NPR College of Engineering & Technology
NPR Nagar, Natham, Dindigul - 624401, Tamil Nadu, India.
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai.
An ISO 9001:2015 Certified Institution.
Phone No: 04544- 246 500, 246501, 246502.
Website : www.nprcolleges.org, www.nprcet.org, Email:nprcetprincipal@nprcolleges.org

CS8501

THEORY OF COMPUTATION

by,
C.Kalpana, M.E(CSE)
Assistant Professor,
Department of Computer Science and Engineering,
NPR College of Engineering & Technology

Syllabus

UNIT V

UNDECIDABILITY

Non Recursive Enumerable (RE) Language – Undecidable Problem with RE – Undecidable Problem about TM – Post’s Correspondence Problem – The Class P and NP.

Turing Machine Problems

-> Algorithm

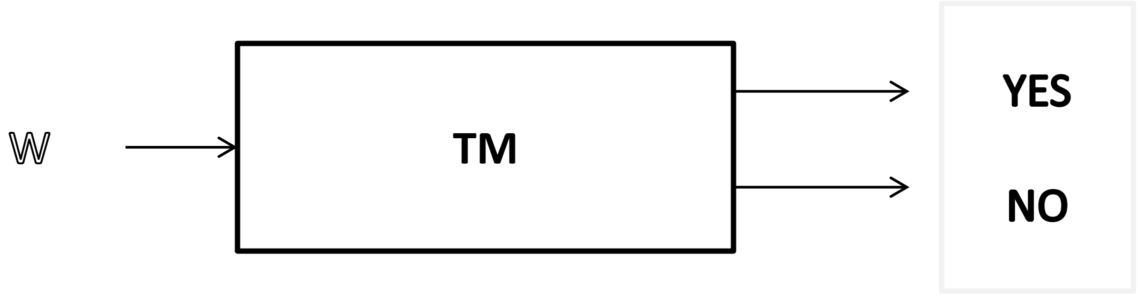
-> Only solved by TM

TM – **Halts -**>(Accept / Rejects)

Run forever -> Input do not accept.

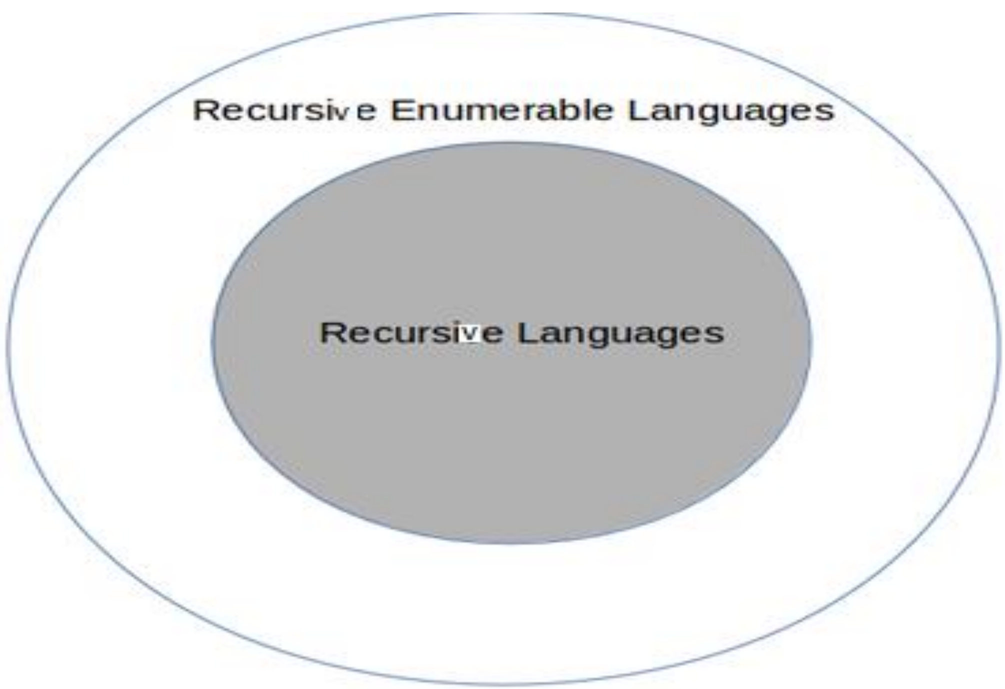
Recursive Language

A language is said to be **recursive** if there exists a Turing Machine that **accepts** every string of the language and **rejects the string** that are **not in the language.**



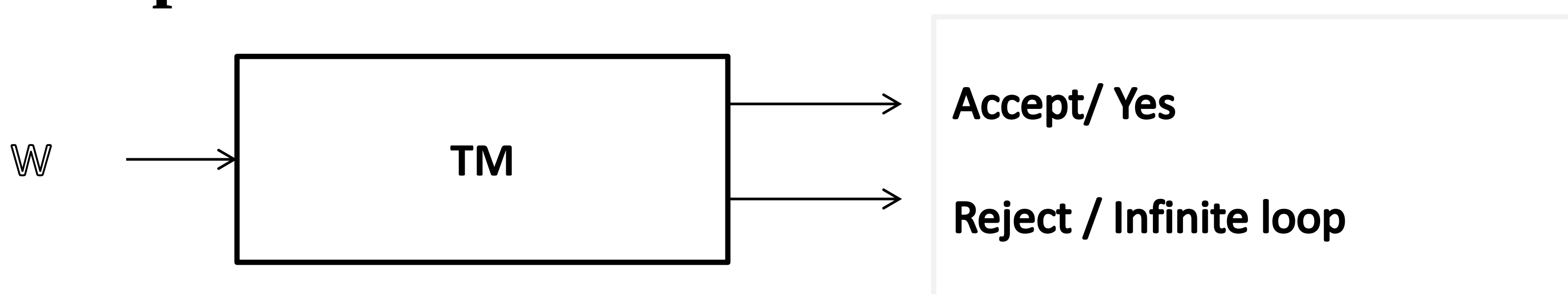
Recursively Enumerable Language

A language defined by **phrase structure grammar** is known as Recursively Enumerable language.



Recursively Enumerable Language

A language is said to be **recursively enumerable** if there exists a Turing Machine that **accepts** every string of the language and **rejects the string** that are **not in the language** and it may cause TM to enter into an **infinite loop**.



Properties of Recursive & RE Languages

Complement

W → M – Yes → No

No → Yes

Properties of Recursive & RE Languages

- 2 Classes
- **First Class**
- TM decides whether the input string belongs to that language or not.
- It halts, whether or not it reaches the accepting state.

Properties of Recursive & RE Languages

- 2 Classes
- **<u>Second Class</u>**
- It consists of RE languages that are not accepted by TM with the guarantee of Halting.

Language that is not recursively enumerable

Aim – To prove the undecidable language consists of pairs (M,W) such that

Ex:-

i) M is a TM with input alphabets { 0 , 1}

ii) W is a string of o’s and 1’s.

Decidable Problems

A Problem whose **language is recursive** is said to be **decidable** otherwise it is not decidable.

Ex:-

The set of strings of equal no. of o’s and 1’s.

0011 , 011010

Undecidable Problems

A Problem is said to be **undecidable** if there is no algorithm and we cannot predict the input is accepted by TM or not.

No algorithm exists to solve a problem in finite time.

Ex:-

Does the TM halt on input w is an undecidable problem.

Theorem 1

If L is recursive then L' is also a recursive language.

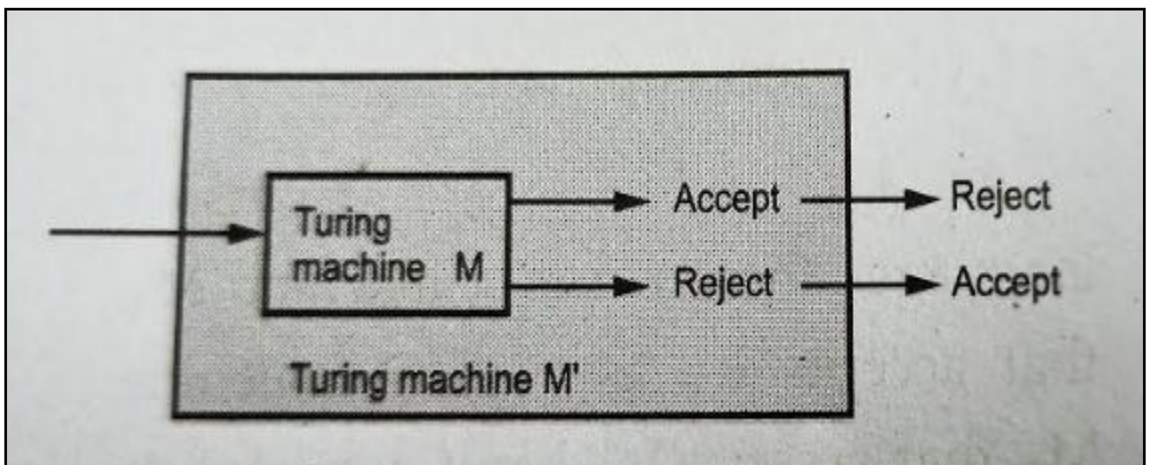
Proof

Language L can be accepted by TM

L=L(M)

L'=(M')

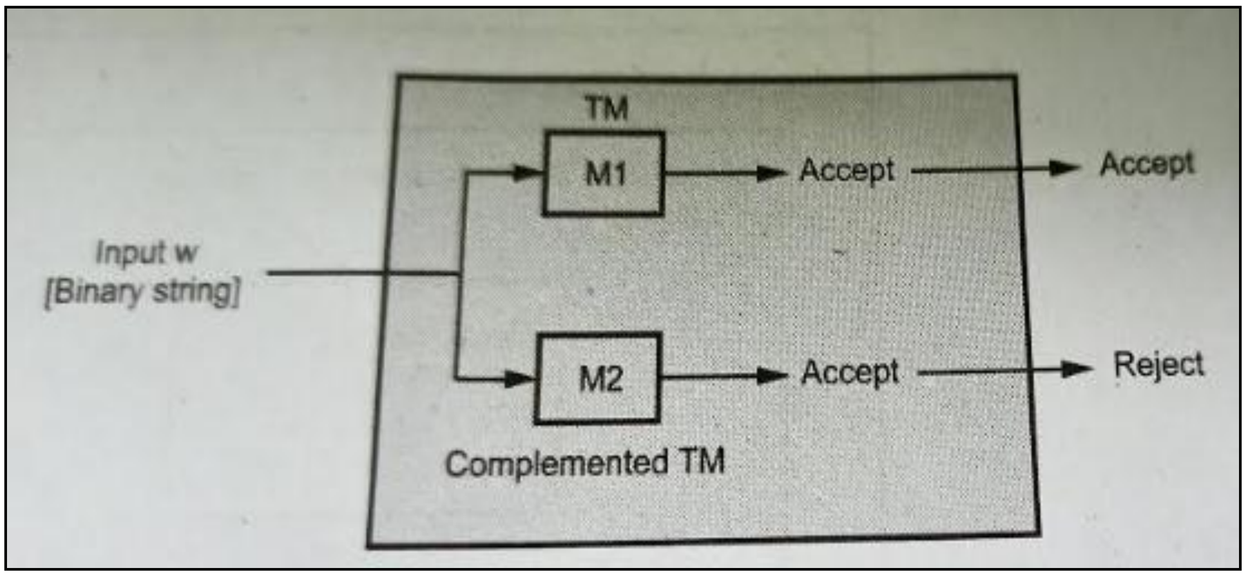
- **Steps**
  - Accepting states and Non accepting states of M’
  - Create new accepting states of M’
  - Make same combination of accepting states and input tape symbol for Machine M and M’



Theorem 2

If a language L and its complement L' both are RE then L is recursive Language.

Proof



Theorem 3

Show that if L1 and L2 are recursive languages then

L1 U L2 also recursive.

Proof

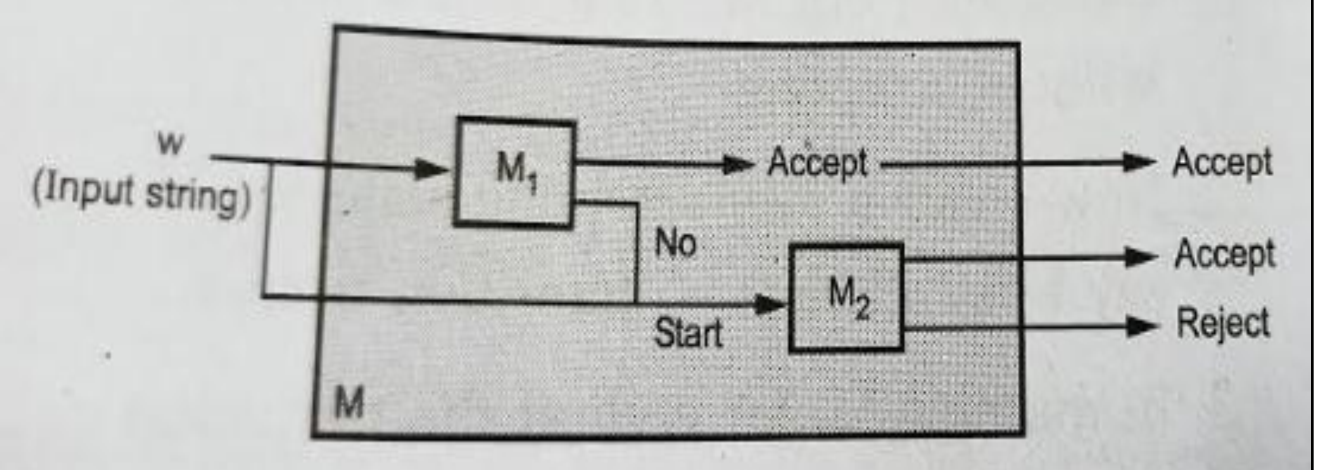
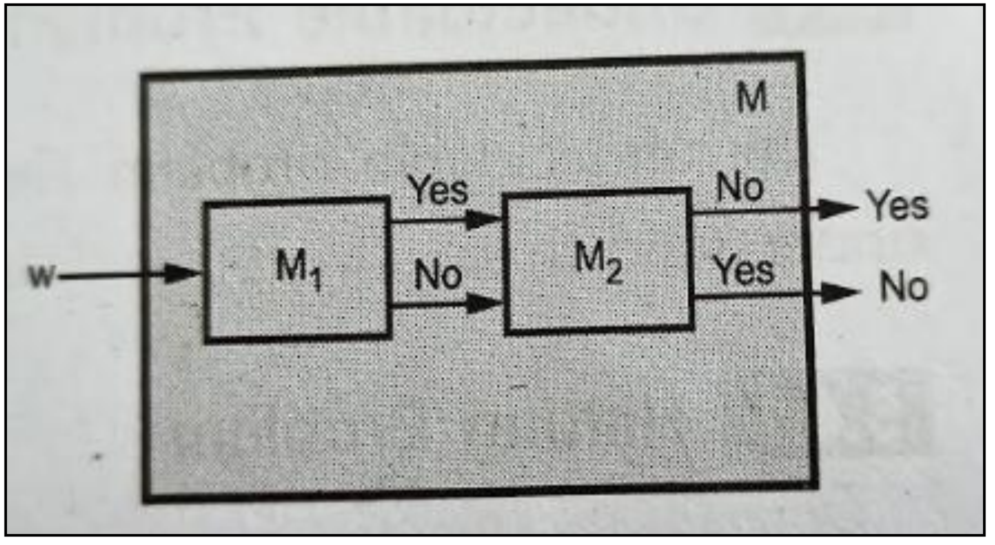


Fig. 5.2.4

Theorem 4

If L1 and L2 are two recursive languages and if L is defined as L={w or w is L1 not in L2 and not in L1}. Prove or disapprove that L is recursive.

Proof



Theorem 5

Show that the set of languages L over {0,1} so that neither L nor L’ is recursively enumerable in uncountable.

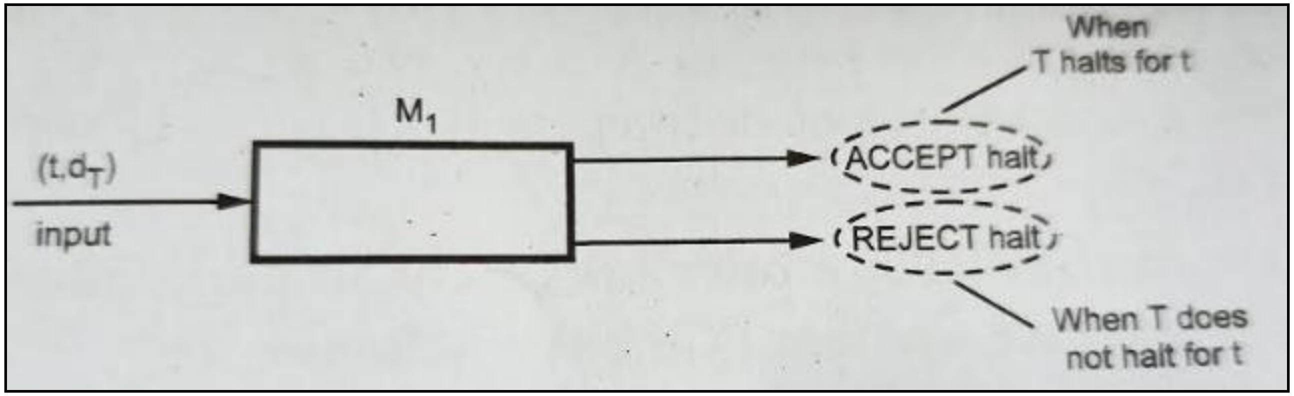
Proof

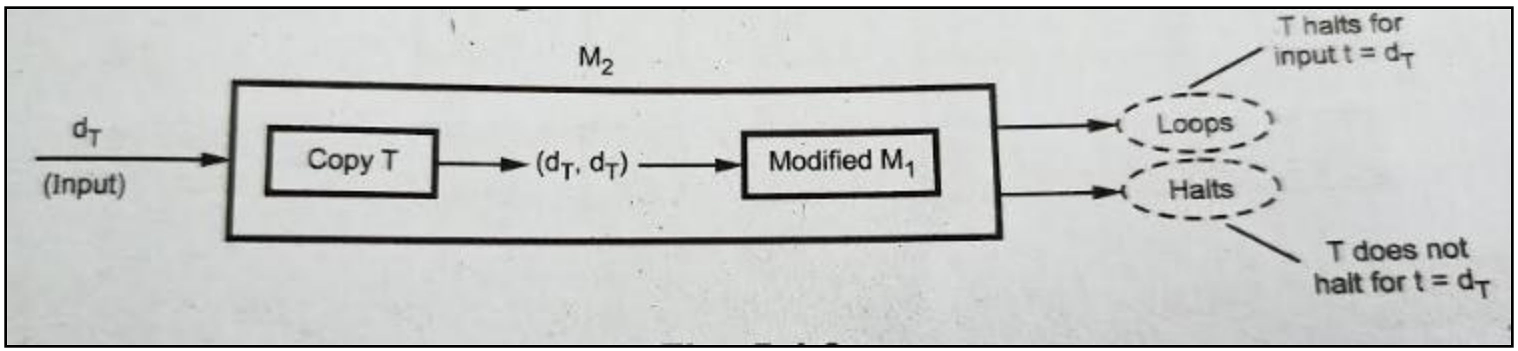
1. If L is recursively enumerable then there exists a TM Which accepts / halts / loops forever.

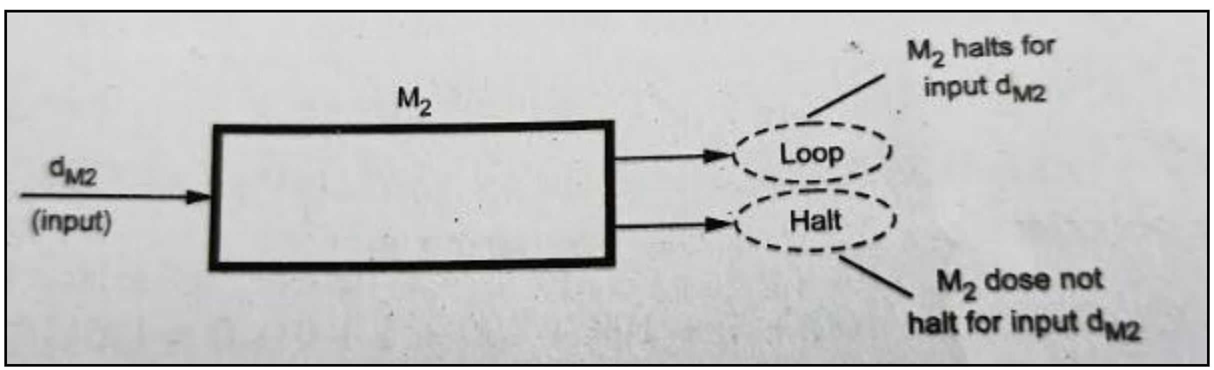
2. TM has finite length.

Undecidable Problem about TM

- **Halting Problem**
  - Halt (Halt after finite number of states)
  - No Halt (Never reaches the halt state, No matter how long it runs)
  - Unsolvable
  - Prove why it is unsolvable.


M1
(t,dT)
input
ACCEPT halt
REJECT halt
When
T halts for t
When T does
not halt for t


M2
dT
(Input)
Copy T
(dT, dT)
Modified M1
Loops
Halts
T halts for
input t = dT
T does not
halt for t = dT


M2
dM2
(input)
Loop
Halt
M2 halts for
input dM2
M2 dose not
halt for input dM2

Diagonalization Language (L_d)

L_d is the language in which the set of strings W_i such that W_i is not in $L(M_i)$.

i.e., input is not in the language of TM.

$L(M_i) = \phi$ if $W_i \not\in$ valid code of TM.

Diagonalization

The Process of complementing the diagonal to construct the characteristic vector of a language that cannot be the language that appears in any row is called **Diagonalization.**

L_d is not Recursively Enumerable

i.e., There is a no TM that accepts **L_d**

$L(M_i)$ = { W_i | M_i does not accept W_i }

2 Possibilities

(i,j)=0 No **$W_i \Sigma$** L

(i,j)= 1 Yes **$W_i \not\Sigma L_d$**

Diagonal String = 0111

Complement = 1000

An Undecidable Problem that is Recursively Enumerable (RE)

Statement – Recall in Slide 7,8.

Recursive Language

Relationship between Recursive , RE & Non RE languages.

Universal Languages (L_u)

Recall the Definition of L_d

$$L_d = \{ W_i \mid W_i \text{ does not accept } L(M_i) \}$$

Universal Languages (L_u)

It is the set of strings representing a TM and an input is **accepted by** the **TM**. Therefore TM **U** is called as **Universal Language**.

$$L_u = \{ (M, w) \mid M \text{ accepts } w \}$$

Undecidable of Universal Language

W→ **TM M** - Yes → Yes

No → No

Undecidable Problems about TM

- Reductions
- Turing Machine that Accept the Empty Language

$Le = \{ M \mid L(M) = \phi \}$

$Lne = \{ M \mid L(M) \neq \phi \}$

Rice Theorem & Properties of RE Languages

Trivial Property

A Property is trivial if it is either empty such that if it is satisfied by **no language (or) all RE language**.

Non Trivial Property

A Property is said to be Non – Trivial if it is not empty.

All Nontrivial properties of the RE language are undecidable.

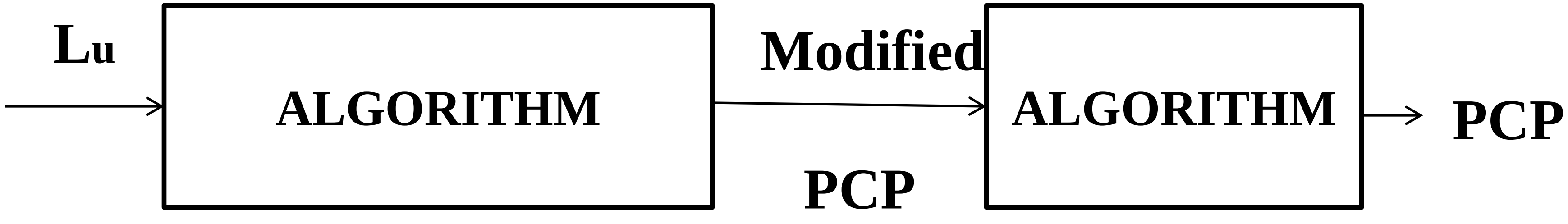
i.e., Not possible to recognize the property by a Turing Machine

i) **Context free -** Set of all CFL's

ii) **Empty** - Set consists of empty language.

POST CORRESPONDENCE PROBLEM (PCP)

- PCP involves **strings rather than TM**.
- **To Prove**
  - Strings to be Undecidable.
  - Use Undecidability concept to prove other problems undecidable by reducing PCP.



Definition - PCP

PCP consists of two lists of string over Σ .

A = w1,w2,…….wk

B= x1,x2…….xk.

For some integer k.

wi1,wi2,wi3……wim = x1,x2….xk.

is a solution to this instance of PCP.

Example

	A	B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

This PCP instance has a solution: 2, 1, 1, 3:

$w_2w_1w_1w_3 = x_2x_1x_1x_3 = 101111110$

Does this PCP instance have a solution?

	A	B
i	w_i	x_i
1	110	110110
2	0011	00
3	0110	110

This PCP instance has a solution: 2,3,1

$w_2w_3w_1 = x_2x_3x_1 = 00110110110$

One more solution:

2,1,1,3,2,1,1,3

Modified PCP

An Intermediation (or) intermediate version of PCP is **Modified PCP.**

An instance of MPCP has two lists.

A=w1,w2……wk.

B=x1,x2…….xk.

Solution

w1wi1wi2…..wm = **x1**xi1xi2…….xim

Where w1 x1 – Beginning of two strings

Modified Post Correspondence Problem (MPCP)

	List A	List B
i	w_i	x_i
1	10	10
2	110	11
3	11	011

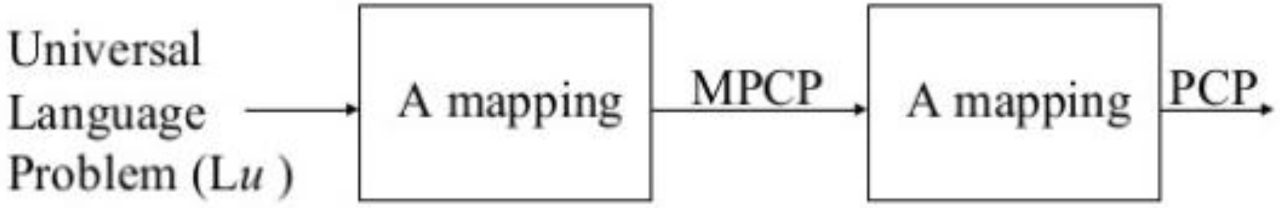
This MPCP instance has a solution: 1,2,3

$w_1 w_2 w_3 = x_1 x_2 x_3$

10 110 11 = 10 11 011

Undecidability of PCP

To show that PCP is undecidable, we will reduce the universal language problem (L*u*) to MPCP and then to PCP:



If PCP can be solved, L*u* can also be solved. L*u* is undecidable, so PCP must also be undecidable.

Reducing MPCP to PCP

- This can be done by inserting a special symbol (*) to the strings in list A and B of to make sure that the first pair will always go first in any solution.
- List A : * follows the symbols of ∑
- List B : * precedes the symbols of ∑
- w_{k+1} = $; x_{k+1} = *$

Mapping MPCP to PCP

Suppose the original MPCP instance is:

	A	B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Mapping MPCP to PCP

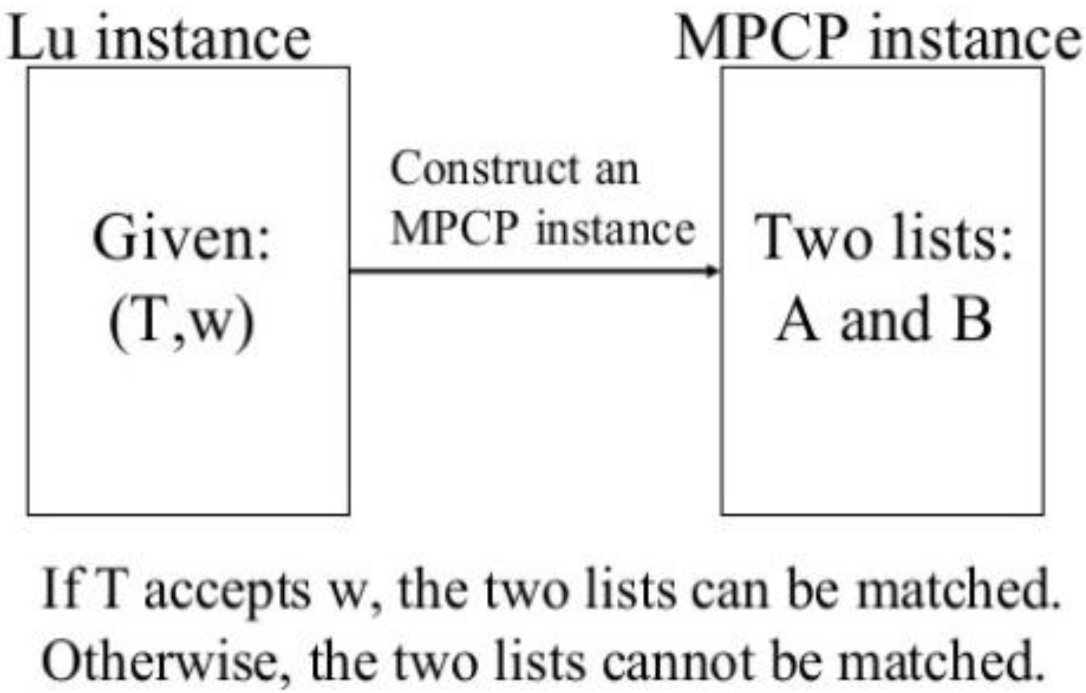
The mapped PCP instance will be:

	A	B
i	w_i	x_i
0	*1*	*1*1*1
1	1*	*1*1*1
2	1*0*1*1*1*	*1*0
3	1*0*	*0
4	$	*$

Mapping Lu to MPCP

- Turing machine M and an input w, we want to determine if M will accept w.
- the mapped MPCP instance should have a solution if and only if M accepts w.

Mapping L*u* to MPCP



Rules of Reducing L*u* to MPCP

- We summarize the mapping as follows. Given T and w, there are five types of strings in list A and B:
- Starting string (first pair):

List A	List B
#	$\#q_0w\#$

where q_0 is the starting state of T.

- Strings for copying:

List A	List B
X	X
#	#

where X is any tape symbol (including the blank).

is a separator can be appended to both the lists

- Strings from the transition function δ:

List A	List B	
qX	Yp	from $\delta(q,X)=(p,Y,R)$
ZqX	pZY	from $\delta(q,X)=(p,Y,L)$
q#	Yp#	from $\delta(q,\#)=(p,Y,R)$
Zq#	pZY#	from $\delta(q,\#)=(p,Y,L)$

where <u>Z is any tape symbol except the blank</u>.

- Ending string:

List A	List B
q##	#

where q is an accepting state.

- Using this mapping, we can show that the original L*u* instance has a solution if and only if the mapped MPCP instance has a solution.

PCP is undecidable

- Theorm: Post's Correspondence Problem is undecidable.
- We have seen the reduction of MPCP to PCP
- now we see how to reduce Lu to MPCP.
  - M accepts w if and only if the constructed MPCP instance has a solution.
  - As Lu is undecidable, MPCP is also undecidable.

CLASS P & NP Problems

- Problems solved by Polynomial Time.
- Dividing line between problems.

Intractable Problems

- Problems not to be solved in polynomial time.

Example - CLASS P Problems

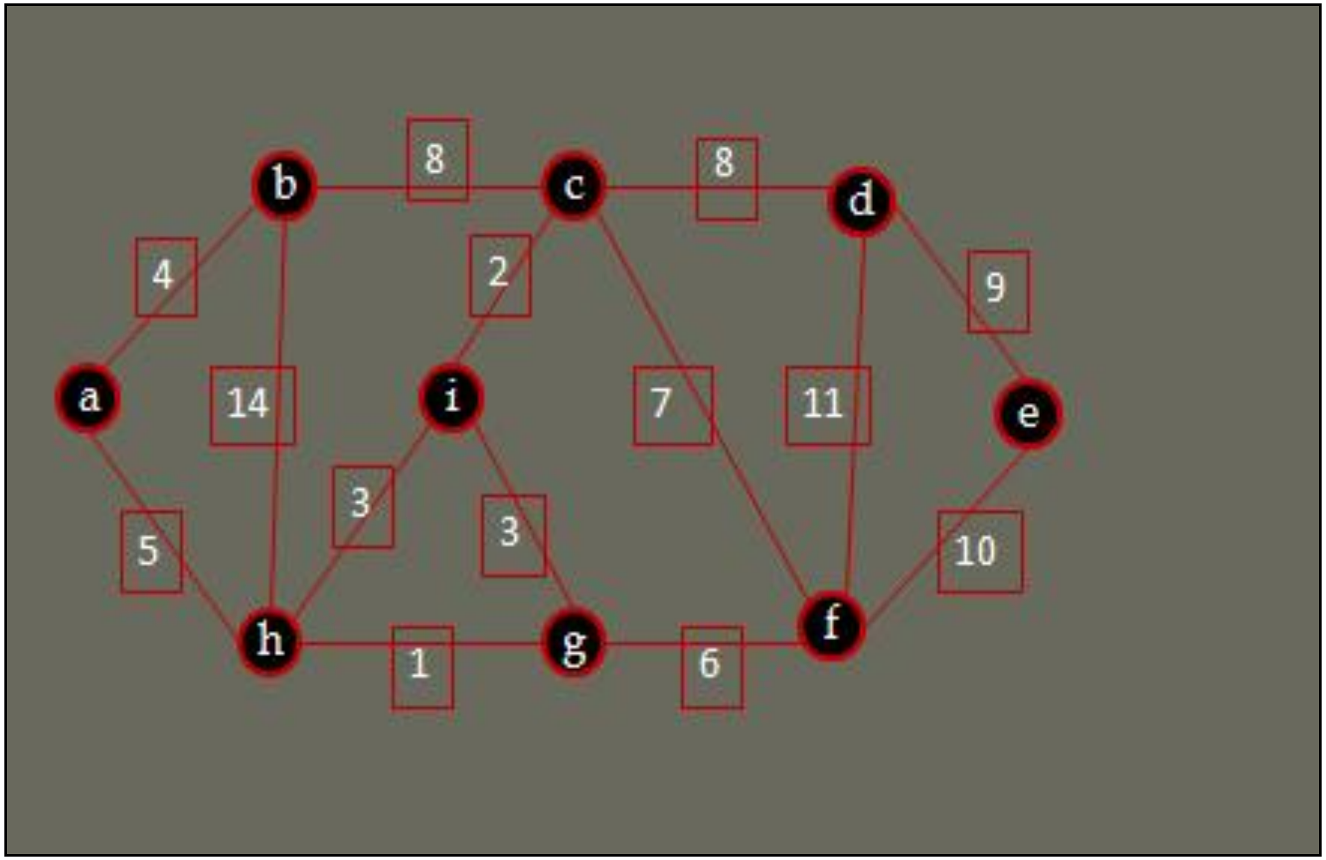
- **Kruskal's Algorithm**

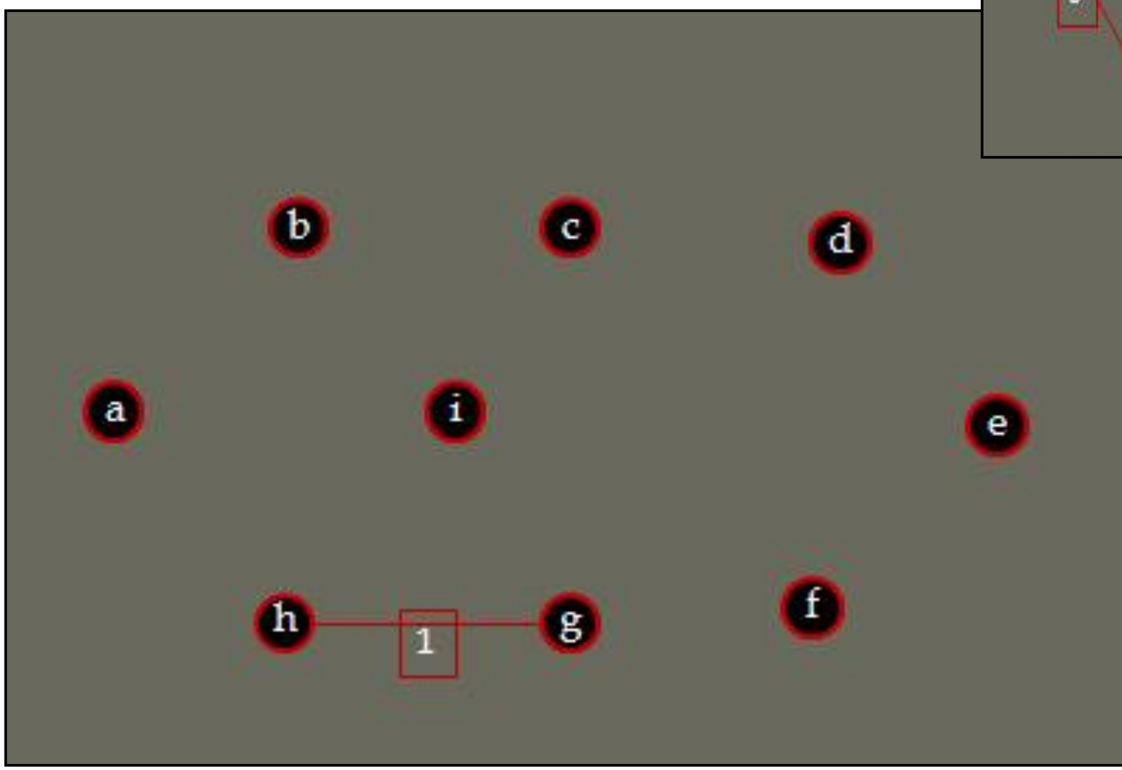
  Minimum Weight Spanning Tree

Example - CLASS NP Problems

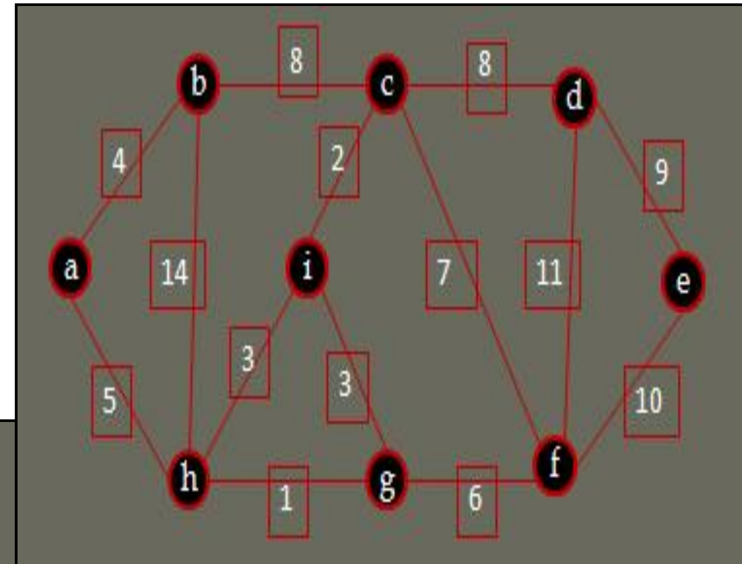
- **Travelling Salesman Problem**

Kruskal's Algorithm

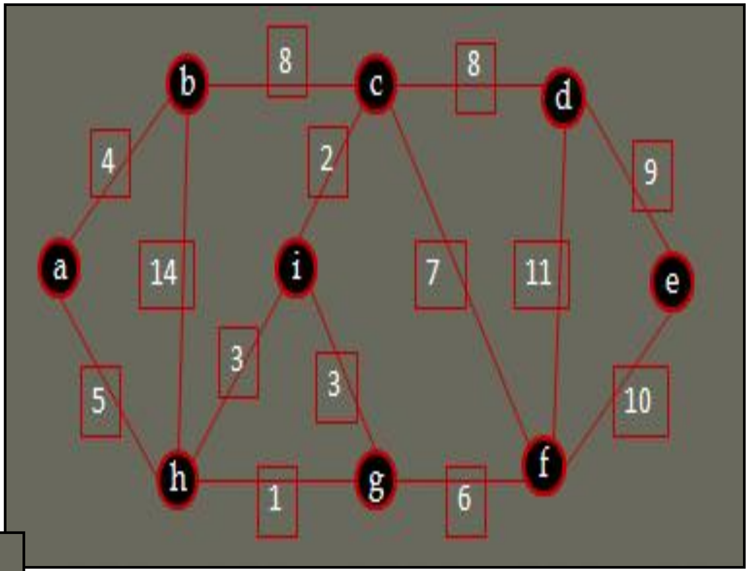




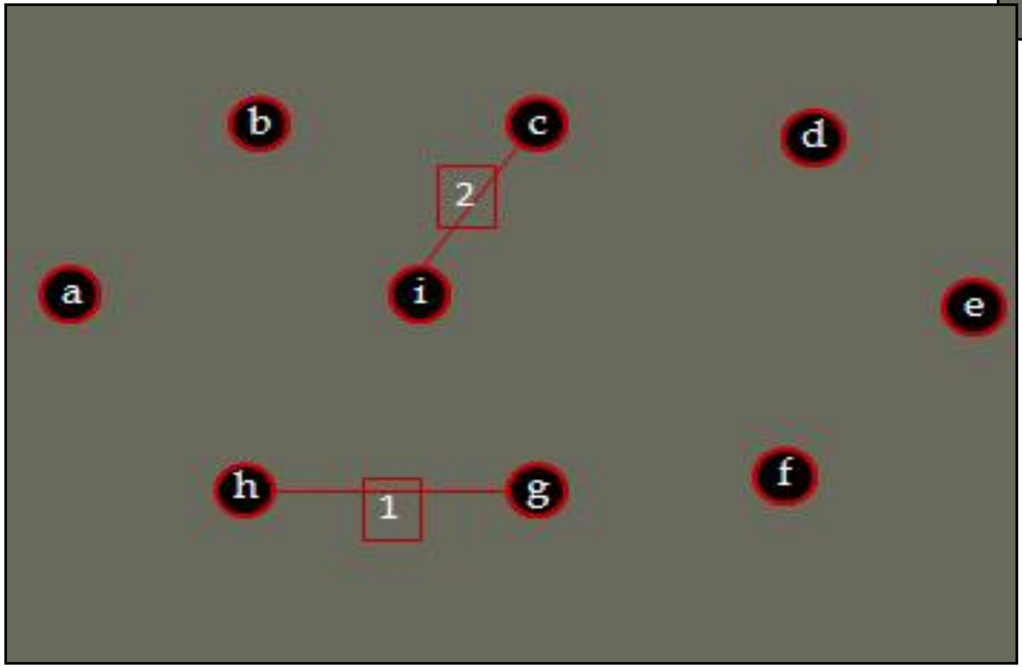
b
c
d
a
i
e
h
1
g
f



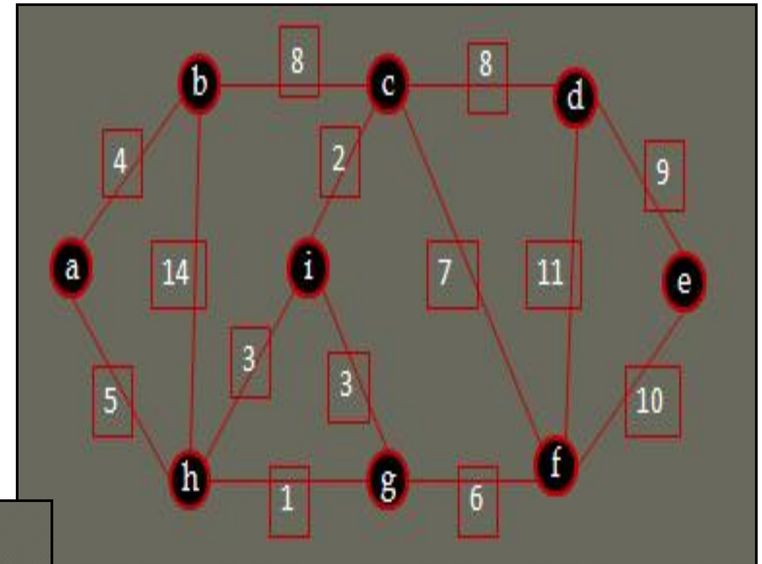
b
8
c
8
d
4
2
9
a
14
i
7
11
e
3
3
5
10
h
1
g
6
f

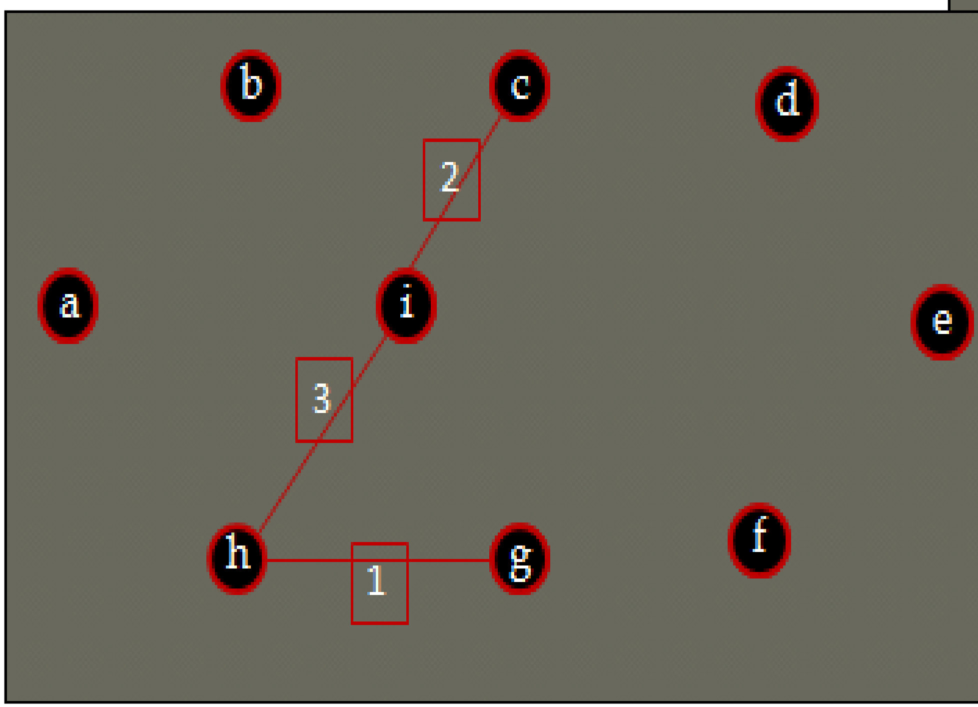


b
8
c
8
d
4
2
9
a
14
i
7
11
e
3
3
5
10
h
1
g
6
f

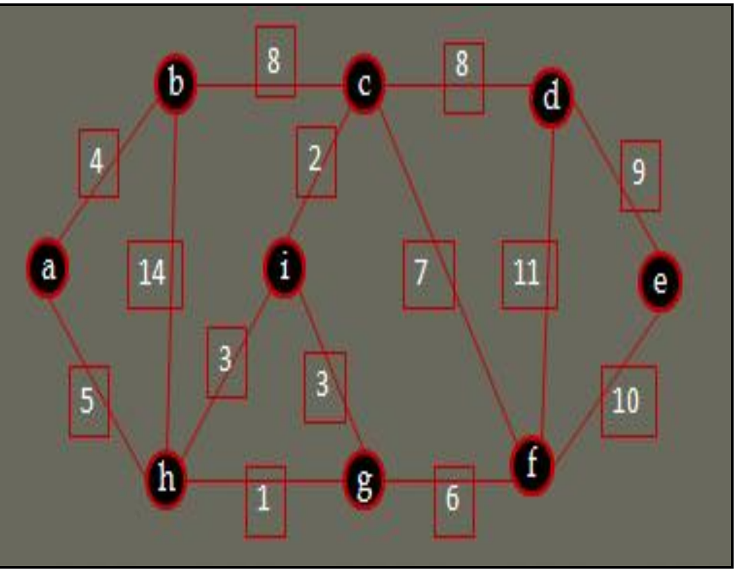


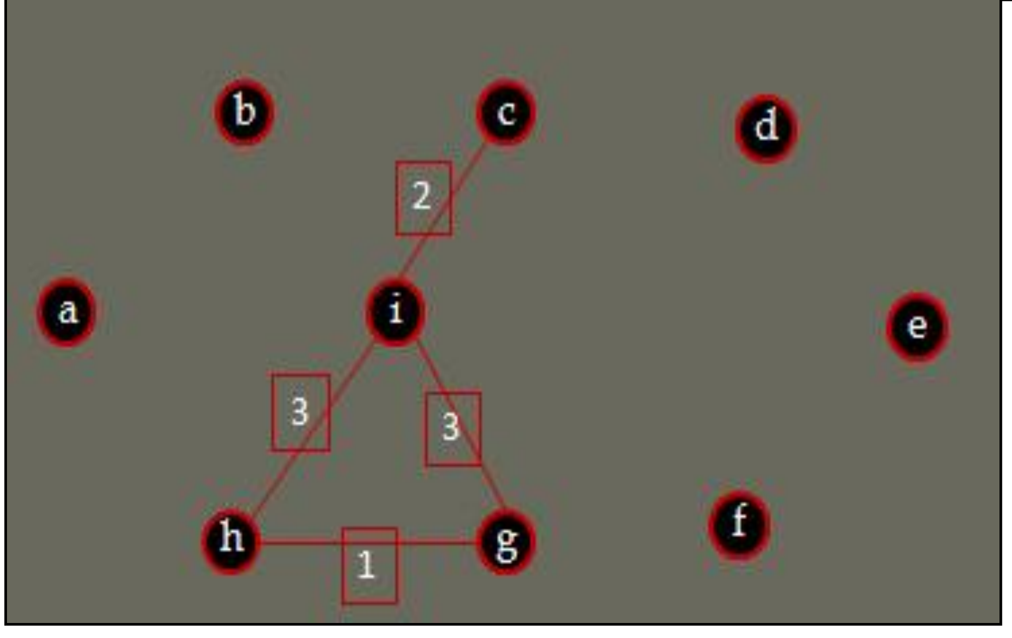
b
c
d
2
a
i
e
h
1
g
f

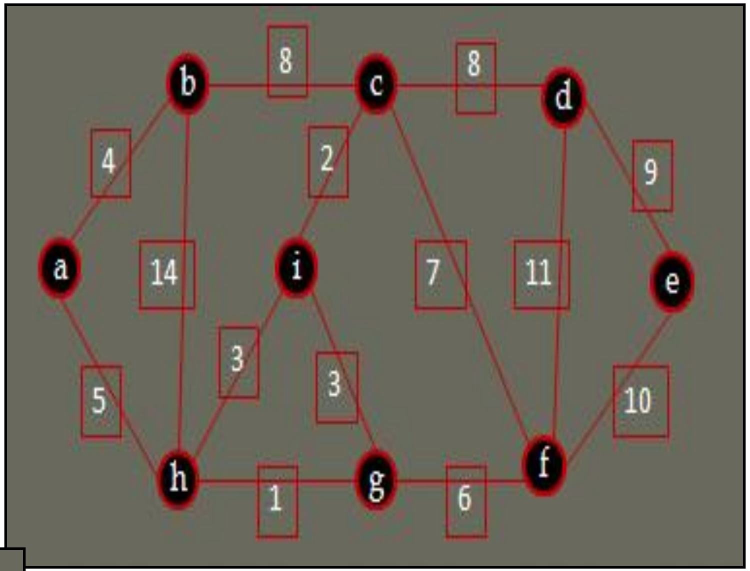

8
8
b
c
d
4
2
9
a
14
i
7
11
e
3
3
5
10
h
g
f
1
6

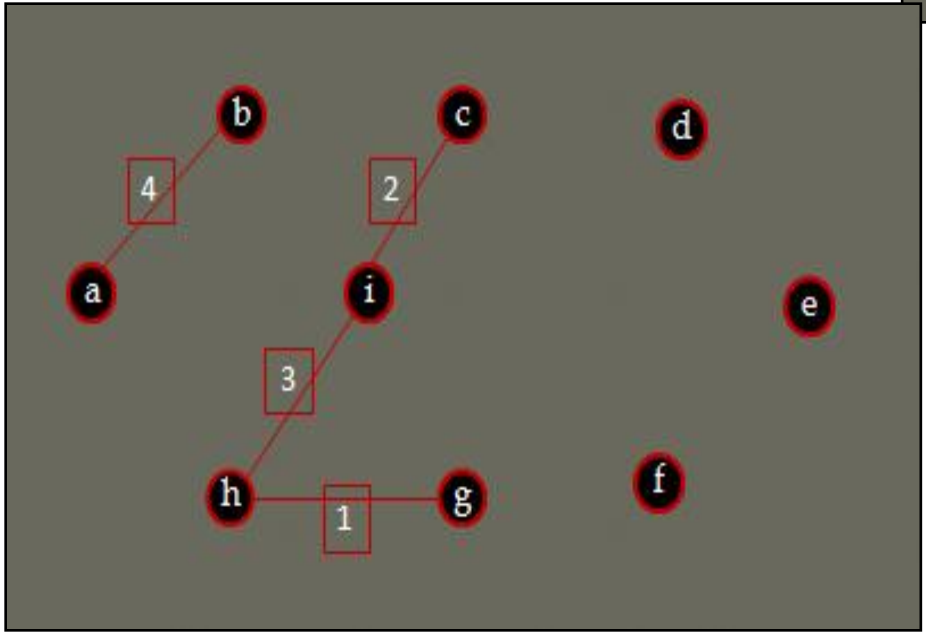

b
c
d
2
a
i
e
3
h
g
f
1

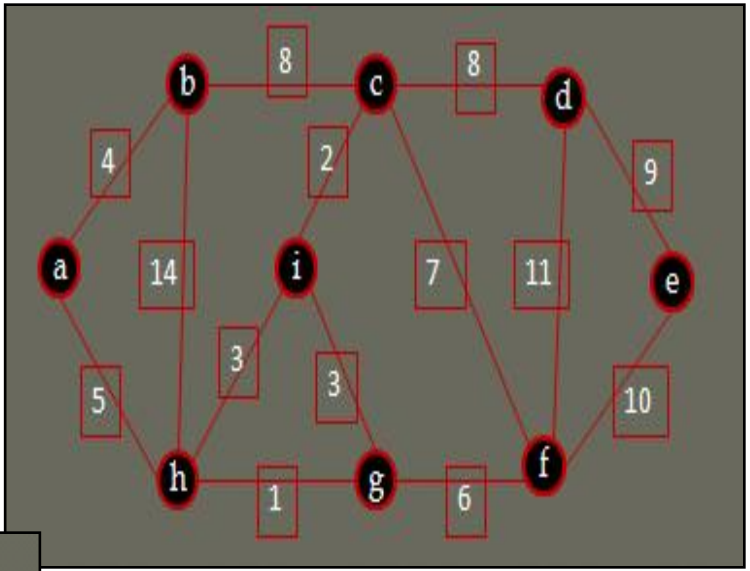
- **Reject (i, g)**
  - forms a cycle

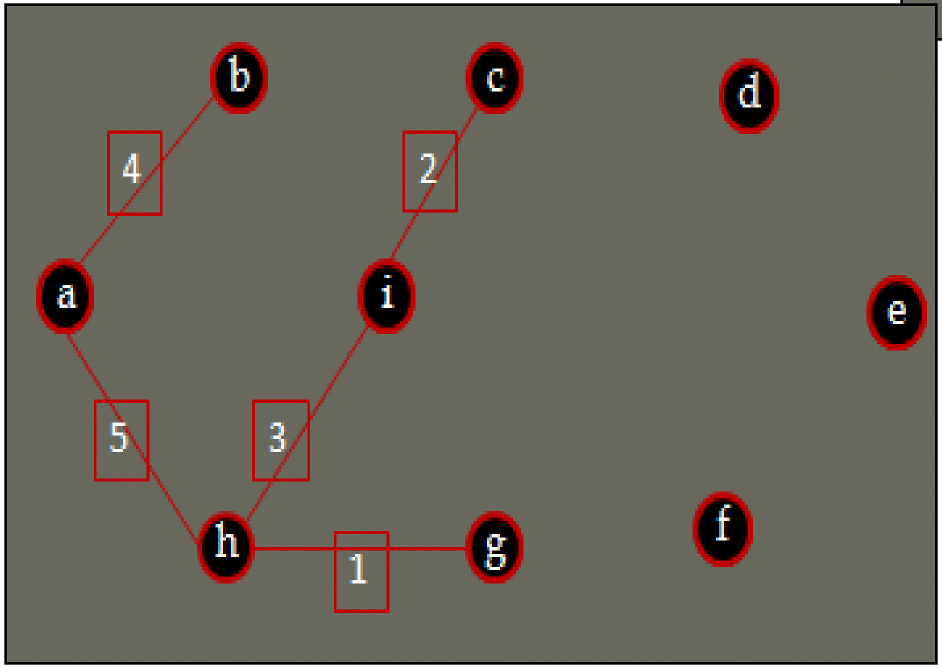


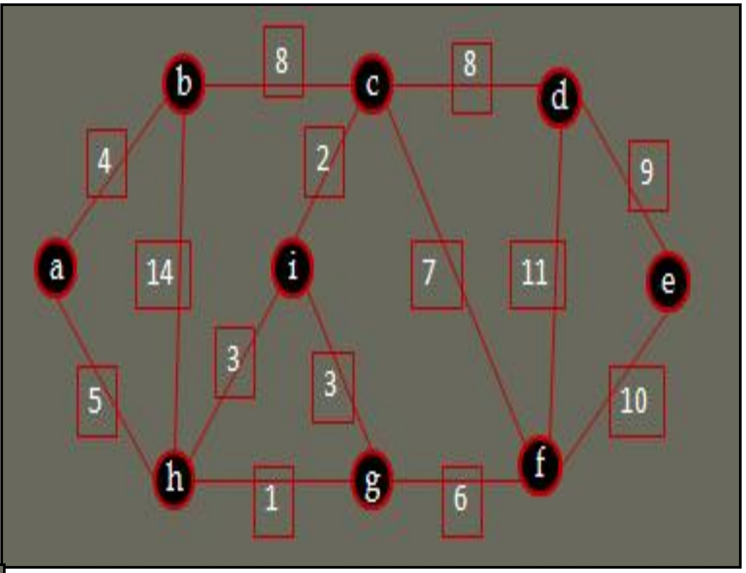


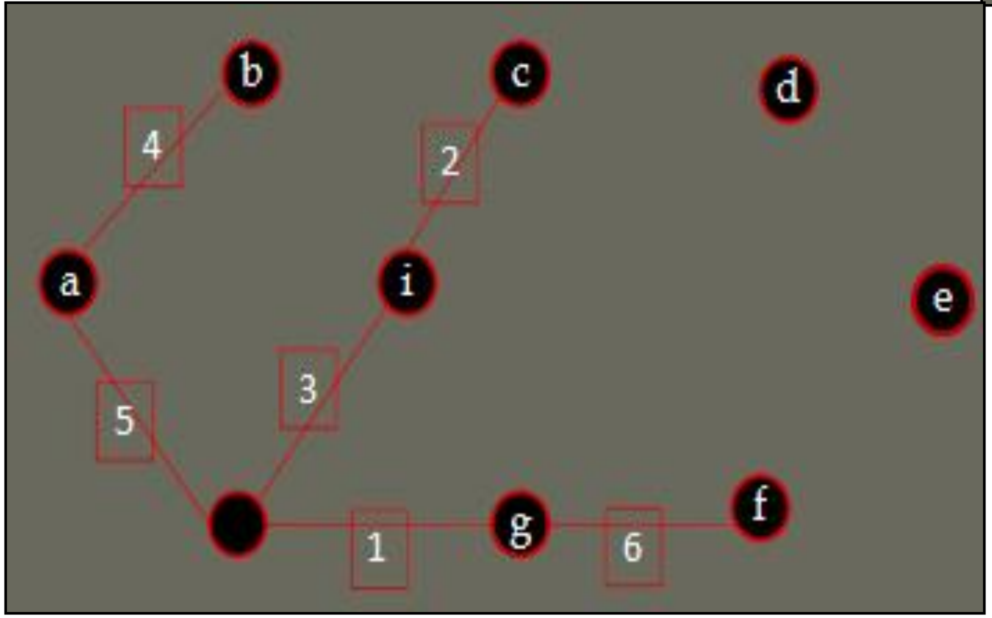

b
8
c
8
d
4
2
9
a
14
i
7
11
e
3
3
5
10
h
1
g
6
f


b
c
d
4
2
a
i
e
3
h
1
g
f

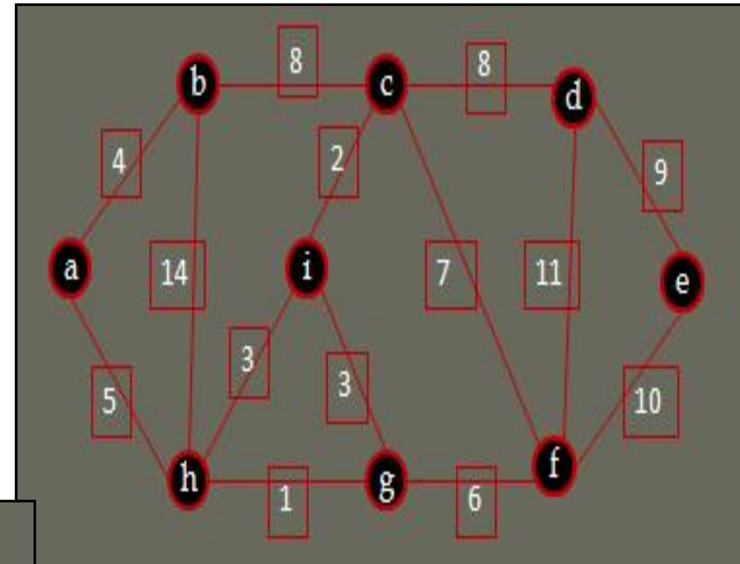

b
c
d
a
i
e
f
h
g
8
8
4
2
9
14
7
11
3
3
5
10
1
6

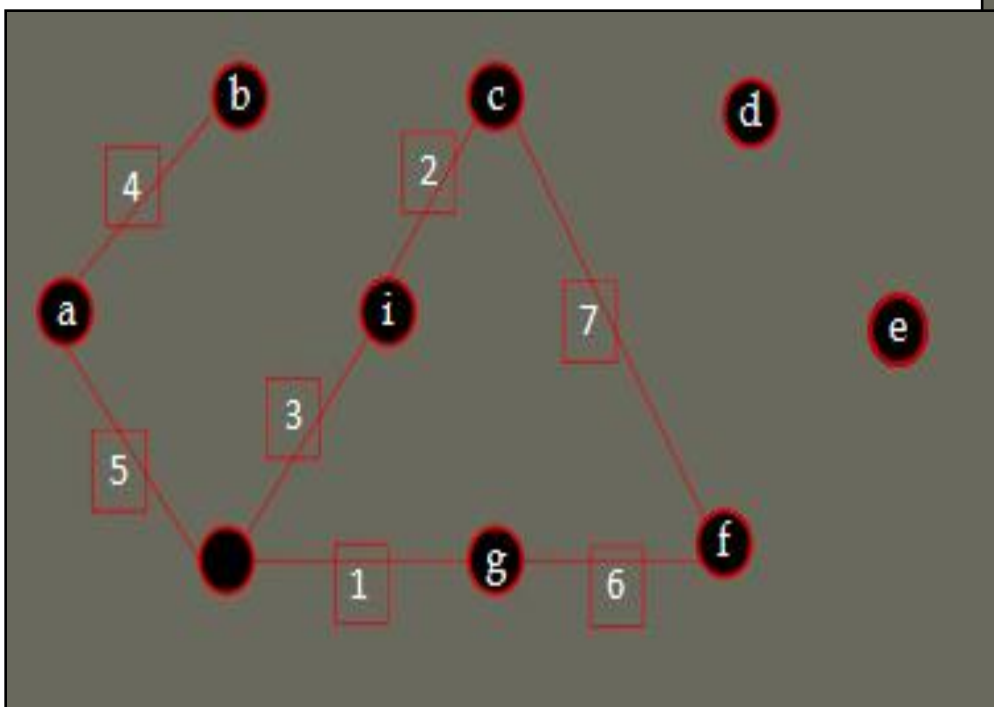

b
c
d
a
i
e
h
g
f
4
2
5
3
1

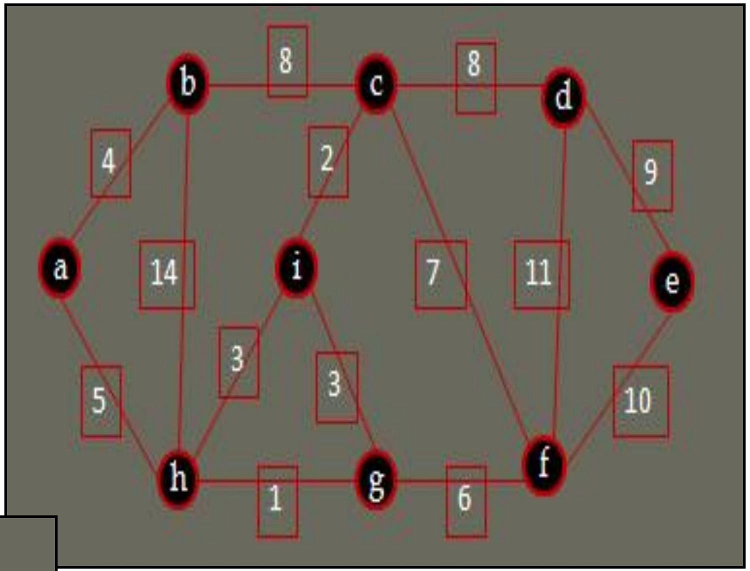

b
8
c
8
d
4
2
9
a
14
i
7
11
e
3
3
5
10
h
1
g
6
f


b
c
d
4
2
a
i
e
3
5
1
g
6
f

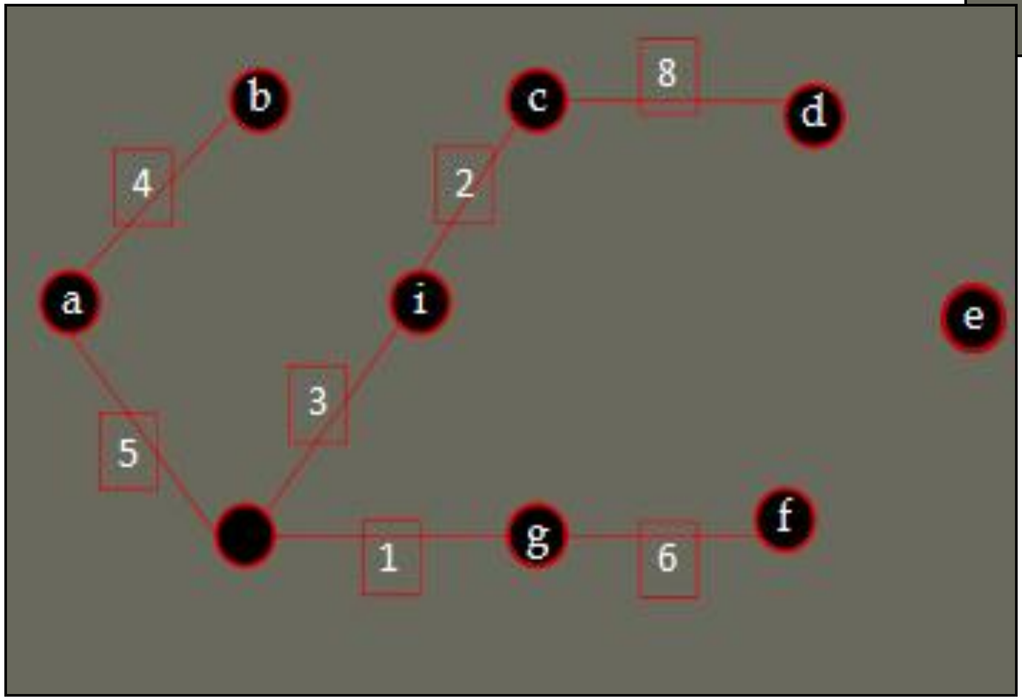
- **Reject (c , f)**
  – forms a cycle





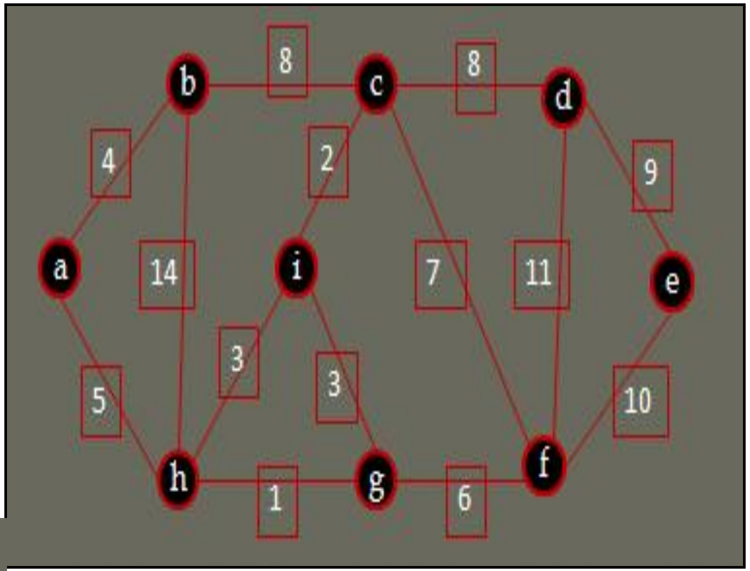


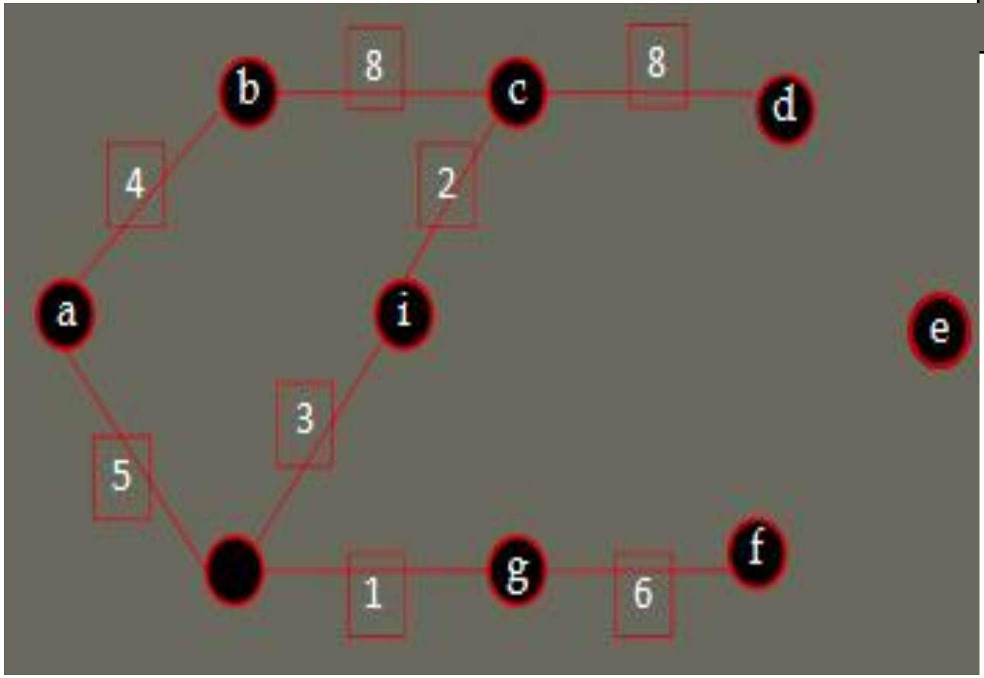
b
8
c
8
d
4
2
9
a
14
i
7
11
e
3
3
5
10
h
1
g
6
f

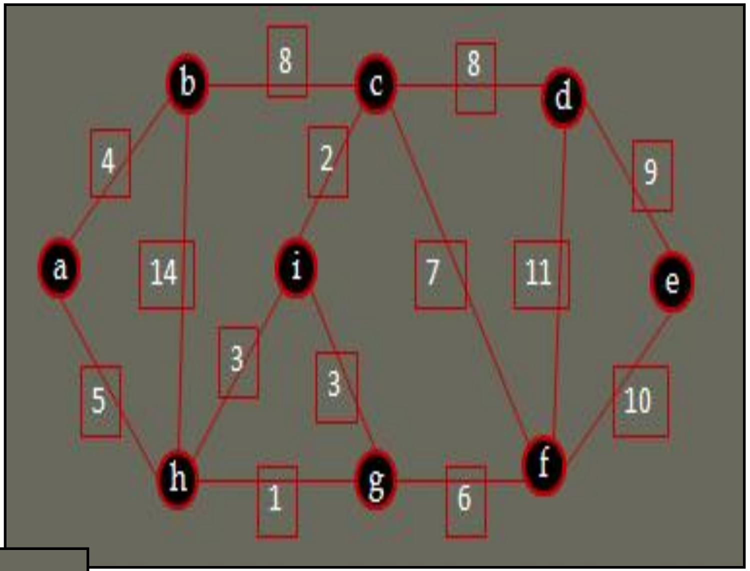


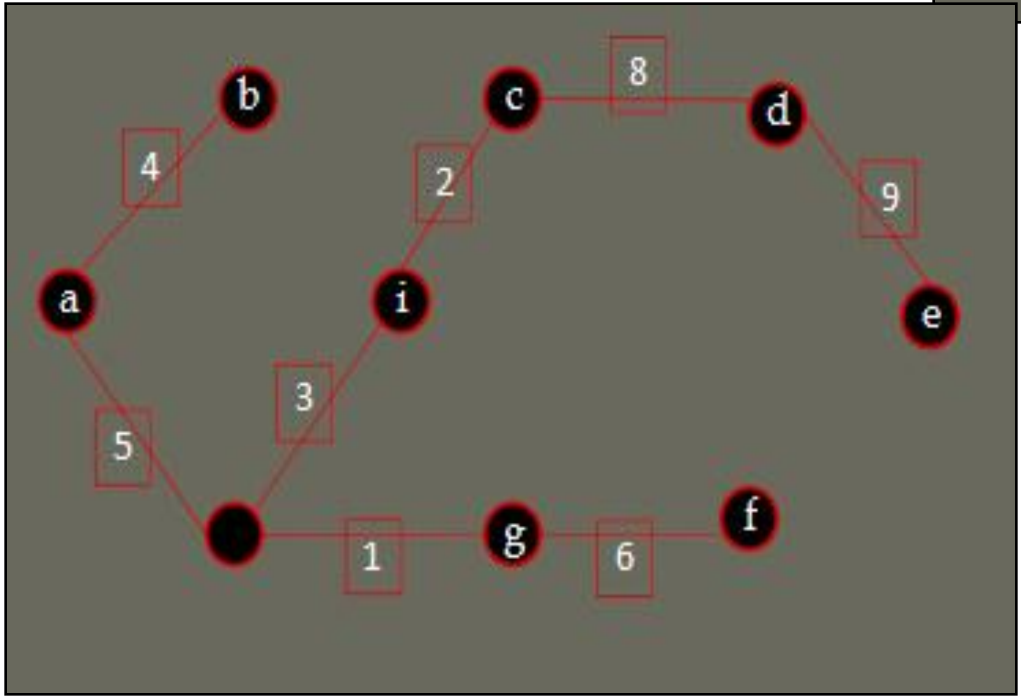
b
c
8
d
4
2
a
i
e
3
5
1
g
6
f

- **Reject (b , c)**
  - forms a cycle

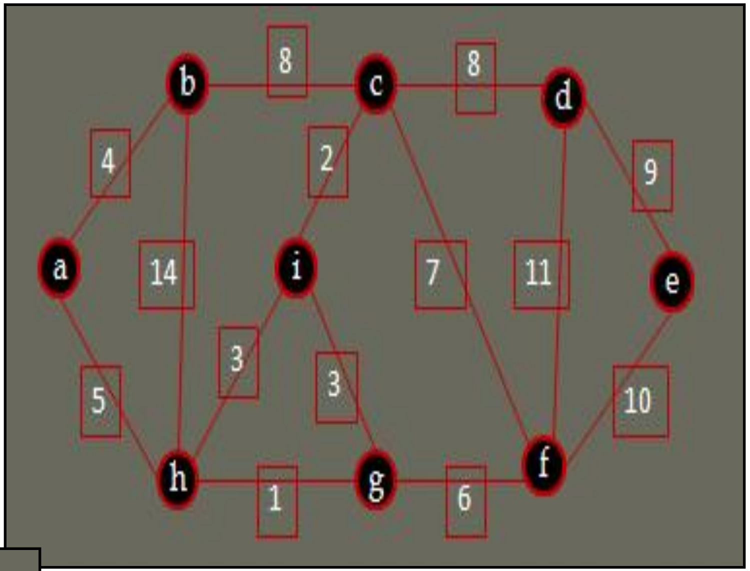


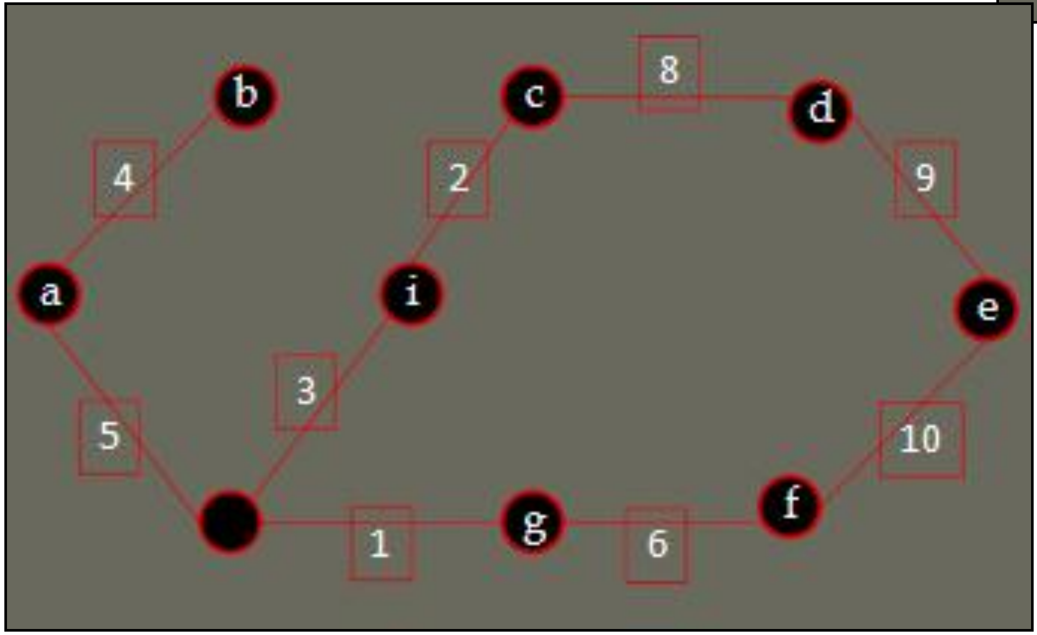



b
c
d
a
i
e
h
g
f
8
8
4
2
9
14
7
11
3
3
5
10
1
6

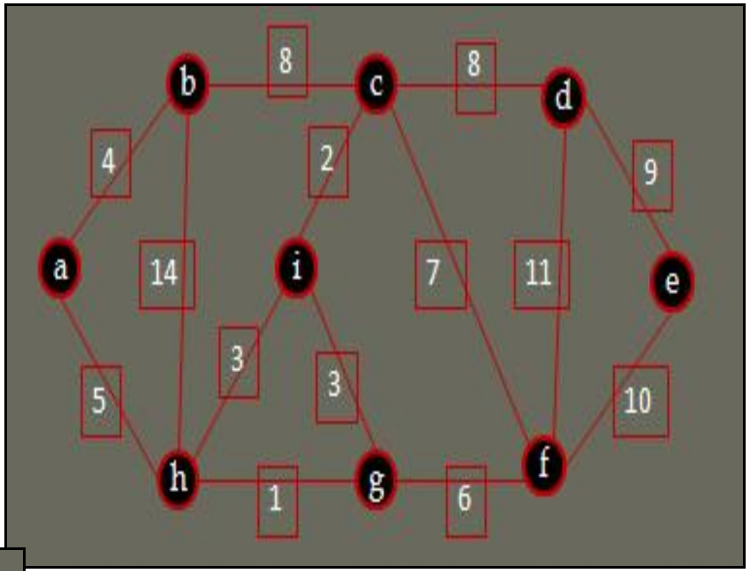

b
c
d
a
i
e
g
f
8
4
2
9
3
5
1
6

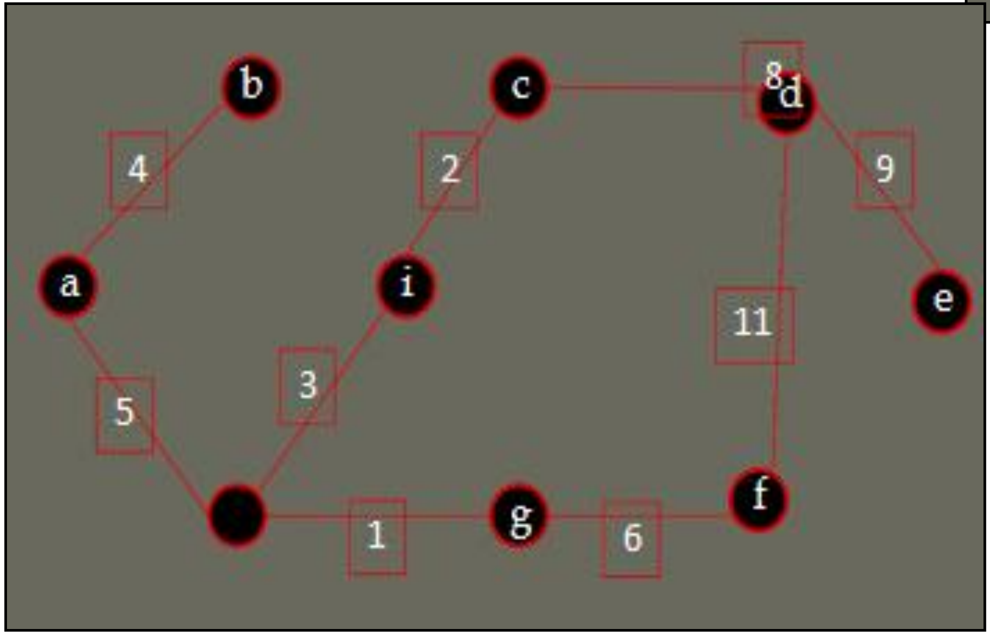
- **Reject (e , f)**
  - forms a cycle



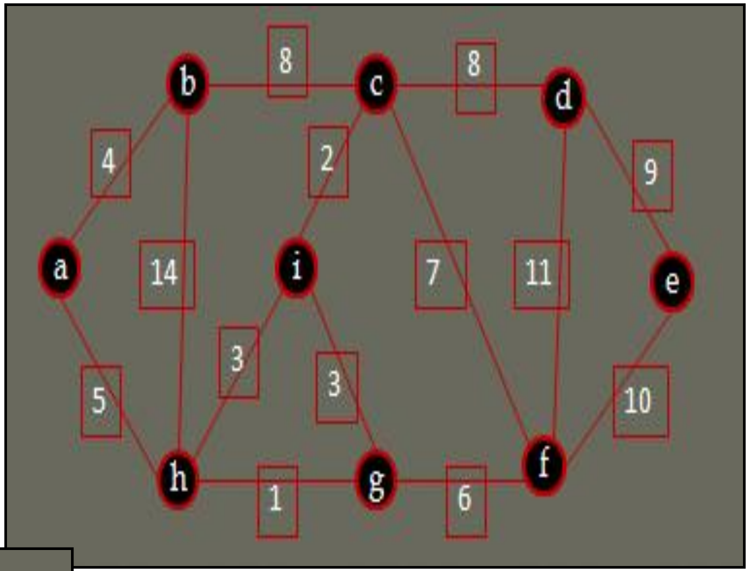


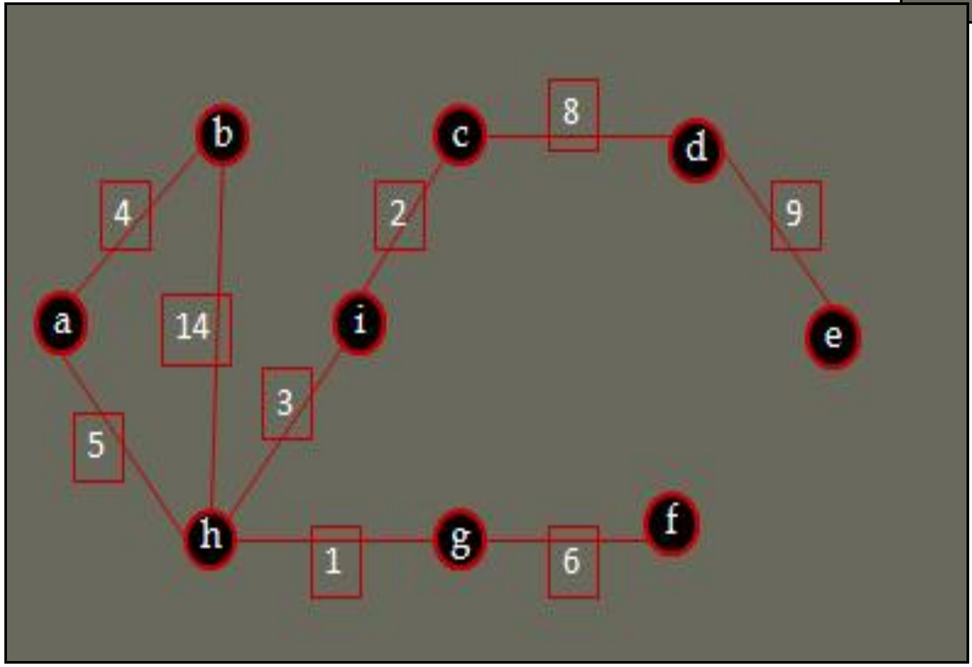
- **Reject (d , f)**
  - – forms a cycle





- **Reject (b , h)**
  - – forms a cycle

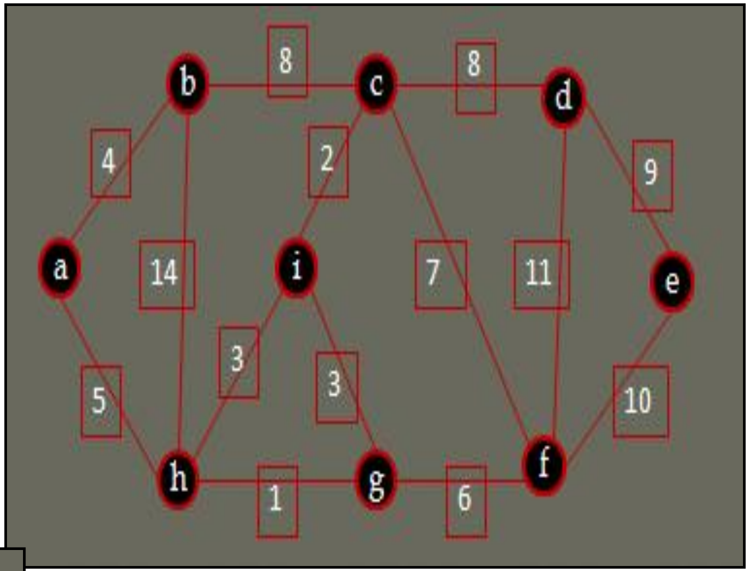


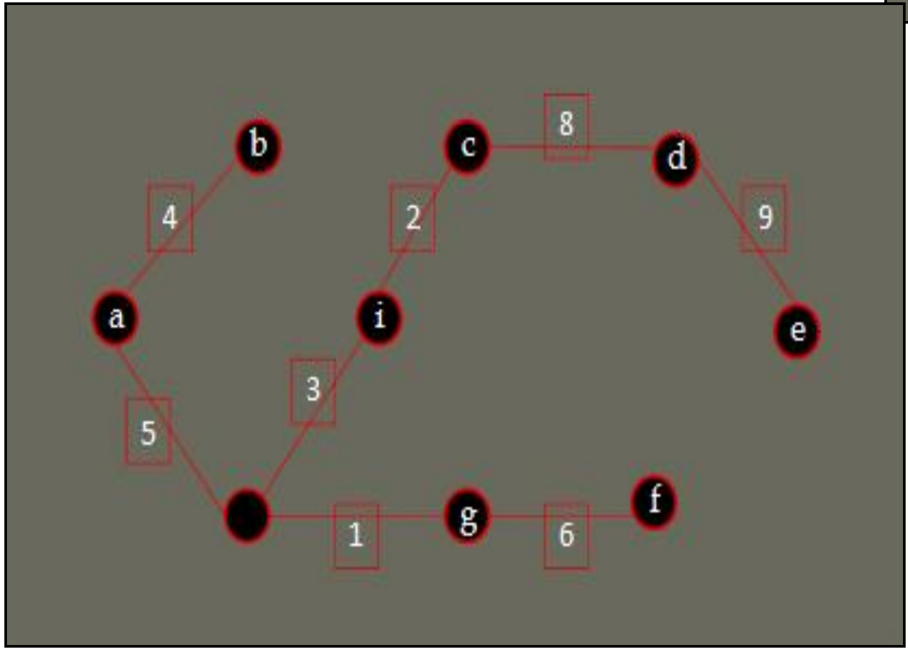


Answer

Minimum Spanning Tree

Total weight(MST)=1+2+3+4+5+6+8+9=**38**





CLASS P & NP Problems

- Problems solved by Polynomial Time.
- Dividing line between problems.

Tractable Problems

- Problems can be solved in reasonable time and space. Ex:- Sorting

Intractable Problems

- Problems not to be solved in polynomial time.Ex:- Class p and np problems

Example - CLASS P Problems

- **Kruskal's Algorithm**

  Minimum Weight Spanning Tree

Example - CLASS NP Problems

- **Travelling Salesman Problem**

Travelling Salesman Problem