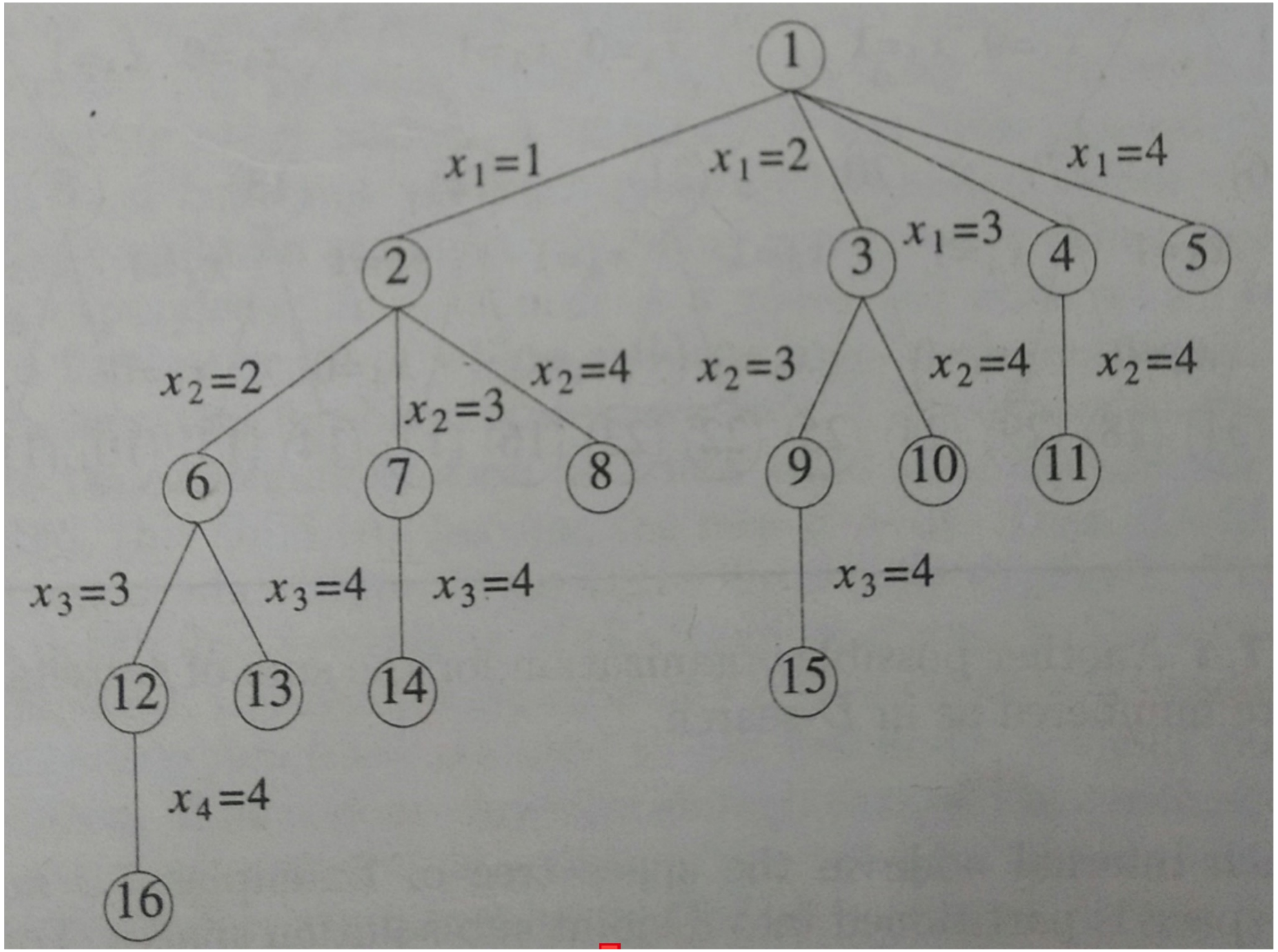
= (11, 13, 24, 7)

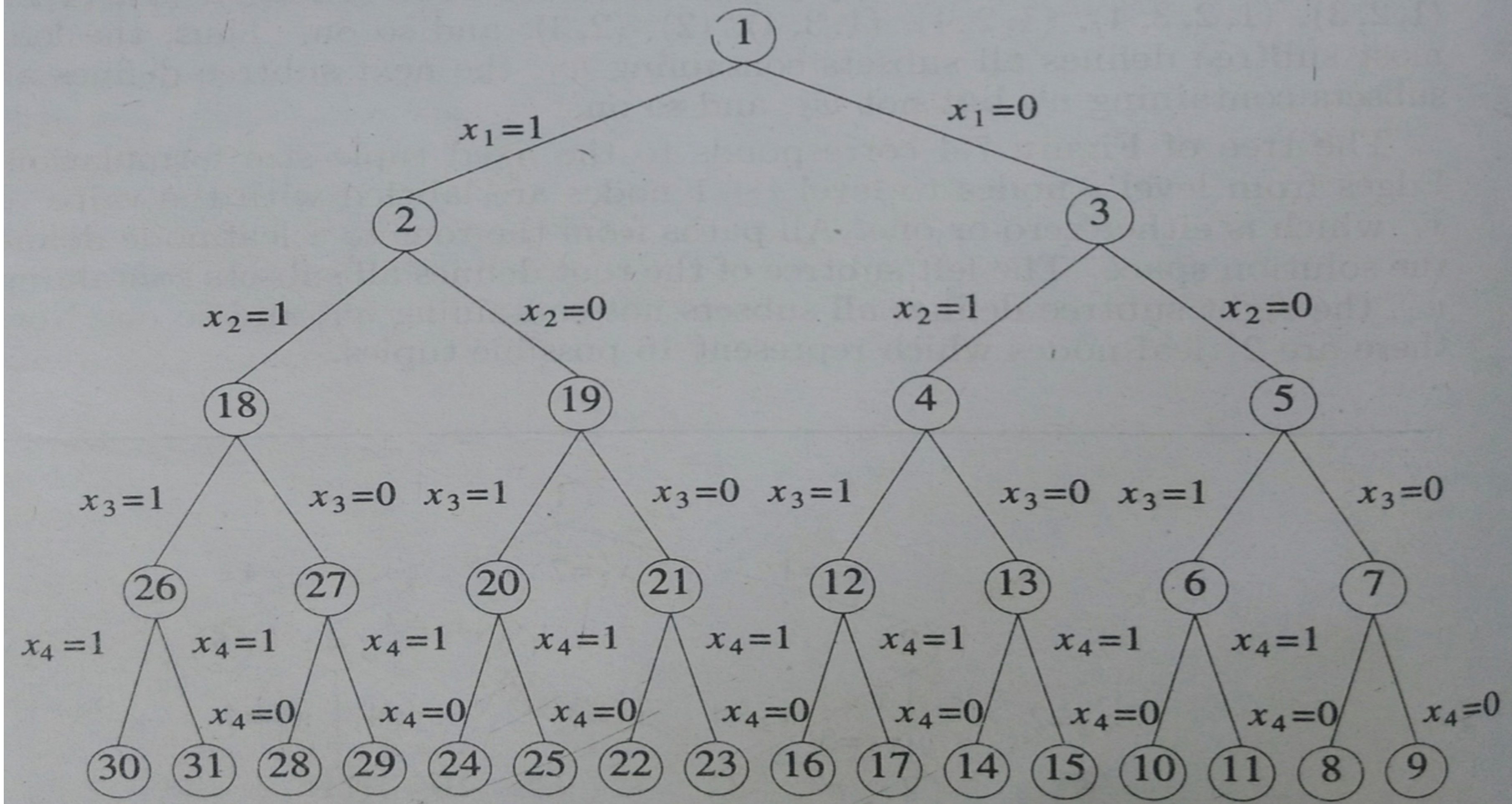
1	2	3	4
11	13	24	7

Solution sets -

(1, 2, 4) OR (1, 1, 0, 1)

(3, 4) OR (0, 0, 1, 1)




1
$x_1=1$
$x_1=0$
2
3
$x_2=1$
$x_2=0$
$x_2=1$
$x_2=0$
18
19
4
5
$x_3=1$
$x_3=0$
$x_3=1$
$x_3=0$
$x_3=1$
$x_3=0$
$x_3=1$
$x_3=0$
26
27
20
21
12
13
6
7
$x_4=1$
$x_4=0$
30
31
28
29
24
25
22
23
16
17
14
15
10
11
8
9

Terminology (1)

Explicit Constraints:

Explicit constraints are rules that restrict each xi the to take on values only from a given set.

Examples –

- xi >= 0
- xi = 0 or 1
- Lbound <= xi <= Ubound

Implicit Constraints:

Implicit constraints are rules that determine which of the tuples in the solution space satisfy criterion function. The Implicit constraints basically describe how xi must relate to each other.

Terminology (2)

State space tree:

A tree constructed from:

1. All of the possible states of the problem as nodes
2. Connected via state transitions from some initial state as root to some terminal state as leaf

Terminology (3)

Live node is a node that has been generated but whose children have not yet been generated.

E-node is a live node whose children are currently being explored. In other words, an E-node is a node currently being expanded.

Dead node is a generated node that is not to be expanded or explored any further. All children of a dead node have already been expanded

Terminology (4)

Solution Space:

Tuples that satisfy the explicit constraints define a solution space.

Problem States:

Each node in the tree defines a problem state.

Solution States:

Solution states are those problem states S for which the path from the root to S defines a tuple in the solution space.

Answer States:

Answer states are solution states S for which the path from the root to S defines a tuple that is a member of the set of solutions of the problem.

Terminology (5)

Bounding Function:

Function or restriction used to kill nodes without generating all its child nodes.

Static Tree:

The tree organizations are independent of the problem instance being solved.

Dynamic Tree:

Tree organizations are problem instance dependent.

Solving the Problem

Once a state space tree is conceived, the problem can be solved by systematically generating the problem states, determining which of the states are solution states and finally determining which solution states are answer states.

2 fundamentally different ways of generating problem states –

Method 1: As soon as new child C of current E-node R is generated, this child node will become the E-node. Then R will become the E-node again, after the subtree rooted at C is fully explored. (Depth first generation)

Method 2: E-node remains the E-node until it becomes a Dead node. (Breadth first generation)

Solving CSP

Backtracking is a technique used to find solution for constraint satisfaction problem.

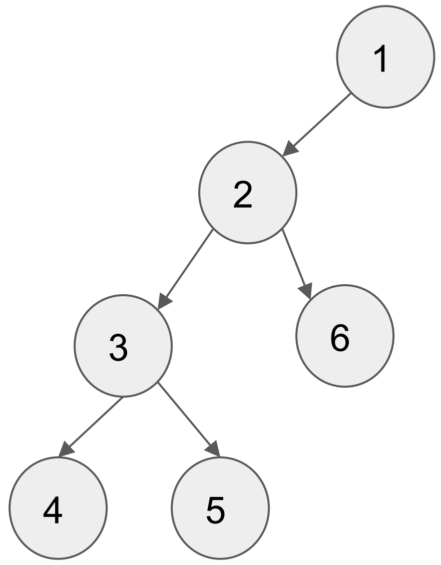
A **constraint satisfaction problem (CSP)** is a problem that requires its solution within some limitations or conditions also known as constraints. It consists of the following:

- A finite set of **variables** which store the solution (V = {V1, V2, V3,....., Vn})
- A set of **discrete** values known as **domain** from which the solution is picked (D = {D1, D2, D3,.....,Dn})
- A finite set of **constraints** (C = {C1, C2, C3,......, Cn})

(https://study.com/academy/lesson/constraint-satisfaction-problems-definition-examples.html)

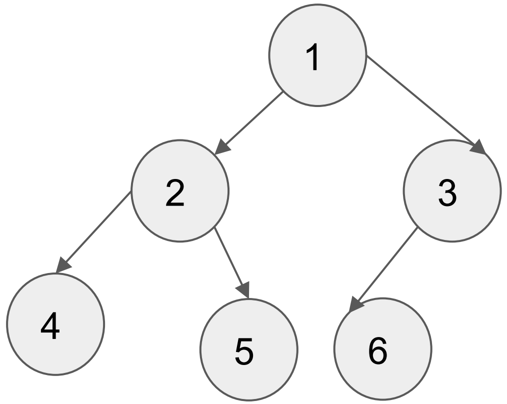
Depth first Generation

Method 1: As soon as new child C of current E-node R is generated, this child node will become the E-node. Then R will become the E-node again, after the subtree rooted at C is fully explored.



Breadth first Generation

Method 2: E-node remains the E-node until it becomes a Dead node.



Backtracking and Branch & Bound

- Depth first node generation with bounding functions is Backtracking.

- Breadth first node generation with bounding functions is Branch & Bound.

Bounding Functions

What is a good bounding function?

The one that substantially reduces the number of nodes needed to be generated.

Good bounding functions can take longer to evaluate.

Since reduction in overall computing time is desirable, there might need to be a trade off as far as the bounding function is concerned.

Feasible Solution and Optimal Solution

In the standard terminology of optimization problems

- **Feasible solution** is one that satisfies all the problem’s constraints.

- **Optimal solution** is a feasible solution with the best value of the objective function.

B & B (1)

- Branch and Bound is an Algorithm Design Paradigm for discrete and combinatorial optimization problems.
- It consists of systematic enumeration of candidate solutions by means of state space search – the set of solutions is thought of as forming a rooted tree with the full set at the root.
- The algorithm explores the branches of the tree, which represent subsets of the solution set.

B & B (2)

- Goal of a B & B algorithm is to find a value x that maximizes or minimizes the value of a real-valued function f(x), called an objective function, among some set S of admissible, or candidate solutions.
- The set S is called the search space, or feasible region.
- It recursively splits the search space into smaller spaces, then minimizing f(x) on these smaller spaces.
- the splitting is called branching.

B &B (3)

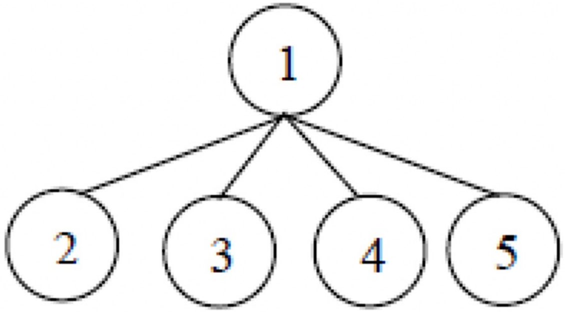
- It is a state space search method in which all the children of a node are generated before expanding any of its children.

3 Variants

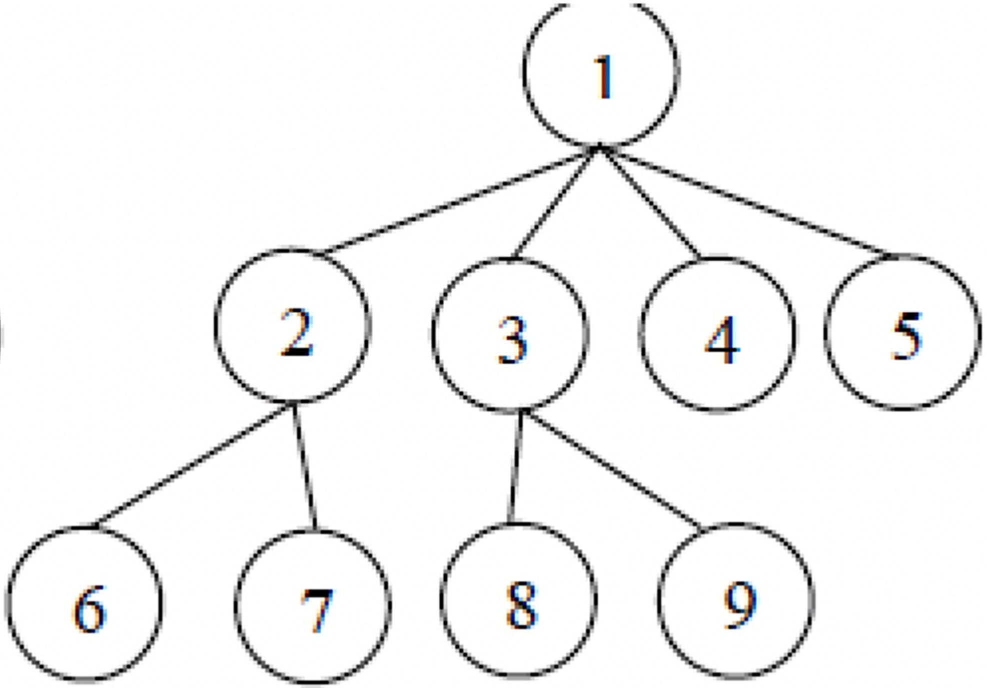
1. FIFO or BFS

2. LIFO or D-search

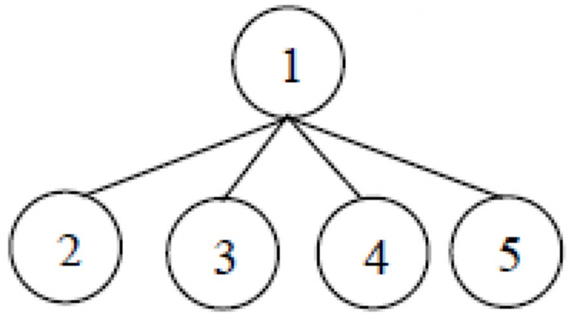
3. LC search



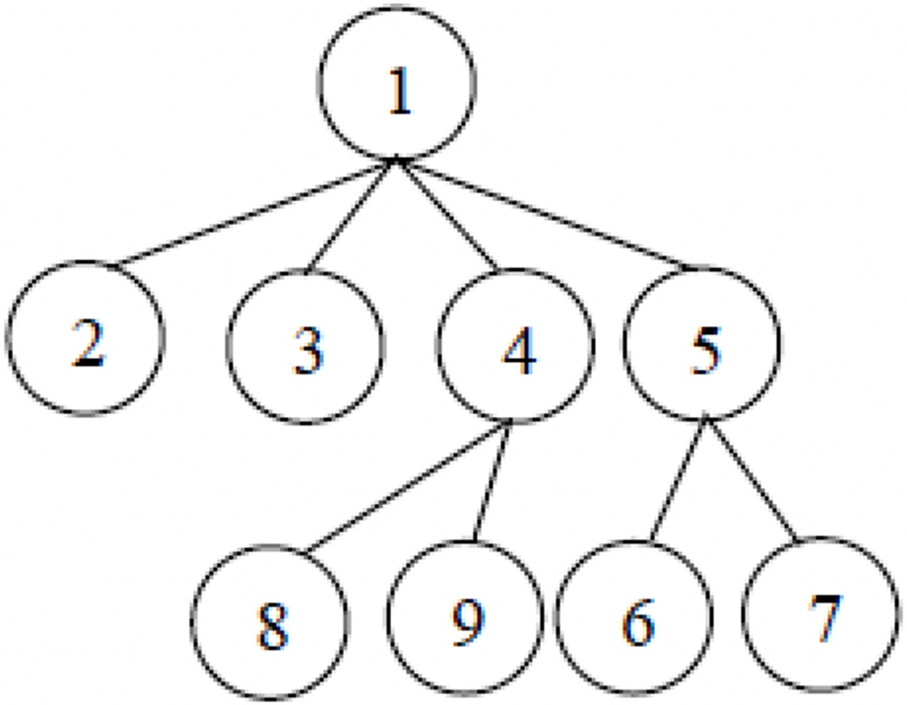
Live Node: 2, 3, 4, and 5



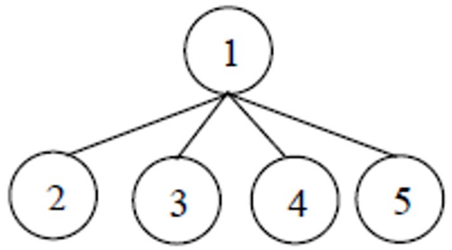
FIFO Branch & Bound (BFS)
Children of E-node are
inserted in a queue.



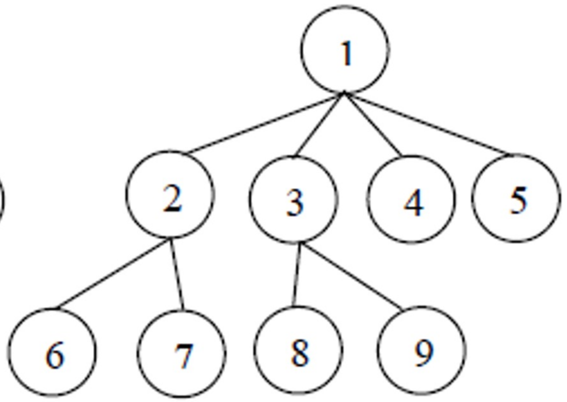
Live Node: 2, 3, 4, and 5



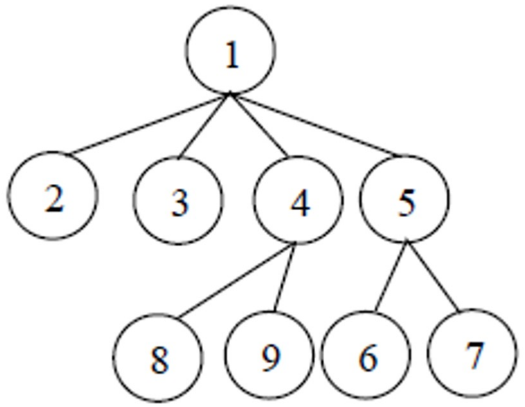
LIFO Branch & Bound (D-Search)
Children of E-node are inserted in a stack.



Live Node: 2, 3, 4, and 5



FIFO Branch & Bound (BFS)
Children of E-node are inserted in a queue.



LIFO Branch & Bound (D-Search)
Children of E-node are inserted in a stack.

LC – Search (Least Cost Search)

- The selection rule for the next E-node in FIFO or LIFO branch-and-bound is sometimes “blind”. i.e. the selection rule does not give any preference to a node that has a very good chance of getting the search to an answer node quickly.

- The search for an answer node can often be speeded by using an “intelligent” ranking function, also called **an approximate cost function ĉ**

- The E-node is the live node with the best ĉ value