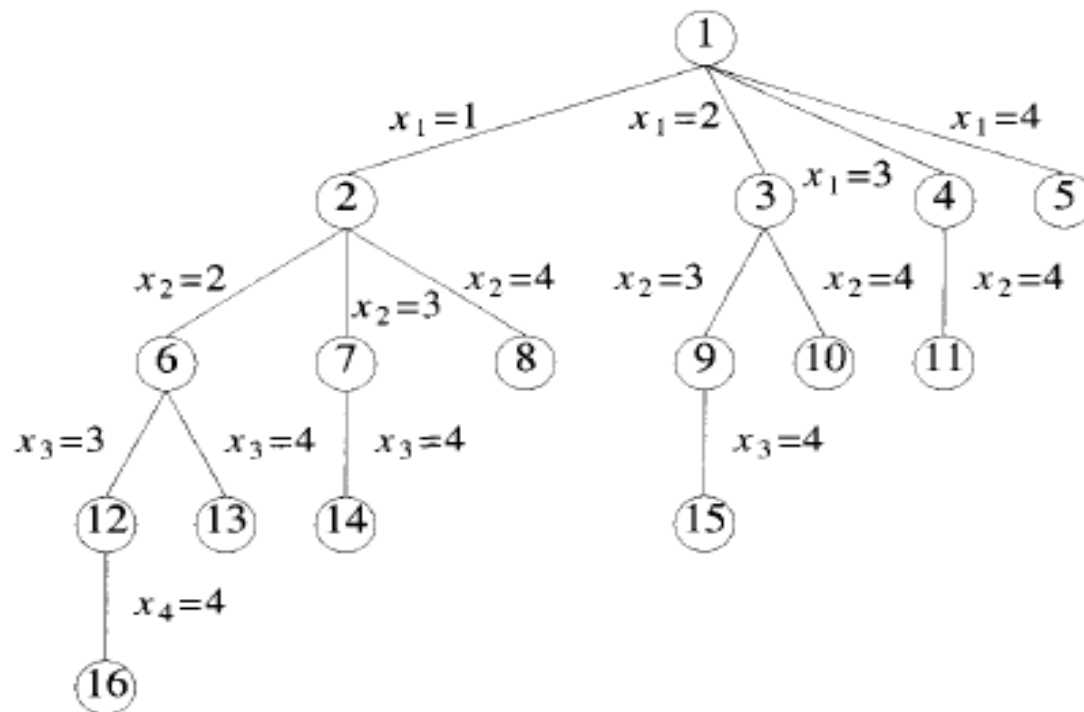


Branch and Bound

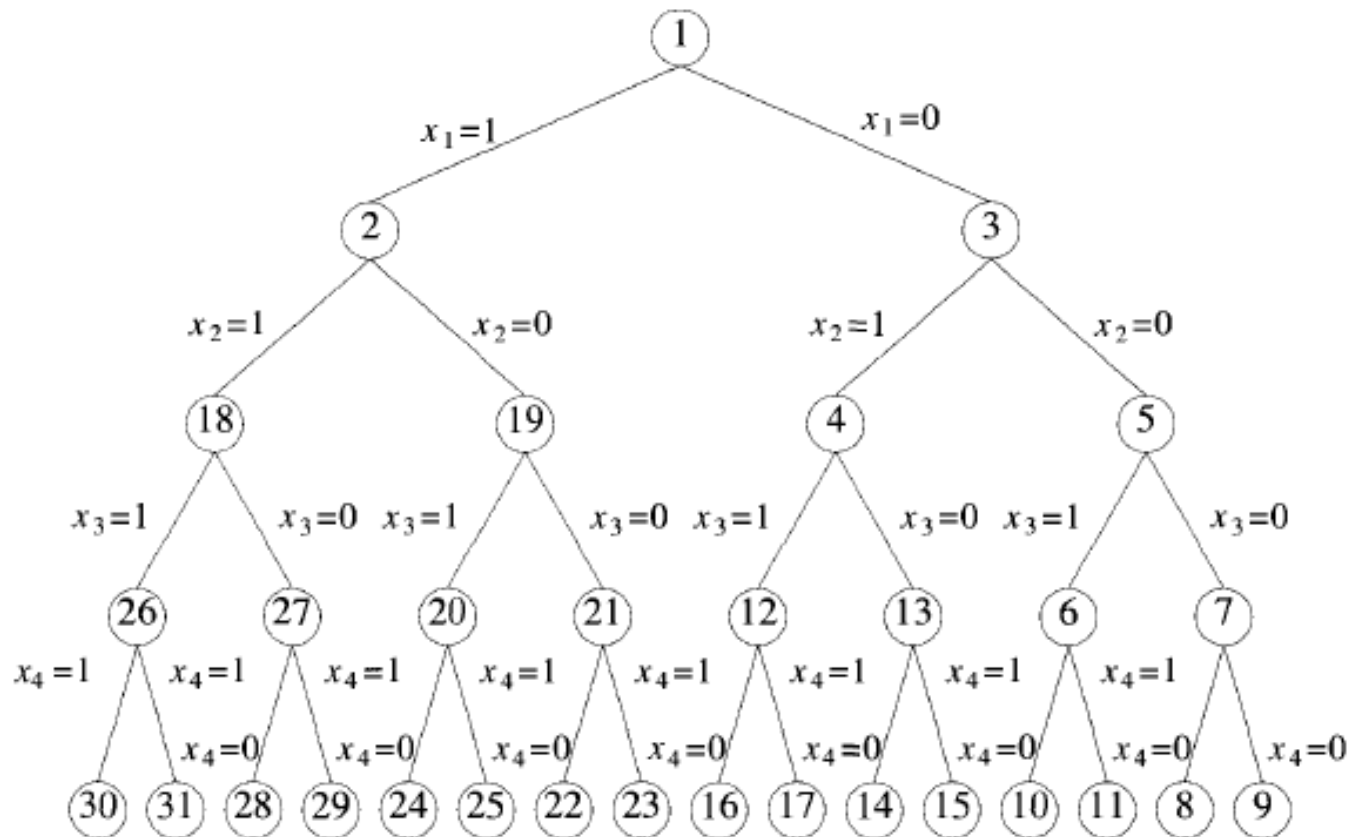
Branch and Bound -Terminology

BREADTH-FIRST-SEARCH : Branch-and Bound with each new node placed in a queue .The front of the queue becomes the new E-node.



Branch and Bound -Terminology

DEPTH-SEARCH (D-Search) : New nodes are placed in to a stack.The last node added is the first to be explored.



Branch and Bound Principal

The term branch-and-bound refers to all state space search methods in which all children of the ϵ -node are generated before any other live node can become the ϵ -node.

We have already seen two graph search strategies, BFS and D-search, in which the exploration of a new node cannot begin until the node currently being explored is fully explored.

Both of these generalize to branch-and-bound strategies.

In branch-and-bound terminology, a BFS-like state space search will be called FIFO (First In First Out) search as the list of live nodes is a first-in-first-out list (or queue).

A D-search-like state space search will be called LIFO (Last In First Out) search as the list of live nodes is a last-in-first-out list (or stack)

Least Cost Branch and Bound(LCBB)

The search for an answer node can often be speeded by using an "intelligent" ranking function, $\hat{c}(\cdot)$, for live nodes. The next E-node is selected based on ranking function

How to assign rank for each node ?

For a node x ,

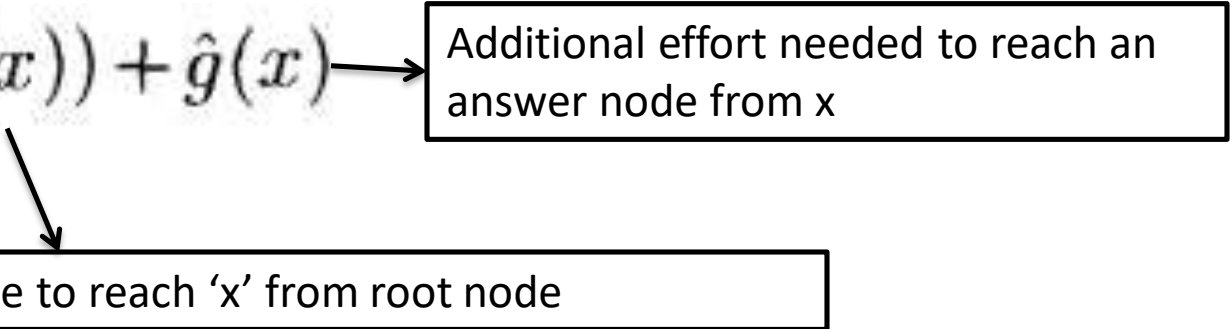
- the number of nodes in the subtree ' x ' that need to be generated before an answer node is generated.
- the number of levels the nearest answer node (in the subtree ' x ') is from ' x ' .

But in both approaches to calculate the cost of node ' x ' we need to search the subtree ' x ' for the answer node .

Hence, by the time the cost of a node is determined, that subtree has been searched and there is no need to explore ' x ' again.

Search algorithms usually rank nodes only on the basis of an estimate $\hat{g}(\cdot)$ of their cost.

Least Cost Branch and Bound(LCBB)

$$\hat{c}(x) = f(h(x)) + \hat{g}(x)$$


Additional effort needed to reach an answer node from x

Effort done to reach 'x' from root node

Node which is having least cost $\hat{c}(\cdot)$ will be the next E-node

Two Special cases can be drawn from LC search

If $\hat{g}(x)=0$ and $f(h(x)) = \text{level of node 'x'}$, then LC search generates nodes by levels. Nothing but BFS-search or FIFO -search

If $f(h(x)) \equiv 0$ and $\hat{g}(x) \geq \hat{g}(y)$ whenever y is a child of x , then the search is essentially a D -search. or LIFO -search

Control Abstraction for LC -Search

```
listnode = record {  
    listnode * next, * parent; float cost;  
}
```

```
1  Algorithm LCSearch(t)  
2  // Search t for an answer node.  
3  {  
4      if *t is an answer node then output *t and return;  
5      E := t; // E-node.  
6      Initialize the list of live nodes to be empty;  
7      repeat  
8      {  
9          for each child x of E do  
10         {  
11             if x is an answer node then output the path  
12                 from x to t and return;  
13             Add(x); // x is a new live node.  
14             (x → parent) := E; // Pointer for path to root.  
15         }  
16         if there are no more live nodes then  
17         {  
18             write ("No answer node"); return;  
19         }  
20         E := Least();  
21     } until (false);  
22 }
```


Problem with the FIFO,LIFO and LC Search

- Generated all the nodes.
- Terminate in a finite state space tree which is having an answer. And process all the nodes in a finite state space tree which does not have a solution.
- Terminates in an infinite state space tree after processing so many nodes which is having a solution. And processing all node in an infinite state space tree which does not have any solution is very difficult.

Solution is Bound the nodes by maintaining the an upper bound

Solution

Maintain an *upper* is the upper bound on the cost of a minimum cost solutions

Now kill all the live nodes whose cost is greater than *upper*

The starting values for *upper* can be obtained by some heuristic or can be set to ∞

As long as the initial value for upper is no less than the cost of a minimum-cost answer node, the above rules to kill live nodes will not result in the killing of a live node that can reach a minimum-cost answer node. Each time a new answer node is found, the value of upper can be updated

This is nothing but **BOUNDING**

Search Techniques with bounding are called as **branch and bound**

LC-Search with Bounding is **LCBB**

FIFO Search with Bounding is **FIFOBB**

LIFO Search with Bounding is **LIFOBB**

For every problem -how to find the cost a node ,upper value at the node and upper bound for the problem to be taken care.