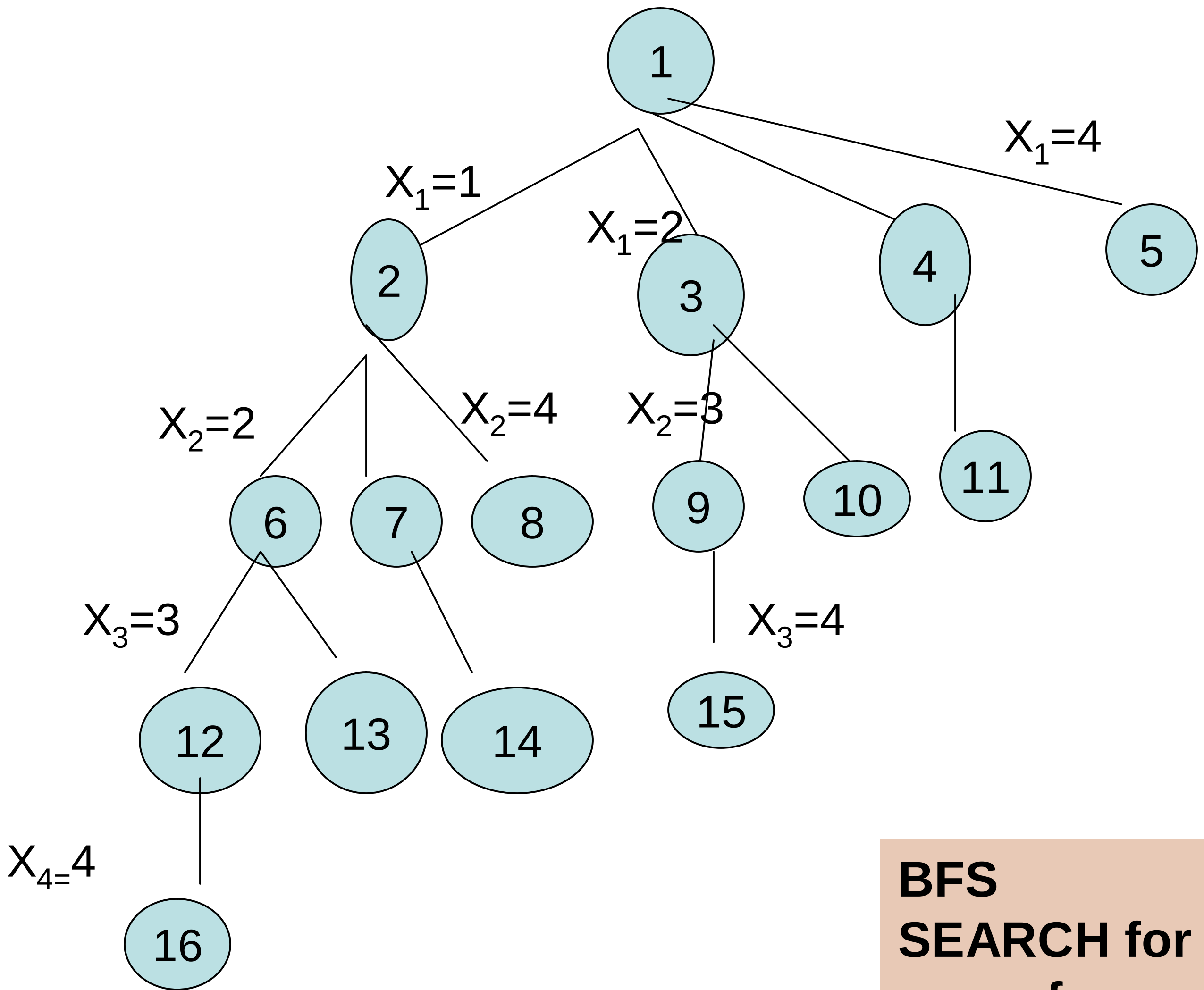
Branch and bound

Branch and Bound Method

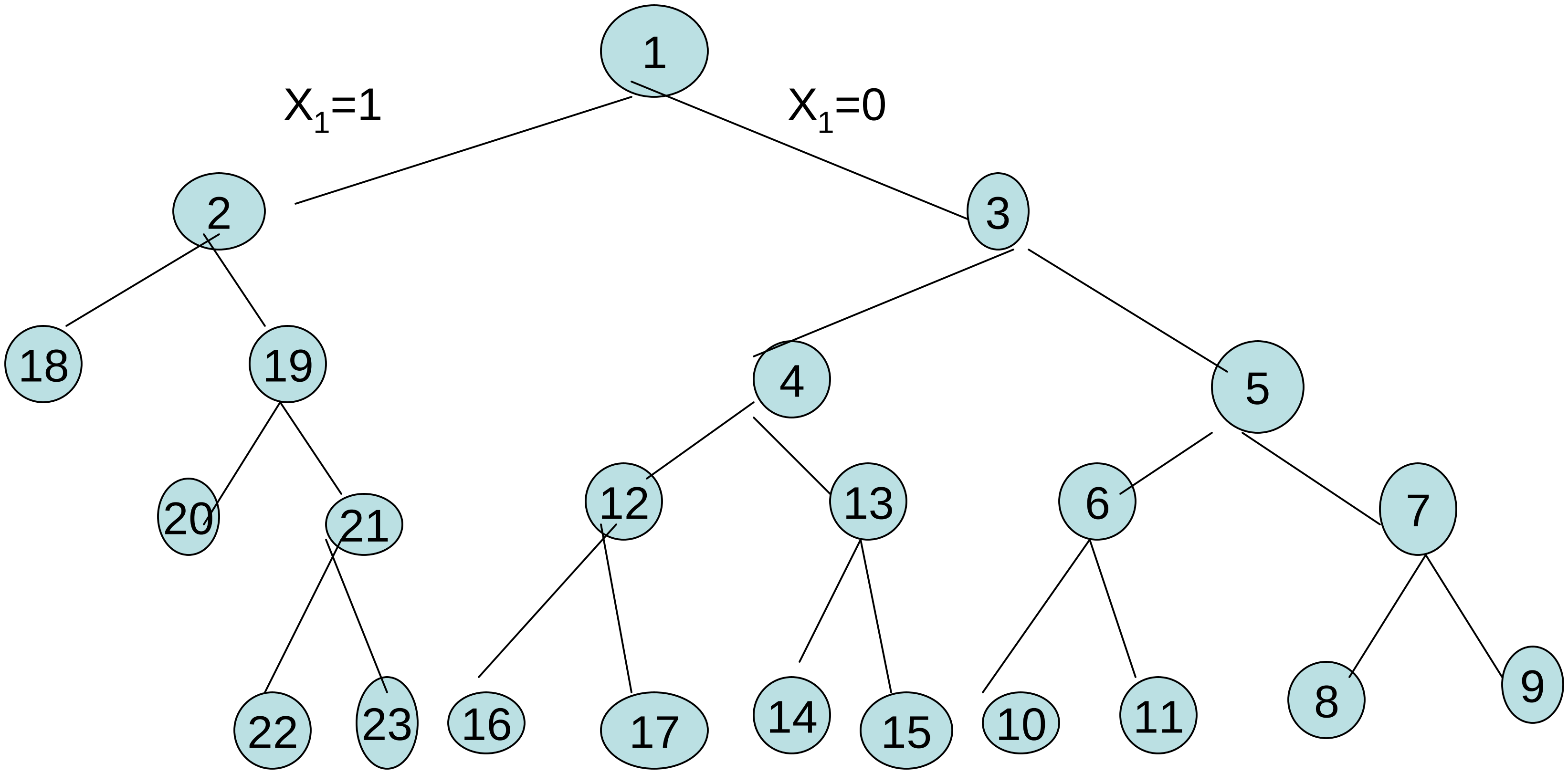
- The design technique known as branch and bound is similar to backtracking in that it searches a tree model of the solution space and is applicable to a wide variety of discrete combinatorial problems.
- Backtracking algorithms try to find one or all configurations modeled as N-tuples, which satisfy certain properties.
- Branch and bound are more oriented towards optimization.

Here all the children of the E- node are generated before any other live node can become E- node.

Here two state space trees can be formed BFS (FIFO)and D search (LIFO).



BFS SEARCH for sum of subset problem



Nodes are generated in D - search manner for sum of subset problem

Traveling salesman problem

- The salesman problem is to find a least cost tour of N cities in his sales region.
- The tour is to visit each city exactly once.
- Salesman has a cost matrix C where the element c_{ij} equals the cost (usually in terms of time, money, or distance) of direct travel between city I and city j. Assume c_{ii}=infinity for all i. Also c_{ij}= infinity if it is not possible to move directly from city I to city j.

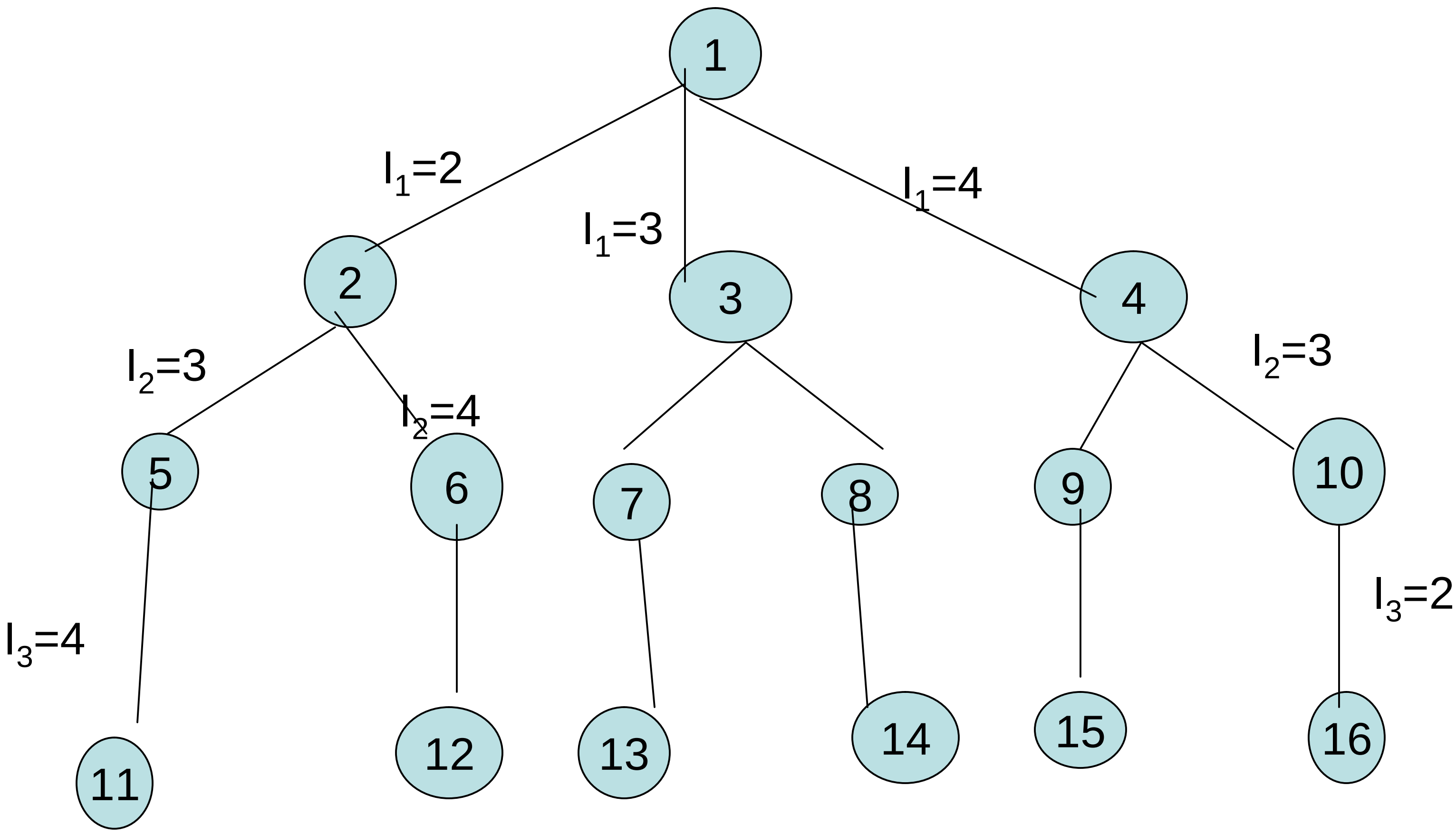
Branch and bound algorithms for traveling salesman problem can be formulated in a variety of ways.

Without loss of generality we can assume that every tour starts and ends at city one.

So the solution space S is given by

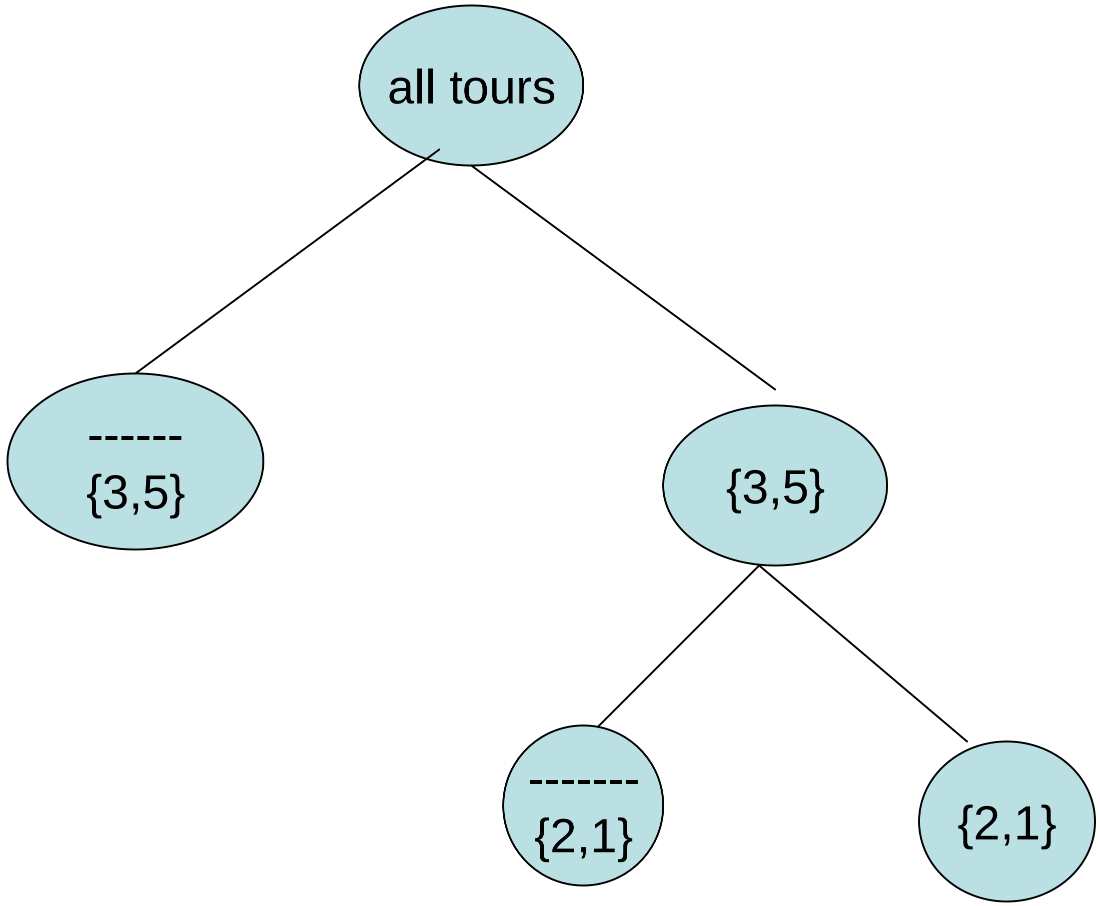
$\{ 1 , \pi , 1 \mid \pi$ is a permutation of $(2,3,4\ldots n)\}$

$|S| = (n-1)!$.



Tour 1 2 3 4 1   1 2 4 3 1

A Sate space tree for traveling salesman problem with n= 4



A branch and bound state space tree for traveling salesman problem

What is meant by bounding?

With each vertex in the tree we associate a lower bound on the cost of any tour in the set represented by the vertex.

The computation of these lower bounds is major labor saving device in any branch and bound algorithm.

There fore much thought should be given to obtain tight bounds.

Assume that we have constructed a specific complete tour with cost m.

If the lower bound associated with the set of tours represented by a vertex v is M.

And

$M >= m$

Then no need to search further for descendants of v for the optimum tour.

Basic steps for the computation of lower bounds

The basic step in the computation of lower bound is known as reduction. It is based on following observations:

1- In the cost matrix C every full tour contains exactly one element from each row and each column.

Note: converse need not be true e.g {(1,5),(5,1),(2,3),(3,4),(4,2)}.

2- If a constant h is subtracted from every entry in any row or column of C , the cost of any tour under the new matrix C' is exactly h less than the cost of the same tour under matrix C.This subtraction is called a row (column)reduction

3- By a reduction of the entire cost matrix C we mean the following: Sequentially go down the rows of C and subtract the value of each row's smallest element h_i from every element in the row. Then do the same for each column.

Let $h = \sum h_i$ summation over all rows and columns

The resulting cost matrix will be called the reduction of C.

h is a lower bound on the cost of any tour.

Let A be the reduced cost matrix for a nose R. Let S be a child of R such that edge (R,S) corresponds to including edge (i,j) in the tour.

1- change all entries in row i and column j of A to α.(so that no edge from this row (column)leaving from I(coming to j), may be included in the tour in future).

2- set A(j,1) = ∞

This prevents A(j,1) since node 1 I should be the last node of the tour.

4-Reduce all rows and columns in the resulting matrix except for rows and columns containing only ∞

.

Let the resulting matrix be B.

Let r be the total amount subtracted then lower bound on S is

lower bound for)R) + A(i,j) + r

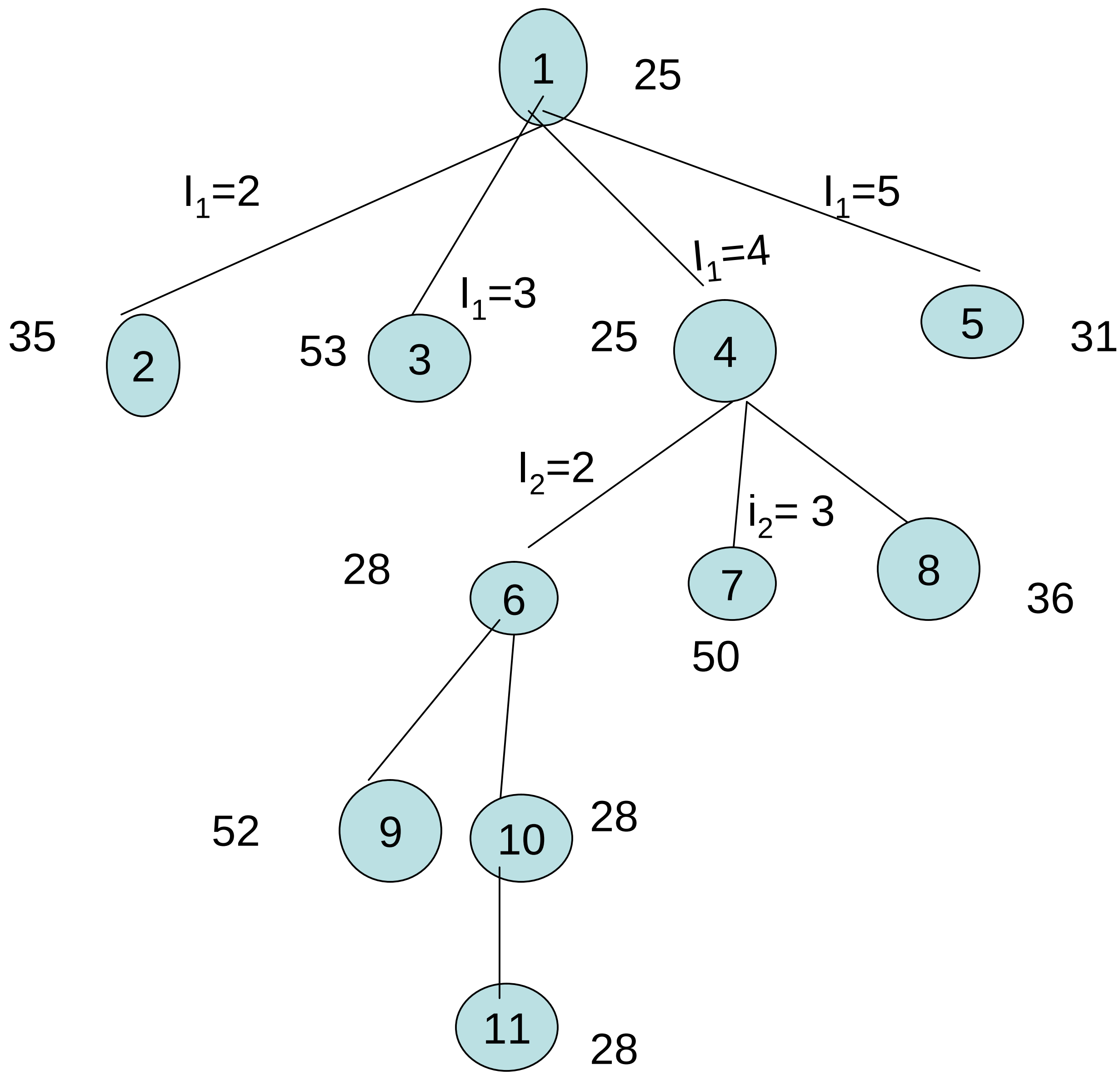
example

∞	20	30	10	11
15	∞	16	2	2
3	5	∞	2	4
19	6	18	∞	3
16	4	7	16	∞

Cost matrix

∞	10	17	0	1
12	∞	11	2	0
0	3	∞	0	2
15	3	12	∞	0
11	0	0	12	∞

Reduced cost matrix lower bound = 25


1
25
I1=2
I1=5
I1=4
I1=3
35
2
53
3
25
4
5
31
I2=2
i2= 3
28
6
7
8
36
50
52
9
10
28
11
28

∞	∞	∞	∞	∞
∞	∞	11	2	0
0	∞	∞	0	2
15	∞	12	∞	0
11	∞	0	12	∞

Path (1,2)

∞	∞	∞	∞	∞
1	∞	∞	2	0
∞	3	∞	0	2
4	3	∞	∞	0
0	0	∞	12	∞

path(1,3)

∞	∞	∞	∞	∞
12	∞	9	0	∞
0	3	∞	0	∞
12	0	9	∞	∞
∞	0	0	12	∞

path(1,5)