1.Basic Operations of fuzzy set:

i)Union

ii)Intersection

iii)Complement

iv)Difference

1.i)Union:

Introduction: Consider 2 Fuzzy Sets denoted by A and B, then let's consider result be the Union of them, then for every member of A and B, result will be:

degree_of_membership(result)= max(degree_of_membership(A), degree_of_membership(B)).

The First Fuzzy Set is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

The Second Fuzzy Set is : {'x1': 0.4, 'x2': 0.2, 'x3': 0.1}

Fuzzy Set Union is : {'x1': 0.4, 'x2': 0.5, 'x3': 0.8}

Pythone code:

```
import numpy as np
A = dict()

B = dict()
result=dict()

n = int(input("enter no.of elements of set A:"))
A = {}

for i in range(n):
keys = input()

values = float(input())
A[keys] = values

n1 = int(input("enter no.of elements of set B:"))
B = {}

for i in range(n1):
keys1 = input()

values1 = float(input())
B[keys1] = values1
```

```
print('The First Fuzzy Set is :', A)
print('The Second Fuzzy Set is :', B)
for A_key, B_key in zip(A, B):

A_value = A[A_key]
B_value = B[B_key]
if A_value > B_value:

result[A_key] = A_value
else:

result[B_key] = B_value
print('Fuzzy Set Union is :', result)
```

Output:

enter no.of elements of set A:3

x1

0.2

x2

0.5

x3

0.8

enter no.of elements of set B:3

x1

0.4

x2

0.2

x3

0.1

The First Fuzzy Set is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

The Second Fuzzy Set is : {'x1': 0.4, 'x2': 0.2, 'x3': 0.1}

Fuzzy Set Union is : {'x1': 0.4, 'x2': 0.5, 'x3': 0.8}

Conclusion: It allows the combination of two or more fuzzy sets to create a new fuzzy set that captures the degree of membership of each element in at least one of the original sets.

1.ii)Intersection:

Introduction: Consider 2 Fuzzy Sets denoted by A and B, then let's consider result be the Intersection of them, then for every member of A and B, result will be:

degree_of_membership(result)= min(degree_of_membership(A), degree_of_membership(B))

The First Fuzzy Set is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

The Second Fuzzy Set is : {'x1': 0.4, 'x2': 0.2, 'x3': 0.1}

Fuzzy Set Intersection is : {'x1': 0.2, 'x2': 0.2, 'x3': 0.1}

Python Code:

```
import numpy as np
A = dict()
B = dict()
result=dict()
n = int(input("enter no.of elements of set A:"))
A = {}
for i in range(n):
 keys = input()
 values = float(input())
 A[keys] = values
n1 = int(input("enter no.of elements of set B:"))
B = {}
for i in range(n1):
 keys1 = input()
 values1 = float(input())
 B[keys1] = values1
print('The First Fuzzy Set is :', A)
print('The Second Fuzzy Set is :', B)
for A_key, B_key in zip(A, B):
```

```
 A_value = A[A_key]
 B_value = B[B_key]
 if A_value < B_value:
   result[A_key] = A_value
 else:
   result[B_key] = B_value
print('Fuzzy Set Intersection is :', result)
```

Output:

```
enter no.of elements of set A:3
x1
0.2
x2
0.5
x3
0.8
enter no.of elements of set B:3
x1
0.4
x2
0.2
x3
0.1
The First Fuzzy Set is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}
The Second Fuzzy Set is : {'x1': 0.4, 'x2': 0.2, 'x3': 0.1}
Fuzzy Set Intersection is : {'x1': 0.2, 'x2': 0.2, 'x3': 0.1}
```

Conclusion: It allows the common element between of two or more fuzzy sets to create a new fuzzy set .

1.iii)Complement:

introduction: Consider a Fuzzy Sets denoted by A , then let's consider result be the Complement of it, then for every member of A , result will be:

degree_of_membership(result)= 1 - degree_of_membership(A)

The First Fuzzy Set is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

Fuzzy Set Complement is : {'x1': 0.8, 'x2': 0.5, 'x3': 0.2 }

Python Code:

```
import numpy as np

A = dict()

result=dict()

n = int(input("enter no.of elements of set A:"))

A = {}

for i in range(n):

 keys = input()

 values = float(input())

 A[keys] = values

print('The First Fuzzy Set is :', A)


for A_key in A:

   result[A_key]= 1-A[A_key]


print('Fuzzy Set Complement is :', result)
```

Output:

enter no.of elements of set A:3

x1

0.2

x2

0.5

x3

0.8

The First Fuzzy Set is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

Fuzzy Set Complement is : {'x1': 0.8, 'x2': 0.5, 'x3': 0.2 }

Conclusion: the fuzzy complement operation is a powerful tool in fuzzy set theory that enables us to handle situations where the distinction between membership and non-membership is not clear-cut.

1.iv)Difference:

Introduction: Consider 2 Fuzzy Sets denoted by A and B, then let's consider result be the Intersection of them, then for every member of A and B, result will be:

degree_of_membership(result)= min(degree_of_membership(A), 1- degree_of_membership(B))

The First Fuzzy Set is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

The Second Fuzzy Set is : {'x1': 0.4, 'x2': 0.2, 'x3': 0.1}

Fuzzy Set Difference is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

Python Code:

```
import numpy as np

A = dict()

B = dict()

result=dict()

n = int(input("enter no.of elements of set A:"))
```

```
A = {}
for i in range(n):
 keys = input()
 values = float(input())
 A[keys] = values
n1 = int(input("enter no.of elements of set B:"))
B = {}
for i in range(n1):
 keys1 = input()
 values1 = float(input())
 B[keys1] = values1
print('The First Fuzzy Set is :', A)
print('The Second Fuzzy Set is :', B)
for A_key, B_key in zip(A, B):
 A_value = A[A_key]
 B_value = B[B_key]
 B_value = 1 - B_value
 if A_value < B_value:
   result[A_key] = A_value
 else:
   result[B_key] = B_value
print('Fuzzy Set Difference is :', result)
```

Output:

```
enter no.of elements of set A:3
x1
0.2
```

x2

0.5

x3

0.8

enter no.of elements of set B:3

x1

0.4

x2

0.2

x3

0.1

The First Fuzzy Set is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

The Second Fuzzy Set is : {'x1': 0.4, 'x2': 0.2, 'x3': 0.1}

Fuzzy Set Difference is : {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

1.Lab Task: Perform the all operations of fuzzy set in a program. Sample output is following:

```
enter no.of elements of set 1:3
x1
0.2
x2
0.5
x3
0.8
enter no.of elements of set 2:3
x1
0.4
x2
0.2
x3
0.1
Menu
1.Union 2.Intersection 3.Complement 4.Difference 5.Exit
Enter your choice:4
Difference of two sets is {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}
```

Introduction: Fuzzy set basic operation are union,intersection,complement,difference.Here Perform all kinds of basic operation.

Python Code:

```
import numpy as np

def union(A,B):

    result={}

    for i in A:

        if(A[i]>B[i]):

            result[i]=A[i]

        else:

            result[i]=B[i]

    print("Union of two sets is",result)


def intersection(A,B):

    result={}

    for i in A:

        if(A[i]<B[i]):

            result[i]=A[i]

        else:

            result[i]=B[i]

    print("Intersection of two sets is",result)


def complement(A,B):

    result={}

    result1={}

    for i in A:
```

```
        result[i]=round(1-A[i],2)
    for i in B:
        result1[i]=round(1-B[i],2)
    print("Complement of  1st set is",result)
    print("Complement of  2nd set is",result1)

def difference(A,B):
    result={}
    for i in A:
        result[i]=round(min(A[i],1-B[i]),2)
    print("Difference of two sets is",result)



def main():

    while True:
        print("Menu ")
        print("1.Union")
        print("2.Intersection")
        print("3.Complement")
        print("4.Difference")
        print("5.Exit")
        choice=int(input("Enter your choice:"))
        if choice==1:
            union(d,d1)

        elif choice==2:
```

```
            intersection(d,d1)

        elif choice==3:
            complement(d,d1)

        elif choice==4:
            difference(d,d1)

        elif choice==5:
            break
        else:
            print("Wrong choice")

if __name__ == "__main__":

    n = int(input("enter  no.of elements of set 1:"))
    d = {}
    for i in range(n):
        keys = input()
        values = float(input())
        d[keys] = values
    n1 = int(input("enter  no.of elements of set 2:"))
    d1 = {}
    for i in range(n1):
        keys1 = input()
        values1 = float(input())
        d1[keys1] = values1
```

main()

Output:

enter no.of elements of set 1:3

x1

0.2

x2

0.5

x3

0.8

enter no.of elements of set 2:3

x1

0.4

x2

0.2

x3

0.1

Menu

1.Union

2.Intersection

3.Complement

4.Difference

5.Exit

Enter your choice:1

Union of two sets is {'x1': 0.4, 'x2': 0.5, 'x3': 0.8}

Menu

1.Union

2.Intersection

3.Complement

4.Difference

5.Exit

Enter your choice:2

Intersection of two sets is {'x1': 0.2, 'x2': 0.2, 'x3': 0.1}

Menu

1.Union

2.Intersection

3.Complement

4.Difference

5.Exit

Enter your choice:3

Complement of 1st set is {'x1': 0.8, 'x2': 0.5, 'x3': 0.2}

Complement of 2nd set is {'x1': 0.6, 'x2': 0.8, 'x3': 0.9}

Menu

1.Union

2.Intersection

3.Complement

4.Difference

5.Exit

Enter your choice:4

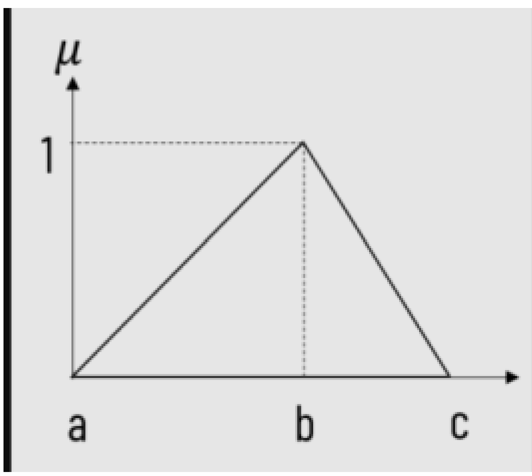
Difference of two sets is {'x1': 0.2, 'x2': 0.5, 'x3': 0.8}

Conclusion: the four fundamental operations of fuzzy set theory provide a powerful set of tools for handling uncertainty and imprecision in complex systems and situations, and have wide-ranging applications in fields such as decision-making, pattern recognition, control systems, and artificial intelligence.

2. **Fuzzy Membership Function: Triangular membership function:**

Introduction:

This is one of the most widely accepted and used membership functions (MF) in fuzzy controller design. The triangle which fuzzifies the input can be defined by three parameters a, b and c, where c defines the base and b defines the height of the triangle.
Trivial case:



Here, in the diagram, X-axis represents the input from the process (such as air conditioner, washing machine, etc.) and the Y axis represents the corresponding fuzzy value.

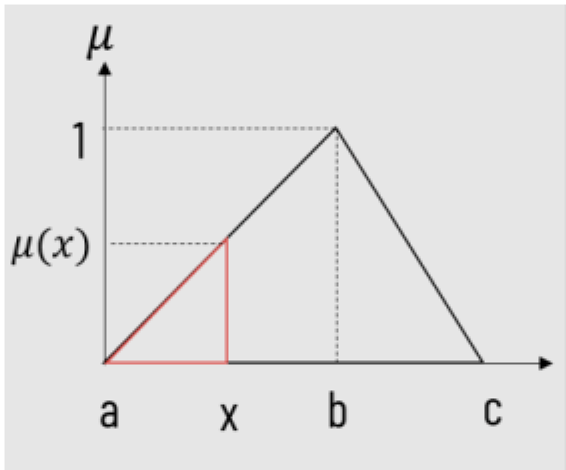
If input x = b, then it is having full membership in the given set. So,

$\mu(x) = 1$, if $x = b$

And if the input is less than a or greater than b, then it does belong to the fuzzy set at all, and its membership value will be 0

$\mu(x) = 0$, $x < a$ or $x > c$

x is between a and b:

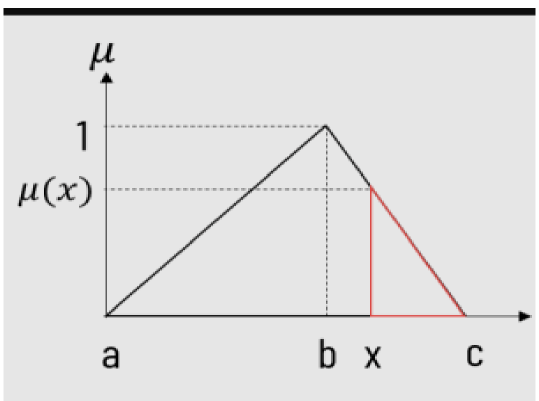


If x is between a and b, as shown in the figure, its membership value varies from 0 to 1. If it is near a, its membership value is close to 0, and if x is near b, its membership value gets close to 1.

We can compute the fuzzy value of x using a similar triangle rule,

$\mu(x) = (x - a) / (b - a)$, $a \leq x \leq b$

x is between b and c:



If x is between b and c, as shown in the figure, its membership value varies from 0 to 1. If it is near b, its membership value is close to 1, and if x is near c, its membership value gets close to 0.

We can compute the fuzzy value of x using a similar triangle rule,

$\mu(x) = (c - x) / (c - b)$, $b \leq x \leq c$

Combine all together:

We can combine all above scenarios in single equation as,

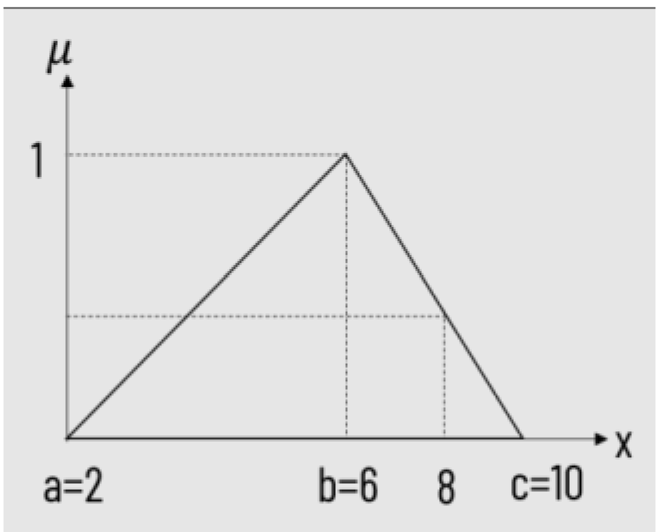
$$\mu_{triangle}(x; a, b, c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases}$$

$$= \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$$

Triangular membership function

Example: Triangular membership function

Determine μ, corresponding to x = 8.0



For the given values of a, b and c, we have to compute the fuzzy value corresponding to x = 8. Using the equation of the triangular membership function

$$\mu_{triangle}(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),0\right)$$

$$= \max\left(\min\left(\frac{x-2}{6-2},\frac{10-x}{10-6}\right),0\right)$$

$$= \max\left(\min\left(\frac{x-2}{4},\frac{10-x}{4}\right),0\right)$$

$We\ put\ x = 8.0$

$$= \max\left(\min\left(\frac{3}{2},\frac{1}{2}\right),0\right) = \frac{1}{2} = 0.5$$

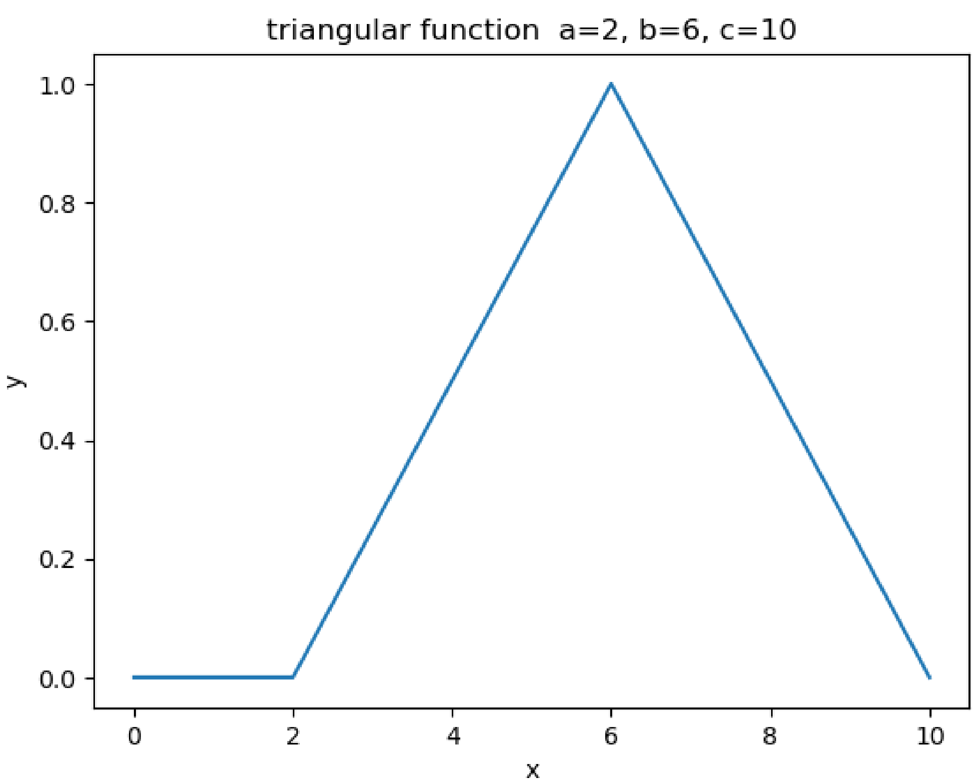
Thus, x = 8 will be mapped to a fuzzy value of 0.5 using the given triangle fuzzy membership function

Triangular membership function program(Python Code):

```
import numpy as np

import matplotlib.pyplot as plt

def trimf(x,a,b,c):

    if x<=a:

        return 0


    elif a<=x<=b:

        return ((x-a)/(b-a))

    elif b<=x<=c:

        return ((c-x)/(c-b))

x=np.arange(0,50,0.1)
```

```
#print(x)
a=2
b=6
c=10
y=[trimf(xi,a,b,c) for xi in x]
plt.plot(x,y)
plt.title("triangular function  a={}, b={}, c={}".format(a,b,c))
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Output:

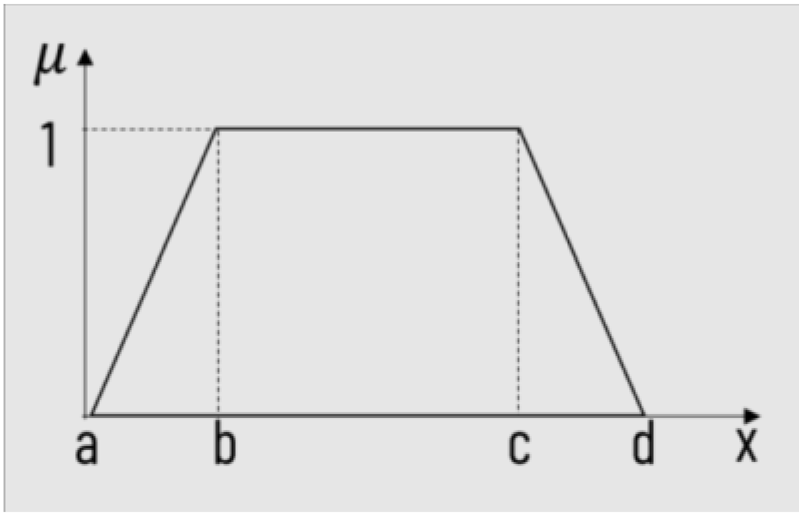


Colclusion: A triangular membership function is a type of fuzzy set membership function that has a triangular shape, characterized by three parameters: the lower limit, the upper limit, and the peak. the triangular membership function is a versatile and widely used tool in fuzzy set theory.

3. **Trapezoidal membership function:**

Introduction:

The trapezoidal membership function is defined by four parameters: a, b, c and d. Span b to c represents the highest membership value that element can take. And if x is between (a, b) or (c, d), then it will have a membership value between 0 and 1



We can apply the triangle MF if elements are in between a to b or c to d. It is quite obvious to combine all together as,

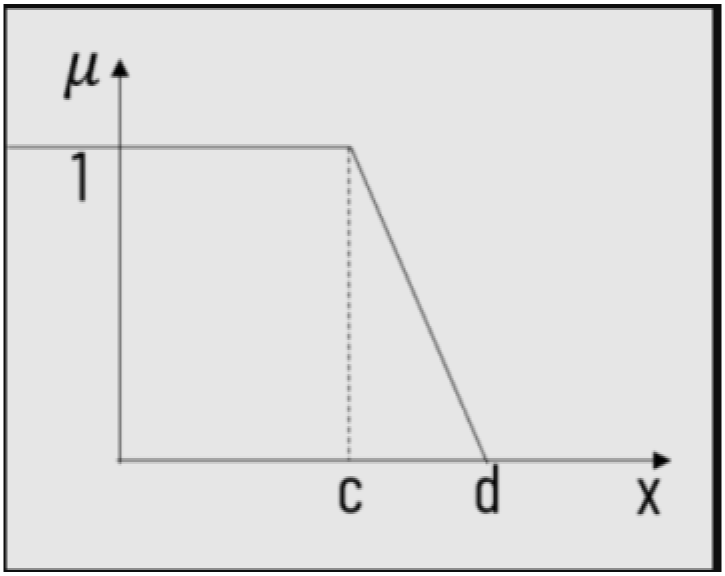
$$\mu_{trapezoidal}(x; a, b, c, d) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{cases}$$

$$= \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$$

There are two special forms of trapezoidal function based on the openness of function. They are known as Rfunction (Open right) and L-function (Left open). Shape and parameters of both functions are depicted here:

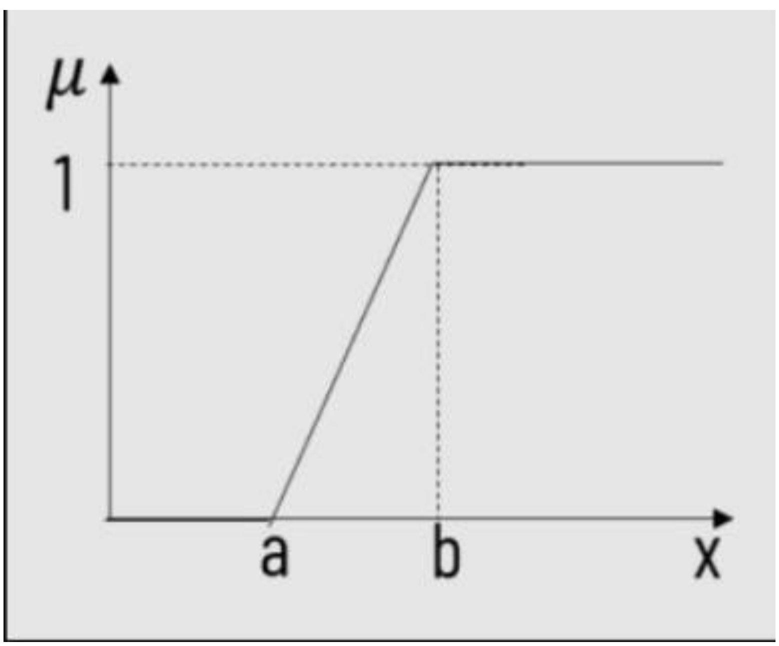
R-function: it has a = b = -∞

R-function



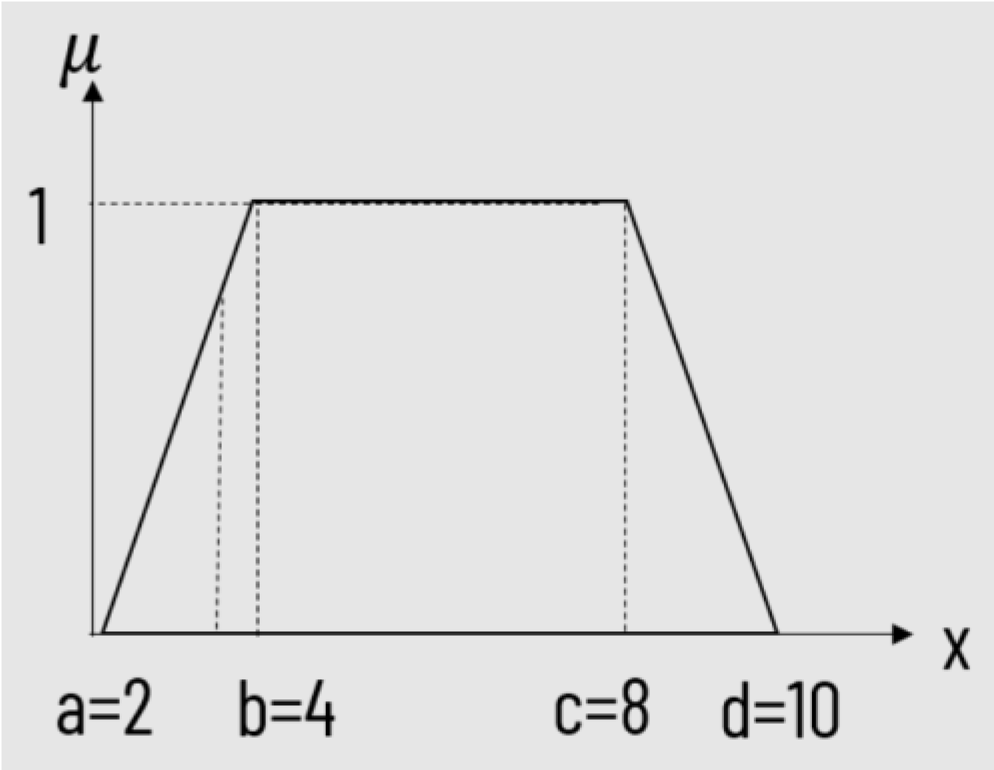
L-function: It has c = d = +∞

L-function



Example: Trapezoidal membership function

Determine μ, corresponding to x = 3.5

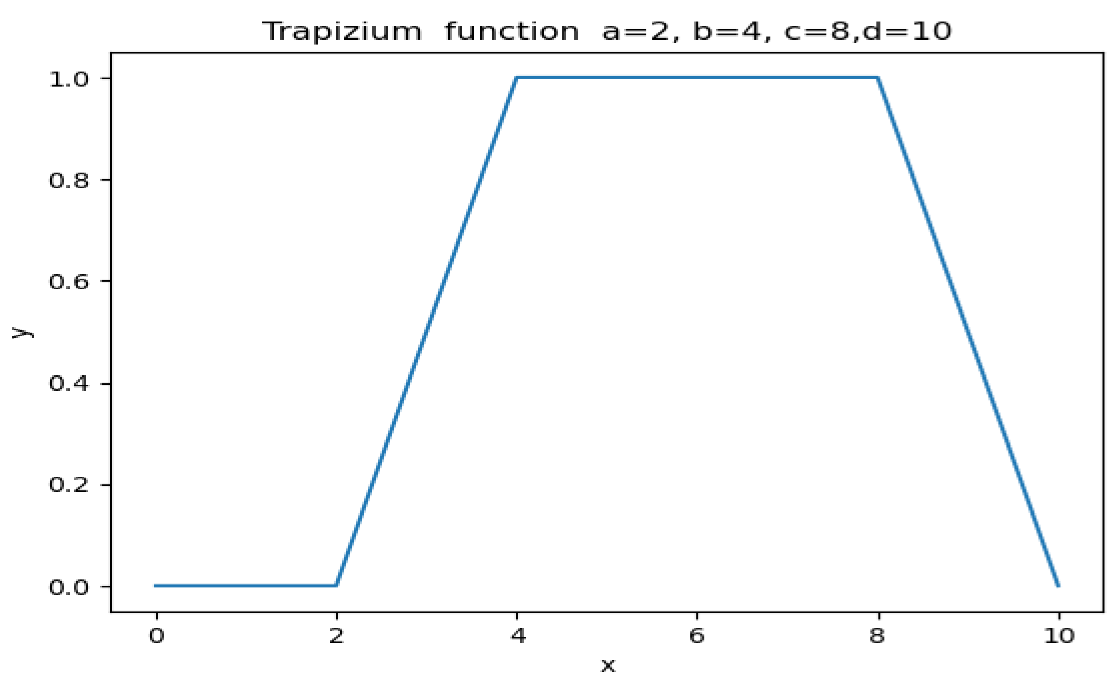


Trapezoidal Membership function(Python Code):

```
import numpy as np

import matplotlib.pyplot as plt

def trimf(x,a,b,c,d):

    if x<=a:

        return 0


    elif a<=x<=b:

        return ((x-a)/(b-a))


    elif b<=x<=c:

        return 1


    elif c<=x<=d:
```

```
        return ((d-x)/(d-c))
x=np.arange(0,50,0.1)
#print(x)
a=1
b=5
c=7
d=8
y=[trimf(xi,a,b,c,d) for xi in x]
plt.plot(x,y)
plt.title("Trapizium  function  a={}, b={}, c={},d={}".format(a,b,c,d))
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Output:



Conclusion: trapezoidal membership functions is that they provide a more flexible modeling fuzzy sets than triangular membership functions.