

Set Theory

Furqan Ahmed | Discrete Mathematics | 16B-117-SE

# SET THEORY IN CS

Most students including myself have studied sets from a very early stage, mostly from grade two, before knowing how tangled the topic is in our daily life, the uses of this diverse and deep topic, I always questioned the motive to teach me such topic, but I wasn’t given a satisfactory answer; Until now!

It is crucial in data structures, data structures, define, organize, search. The queries you make are in form of set notation. Without which even if data is available the “extraction” would be next to impossible

Databases are at the root of most real-world applications. You can’t imagine life where every time you login or do something you have to reinsert the data. Databases are built to process data in Sets, and the applications that use them deals with the databases using sets.

Many algorithms work on sets. In machine learning, you use a group of results to ‘train’ your machine, then the algorithm is used in the ‘real’ world.

Statistics provide a mathematical description of sets of data & Stats are key to rule the internet. You must have seen ads that surprisingly show exactly what you wanted or what you searched for a week ago. This is how you make better sums of money.

Understanding sets is so CRITICAL in computer science that most universities teach this course as part of freshman year, to students studying in CS fields.

This decade is period of 3D, Virtual and High-End UHD games. Not to mention that this is a Billion Dollar Market, Employing a substantial number of people under it.

It won’t be an overstatement to say that this all wouldn’t had been possible without Sets, Sets are the at the core of movements of points and their proper application renders smooth motion, which is like the most demanded feature in every electronic device.

IT PLAYS A CORE ROLE IN AUTOMATA, A CORE OF CS ITSELF!

~There are many uses of infinite sets and their properties. One important task in computer science is proving or verifying that programs do what they are supposed to do. When such programs involve loops and recursive calls, we need methods for showing that the loops and recursive calls terminate, i.e., that the program won't run forever. The usual induction principle for natural numbers suffices for showing that a single loop terminates, but we need double induction for double loops, triple induction for triple loops, etc. The whole business can get very complicated when the program is more than just a simple combination of loops. Set theory helps sort it all out with the principle of transfinite induction and the calculus of (infinite) ordinal numbers. Transfinite induction covers all ways in which one could show that a program terminates, while the ordinal numbers are used to express how complex the proof of termination is!

Set theory is an extremely taken to be a language for being able to rigorously define and manipulate various "completed infinities" - not only just infinite sets such as the natural numbers or real numbers, but much "larger" completed infinities, one can often get by in applications using various "incomplete" and/or "finitary" substitutes for these objects, which require less set-theoretic machinery to set up (e.g. one may be able to largely avoid use of the axiom of choice).

Once one has set up a non-trivial amount of mathematics in the realm of infinite or continuous spaces, one can often derive finitary consequences (at least at a qualitative level) by using further tools such as compactness arguments or nonstandard analysis, which again are most easily discussed if one is working within a set theoretic framework. A good example of this is the Furstenberg correspondence principle that allows one to derive combinatorial statements about finite sets of integers using the infinitary language of ergodic theory, which can require a non-trivial amount of set theory to work with (e.g. when using tools such as disintegration of measures with respect to a sigma algebra.

In conclusion Set theory is important for topics of Theory of automata, which in turn is crucial in CS, without which you can’t really get a degree in CS.

# ABSTRACT

The project under discussion is based upon the basic idea of set operations, it aims to automate the entered set members, it targets Union, Intersection, Subtraction, and DeMorgan’s Law for two sets.

The program uses separate methods, for each functionality, high end logic for computations are stored in separate class, governing all matters concerning operations on set, the Central class contains basic value passing methods, that calls the crucial methods in other class,

The number of members in a set is made virtually unlimited using params.

The members of each set are required to be separated using ‘,’ a comma.

Limitations

The only know limitation in the program is the members of any set could be a single character long, which is only due to the programmer’s inability to take data input as multiple character, as the input is taken in as a string and later converted to a character array.

The program was so limited that only ‘new’ thing the programmer learnt was that params should be the last one in a method’s parameter list. Programmer honestly can’t recall any other.

CODE:

using System;

using System.Collections.Generic;

namespace DS\_Project

{

class SetOperations

{

public char[] Sub(int c ,char[] a,params char[] b)

{

var aa = new List<char>();

aa.AddRange(a);

foreach (var VAR1 in a)

{

foreach (var VAR2 in b)

{

var cast = 0;

var cast2 =0;

cast = (int)VAR1;

cast2 = (int)VAR2;

if (cast==44)

{

break;

}

if (cast2 == 44)

{

continue;

}

if (VAR1==VAR2)

{

aa.Remove(VAR1);

}

}

}

var index = 0;

while (index < aa.Count - 1)

{

if (aa[index] == aa[index + 1])

aa.RemoveAt(index);

else

index++;

}

var charray = aa.ToArray();

if (c == 1)

{

return charray;

}

foreach (var VARIABLE in aa)

{

if (VARIABLE==',')

{

continue;

}

Console.Write(VARIABLE+",");

}

Console.WriteLine();

return charray;

}

public char[] Union(int c,char[] a, params char[] b)

{

var aa = new List<char>();

foreach (var VAR1 in a)

{

foreach (var VAR2 in b)

{

var cast = 0;

var cast2 = 0;

cast = (int)VAR1;

cast2 = (int)VAR2;

if (cast == 44)

{

break;

}

if (cast2 == 44)

{

continue;

}

if (VAR1 == VAR2)

{

aa.Add(VAR1);

}

else

{

aa.Add(VAR1);

aa.Add(VAR2);

}

}

}

for (var VARIABLE = 0; VARIABLE < aa.Count; VARIABLE++)

{

for (var VARIABLE1 = 0; VARIABLE1 < aa.Count; VARIABLE1++)

{

if (aa[VARIABLE] < aa[VARIABLE1])

{

var temp = aa[VARIABLE1];

aa[VARIABLE1] = aa[VARIABLE];

aa[VARIABLE] = temp;

}

}

}

var index = 0;

while ( index < aa.Count - 1)

{

if (aa[index] == aa[index + 1])

aa.RemoveAt(index);

else

index++;

}

var charray = aa.ToArray();

if (c==1)

{

return charray;

}

foreach (var VARIABLE in aa)

{

Console.Write(VARIABLE + ",");

}

Console.WriteLine();

return charray;

}

public void Intersection(char[] a, params char[] b)

{

var aa = new List<char>();

foreach (var VAR1 in a)

{

foreach (var VAR2 in b)

{

var cast = 0;

var cast2 = 0;

cast = (int)VAR1;

cast2 = (int)VAR2;

if (cast == 44)

{

break;

}

if (cast2 == 44)

{

continue;

}

if (VAR1 == VAR2)

{

aa.Add(VAR1);

}

}

}

var index = 0;

while (index < aa.Count - 1)

{

if (aa[index] == aa[index + 1])

aa.RemoveAt(index);

else

index++;

}

foreach (var VARIABLE in aa)

{

Console.Write(VARIABLE + ",");

}

Console.WriteLine();

}

}

}

using System;

namespace DS\_Project

{

static class Program

{

static char[] a, b,u;

static readonly SetOperations Apply = new SetOperations();

static void Main(string[] args)

{

Start();

Console.Write("Answers:\n\n(A - B) =");

AMinusB();

Console.Write("(B - A) =");

BMinusA();

Console.Write("(A U B) =");

AUnionB();

Console.Write("(B U A) =");

AUnionB();

Console.Write("(A ∩ B) = ");

AIntB();

Console.Write("(B ∩ A) = ");

AIntB();

Console.Write("(A)'= ");

AComp();

Console.Write("(B)'= ");

BComp();

Console.Write("\n\nDeMorgan's Laws:" +

"\n(A U B)' == A' ∩ B'" +

"\n" +

"(A U B)' = ");

AUnionBComp();

Console.Write("\nA' ∩ B' = ");

ACompIntBComp();

Console.WriteLine("Hence Proved!\n");

}

static void Start()

{

Console.WriteLine("Furqan Ahmed 16B-117-SE");

Console.WriteLine("//////DISCLAIMER: Set members should be single Character\\\\\\\n");

Console.Write("Enter Elements of set U and press enter(Leave it blank," +

"if you want to)\nU= ");

var setU = Console.ReadLine();

Console.Write("Enter Elements of set A and press enter\nA= ");

var setA = Console.ReadLine();

Console.Write("Enter Elements of set B and press enter\nB= ");

var setB = Console.ReadLine();

u = setU.ToCharArray();

a = setA.ToCharArray();

b = setB.ToCharArray();

}

static void AMinusB()

{

Apply.Sub(0,a, b);

}

static void BMinusA()

{

Apply.Sub(0,b, a);

}

static void AUnionB()

{

Apply.Union(0,a,b);

}

static void AIntB()

{

Apply.Intersection(a,b);

}

static void AComp()

{

Apply.Sub(0,u,a);

}

static void BComp()

{

Apply.Sub(0,u, b);

}

static void AUnionBComp()

{

var charray = Apply.Union(1,a, b);

Apply.Sub(0,u,charray);

}

static void ACompIntBComp()

{

var charray1=Apply.Sub(1,u,a);

var charray2= Apply.Sub(1, u, b);

Apply.Intersection(charray1,charray2);

}

}

}

# Sample Outputs:



