Design and Analysis of Algorithms

Lecture – 24

Backtracking

Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Learning Objective

- Learn a Brute Force Technique used to generate all possible combinations

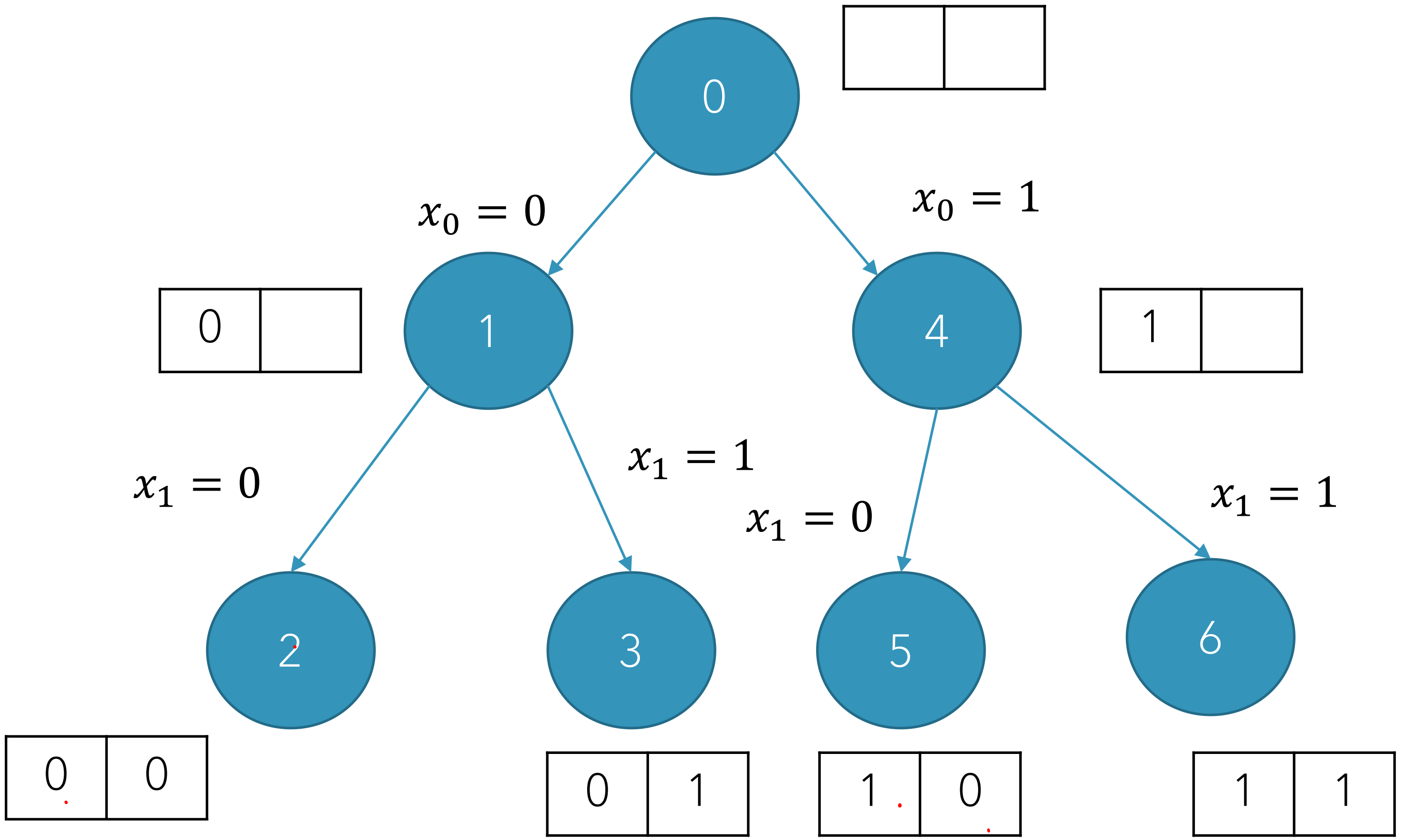
Solution Space / State Space Tree

- Solution space represent all possible combinations
- Starts with defining a solution vector x with n components and starts to assign values to the component one by one.
- To generate all possible binary strings of length n , What will be the solution vector and What is the set of values that each component in a solution vector take ?

Solution Space

- Depicted as a tree
- Root node indicate the empty solution vector
- Number of branches in a node indicate possible values a component can take
- Number of levels will be equal to number of components + 1
- Internal node represent vectors with partial solution
- Leaf node represent vectors with complete solution

Binary String with length 2



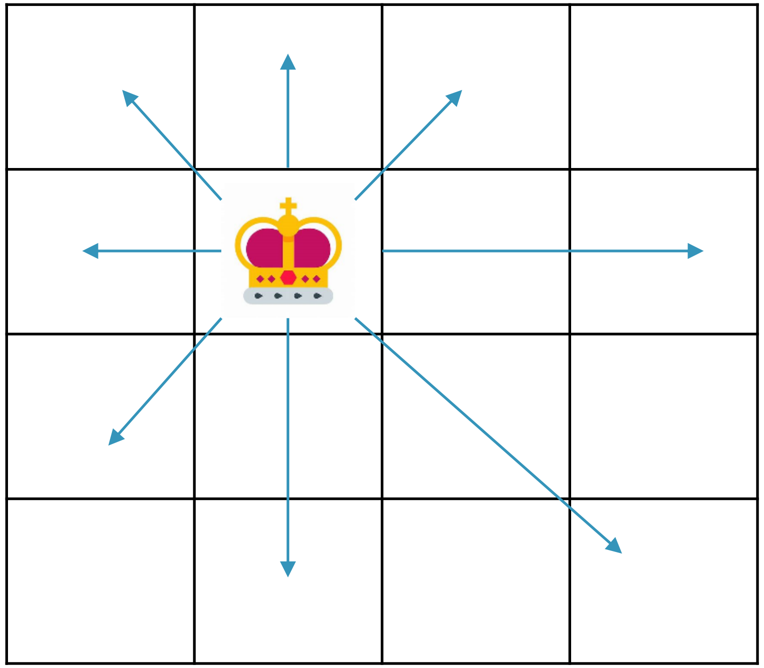
X[] is a solution vector containing components from 1 to n

Function Btrack (i, n)

```
if( i ! = n):
    for j = 0 to 1:
        X[i] = j
        Btrack (i+1, n)
```

N Queen Problem

- Place N queens on a N*N chessboard such that no two queen attack each other



- Design a solution vector - matrix equal to size of the chessboard

0	1	0	0
0	0	0	1
1	0	0	0
0	0	1	0

- Each component can be assigned with value either 0 (or) 1 depending on whether queen is placed in the cell

- How many components will be there in solution vector

  rows * columns in the matrix

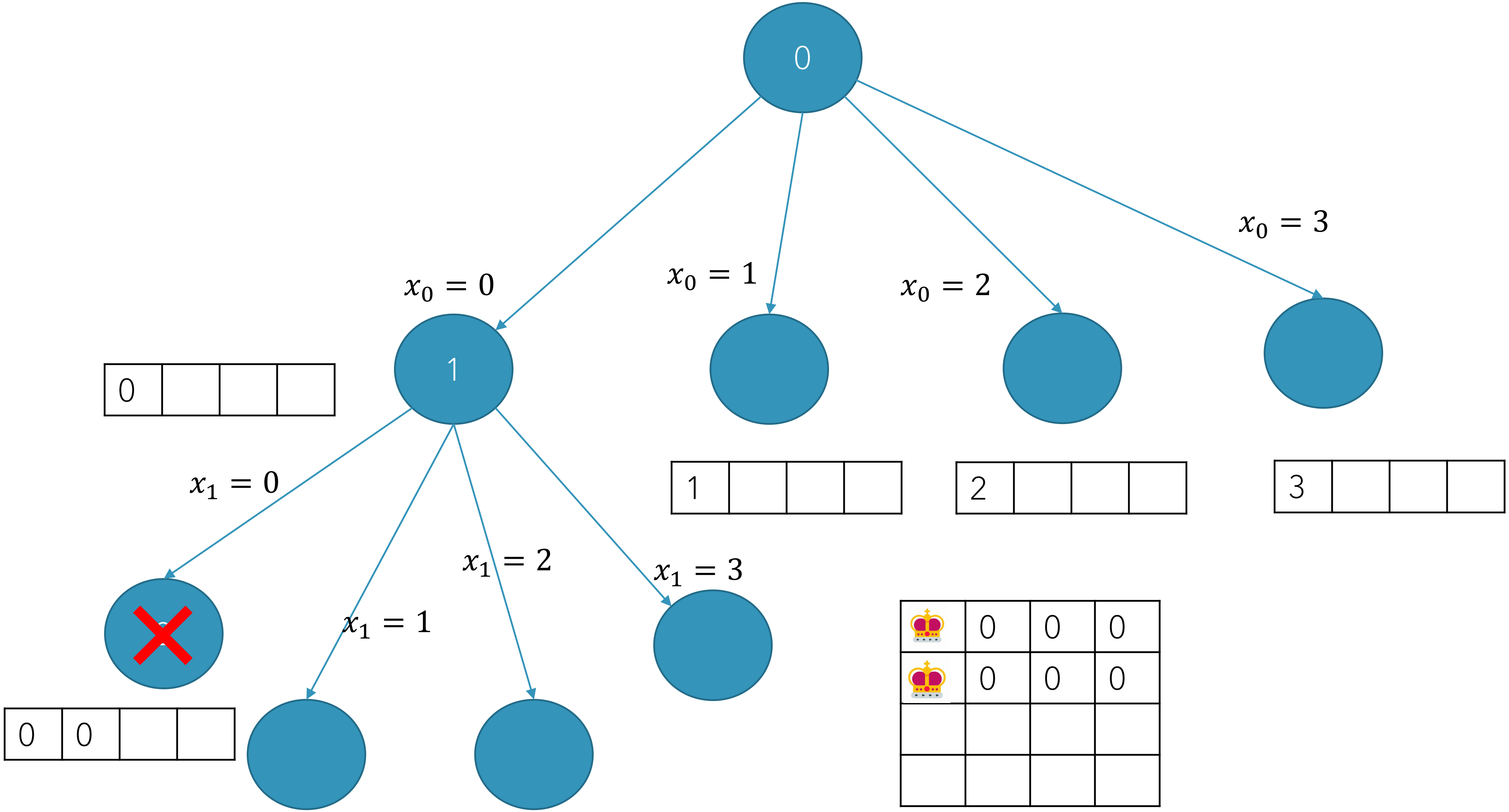
- Optimum Solution Vector

	0	1	2	3
0	0	1	0	0
1	0	0	0	1
2	1	0	0	0
3	0	0	1	0

- Two queens cannot be placed in a single row
- Let us assume that i^{th} queen will always be placed in the i^{th} row

Index	0	1	2	3
Solution Vector	1	3	0	2

- Solution vector is modified to encode the column in which the queen can be placed


0
$x_0 = 0$
$x_0 = 1$
$x_0 = 2$
$x_0 = 3$
1
0
1
2
3
$x_1 = 0$
$x_1 = 1$
$x_1 = 2$
$x_1 = 3$
0 0
0 0 0
0 0 0

X[] is a solution vector containing components from 1 to n

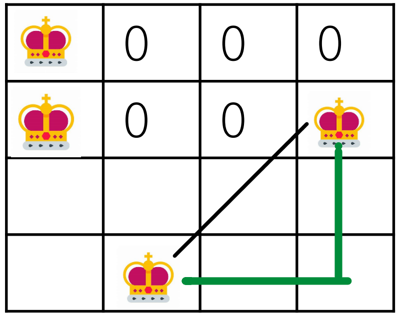
Function Btrack (i, n)

```
if( i ! = n):
    for j = 0 to n:
        if safe(i,j) {
            X[i] = j
            Btrack (i+1, n)
        }
```

Function safe (i, j)

```
for k= 0 to i-1:
    if(x[k]==x[j] || (k-j) == (x[k]-x[j]))
        return false

return true
```



Sum of subset problem

- Given n distinct positive numbers generate all combination of these numbers whose sum is m
- Consider 4 integers namely {3,5,6,7} and sum m = 15
- Solution {3,5,7}

- Design Solution vector – variable sized
- Convert it to fixed size tuple

0	1	2	3
1	1	0	1

Pause & Think

- What are the values that the component can take?

    0 (or) 1

- How many levels will a state space tree contain?

    Equal to value of n + 1

Sum of subset problem

- Bounding Condition

Is there a way to determine non-promising nodes?

Suppose for {3,5,6,7} if there exist a node with partial solution where 3 and 5 are selected

1	1	-	-

Sum of node = 3+5 = 8 It can lead us to correct solution only if

sum of node + sum of remaining elements >=m

Pause & Think

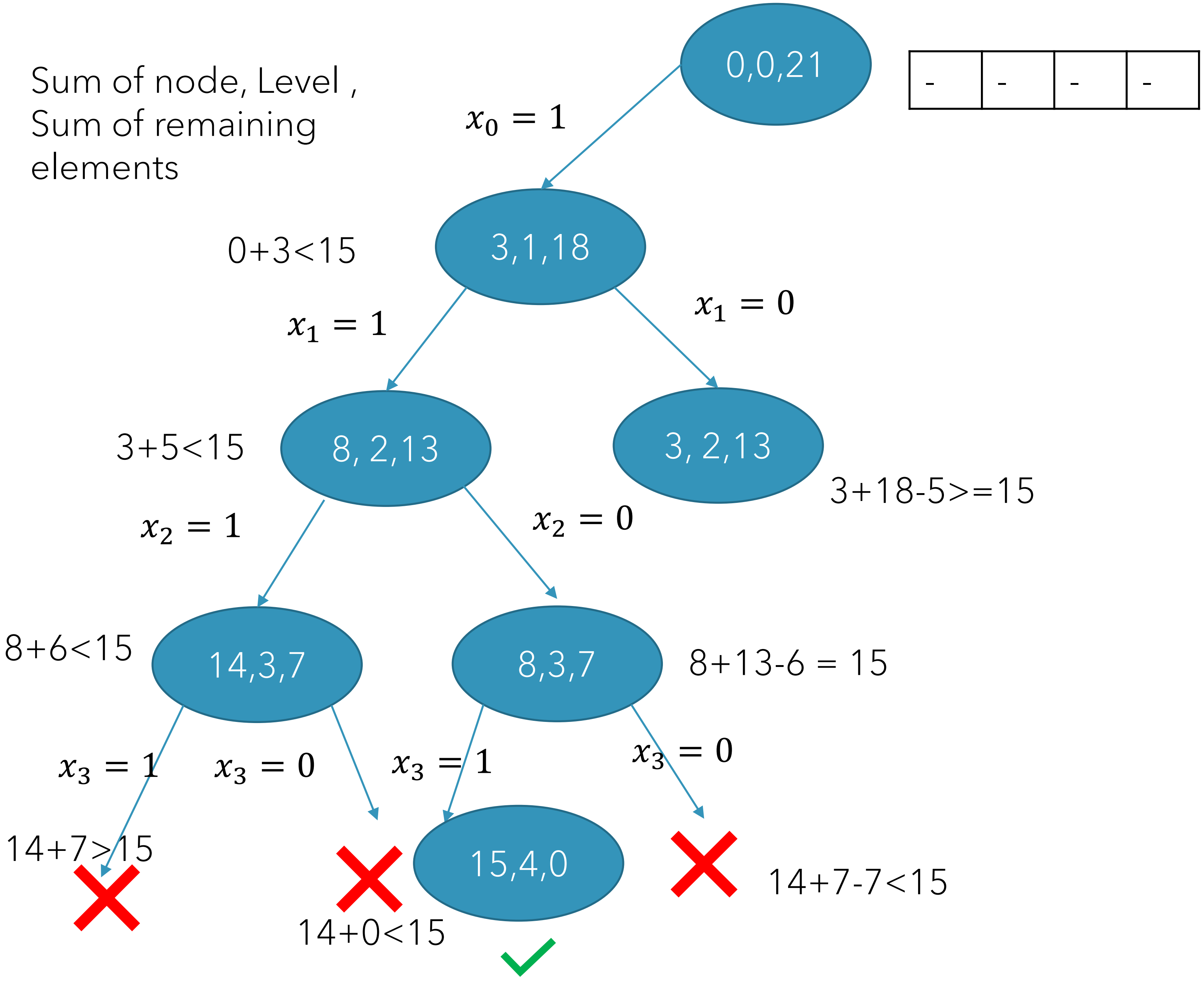
Consider a node

Inclusion of A3 sum of node + A3 <=m

Exclusion of A3 sum of node + sum of remaining elements > = m

1	1	-	-

For easier computation, sum of elements considered and sum of remaining elements are passed as parameters to the node.


Sum of node, Level , Sum of remaining elements
0,0,21
- | - | - | -
$x_0 = 1$
3,1,18
0+3<15
$x_1 = 1$
$x_1 = 0$
8, 2,13
3, 2,13
3+5<15
3+18-5>=15
$x_2 = 1$
$x_2 = 0$
14,3,7
8,3,7
8+6<15
8+13-6 = 15
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
14+7>15
15,4,0
14+0<15
14+7-7<15

X[] is a solution vector containing components from 1 to n

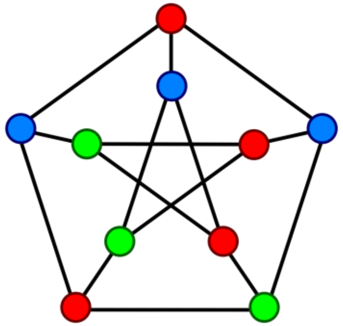
Function Btrack (i, n, s, r, elements, m)

n number of elements
m required sum
s sum of elements included
r sum of remaining elements
Elements -------- One dimensional array of elements

```
if (i!=n):
    x[i] = 1
    if(s+elements[i]<=m)
        Btrack(i+1,n,s+elements[i],r-elements[i], elements, m)
    x[i]=0
    if(s+r-elements[i]>=m)
        Btrack(i+1,n,s,r-elements[i], elements,m)
```

Graph Coloring Problem

- Given a graph G , try to assign a color to the vertices such that adjacent vertices do not take a similar color.
- Problem: Given m colors , find whether an assignment possible for the graph subject to the condition that adjacent vertices take distinct colors



n Vertices
m Available Colors

Function Btrack (i, n, m)

```
if( i ! = n):
    for j = 1 to m:
        if safe(i,j,m) {
            X[i] = j
            Btrack (i+1, n)
            }
```

Function safe (i, j)

```
for k= 0 to n:
    if(G[i][k]==1 && (x[k]==j) )
        return false

return true
```

Pause & Think

- How many levels in the tree

  n+1 levels [n denote the number of components in a tree]

- How many nodes will be there in the state space tree of the proposed solution (graph coloring problem)

  m-ary tree (m –number of colors)

  $= \frac{m^{n+1}-1}{m-1}$ = O(m^{n+1})

Summary

- Discussed the general outline of any backtracking algorithm.

Thank You
Happy Learning

Success is always inevitable with Hard Work and Perseverance