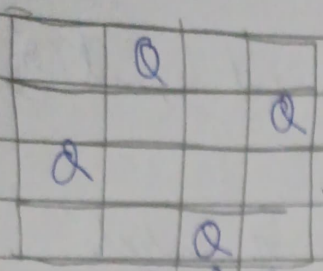
53 N-Queen's problem

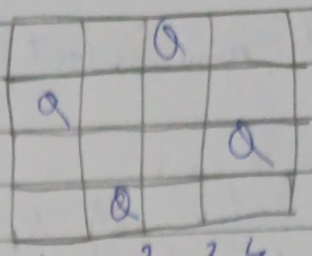
→ Objective is to place n-queens on a chess-board of dimension nxn such that no two queens attack each other

→ Queens attack each other if they are in the same row, same column or same diagonal to each other

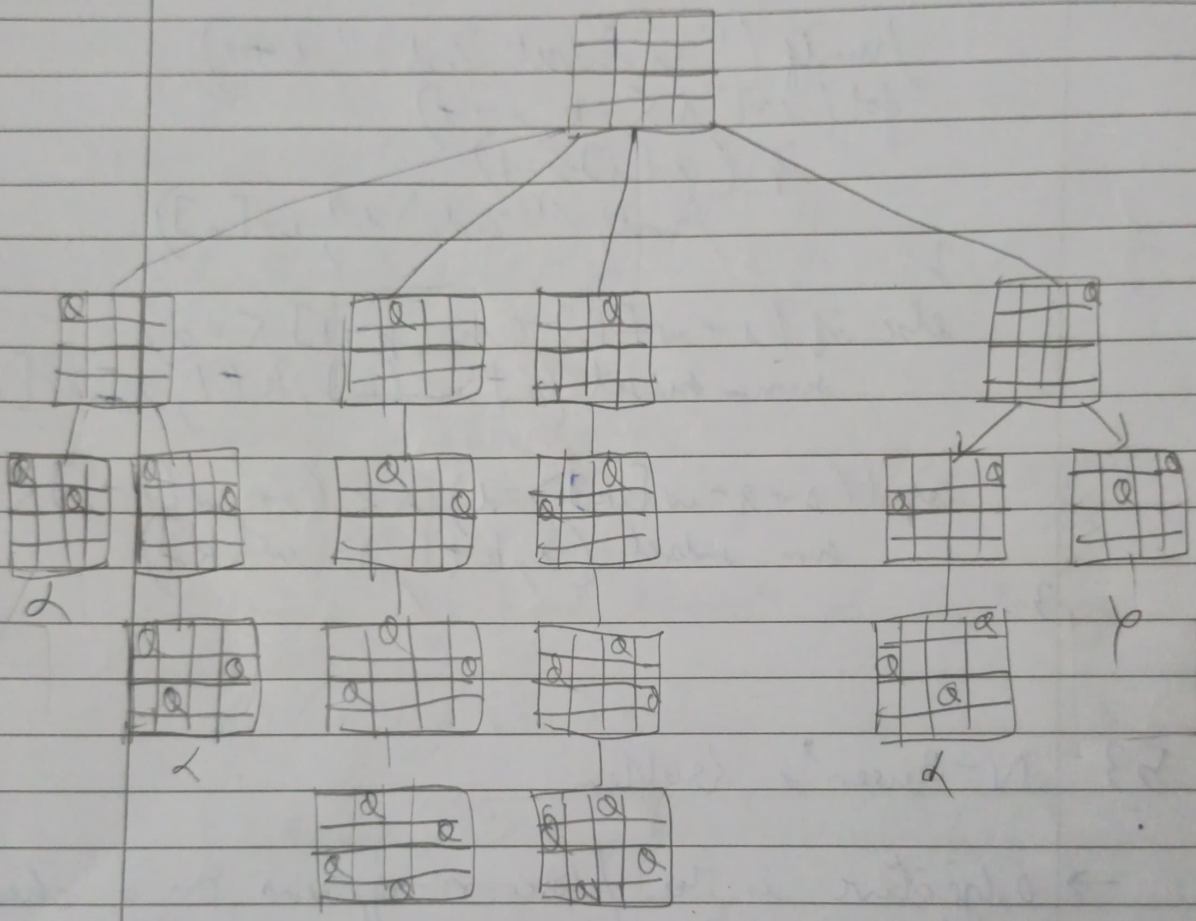
4x4



	1	2	3	4
X	2	4	1	3

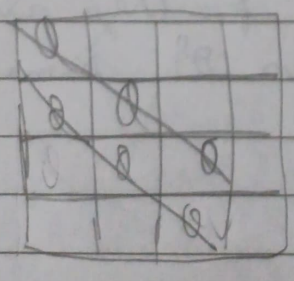


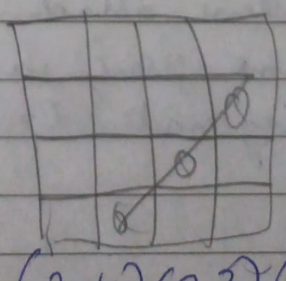
	1	2	3	4
X	3	1	4	2



	1	2	3	4
X	2	4	1	3

	1	2	3	4
X	3	1	4	2





(1,1) (2,2) (3,3) (4,4)
(2,1) (3,2) (4,3)

(2,4) (3,3) (4,2)

Let us consider any 2 points

$(3,2) \rightarrow (i,j)$

$(4,3) \rightarrow (k,l)$

$i - j = k - l$

$3 - 2 = 4 - 3$

$1 = 1$

$j - l = i - k$

Let us consider 2 points

$(3,3) = (i,j)$

$(4,2) = (k,l)$

$i + j = k + l$

$3 + 3 = 4 + 2$

$6 = 6$

$j - l = k - i$

$abs(j-l) = abs(k-i)$

Algorithm NQueens (k, n) → Initial value → no. of queens

// using backtracking, the algorithm finds all possible solutions

```
for i ← 1 to n do
    if (place(k, i))
        x[k] ← i
        if k == n
            print x[1 ... n]
        else
            NQueens(k+1, n)
```

Algorithm place (k, i)

// Returns true if Q is placed at k^{th} row & i^{th} column, otherwise returns False

__/__/____

```
for j <- 1 to (k-1) do
    if (x[j] == i) or (abs(x[j] - i) == abs(j - k))
        return False
return True
```

Backtracking

→ We find all feasible solutions

→ Once a decision is taken, we can revoke the decision

→ To solve any problem using backtracking we construct a tree, this tree is termed as "State-Space Tree"

→ Tree is constructed using DFS traversal

$S = \{1, 2, 3, 4, 5\}$

$d = 7$

↑

Max value of subset

Output → all subsets $S_1, S_2 \ldots$ (Sum of element in the subset should be equal to d)

$S_1 = \{1, 2, 4\}$

$S_2 = \{2, 5\}$

$S_3 = \{3, 4\}$

Sum of subset problem

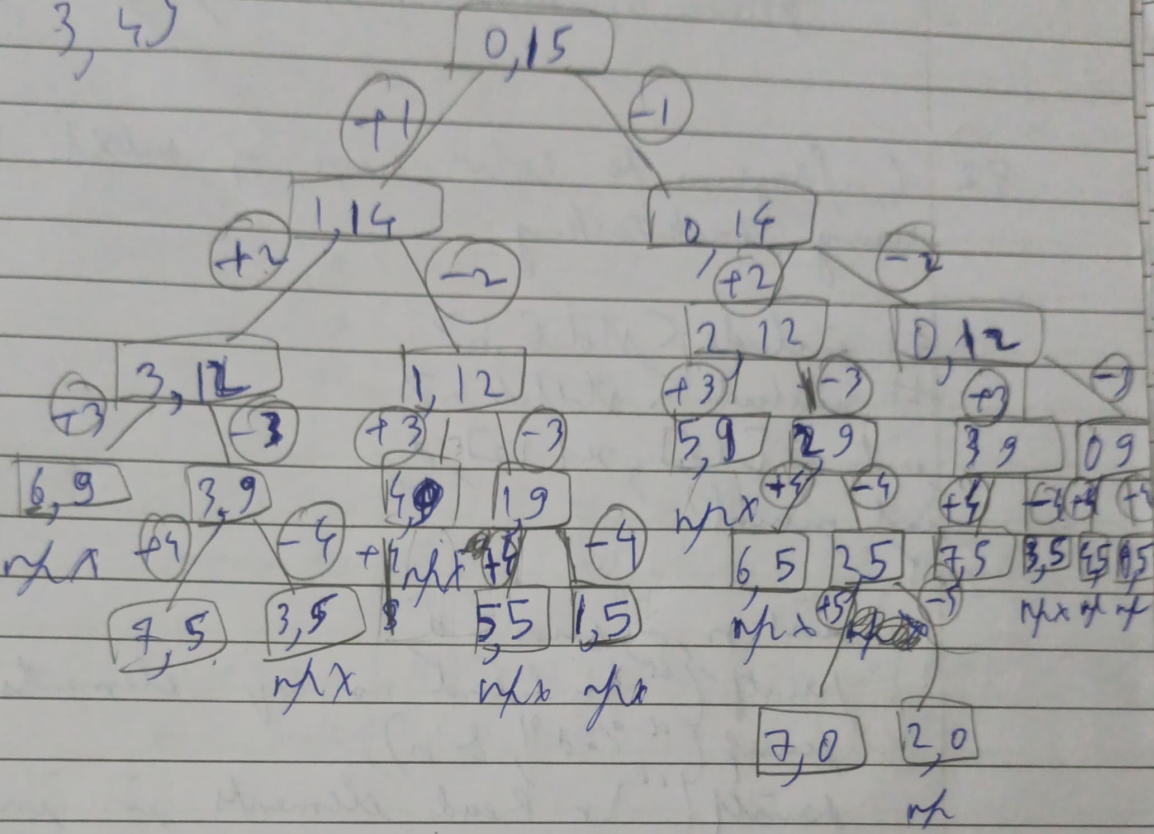
$S = \{1, 2, 3, 4, 5\}$ → Elements in sets should always be in increasing order

$d = 7$

{1, 2, 4},

{2, 5}

{3, 4}



Time complexity $= O(2^n)$

Algorithm Sum_of_subsets (S, k, d)
// Input: w[1..n] which holds the element in increasing order & d
// output: Subsets whose summation is d

x[k] = 1
if (w[k] + s == d)
    write(x[1..n])

else if (s + w[k] + w[k+1] <= d)
    Sum_of_subsets (s + w[k], k+1, d - w[k])

if ((s + w[k] >= d) and (s + w[k+1] <= d))

x[k] = 0

sum_of_subsets (s, k+1, r - w[k])

52 /* Program to solve sum of subset Problem using backtracking */

```
#include <stdio.h>
#include <stdlib.h>
int w[10], x[12], d;
int main()
{
    int n, i, sum = 0;
    printf("\n Read no. of elements :");
    scanf("%d", &n);
    printf("\n Read elements in increasing order \n");
    for (i = 1; i <= n; i++)
    {
        scanf("%d", &w[i]);
        sum = sum + w[i];
    }
    printf("\n Read the subset max value :");
    scanf("%d", &d);
    if (sum < d || w[1] > d)
    {
        printf("\n No solutions");
        exit(0);
    }
    sum_subset(0, 1, sum);
    return 0;
}
```

```
void sum_subset(int s, int k, int r)
{
    int i;
    static int t=1; // No. of subsets
    x[k]=1
    if(s+w[k]==d)
    {
        printf("\n Subset %d :", t++);
        for(i=1; i<=k; i++)
            if(x[i]==1)
                printf("%d \t", w[i]);
    }
    else if(s+w[k]+w[k+1]<=d)
        sum_subset(s+w[k], k+1, r-w[k]);

    if((s+r-w[k]>=d) && (s+w[k+1]<=d))
        sum_subset(s, k+1, r-w[k]);
}
```