NP-Complete Problems

Tractable Problems vs. Intractable Problems

- An algorithm for a given problem is said to be a *polynomial time* algorithm if it's worst case complexity belongs to **$O(n^k)$** for a fixed integer **k** and an input size of n.

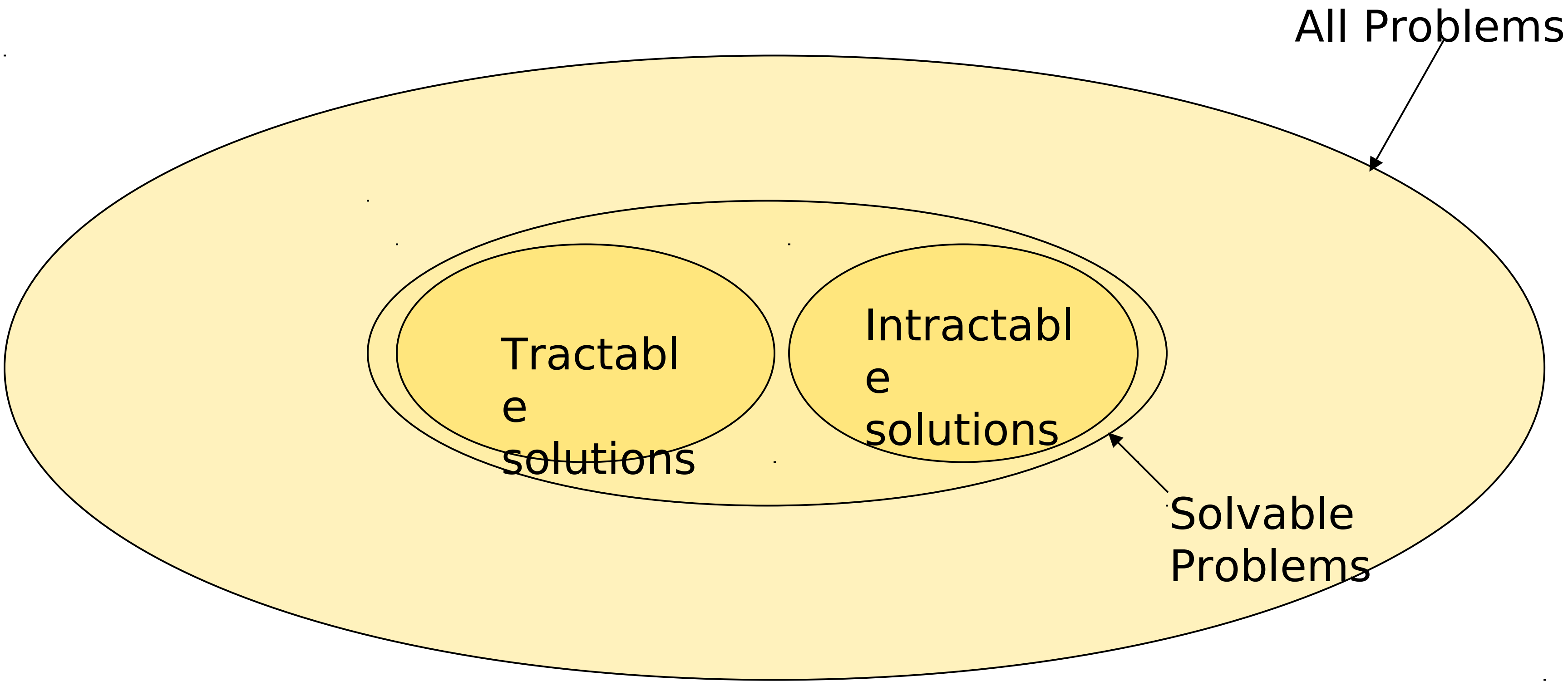
- The set of all problems that can be solved in *polynomial amount of time* are called **Tractable Problem**s. These problems can run in a reasonable amount of time for even very large amounts of input data.

- The set of all problems that cannot be solved in polynomial amount of time are called **Intractable Problem**s. It is of type **$O(k^n)$.** Intractable problems require huge amounts of time for even modest input sizes.

Tractable Problems vs. Intractable Problems

1	1	1	1	1
log n	3.3	4.3	4.9	5.3
n	10	20	30	40
nlog n	33	86	147	212
n^2	100	400	900	1600
n^3	1000	8000	27000	64000
2^n	1024	1 million	1.1 billion	1.1 trillion

Tractable Problems vs. Intractable Problems



Deterministic Vs Non-Deterministic Machines

- Deterministic machines:

  Conventional Digital machines are Deterministic in nature.

  Serialization of resource access

- Non – Deterministic machines:

  Hypothetical machine.

  More than one job can be done in one unit of time.

Deterministic Vs Non-Deterministic Machines

Conventional Digital Machines do a Sequential Execution. This execution is based on

- Von Neumann Architecture
- Serialization of resource access

Such machines are called Deterministic Machines.

Deterministic Vs Non-Deterministic Machines

- In Non-deterministic machines there will be only one hypothetical processor which can do more than one job at any instance of time.

Deterministic Vs Non-Deterministic Machines

- Example:

Consider Linear search. Let us consider that the scanning of an element takes 1 unit of time.

In deterministic machines, searching is done by scanning every element. If there are n elements, then the average time taken will be of O(n).

In non-deterministic machines, searching is done in parallel fashion. So the time taken will be of O(1).

Problems

Ex: Suppose we have an unlimited number of bins each of capacity one, and n objects with sizes $s_1, s_2, \ldots\ldots, s_n$ where $0 < s_i <= 1$

Optimization Problem: Determine the smallest number of bins into which the objects can be packed (find an optimal packing)

Decision Problem: Given, in addition to the inputs described, an integer k, do the objects fit in k bins?

P-Class problems

Definition

Polynomial problems are the set of problems which have polynomial time

Algorithms

- The class of decision problems that can be solved in polynomial time **by**

deterministic algorithms is called the **P class** or **Polynomial problem**s.

Polynomial problems

O(1) -- Constant

O(log n) -- Sub-linear

O(n) -- Linear

O(*n* log n) -- Nearly linear

O(n^2) – Quadratic

Decision Problems are problems with **yes/no** answers.

NP Class

Definition

- NP problems are the set of problems which have nondeterministic polynomial time algorithms

The class of decision problems that can be solved in polynomial time by nondeterministic algorithms is called the **NP class** or **Nondeterministic Polynomial problem**s.

- Algorithms which run in Polynomial time on a nondeterministic machine are called nondeterministic polynomial time algorithms.

NP Class

- Graph coloring, Hamiltonian cycle, Hamiltonian path, job scheduling with penalties, bin packing, the subset sum problem, satisfiability problem and the traveling salesperson problem are all in NP

Nondeterministic algorithm

A nondeterministic algorithm has two phases and an output step

- The nondeterministic *guessing phase.* Some completely arbitrary string of characters *s*, is written beginning at some designated place in memory. Each time the algorithm is run, the string written may differ.
- The deterministic *verifying phase*. A deterministic subroutine begins execution. In addition to the decision problem's input, the subroutine may use *s*, or it may ignore *s*. Eventually it returns a value true or false – or it may get in an infinite loop and never halt.
- The Output step: If the verifying phase returned true, the algorithm outputs yes. Otherwise, there is no

NP complete problems

- A NP Complete problem is one which belongs to the NP class
- Every problem in this NP class can be reduced to another NP problem, in a polynomial way
- If a problem in a class of NP can be solved in Polynomial way then all the problems in that class will be solved in a polynomial way using the same algorithm

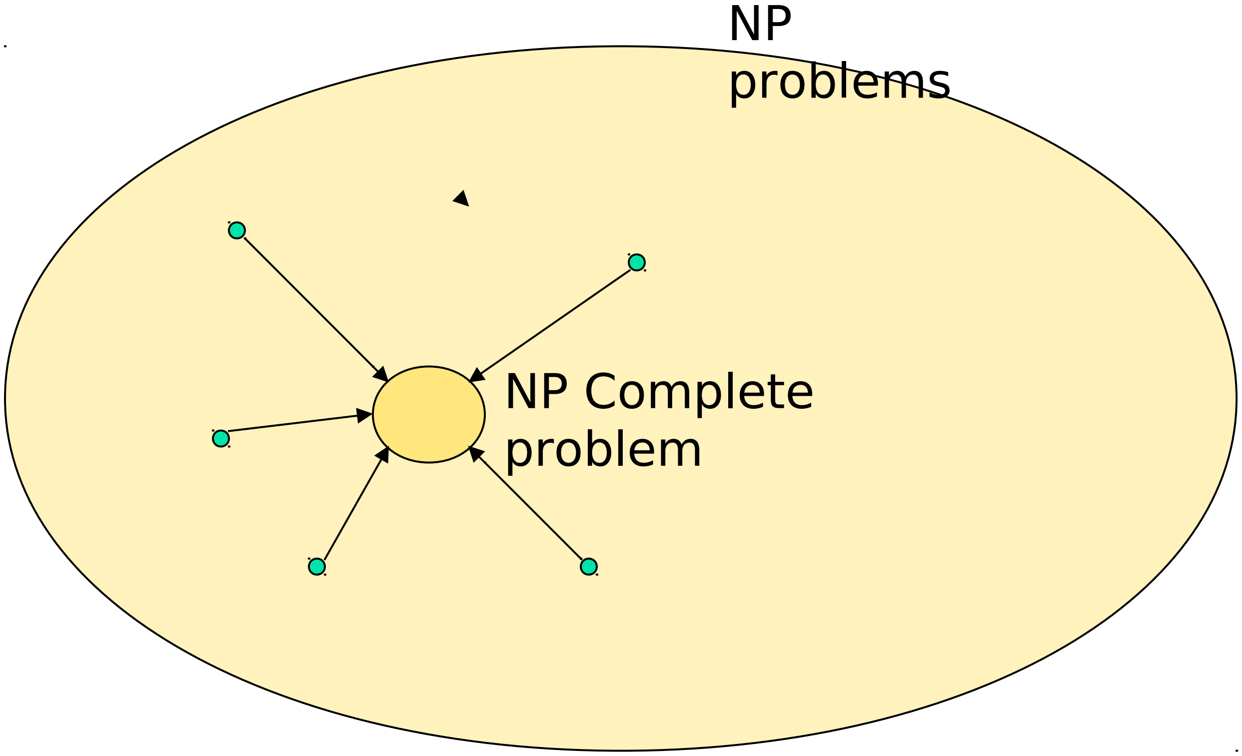
NP complete problems

A formal definition is given below

A decision problem *D* is said to be **NP-Complete** if

1. it belongs to NP class
2. Every problem in NP is polynomially reducible to D

NP complete problems



NP-Hard problems

- A problem Q is NP-hard if every problem K in NP is reducible to Q.
- A problem Q is NP-Complete if it is in NP and is NP-hard

- The problems that cannot be solved by any algorithms are called **Undecidable Problems**

Traveling Salesperson Problem

- The salesperson wants to minimize the total traveling cost (time or distance) required to visit all the cities in a territory and return to the starting point.

- This problem is known as traveling salesperson problem (TSP) or minimum tour problem

Traveling Salesperson Problem

- <u>The Nearest-Neighbor Strategy</u>

nearestTSP(V,E,W)

select an arbitrary vertex *s* to start the cycle C

v=s;

While there are vertices not yet in C

select an edge *vw* of minimum weight, where *w* is not in C

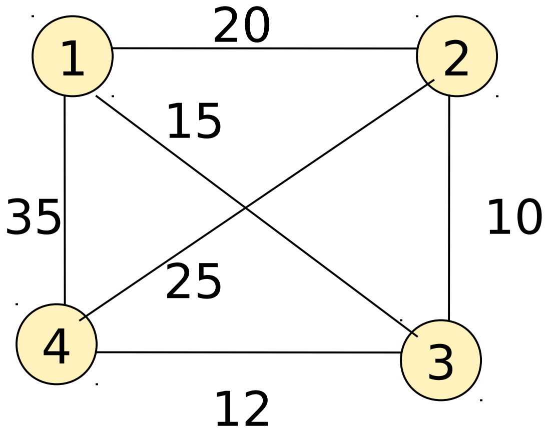
Add edge *vw* to C

v=w;

Add the edge *vs* to C

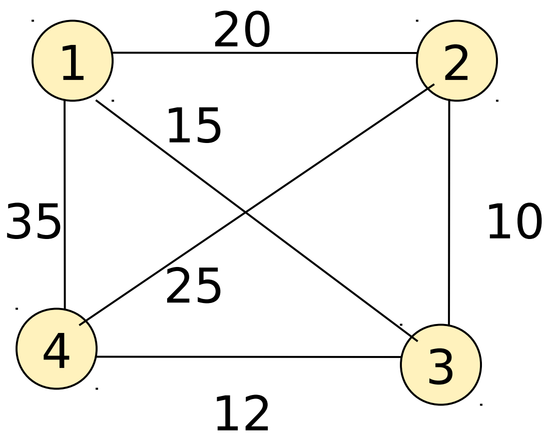
return C;

Traveling Salesperson Problem



Starting at vertex 1, the algorithm gives the cycle

Traveling Salesperson Problem



Starting at vertex 1, the algorithm gives the cycle 1, 3, 2, 4, 1 with total weight 85

Traveling Salesperson Problem

<u>The Shortest-Link Strategy</u>

ShortestlinkTSP(V,E,W)

R=E;   //R is remaining edges

C=Ø;   //C is cycle edges

While R is not empty

    Remove the lightest edge, vw, from R

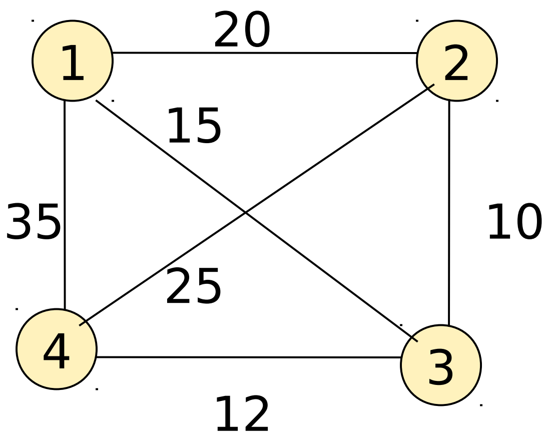
    if vw does not make a cycle with edges in C and vw would not be the third edge in C incident on v or w;

    Add vw to C

Add the edge connecting the endpoints of the path in C

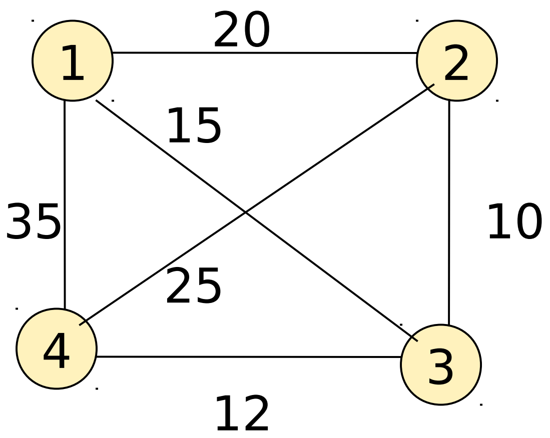
return C;

Traveling Salesperson Problem



The algorithm selects the edges

Traveling Salesperson Problem



The algorithm selects the edges (2,3), (3,4), (1,2), (1,4)

weight =77

Thank You