Design and Analysis of Algorithm

Unit 8:Introduction to Complexity Theory

The class P and NP, Polynomial reduction
NP- Complete Problems, NP-Hard Problems
Travelling Salesman problem

Compiled By

Prof. Dharmesh R. Tank

CE\IT Department, LDRP-ITR

Outline

- **The class P and NP,**
- **Polynomial Reduction**
- **NP- Complete Problems,**
- **NP-Hard Problems**
- **Travelling Salesman Problem**

The Class P

- Most (but not all) of the algorithms we have studied so far are easy, in that they can be solved in polynomial time, be it linear, quadratic, cubic, etc.
- Cubic may not sound very fast, and isn't when compared to linear, but compared to exponential we have seen that it has a much better asymptotic behaviour.
- There are many algorithms which are not polynomial. In general, if the space of possible solutions grows exponentially as n increases, then we should not hope for a polynomial time algorithm. These are the exception rather than the rule.

The Class P

- The class P consists of those problems that are solvable in **polynomial time**.
- More specifically, they are problems that can be solved in time O(nk) for some constant k, where n is the size of the input to the problem
- The key is that n is the **size of input**
- **Example:** All sorting and searching algorithms

The Class NP

- NP is not the same as non-polynomial complexity/running time. NP does not stand for not polynomial.
- **NP = Non-Deterministic polynomial time**
- NP means verifiable in polynomial time
- Verifiable?
- If we are somehow given a ‘certificate’ of a solution we can verify the legitimacy in polynomial time
- **Example:** Su-Do-Ku, Prime Factor, Scheduling, Travelling Salesman

NP Problem

- The set of NP problems that can be mapped to each other in polynomial time is called NP-Complete.
- It is difficult to prove that something can not be done.
- Let me repeat that, it is difficult to prove that something can not be done.
- So, while we know how to solve many of these algorithms in exponential time, whose to say that one of you bright students won't come up with a clever polynomial time algorithm!
- NP-Complete is useful from that standpoint, if any of them turns out to have a polynomial time algorithm, they all do (hence P=NP).
- Of course, us old professors feel that these problems have been looked at from every angle and no such solution will ever be found (hence, $P \neq NP$).

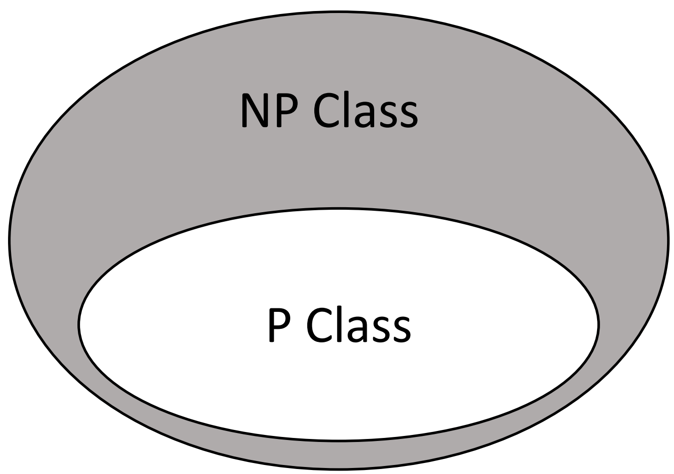
NP Problem

- It is also useful to know these algorithms, as they occur frequently in real applications and tackling them in a brute force fashion may be disastrous.
  - SAT problem
  - Traveling Salesman problem
  - Knapsack Problem
  - Longest Path
  - Graph Clique

Class P vs Class NP

- The NP Class Problems, it is verified in polynomial time.
- The P Class problems, not only it is solved on polynomial time but it is verified also in polynomial time.
- The P class problem called **Tractable(Easy to control) problem** and NP Class problem called **Intractable problem(Hard to control).**

NP Class
Hard to solve
&
Easy to verify
exponential
time



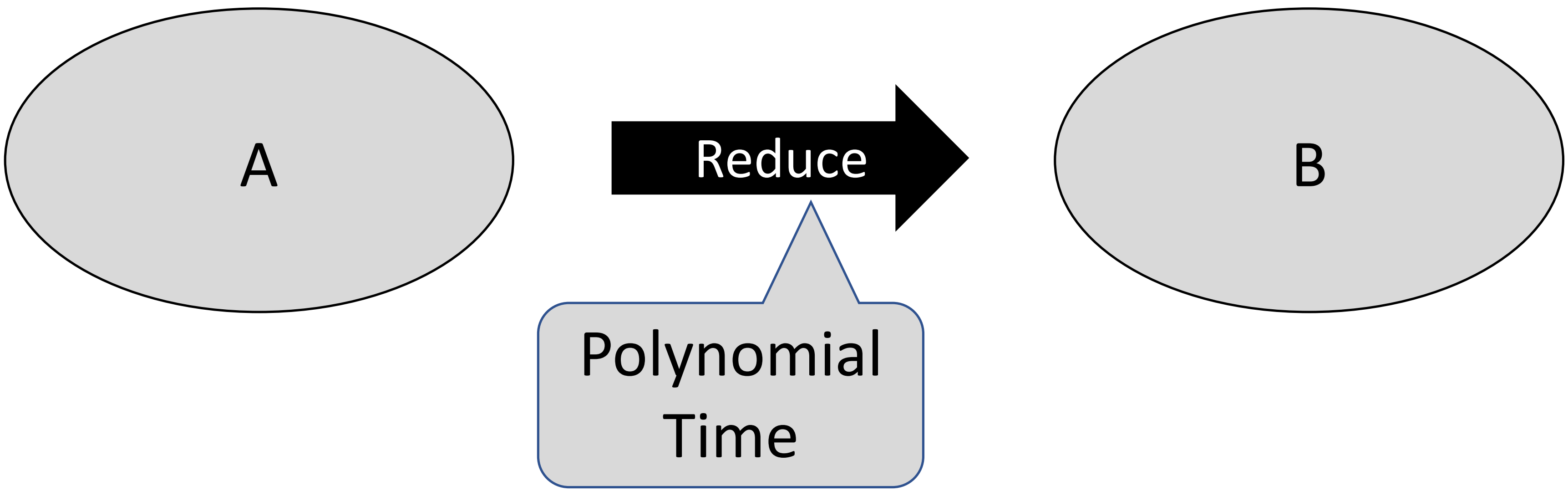
P Class
Easy to solve
&
Easy to verify
Polynomial
time

That's P is subset of NP

Is P = NP ?

- ➢It is a million dollar questions.
- ➢Till now no body solve it.
- ➢What is reason behind it…
- ➢Let assume that if you can prove P = NP then
  - Information security or online security is vulnerable to attack, Everything become more efficient such as Transportation, Scheduling, Understanding DNA etc.
- ➢Let assume that if you can prove P ≠ NP then
  - You can prove that there are some problems that can never be solved.

Reduction



- Let A and B are two problems then problem A reduces to problem B if and only if there is a way to solve A by deterministic algorithm that solve B in polynomial time.
- If A reducible to B we denote it by A $\propto$ B

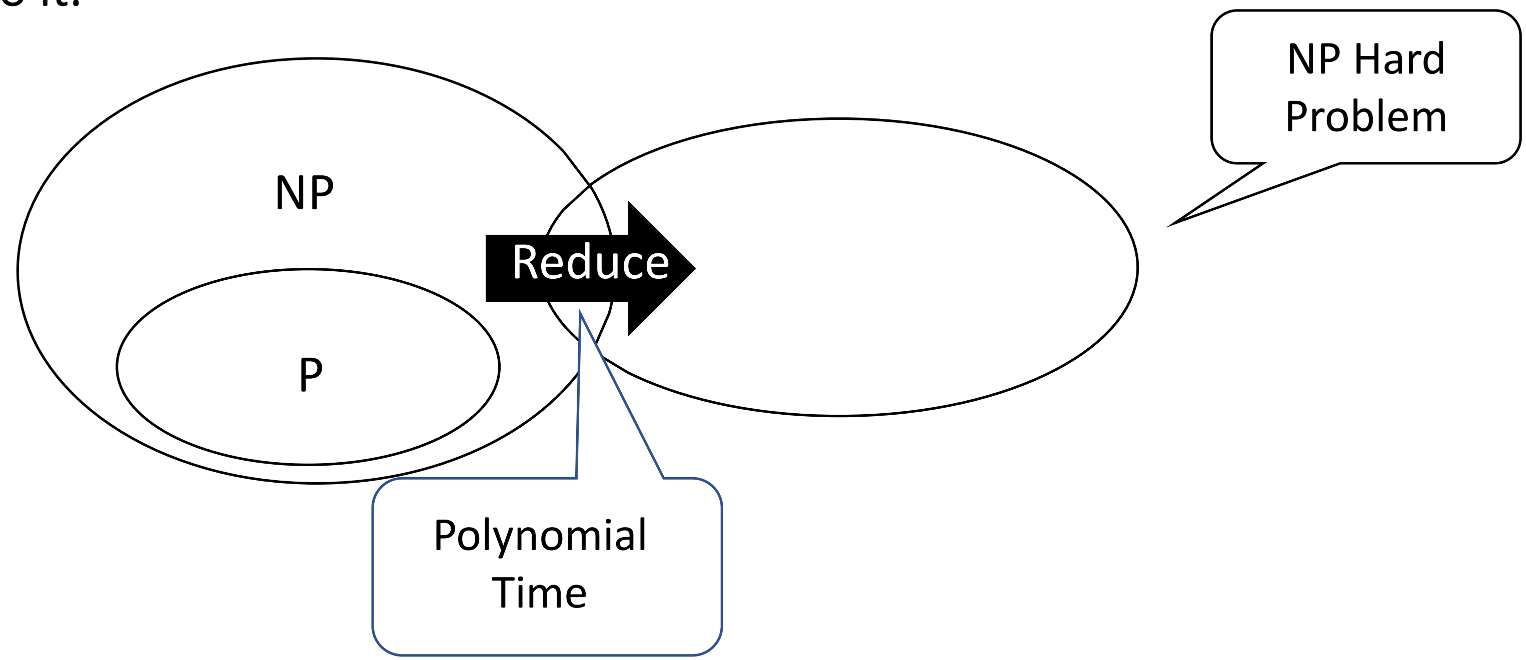
Reduction

- Properties :

1. If A reducible to B and B in P then A in P.
2. A is not in P implies B is not in P.

NP Hard Problem

- A problem is NP-Hard if every problem in NP can be polynomial reduced to it.



- If i having one set, then all problem on NP can be reduced to this new set and the reduction will take polynomial time, then it is called NP Hard problem.

NP Complete Problem

- A language B is NP-complete if it satisfies two conditions
  - B is in NP
  - Every A in NP is polynomial time reducible to B
- If a language satisfies the second property, but not necessarily the first one, the language B is known as NP-Hard.
- Informally, a search problem B is NP-Hard if there exists some NP-Complete problem A that Turing reduces to B.
- The problem in NP-Hard cannot be solved in polynomial time, until P = NP.
- If a problem is proved to be NPC, there is no need to waste time on trying to find an efficient algorithm for it.
- Instead, we can focus on design approximation algorithm.

NP Complete Problem

- Following are some NP-Complete problems, for which no polynomial time algorithm is known.
  - Determining whether a graph has a Hamiltonian cycle
  - Determining whether a Boolean formula is satisfiable, etc.

Travelling Salesman Problem

- The travelling salesman problem (also called the traveling salesperson problem or TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?“
- It is an NP-hard problem in combinatorial optimization, important in theoretical computer science and operations research.
- In the traveling salesman Problem, a salesman must visits n cities.
- We can say that salesman wishes to make a tour or Hamiltonian cycle, visiting each city exactly once and finishing at the city he starts from.
- There is a non-negative cost c (i, j) to travel from the city i to city j.
- The goal is to find a tour of minimum cost. We assume that every two cities are connected.
- Such problems are called Traveling-salesman problem (TSP).

Travelling Salesman Problem

- We can model the cities as a complete graph of n vertices, where each vertex represents a city.
- It can be shown that TSP is NPC.
- If we assume the cost function c satisfies the triangle inequality, then we can use the following approximate algorithm.

$$c(u, w) \leq c(u, v) + c(v, w)$$

- **Algorithm**

```
Approx-TSP (G= (V, E))
{
 1. Compute a MST T of G;
 2. Select any vertex r is the root of the tree;
 3. Let L be the list of vertices visited in a preorder tree walk of T;
 4. Return the Hamiltonian cycle H that visits the vertices in the order L;
}
```

TSP using Branch-and-Bound

T(k) = a tour on k cities

Search(k, T(k-1))

  if k=n

    record the tour details

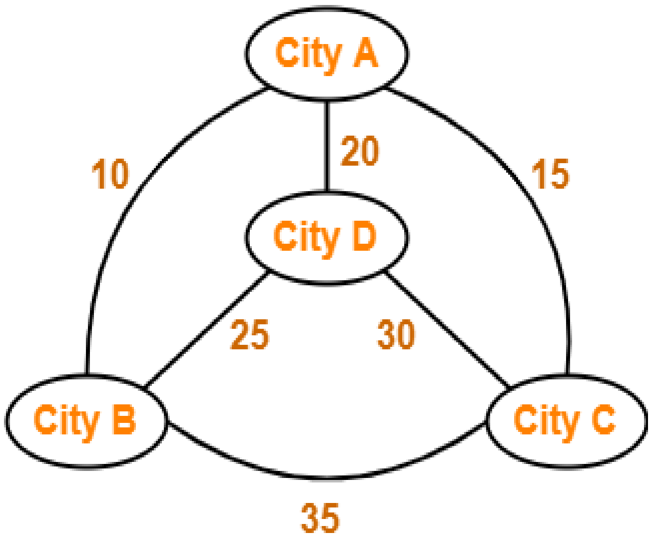
    the bound B=length of the tour

  else

    Find the k-1 possibilities of adding k to all of the possible places in the tour

    For every tour where the tour length is less then B, Search(k+1,T(k))

Example



Travelling Salesman Problem

If salesman starting city is A, then a TSP tour in the graph is-

A → B → D → C → A

Cost of the tour

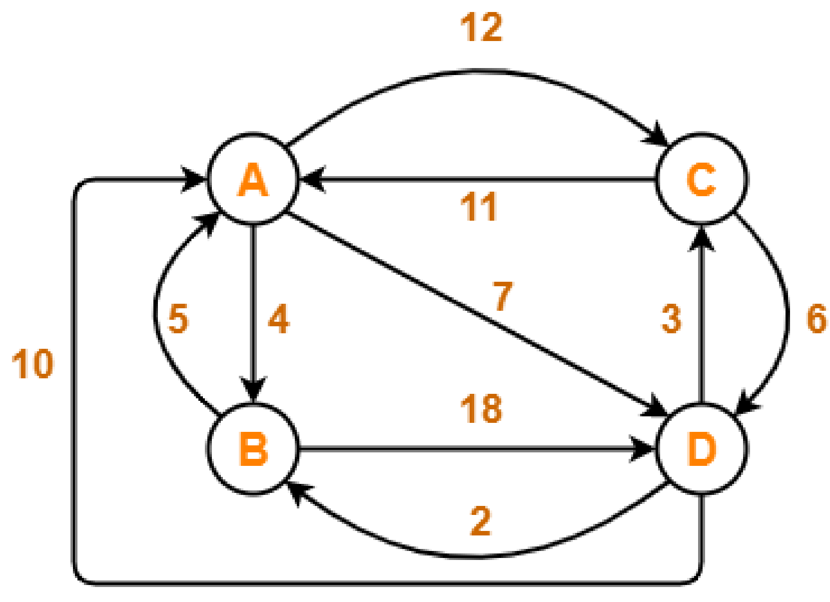
= 10 + 25 + 30 + 15 = **80 units**

Source : https://www.gatevidyalay.com/

TSP using Branch-and-Bound

Problem:

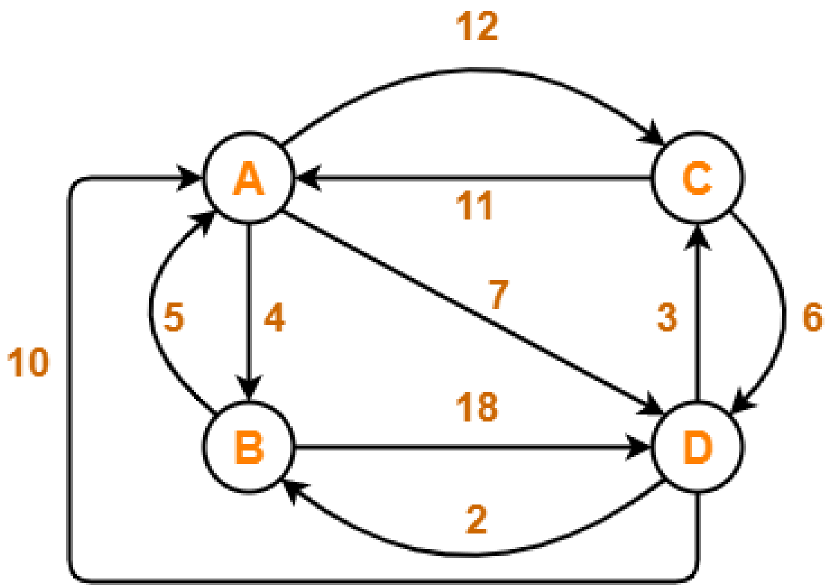
Solve Travelling Salesman Problem using Branch and Bound Algorithm in the following graph-



TSP using Branch-and-Bound

Solution:

Step-01: Write the initial cost matrix and reduce it.



$$
\begin{array}{c|cccc}
 & A & B & C & D \\
\hline
A & \infty & 4 & 12 & 7 \\
B & 5 & \infty & \infty & 18 \\
C & 11 & \infty & \infty & 6 \\
D & 10 & 2 & 3 & \infty
\end{array}
$$

Rules

- To reduce a matrix, perform the row reduction and column reduction of the matrix separately.
- A row or a column is said to be reduced if it contains at least one entry '0' in it.

TSP using Branch-and-Bound

Solution:

Row Reduction:

Consider the rows of above matrix one by one.

If the row already contains an entry ‘0’, then-

- There is no need to reduce that row.

If the row does not contains an entry ‘0’, then-

- Reduce that particular row.
- Select the least value element from that row.
- Subtract that element from each element of that row.
- This will create an entry ‘0’ in that row, thus reducing that row.

TSP using Branch-and-Bound

Solution:

Column Reduction:

Consider the columns of above matrix one by one.

If the column already contains an entry ‘0’, then-

- There is no need to reduce that column.

If the column does not contains an entry ‘0’, then-

- Reduce that particular column.
- Select the least value element from that column.
- Subtract that element from each element of that column.
- This will create an entry ‘0’ in that column, thus reducing that column.

TSP using Branch-and-Bound

Solution:

Step-02: Perform row reduction and column reduction.

Row Reduction:

Following this, we have-

$$\begin{array}{c|c} & \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{bmatrix} \infty & 4 & 12 & 7 \\ 5 & \infty & \infty & 18 \\ 11 & \infty & \infty & 6 \\ 10 & 2 & 3 & \infty \end{bmatrix} \end{array} \begin{array}{c} 4 \\ 5 \\ 6 \\ 2 \end{array}$$

- Reduce the elements of row-1 by 4.
- Reduce the elements of row-2 by 5.
- Reduce the elements of row-3 by 6.
- Reduce the elements of row-4 by 2.

So new matrix will form as

$$\begin{array}{c|c} & \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{bmatrix} \infty & 0 & 8 & 3 \\ 0 & \infty & \infty & 13 \\ 5 & \infty & \infty & 0 \\ 8 & 0 & 1 & \infty \end{bmatrix} \end{array}$$

TSP using Branch-and-Bound

Solution:

Step-02: After row reduction perform column reduction on new matrix

Column Reduction:

$$
\begin{array}{c|cccc}
 & A & B & C & D \\
\hline
A & \infty & 0 & 8 & 3 \\
B & 0 & \infty & \infty & 13 \\
C & 5 & \infty & \infty & 0 \\
D & 8 & 0 & 1 & \infty \\
\hline
 & 0 & 0 & 1 & 0
\end{array}
$$

Following this, we have-

- There is no need to reduce column-1.
- There is no need to reduce column-2.
- Reduce the elements of column-3 by 1.
- There is no need to reduce column-4.

So new matrix will form as

$$
\begin{array}{c|cccc}
 & A & B & C & D \\
\hline
A & \infty & 0 & 7 & 3 \\
B & 0 & \infty & \infty & 13 \\
C & 5 & \infty & \infty & 0 \\
D & 8 & 0 & 0 & \infty
\end{array}
$$

TSP using Branch-and-Bound

Solution:

Step-02: Find the reduced cost will be

Reduction Cost:

Row wise Minimum Value	4 5 6 2	+	0 0 1 0	Column wise Minimum Value

Reduced Cost = 4+5+6+2+1 =18

TSP using Branch-and-Bound

Solution:

Step-03:

We consider all other vertices one by one.

We select the best vertex where we can land upon to minimize the tour cost.

Reduced Cost = 4+5+6+2+1 =18

For further steps refer below link

https://www.gatevidyalay.com/tag/travelling-salesman-problem-example-with-solution-ppt/

TSP Applications

- The TSP has many practical applications
- manufacturing
- plane routing
- telephone routing
- networks
- traveling salespeople
- structure of crystals

Thank You