NP – Hard
and
NP – Complete Problems

- Definition of P:
  - Set of all decision problems solvable in polynomial time by a deterministic algorithm (Turing machine)
- Examples:
  - MULTIPLE: Is the integer y a multiple of x?
    - YES: (x, y) = (17, 51).
  - RELPRIME: Are the integers x and y relatively prime?
    - YES: (x, y) = (34, 39).
  - MEDIAN: Given integers x_1 , …, x_n , is the median value < M?
    - YES: (M, x_1 , x_2 , x_3 , x_4 , x_5) = (17, 2, 5, 17, 22, 104)

Def: P is the set of all decision problems solvable in polynomial time on **REAL** computers.

Example: ordered searching O(n)

Evaluation of polynomial O(n)

Sorting O(n log n)

There is another group of problems whose best known algorithms are non polynomial.

Example: traveling salesman problem,

Knapsack problem

Decision Problem

- *Any problem for which the answer is yes or no, is called a decision problem*.

Example: Knapsack problem:

Optimization Problem: Find the largest total profit of any subset of objects that fits in the knapsack.

Decision Problem: Given k, is there a subset of objects that fits in the knapsack and has total profit at least k?

Example : Subset sum problem:

Input is a positive integer C and n objects whose sizes are positive integers $s_1, s_2, \ldots s_n$

Optimization problem: Among subsets of the objects with sum atmost C what is largest subset sum.

Decision Problem: Is there a subset of the objects whose sizes add up to exactly C?

Idea of decision problems

- The decision problem can be solved in polynomial time if and only if the corresponding optimization problem can.

- If the decision problem cannot be solved in polynomial time then the optimization problem cannot either.

we can make the statement that

If the decision problem cannot be solved in polynomial time then the optimization problem cannot either.

Non deterministic algorithms

Algorithms having operations whose outcomes are not uniquely defined but are limited to specified sets of possibilities.

it has functions like:

Choice (S): arbitrarily chooses one of the elements of the set S.

Failure(): signals an unsuccessful case

Success(): signals a successful completion.

Example (Searching for an element x in a given set of elements A[1 : n] n ≥ 1.)

Non deterministic algorithm:

```
1    J:= choice(1,n);
2    If A[j] = x then {write (j); success;}
3    Write (0); failure();
```

Number 0 can be a output if anf only if there is no j such that A[j] = x

Algorithm is of nondeterministic complexity O(1)

However any deterministic search algo on unordered data is of Ω(n)

Whenever there is a set of choices that leads to a successful completion then one such set of choices is always made and the algorithms terminates successfully/

When ever successful termination is possible , a nondeterministic machine makes a sequence of choices that is a shortest sequence leading to a successful termination.(remember machine is fictitious)

A nondeterministic algorithm terminates unsuccessfully if and only if there exists no set of choices leading to success signal

Time of non deterministic algorithms

The time required by a nondeterministic algorithm performing on any given input is the minimum number of steps needed to reach a successful completion if there exists a sequence of choices leading to such a completion.

In case successful completion is not possible then the time required is O(1).

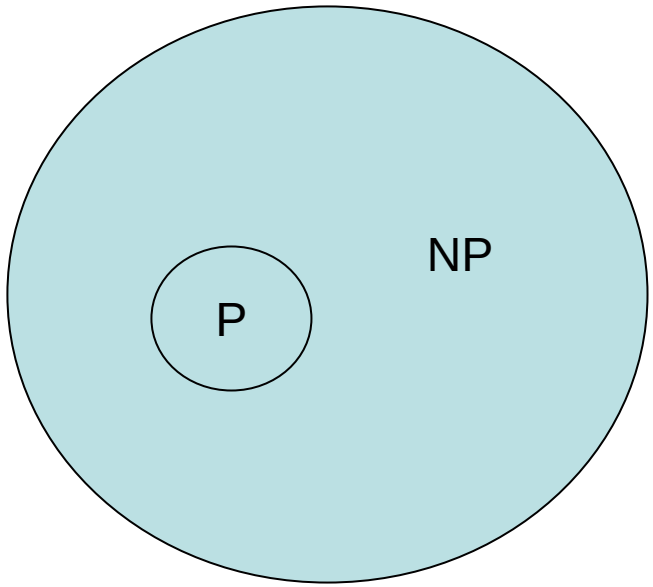
A nondeterministic algorithm is of complexity (f(n)) if for all inputs of size n

$n \geq n_0$ that results is a successful completion the time required is

atmost cf(n) for some constant c and n_0

- Definition of NP:
  - Set of all decision problems solvable in polynomial time by a NONDETERMINISTIC Algorithm(Turing machine).
  - Definition is important because it links many fundamental problems
- Useful alternative definition
  - Set of all decision problems with efficient verification algorithms
    - efficient = polynomial number of steps on deterministic TM
  - Verifier: algorithm for decision problem with extra input

Relation between P and NP



Example

```
Algorithm Nsort(A,n)
// Sort n positive integers
{
  for i:= 1 to n do B[i] :=0;
  for i:= 1 to n do
{ j:= choice (1,n)
If B[j]  0 then failure();
B[j]:= A[i];
}
For i:= 1 to n-1 do
     if B[i]> B[i+1] then failure();
Write(B([1:n]);
Success();
}
```

It is O(n) while all determinisitic sorting algorithms have a complexity $\Omega(n \log n)$

NP = set of decision problems with efficient verification algorithms

- Why doesn’t this imply that all problems in NP can be solved efficiently?
  - BIG PROBLEM: need to know certificate ahead of time
    - real computers can simulate by guessing all possible certificates and verifying
    - naïve simulation takes exponential time unless you get "lucky"

Satisfiability Problem

The satisfiability problem is to determine whether a boolean formula is true for some assignment of truth values to the variables.

{boolean formula is an expression that can be constructed using literals and operators “and” , “or”.

CNF- satisfiabilty is the satisfiability problem for CNF(conjunctive)formula.

Nondeterministic algorithm for satisfiability problem

Algoritm Eval (E,n)

{

For i:=1 to n do

   x_i := choice (false,true);

If E(x_1,x_2…,x_n) then success();

Else failure();

}

Time of the nondeterministic algorithm

O(n) time is required to choose the value of $(x_1,x_2 \ldots x_n)$

+

Time needed to evaluate E for that assignment

(time is proportional to the length of E)