RV College of Engineering®

Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi

Approved by AICTE New Delhi

Backtracking
Branch-and-Bound
Decision Trees
NP and NP-Complete Problems

UNIT 5

Prof. Rajesh R M
Dept. of AIML
RV College of Engineering
Bengaluru

Go, Change the World

Backtracking

Backtracking is one of the techniques that can be used to solve the problem.

When we have multiple choices, then we make the decisions from the available choices. In the following cases, we need to use the backtracking algorithm:

- A piece of sufficient information is not available to make the best choice, so we use the backtracking strategy to try out all the possible solutions.
- Each decision leads to a new set of choices. Then again, we backtrack to make new decisions. In this case, we need to use the backtracking strategy.

It is convenient to implement this kind of processing by constructing a tree of choices being made, called the state-space tree. Its root represents an initial state before the search for a solution begins. The nodes of the first level in the tree represent the choices made for the first component of a solution, the nodes of the second level represent the choices for the second component, and so on.

Backtracking

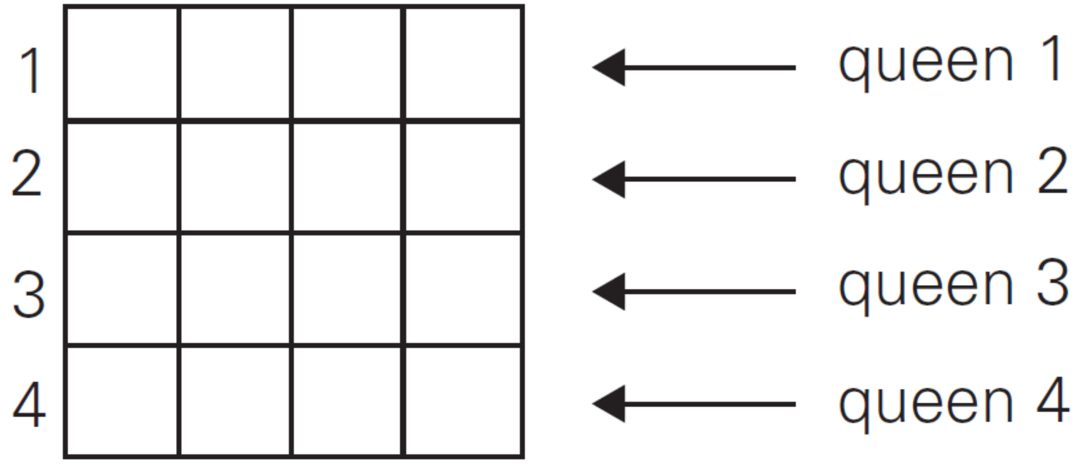
Applications of Backtracking are

- **N-queen problem**
- **Sum of subset problem**

Backtracking

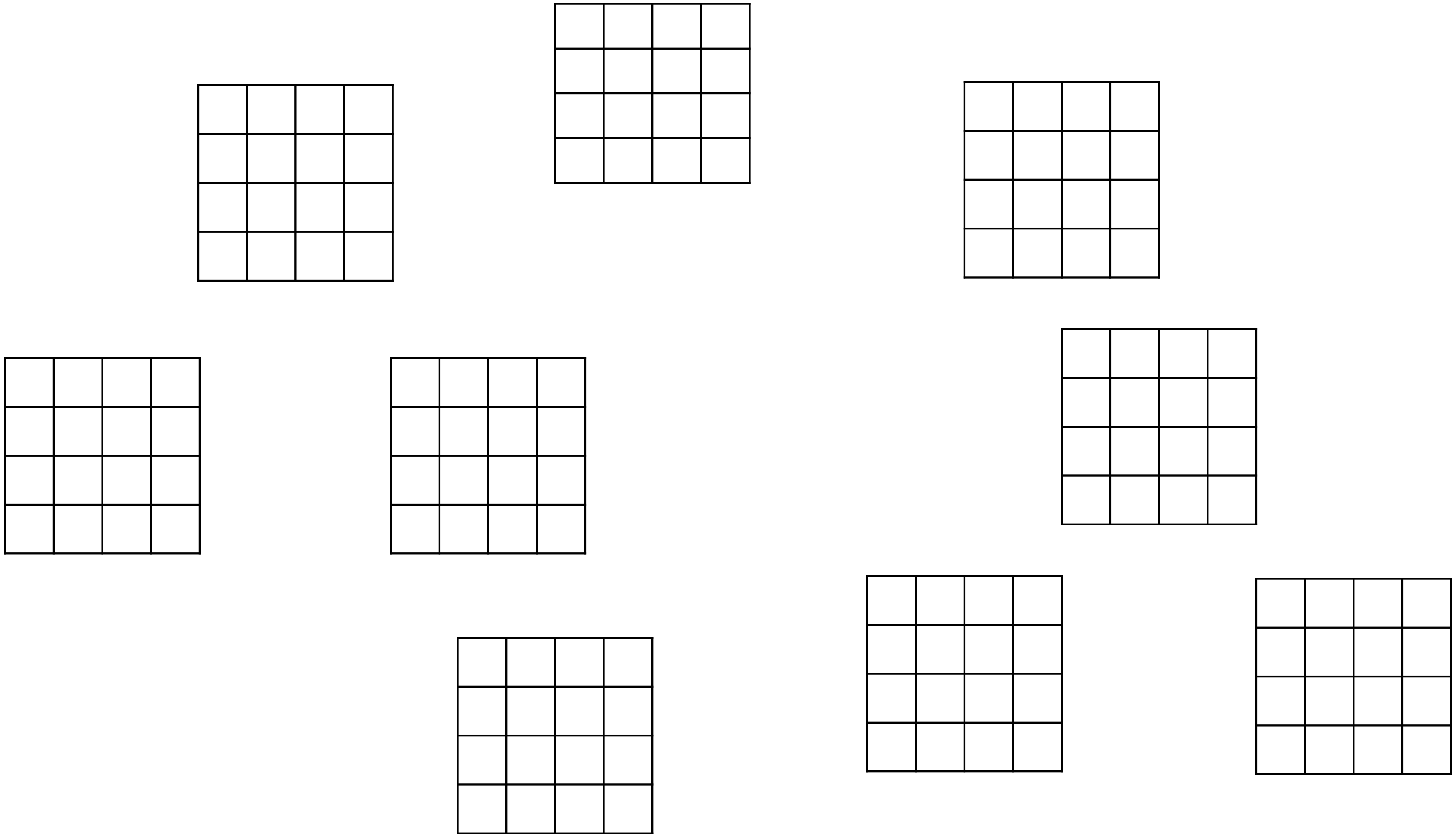
N-queen problem

The problem is to place *n* queens on an *n* × *n* chessboard so that no two queens attack each other by being in the same row or in the same column or on the same diagonal.



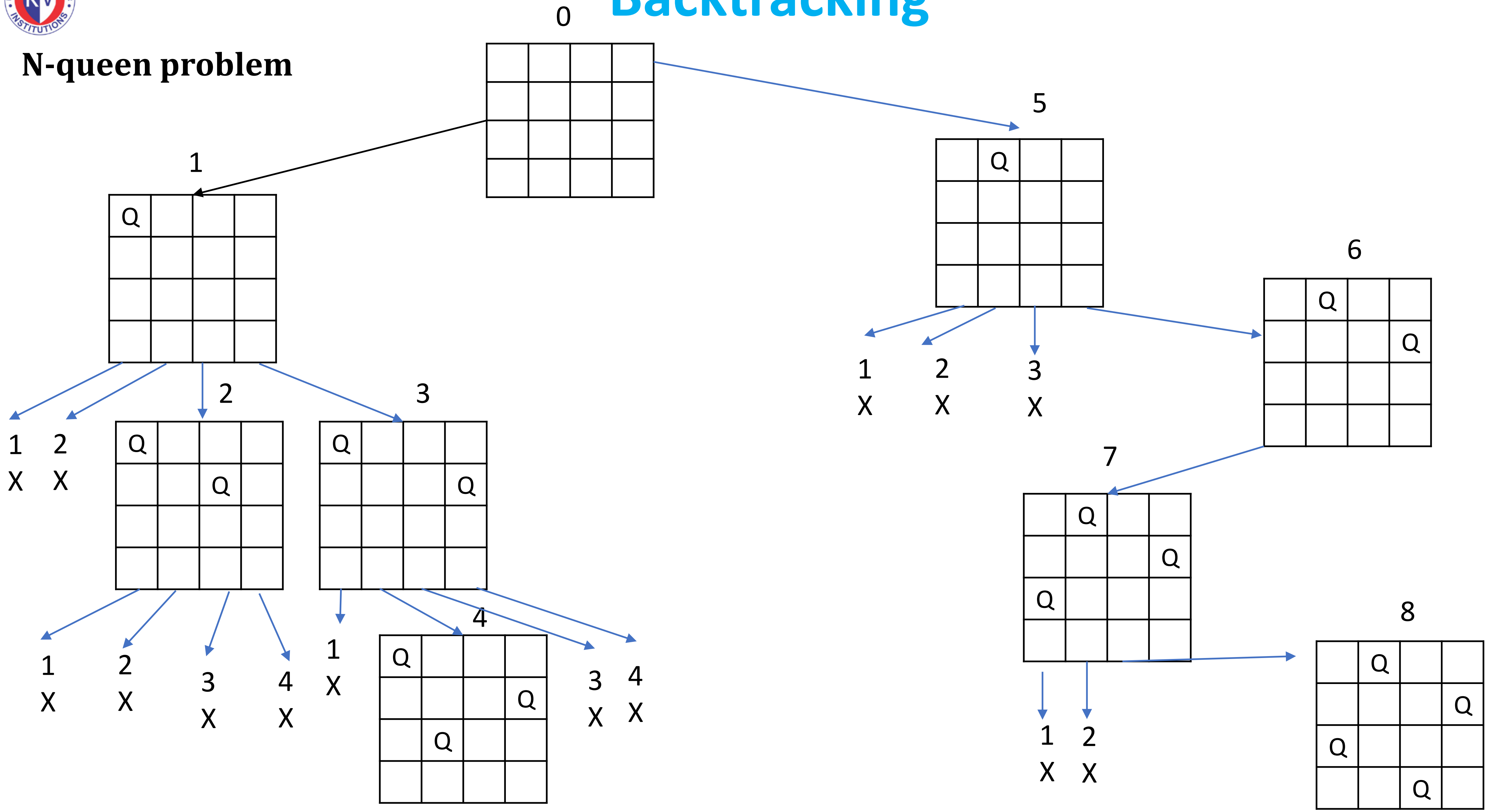
Board for the four-queens problem.

Backtracking



Backtracking

N-queen problem



Backtracking

```
N - Queens (k, n)
{
  For i ← 1 to n
      do if Place (k, i) then
  {
    x [k] ← i;
    if (k ==n) then
      write (x [1....n));
    else
    N - Queens (k + 1, n);
  }
}
```

```
Place (k, i)
  {
    For j ← 1 to k - 1
     do if (x [j] = i)
      or (Abs x [j]) - i) = (Abs (j - k))
   then return false;
    return true;
}
```

Backtracking

subset-sum problem: find a subset of a given set $A = \{a1, \ldots, an\}$ of n positive integers whose sum is equal to a given positive integer d.

For example, for $A = \{1, 2, 5, 6, 8\}$ and $d = 9$, there are two solutions: $\{1, 2, 6\}$ and $\{1, 8\}$.

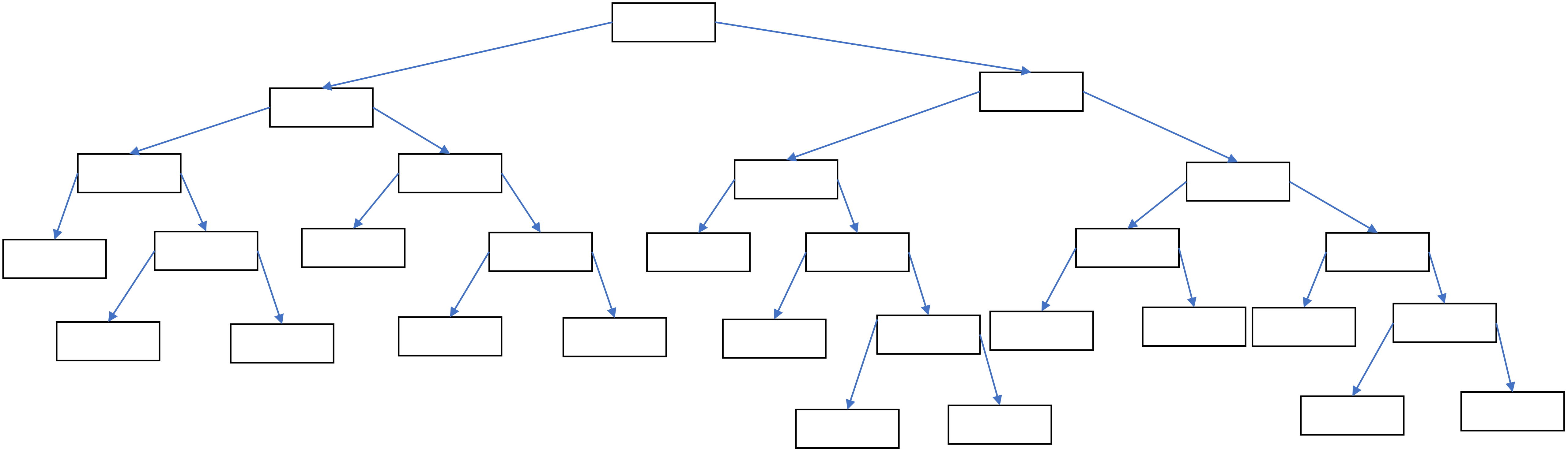
It is convenient to sort the set's elements in increasing order. So, we will assume that
$a1 < a2 < \ldots < an$.

Backtracking

Subset-sum problem: Example S = {2,4,1,5,3}

Sort the set elements {1,2,3,4,5}

Find the sum of all the element in the set = 15

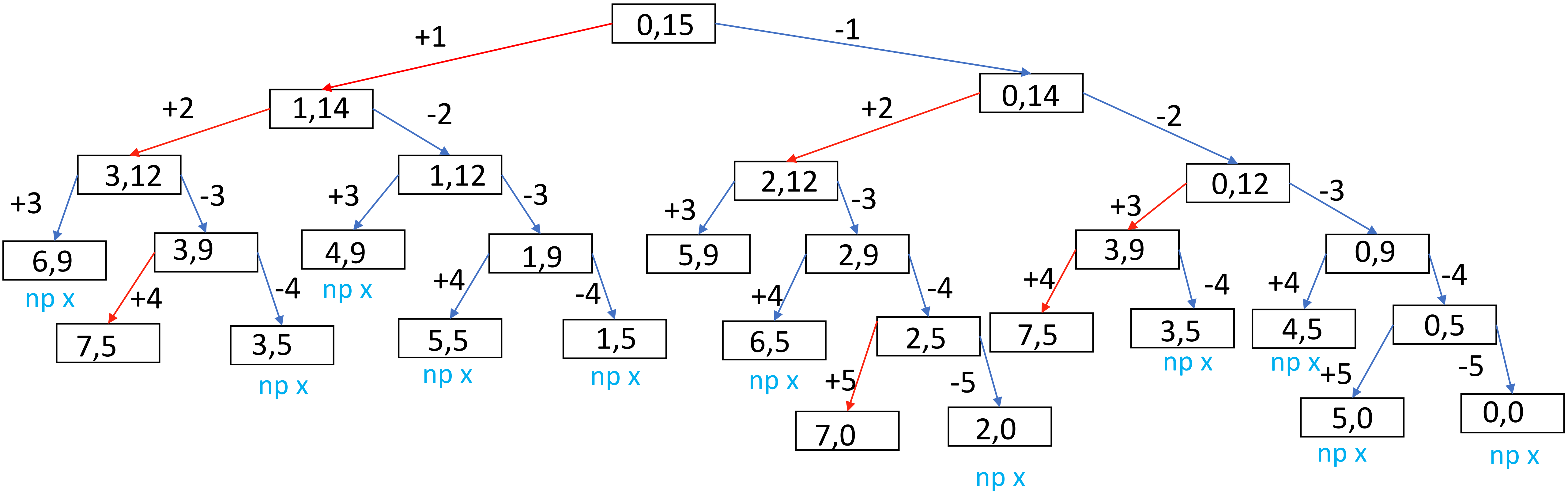


Backtracking

Subset-sum problem: Example S = {2,4,1,5,3}

Sort the set elements {1,2,3,4,5}

Find the sum of all the element in the set = 15



Backtracking

Subset-sum problem:

```
ALGORITHM  Backtrack(X[1..i])
    //Gives a template of a generic backtracking algorithm
    //Input: X[1..i] specifies first i promising components of a solution
    //Output: All the tuples representing the problem's solutions
    if X[1..i] is a solution write X[1..i]
    else    //see Problem 9 in this section's exercises
        for each element x ∈ S_{i+1} consistent with X[1..i] and the constraints do
            X[i + 1] ← x
            Backtrack(X[1..i + 1])
```

Branch and Bound

Branch and Bound

Branch and bound is one of the techniques used for problem solving.
It is similar to the backtracking since it also uses the state space tree.
It is used for solving the optimization problems and minimization problems.

In order to solve any problem using Branch and Bound we need to use Bounding function

1. Lower Bound
2. Upper Bound

Branch and Bound

Backtracking and Branch and bound

SL	Backtracking	Branch and Bound
1	Uses State space tree	Uses State space tree
2	Feasible solution (All combination)	Optimal Solution
3	Uses DFS Traversal	Uses Both DFS and BFS
4	No bounding function is used	Bounding Function is used (Lower & Upper)

Branch and Bound

Assignment Problem

Assignment Problem is the problem of assigning *n* people to *n* jobs so that the total cost of the assignment is as small as possible.

This is the example problem of lower bound

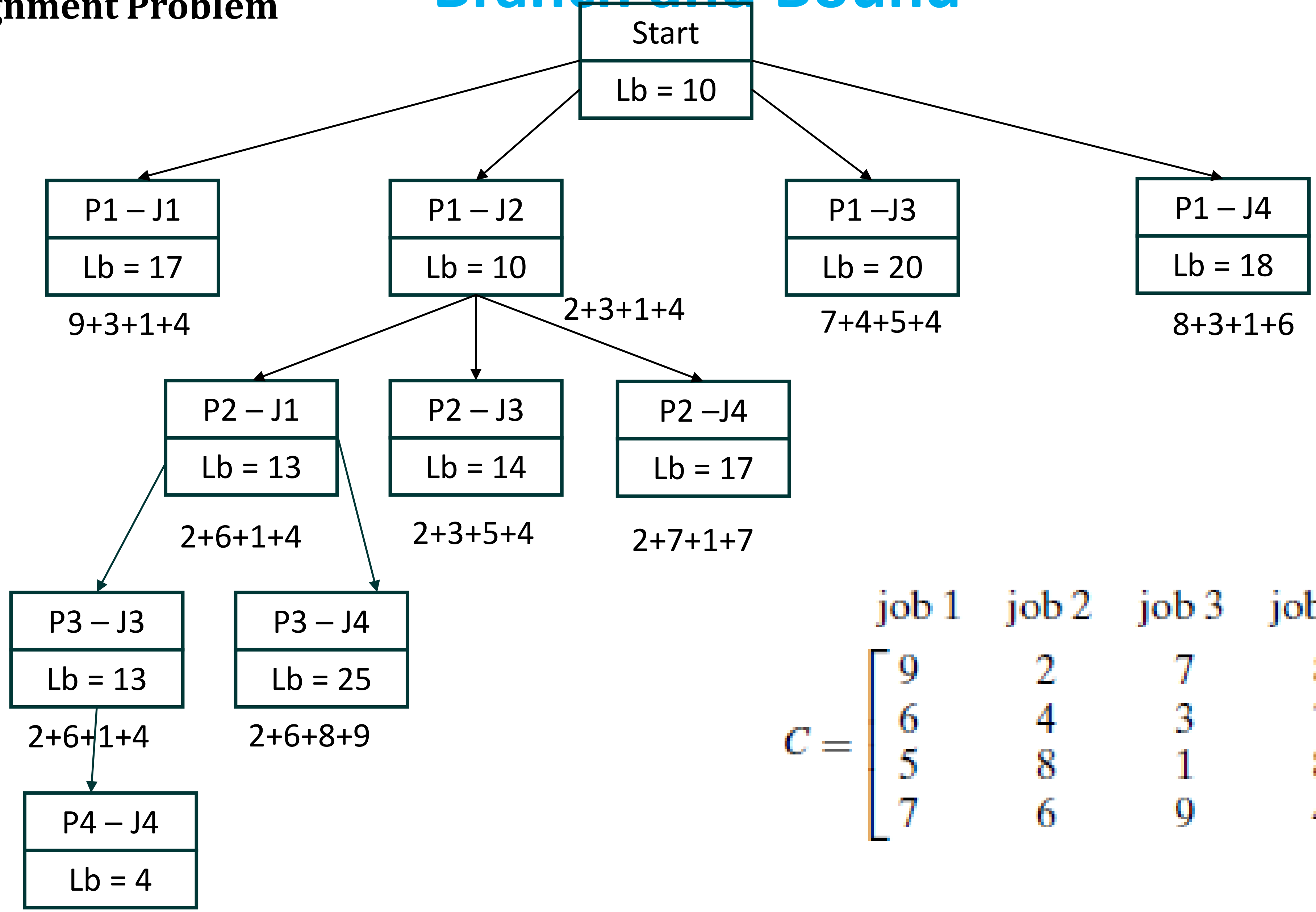
Lower bound is calculated as the summation of the smallest value in the each row

Lower Bound : `lb=2+3+1+4 = 10`

$$
C = \begin{array}{cc}
 & \begin{array}{cccc} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{array} \\
 & \left[\begin{array}{cccc} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{array}\right] \begin{array}{l} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{array}
\end{array}
$$

Assignment Problem

Branch and Bound



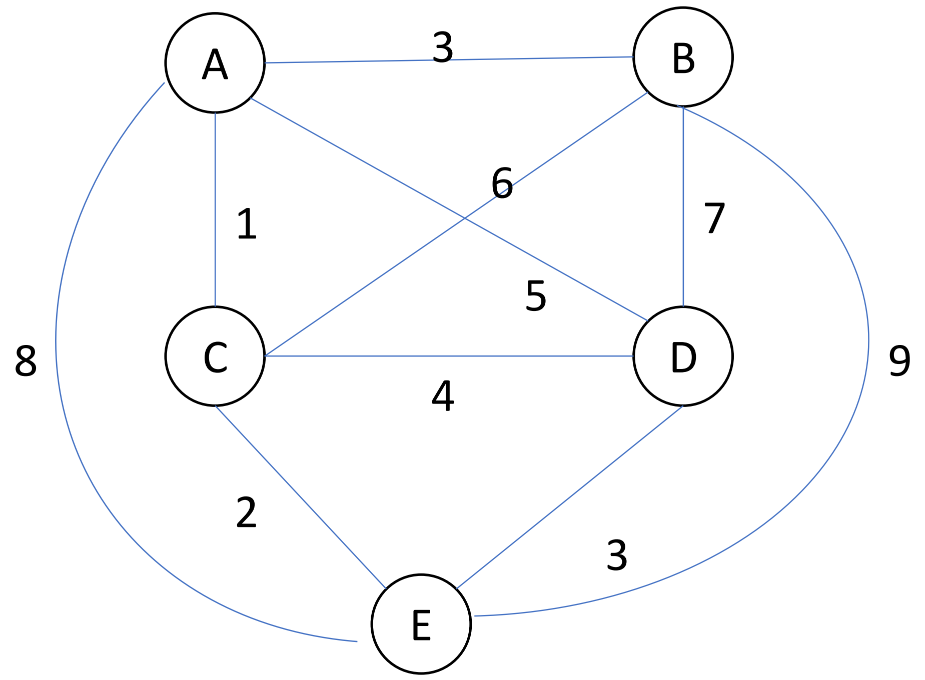
$$C = \begin{array}{c} \\ \\ \\ \\ \end{array}\begin{array}{cccc} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \\ \end{array}$$

$$C = \left[\begin{array}{cccc} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{array}\right] \begin{array}{l} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{array}$$

Branch and Bound

Travelling Salesman Problem

Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible tour that visits every city exactly once and returns to the starting point.



We need to use lower bound function.

Branch and Bound

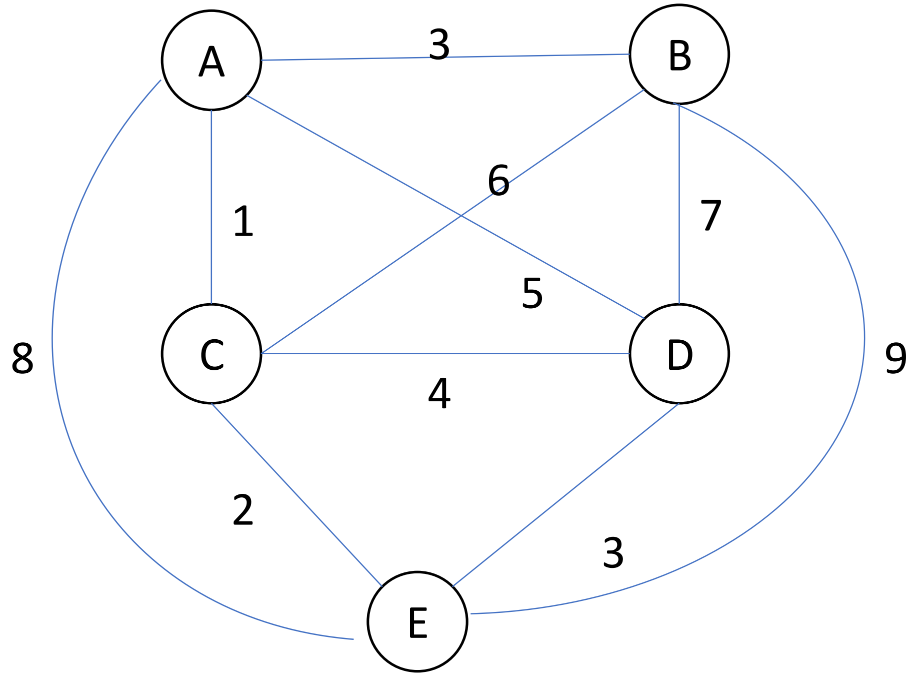
Travelling Salesman Problem

We need to use lower bound function.

Lower bound can be denoted as [s/2]

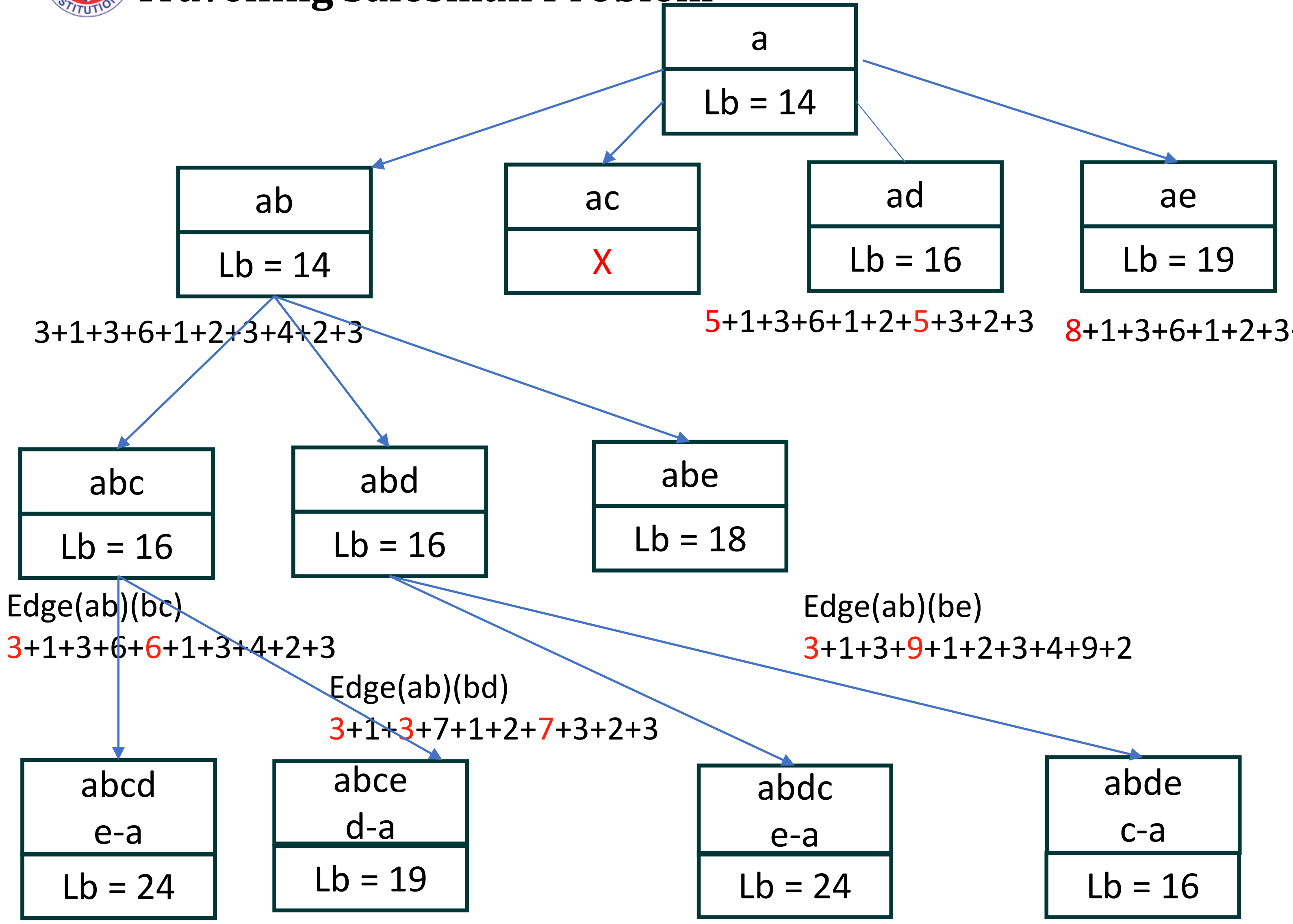
S is the summation of 2 closest cities for all cities

$$Lb = \frac{1+3+3+6+1+2+3+4+2+3}{2} = 28/2=14$$



Branch and Bound

Travelling Salesman Problem



3+1+3+6+1+2+3+4+2+3

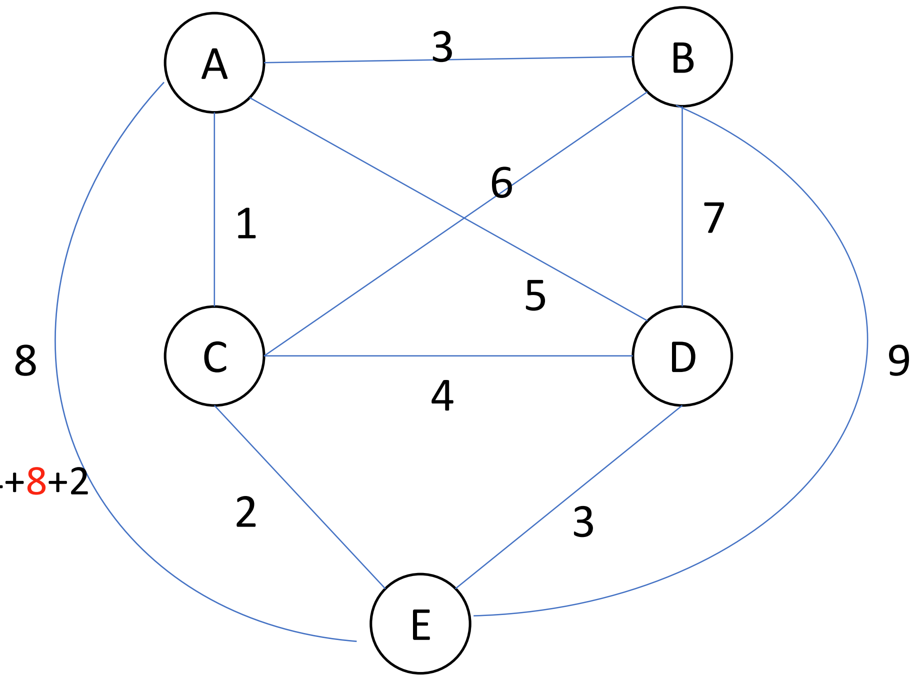
5+1+3+6+1+2+5+3+2+3

8+1+3+6+1+2+3+4+8+2

Edge(ab)(bc)
3+1+3+6+6+1+3+4+2+3

Edge(ab)(bd)
3+1+3+7+1+2+7+3+2+3

Edge(ab)(be)
3+1+3+9+1+2+3+4+9+2



Assumptions : A is the starting node
City b is visited before visiting c

Edge(ab)(bd)(de)(ec)(ca)

a	b	c	d	e
3+1 +	3+7 +	2+1 +	7+3 +	3+2