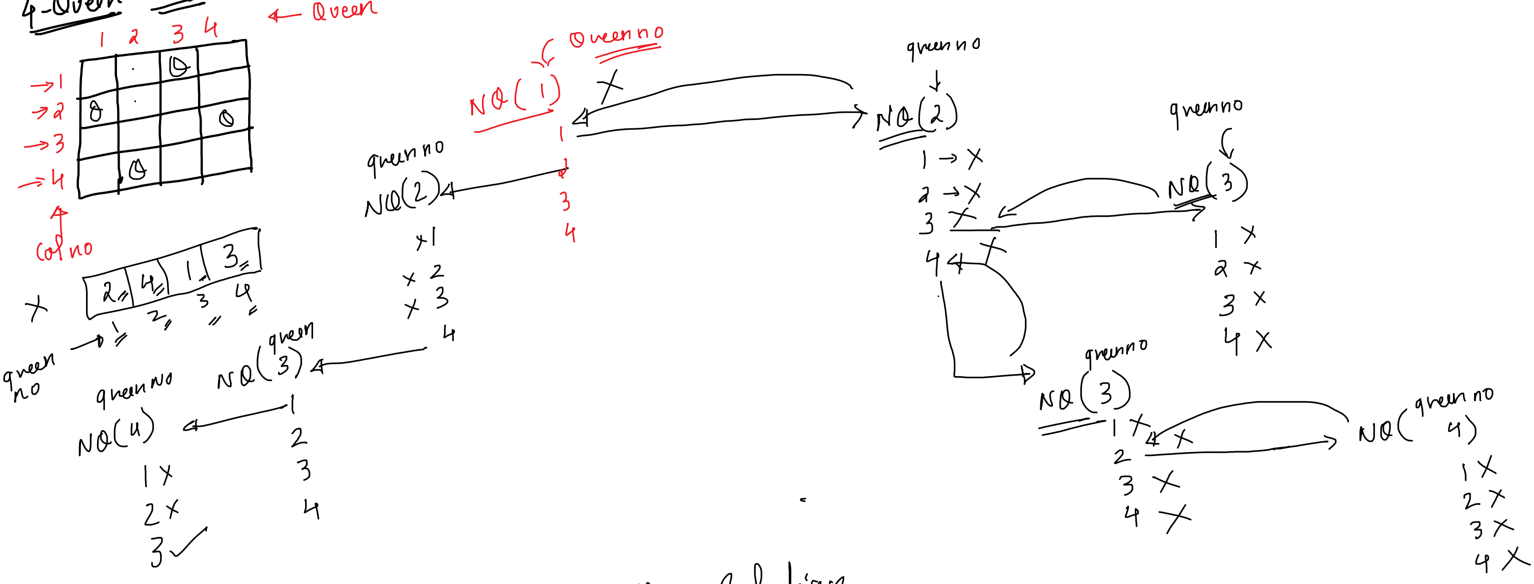
NQueen Problem →

N×N chessboard

N Queen

[None of the queens are attacking position]

4-Queen. N=4



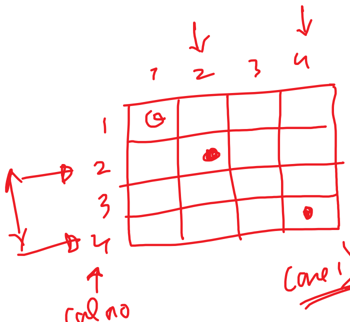
Here the solution array X contains the solution

Here $\boxed{X[i] \text{ gives column position of queen } i}$

X = | 2 | 4 | 1 | 3 | ⇐ Solution array X
1 2 3 4

① Every value in array X must be unique
[Every queen must be placed at diff colⁿ]

② Every value must be between 1 to 4 (inclusive)



X | 1 | | | |
1 2 3 4

Case 1) place(2, 1)

chk for previous ... ⇒ j=1   x[j] i.e. x[1] = 1 ⇒ poorane queen ka col

and i = 1 ⇐ current queen ...

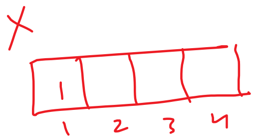
chk for previous queen ⇒ $j=1$ $x[j]$ i.e $x[1] = 1$ .. ka col

and $i = 1$ ⇐ current queen ka proposed col same hai

Nahi chalega.

Case 2) chk diagonal

place $(\overset{k}{2}, \overset{i}{2})$



Previous queen $= j=1$

Here $x[j] = 1$ $i=2$ not same

$|x[j] - i| \overset{?}{==} |k-j|$ diag attack hai

$|1-2|$ $|2-1|$

1 same hai 1

Current & poorane queen no ka diff = poorane and current queen ke col ka diff (diag)

AOA17-4

BACKTRACKING

Date
Page

★ Backtracking :

- It is a method of solving a problem (Eg. N queen)
- The principle idea is to construct solutions by taking one component at a time. The component is added in the solution if all the constraints in the problem definition are satisfied.
- If constraints are not satisfied then the latest component is dropped from the solution.

  i.e. while finding the solution to the problem, we find some partial solution. step 1, step 2, ... k
- At step k+1, the program discovers that it cannot go further with the solution due to some mistake in the previous step in such situation the program can be made to backtrack and repair the previous step solution.

★ N-QUEEN PROBLEM :

- n×n chess board and given n queen.

  We want to place all the n queen on the board in non-attacking position.

Note: Queen can attack horizontally, vertically & diagonally.

- Solution can be found using $\boxed{\text{1-D array of 'n' elements}}$
- k^{th} element of array x will give column of k^{th} queen.
- The constraints on array x are every value stored
  1. in ary. x stored should be unique.
  2. Every value stored in array x should be between 1 to n.

$x[k]$ = col position of k^{th} queen

Scanned by CamScanner

Algorithm :

1. Let n be number of queens.
2. Let x[n+1] be array that stores column of queen.
   (because array index starts at 0)

X				
0	1	2	3	4

zero index not used

Algo for function read()

i. Accept no. of queens. i.e. n
ii. Create array x of n+1 elements.

Algo for function place (k, i)

It indicates whether kth queen can be placed in ith column.

```
int place(int k, int i)      ← col  col no
                current queen no
{
    int j;                   // j refers to previous queens.
    for(j=1 ; j<=k-1 ; j++)
    {
        if((x[j]==i) || (|x[j]-i|==(k-j)))
            return 0;
    }
    return 1;
}
```

chk pos^n of all previous k-1 queens.

returns 1 ⇒ queen k can be placed at col i
returns 0 ⇒ queen k cannot be placed at col i

Algo for function nqueen(int k)

```
void nqueen(int k)       ← queen no to set
{
    for(i=1 ; i<=n ; i++)    ← for every queen k we have possible col from 1 to n
    {
        if (place(k,i))      ← kya queen k ko col i pe rakh sakte hai
        {
            ⇒ x[k]=i ;
            if (k==n)        // To print solution.
            {
                for(j=1 ; j<=n ; j++)
                    print x[j];      } toh print sol^n
            }
```

agar yes queen k ko col i pe rakho.

agar ye last queen hai

agar ye last queen hai

```
{   for(j=1; j<=n; j++)   } toh print soln
        print x[j];
}
```

Scanned by CamScanner

```
        else
          nqueen (k+1);   → agar queen no n nahi hai (last queen nahi hai)
      } // end of if      ← call for next queen.
    } // for ends
  } // end of nqueen.
```

NOTE: In place function.

- j refers to previous queens.
- x[j] : column of previous queen j.
- i : purposed column of current queen k.
- |x[j] − i| : difference in column of previous & current queen.
- (k − j) : difference in queen number.

if((x[j] == i) || (|x[j]−i| == (k−j)))

x[j]	i	|x[j]−i|	(k−j)
previous jth queen ka column number.	current kth queen ka column number. (proposed)	Previous jth queen ka and column current kth queen ka col. ka difference.	kth queen ka and previous jth queen ka difference.

Agar column no. same hai to Nahinn chalega !!

Agar same hai to diagonal problem Hai, Nahinn ! chalega

Scanned by CamScanner

PROGRAM: <math.h>

```c
# include <stdio.h>
int n, x[16]={0};
int place (int k, int i)
{
    int j;
    for (j=1; j<=k-1; j++)
        if ((x[j]==i) || (abs(x[j]-i)==(k-j)))
            return 0;
    return 1;
}

void nqueen (int k)
{
    int i,j;
    for(i=1; i<=n; i++)
        if (place(k, i))
        {
            x[k]=i;
            if (k==n)
            {
                for(j=1; j<=n; j++)
                    printf("%d ", x[j]);
                printf("\n");
            }
            else nqueen(k+1);
        }
}
void main()
{   printf ("Enter the number of Queens\n");
    scanf ("%d", &n);
    printf ("All posible solutions are :\n");
    nqueen (1);
}
```

Page 5

19 April 2021 09:51

classmate

Date

Page

K, i

nqueen(1) → i=1 → place (1, 1) → nqueen(2) | i=1 → place (2, 1) X return 0.

x[1]=1 ← Backtrack

return 1

| i=2 → place (2,2) X return 0

i=2 → place (1,2)

return 1

| i=3 → place (2, 3) return 1 x[2]=3 ← Backtrack →

| i=4 → place(2,4) return 1 x[2] = 4

nqueen(2) →

i=1 → place (2, 1) x return 0

i=2 → place (2,2) X return 0

i=3 → place (2,3) x return 0

Back

i=4 → place (2,4) return 1

classmate

Date

Page

K, i

nqueen(3) i=1 → place (3,1) x return 0

i=2 → place (3,2) x return 0

i=3 → place (3,3) x return 0

i=4 → place (3,4) x return 0

nqueen(3) i=1 → place (3,1) X return 0 Backtrack

i=2 → place (3,2) x return 1 → nqueen(4) i=1 → place (4,1) x return 0

i=2 → place (4,2) x return 0

i=3 → place (4,3) x return 0

i=4 → place (4,4) x return 0

i=3 → place (3,3) x return 0

i=4 → place (3,4) x return 0.

nqueen(3) i=1 → place(3,1) → nqueen(4) → i=1 → place(4,1) x return 0

return 1

i=2 → place (4,2) x return 0

i=3 → place (4,3) return 1.

Page 6

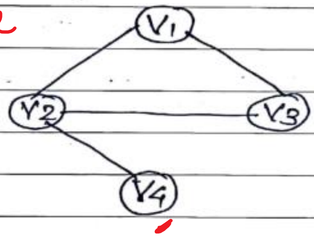
19 April 2021 09:51

★ GRAPH COLOURING PROBLEM :

- Given a graph of 'v' vertices and 'e' edges and 'n' different colours.
- We want to colour the graph with n colours in such a way that the adjustant vertices must not be of same colour.
- A graph is said to be 'm' colourable graph, if 'm' is the minimum no. of colour necessarily to colour all the vertices ensuring adjacent vertices are not of same colour.

- consider a graph :

Problem?
Agar ek graph hai to minimum kitne color we kaune package to ensure no two vertex are of same color adjacent



Here if we take 4 colors It's feasible as all vertex will be of diff color but not optimal because 4 is minimal value.

- The above graph is 3 colourable graph. Let, the colours be 1, 2 & 3.
- possible solutions are :

V_1	V_2	V_3	V_4	
1	2	3	1	✓
1	2	3	3	✓
3	2	1	3	
3	2	1	1	
⋮				

Scanned by CamScanner

Page 7

19 April 2021 09:51

color numbers

X | 1 | 2 | 3 | 1 |
  1 2 3 4

→ vertex NO (For above graph)

Solution :

Consider 1-D array X that has V elements where every element in X store color number.

Constrainst on array X :

(1) If i and j are adjacent vertices then x[i] != x[j]

(2) Every value in X must be between 1 to n

Algorithm :

Let g represents adjacency matrix of graph with v vertices and e edges

Let int x[v] stores solution.

Algo for color( )

[kth vertex pe ith color chalega kya ?]

```
int color (int k, int i)   ← k
{
    int j ;
    for ( j=1 ; j <= k-1 ; j++)
    {
        if ( g[k][j] != 0 && x[j] == i)
            return 0 ;
    }
    return 1 ;
}
```

current vertex NO   current color no

for all old vertex check

agar jth vertex kth vertex ko adjacent hai

x[j] = jth vertex ka color

i = kth vertex ka proposed color

Agar same hai Toh nahi chalega

k : (current) vertex number.

i : colour number. (current proposed)

g[k][j] != 0 : kth vertex aur jth vertex adjacent hai

x[j] = = i

colour of vertex j   proposed colour of kth vertex

Scanned by CamScanner

Yaha

Algo. for graphcolor()

```
void graphcolor (int k)
{
    int i;
    for (i=1; i<=n; i++)
    {
        if (color (k,i))
        {
            x[k] = i;
            if (k==v)
            {
                for (j=1 to v)      // solution.
                    print x[j]
            }
            else graphcolor (k+1);
        }
    }
}
```

k^{th} vertex ko color karo

possible n colors from 1 to n.

kya vertex k ko color i se color kar sakte hai kya

If yes

chk agar last vertex hai

Print

agar last vertex nahi hai

color next vertex

* SUM OF SUBSET :

Problem statement :

We are given 'n' positive integers and a positive integer number 's'.

We have to find all posible subset of 'n' such that their sum is 's'.

N = {1, 2, 3, 10, 12, 13, 15} ⇐ Set of +ve integers

s = 15. ⇐ Sum

{1, 2, 12} ⇒ 15

{3, 12} = 15

{15} = 15

{2, 3, 10} = 15

Possible solutions

Scanned by CamScanner

Possible solutions are:

{15} ⇒ X [15]
{2, 13} ⇒ X [2 | 13]
{3, 12} ⇒ X [3 | 12]
{1, 2, 12} ⇒ X [1 | 2 | 12]
{2, 3, 10} ⇒ X [2 | 3 | 10]
⋮

$$\sum_{i=0}^{n-1} x[i] = sum$$

The solution for the problem can be found using Backtracking approach. Let their be 1-D array X that has n locations. The array X will store those integers whose sum will be S.

Constrains on array X[] are:

1. Sum of elements in array X should not exceed S.
2. The values stored in array X should be unique as we do not want to repeat the subset.

Program:

```
#include <stdio.h>
#include <conio.h>
void sumset(int);
void proper (int,int);
int sum (int)

int v[10];          // To store elements
int x[10];          // To store solution.
int n, sum_req;
```

19 April 2021 09:51

sumreq is global (outside fn)

Assumption.

```
void main()
{
  int i;
  clrscr();
  printf("Enter number of elements\n");
  scanf("%d", &n);
  printf("Enter unique elements in increasing order\n");
    for(i=1; i<=n; i++)
      scanf("%d", &v[i]);
  Printf("Enter sum required\n");
  scanf("%d", &sumreq);
  sumset(1);
  getch();
}
```

Input that has unique and sorted elements

```
int sum (int n)          // gives sum of first n elements
{                        of array x.
  int s, i;
  s=0;
  for (i=1; i<=n; i++)
     s = s + x[i];
  return s;
}
```

element

x | 2 | 13 | | | |
1 2 3 4 5

Solution array

```
int proper(int k, int i)
{
  int j;
  for(j=1; j<=k-1; j++)    // If same element is present before
  {  if (x[j]==v[i])       // Repeated toh nahi
       return 0;
  }
```

chk karo array x main posn k par array v ka ith posn ka element chalega kya?

Scanned by CamScanner

Saare poorane k-1 posⁿ ke element ka sum in X

```
    if ((sum(k-1) + v[i]) > sumreq)     // purane kth element
        return 0;                       ka sum.
    else return 1;
}

int sumset(int k)
{
    int i, s, j;
    for(i=1; i<=n; i++)
    {
        if (proper(k,i))
        {
            x[k] = v[i];
            s = sum(k);
            if (s == sumreq)
            {                           // Print solution
                for(j=1; j<=k; j++)
                    printf("%d", x[j]);
            }
            else
                sumset(k+1);
        }
    }
}
```

current element V ka

In array X at posⁿ K place some element of array V Ensuring the total sum of elements of array X does not exceed sum req

In array X at pos K any of n elements from array V

fill next posⁿ of array X

NOTE:
K is position in array x.
i is position in array v.
Proper () specifies whether element at ith element position can be placed at kth position in array x.

Sumreq = 15

V = (1, 2, 3, 10, 12, 13, 14, 15)
1 2 3 4 5 6 7 8

X = | 1 | 2 | 3 | | | | | |
1 2 3 4 5 6 7 8

K

s = 1
s = 3
s = 6

proper(1,1) ✓
x V

proper(2,
K
proper(4, 4)

Sum(3) of X + V[i] ie V[4]
= 6 + 10 = 16 > Sumreq

19 April 2021 11:36

0 0 0
~~1~~ ~~1~~ 1

1 2 3 4 5 6 7 8 9 10 11 12

$l = 12 \Rightarrow T =$ a b a b c b a b a b a b

$m = \cancel{1} = 3 \Rightarrow P =$ a b c

j: 0 1 2 3

a b a b c b a b a b a d

Slide Size = 1

AOA17-6

19 April 2021 11:31

AOA17-6

Module 6 →

STRING MATCHING ALGORITHMS

★ BRUTEFORCE ALGORITHM / NAIVE METHOD:

Given a text T of length L and given pattern P of length m, we want to find index of text T where pattern P occurs.

Ex.

	i												
		0	1	2	3	4	5	6	7	8	9	10	11
T =		a	b	a	b	c	b	a	b	a	b	a	b

∴ L = 12

P =	a	b	c
	0	1	2
	j		

∴ m = 3

```
Algo Bruteforce (T, P)
{
    for (i=0 to L-m)          9    i=1,2
    {
        j=0;                  ~~1~~ ~~0~~ 2      ~~1~~ 2
                              2+~~0~~=~~2~~ 3 4
        while ( P[j] == T[i+j] )
        {
                              2 ~~1~~ ~~0~~ 2
            if (j == m-1)
                return i;     ⇒ 2
            else
                j = j+1       ⇒ ~~1~~ 2
        }
    }
    return -1; // P is not found in T.
}
```