Course Name: Analysis and Design of Algorithms

Topics to be covered

- Types of Problems/ Algorithms
- *NP*-Hard
- *NP*-Completeness

TWO KINDS OF ALGORITHMS

- *Polynomial Time: Algorithms whose solution is found in polynomial time*

eg., Sorting, searching etc.,

Non-polynomial Time: Algorithms which DO NOT take polynomial time to find the solution

eg., Travelling salesperson problem ($O(n^2 2^n)$)

TWO KINDS OF ALGORITHMS

- ***Problems for which there is no polynomial time complexity, are computationally related***
- ***There are two classes of such problems***
- ***1. NP HARD AND***
  2. ***NP COMPLETE***

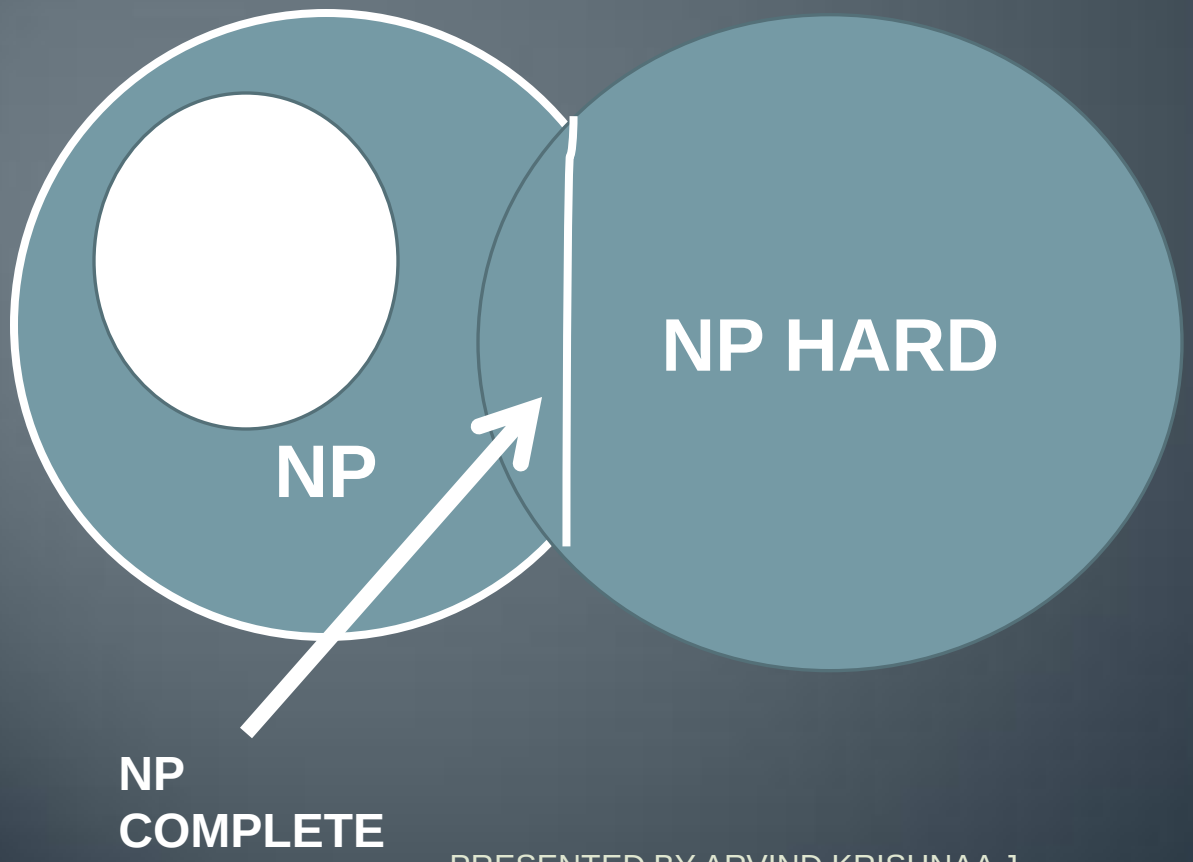
NP COMPLETE

The problem that is NP complete has the property that it can be solved in polynomial time if and only if all the other NP complete problems can be solved in polynomial time

NP HARD

If an NP-hard problem can be solved in ploynomial time, then all NP complete problems can also be solved in polynomial time.

SET REPRESENTATION OF THE ABOVE STATEMENTS



NON DETERMINISTIC ALGORITHMS

Deterministic Algorithm: Has the property that result of every operation is uniquely defined

Non-Deterministic Algorithm: Outcome of an operation is not uniquely defined

THREE NEW FUNCTIONS

Every non-deterministic algorithm makes use of three functions

- *choice(S): arbitrarily choosing an element in a s et S*

  for eg., x=choice(1,n) | means x€[1,n]
- *failure(): unsuccessful completion*
- *success(): successful completion*

SOME EXAMPLES OF NON-DETERMINISTIC ALGORITHMS

- NON DETERMINISTIC SEARCHING
- ND SORTING
- MAXIMUM CLIQUE PROBLEM
- 0/1 KANPSACK PROBLEM
- SATISFIABLITY

and many, many more

NON DETERMINISTIC SEARCHING

```
nondeterministic_search(x)
{
      int j=choice(1,n);
      if(A[j]==x)
      {
            cout<<j;
            success();
      }
      cout<<'0';
      failure();
}
```

NON DETERMINISTIC SORTING

```
voidnondeterministic_sort(int A[],int n)
{
        int B[SIZE],i,j;
        for(i=1;i<=n;i++) B[i]=0;
        for(i=1;i<=n;i++)
        {
                j=choice(1,n);
                if(B[j])
                        failure();
                B[j]=A[i];
        }
        for(i=1;i<=n;i++)//verify order
                if(B[i] > B[i+1])
                        failure();
        for(i=1;i<=n;i++)
                cout<<B[i]<<' ';
        success();
}
```

NON DETERMINISTIC Clique

```
void DCK(int G[]{SIZE],int n,int k)
{
        S=null;//initially empty
        for(int i=1;i<=k;i++)
        {
                int t=choice(1,n);
                if(t is in S)
                        failure();
                S=S U {t};
        }
        //now S contains k distinct vertices
        for(all pairs (i,j) such that i is in S,j is in S
and i!=j)
                if((i,j) is not an edge of G)
                        failure();
        success();
}
```

NON DETERMINISTIC KNAPSACK

```
void DKP(int p[],int w[],int n,int m,int r,int x[])
{
        int W=0,P=0;
        for(int i=1;i<=n;i++)
        {
                x[i]=choice(0,1);
                W+=x[i]*w[i];
                P+=x[i]*p[i];
        }
        if((W>m) || (P < r))
                failure();
        else
                success();

}
```

NON DETERMINISTIC SATISFIABLITY

```
void eval(cnf E,int n)
//determine whether the prop. formula is
satisfiable
//variables are x[1],x[2],x[3],...x[n]
{
int x[SIZE];
//choose a truth value assignment
for(int i=1;i<=n;i++)
      x[i]=choice(0,1);
if(E(x,n))
      success();
else
      failure();
}
```

SOME EXAMPLES OF NP HARD PROBLEMS

- GRAPH PROBLEMS LIKE
  - NODE COVERING PROBLEM
  - GRAPH COMPLEMENT PROBLEM

- SCHEDULING PROBLEMS
- CODE GENERATION PROBLEM

and many more

SOME EXAMPLES OF NP complete PROBLEMS

CHECK OUT THE TEXTBOOK

PAGE NO571

BASIC TEHNIQUES INVOLVED IN SOLVING *NP* COMPLETE PROBLEMS

- APPROXIMATION
- RANDOMIZATION
- RESTRICTION
- PARAMETRIZATION
- HEURISTICS

OR IN SHORT PERFORMING A NON-DETERMINISTIC ANALYSIS ON THE PROBLEM