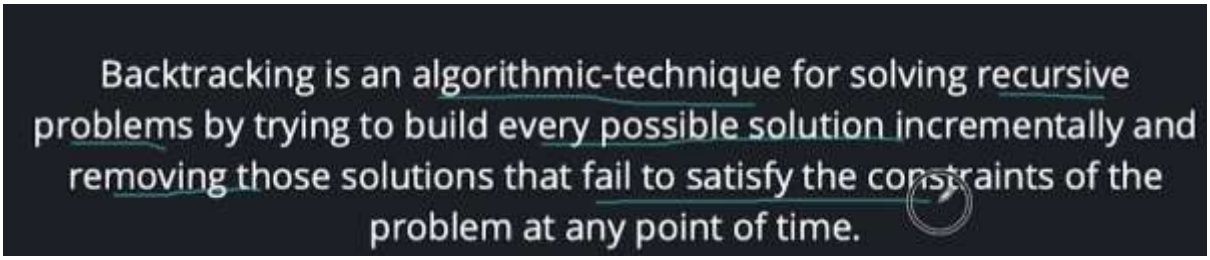
Day-25 of the 101 days Coding challenge

- **Backtracking**
  - **➔ It's a Concept where we are finding the feasible Solutions**
  - **➔ Like-> If one node goes to the one solution and it does not fit the node, then it will come to its previous positions, and from there again, try another option, so this situation is known as backtracking.**



Problem:

Input:

0 denotes wall, 1 denotes free path

two numbers n, m

Next n lines contain m numbers (0 or 1)

Output:

Print 1 if rat can reach (n-1,m-1)

Print 0 if it can not reach (n-1,m-1)

Test Case 1:

Input:

5 5

1 0 1 0 1

1 1 1 1 1

0 1 0 1 0

1 0 0 1 1

1 1 1 0 1

Output:

1

Code:

```
 #include<iostream>
using namespace std;

// checking our position is right or wrong
bool isSafe(int **arr, int x, int y, int n)
{
	if(x<n && y<n && arr[x][y] == 1) // if it is in right position
	{
		return true;
	}

	return false; // if not in right position(if arr[x][y] == 0) return false
}
```

```
bool inBoxposition(int **arr, int x, int y, int n, int **outArr)
{
        // base condition
        if(x == n-1 && y == n-1)
        {
                outArr[x][y] = 1; // stores 1 on sucessful
                return true;
   }
   if(isSafe(arr,x,y,n)){
           outArr[x][y] = 1;
           if(inBoxposition(arr, x+1, y, n, outArr)) // recursivly calling if condition is satisfied then incrementing the path on x axis
              return true;
           if(inBoxposition(arr, x, y+1, n, outArr)) // recursivly calling if condition is satisfied then incrementing the path on y axis
              return true;
           }
           arr[x][y] = 0; // backtracking
           return false;
        }
```

```
int main()
{
	int n;
	cout<<"Enter the size of the elements"<<endl;
	cin>>n;

	// dynamically alocating the memory
	int **arr = new int*[n];

	for(int i = 0; i<n; i++)
	{
		arr[i] = new int [n]; // one D
	}

	// user Input
	for(int i = 0; i<n; i++)
	{
		for(int j = 0; j<n; j++)
		{
			cin>>arr[i][j];
		}
```

```
    }
    cout<<endl;

    // Solution array
    int **outarr = new int*[n];

    for(int i = 0; i<n; i++)
    {
        outarr[i] = new int [n];
        // intilizing the initial value 0 so that garbage value will not occur into the array
        for(int j = 0; j<n; j++)
        {
          outarr[i][j] = 0;
        }
    }

    // now calling the function

    cout<<"Output :"<<endl;
    if(inBoxposition(arr, 0, 0, n,outarr))
    {
```

```
            for(int i = 0; i<n; i++)
            {
                for(int j = 0; j<n;j++)
                {
                    cout<<outarr[i][j]<<" ";
                } cout<<endl;
            }
        }
        return 0;
}
```

Output:

```
Enter the size of the elements
5
1 0 1 0 1
1 1 1 1 1
0 1 0 1 0
1 0 0 1 1
1 1 1 0 1

Output :
1 0 0 0 0
1 1 1 1 0
0 1 0 1 0
0 0 0 1 1
0 0 0 0 1

--------------------------------
Process exited after 38.01 seconds with return value 0
Press any key to continue . . .
```