data structure

data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently

classification->

1. linear data structure (link , list, stack , queue)

yani es me data sequence me rakha jata hai

1. non linear data structure (tree , graph)

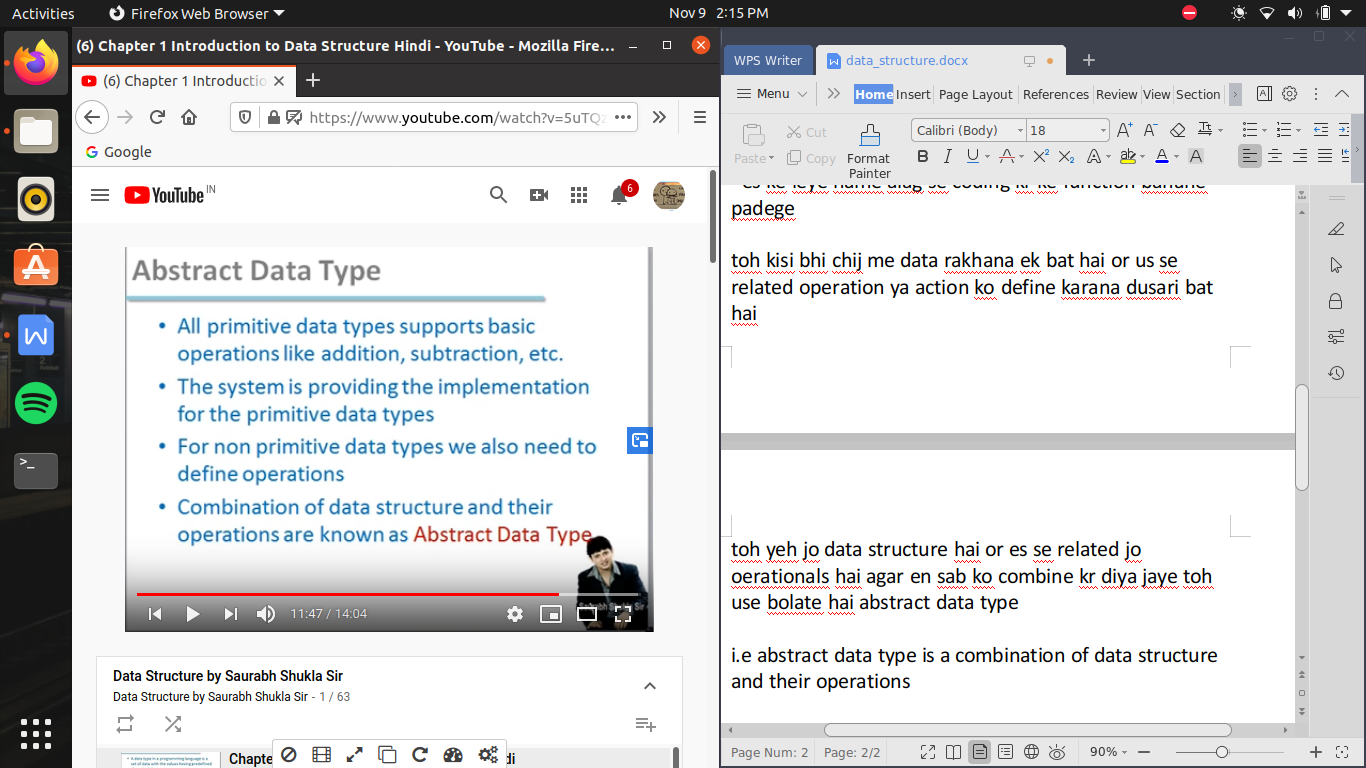
jab hum koe data type banate hai struct ki help se or us me bahot sare variables banate hai toh ab , wo data type capable hai wo sari information store karane ke but agar hame es data type me koe operation perform karane ho jaise hame modify or edit karana hai , hum es ke leye ek dusara function banana padega ya ek or operation define karana padega , toh jab tak maine structure banaya or us ke andar variable store kr diye hai , toh maine data structure ko bana liya hai , jis me hum us ke regarding data store kr sakate hai lekin hum us operation me kisi type a operation perform nhi kr sakate

es ke leye hame alag se coding kr ke function banane padege

toh kisi bhi chij me data rakhana ek bat hai or us se related operation ya action ko define karana dusari bat hai

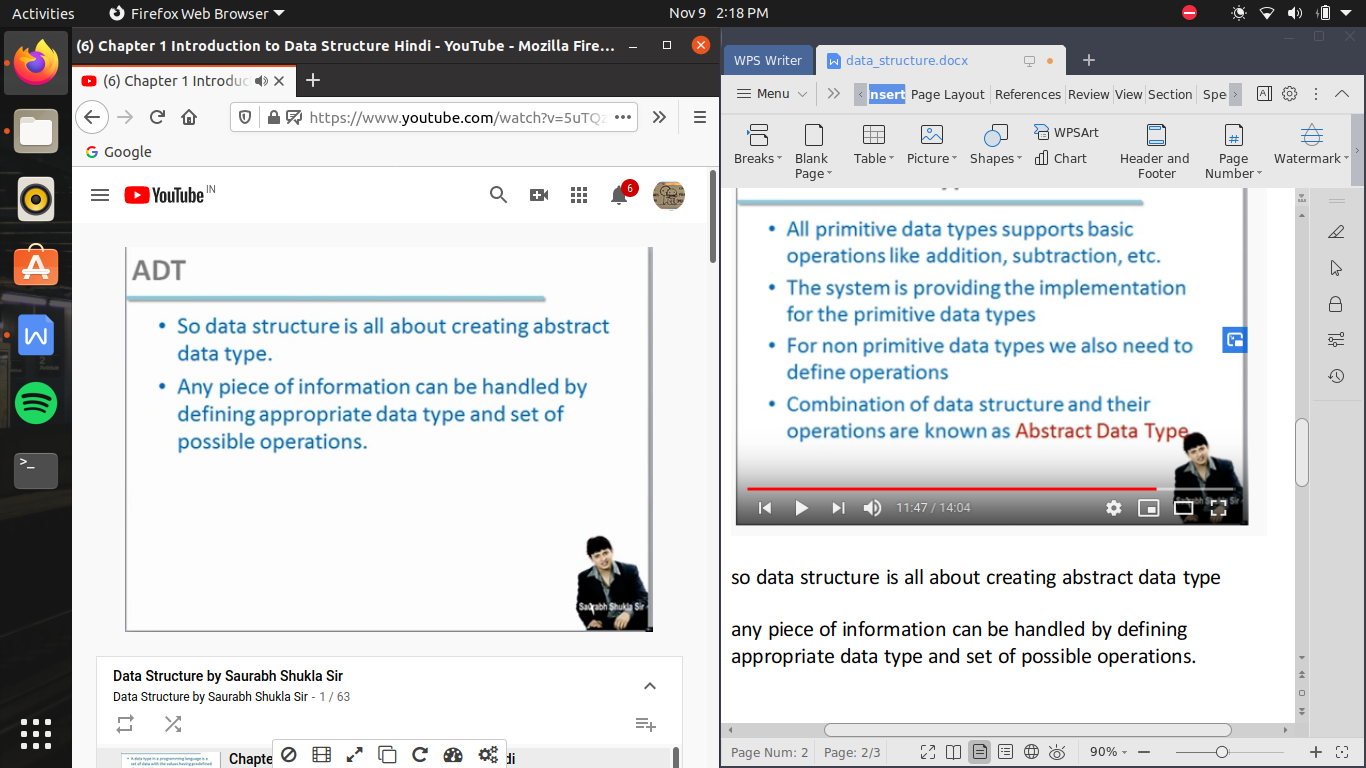
toh yeh jo data structure hai or es se related jo oerationals hai agar en sab ko combine kr diya jaye toh use bolate hai abstract data type

i.e abstract data type is a combination of data structure and their operations



so data structure is all about creating abstract data type

any piece of information can be handled by defining appropriate data type and set of possible operations.



analysis of algorithm ->

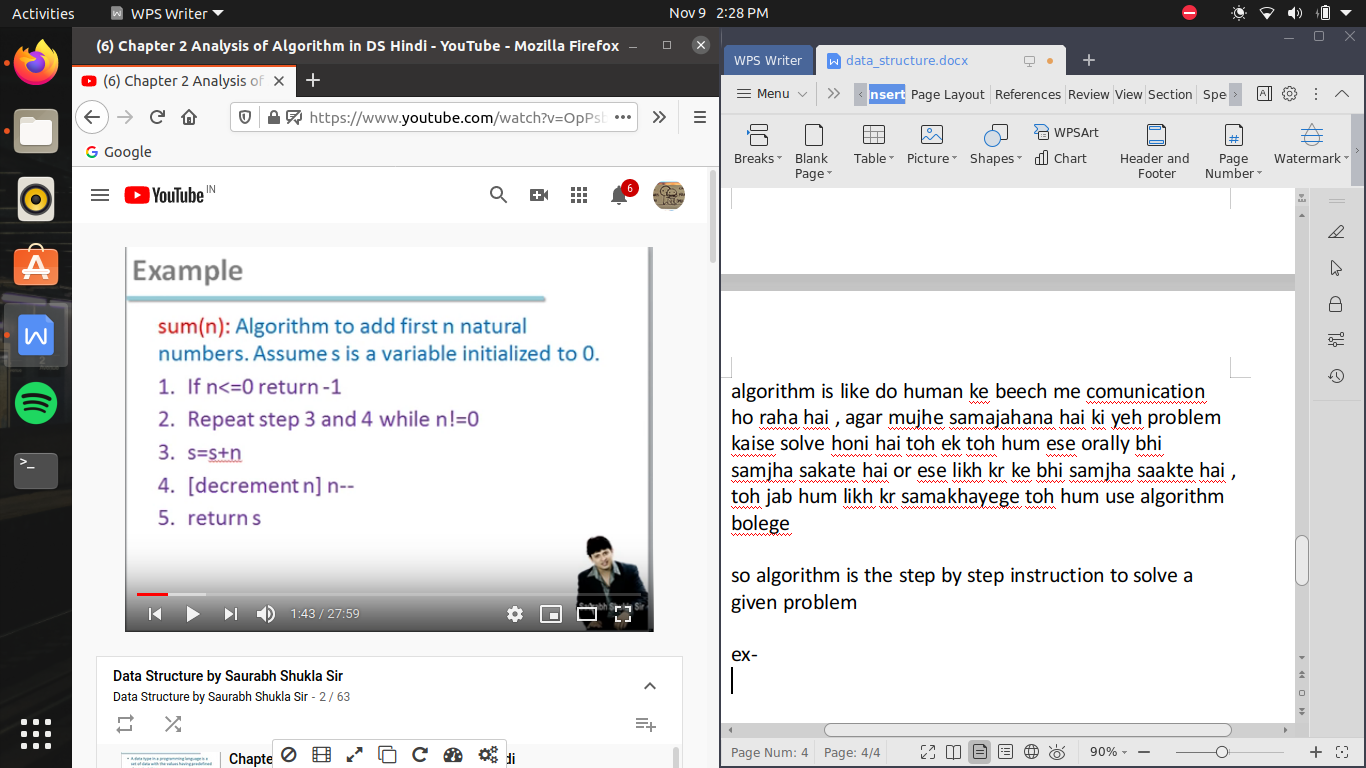
jab bhi hum pahale koe program banate hai toh us se pahle us ke logic ke bare me sochate hai yani hum us problem ko kaise solve karege , pahale kya karege , fir alagali step me kya karege yeh hum sochate hai ,

or agar hum yeh soch sakate hai toh hum es ko likh bhi sakate hai , toh hum program banane se pahale simple jo hum ne logic socha hai use english language me likhana chahate hai , jo program hum likhate hai wo kisi machine ko likhane ke leye likha jata hai ,or specifoc bole toh compiler ko samajhane ke leye likha jata hai

but jab hum algorithm likhatre hai toh keval human ke dawara hi readable or understandable hona chahiye so algorithm is like do human ke beech me comunication ho raha hai , agar mujhe samajahana hai ki yeh problem kaise solve honi hai toh ek toh hum ese orally bhi samjha sakate hai or ese likh kr ke bhi samjha saakte hai , toh jab hum likh kr samakhayege toh hum use algorithm bolege

so algorithm is the step by step instruction to solve a given problem

ex-



jaise n natural no. ka sum nikalane ke leye hame program likhana hai , toh us se pahale hum ne us ka logic socha , ab wo logic ko hum step me likhege yani step no. 1 me kya karege fir step no. 2 me kya karege

asia bhi ho sakata hai ki ek problem ko solve karane ke leye ek se jayada logic ho toh ek se jayada algorithm banegi , toh agar hamare pass kayi sari algorithm hai toh yeh bhi jaruri hai ki analyse kare ki kon si algorithm jayada better hai

so es ke leye hame karana pdata hai analysis of algorithm

analysis of algorithm hum es leye karate hai ki kon se algorithm se hamara effecient porgram bane us ko decide kr sake , toh es me major factors jo decide karate hai kon sa algorithm better hai wo hota hai , memory and time consumption ,

jab hum algorithm ka analysis karte hai toh hum bahot sare factors ko ignore kar dete hai jaise yeh program kon se hardware me chalane wala hai , or yeh program kon se environment me chalane wala hai , yeh sab chijo ko ignore kr dete hai ,

or yhe man ke chalate hai ki hum sirrf logic ki testing kr rahe hai ki kon sa logic better hai , yani do algorithm ko agar same environment me test kiya jaye toh kon better hai

toh yaha pr ek important karika hai jise bolate hai hum rate of growth

the rate at which the running time increases as as function of input is called rate of growth

ab or jiase koe algorithm me kitane data pr operation perform karana hai ese bhi time pe fark padata hai

yaha do bat kr rahe hai hum

first kitana time algorithm ne liya or hum ne us ko kam karne ke leye kitana data diya jaise hum 10 no. ko le kr us me sab se badi value findout karani hai toh hame time lagega or jaise hame 100 no. ko le kr badi value find karani hai toh or samay lagega , toh no. of input ke change hone se , time consume bhin change hoga toh es se hum ek graph plot kr sakate hai , jis me x axis hai no. of input

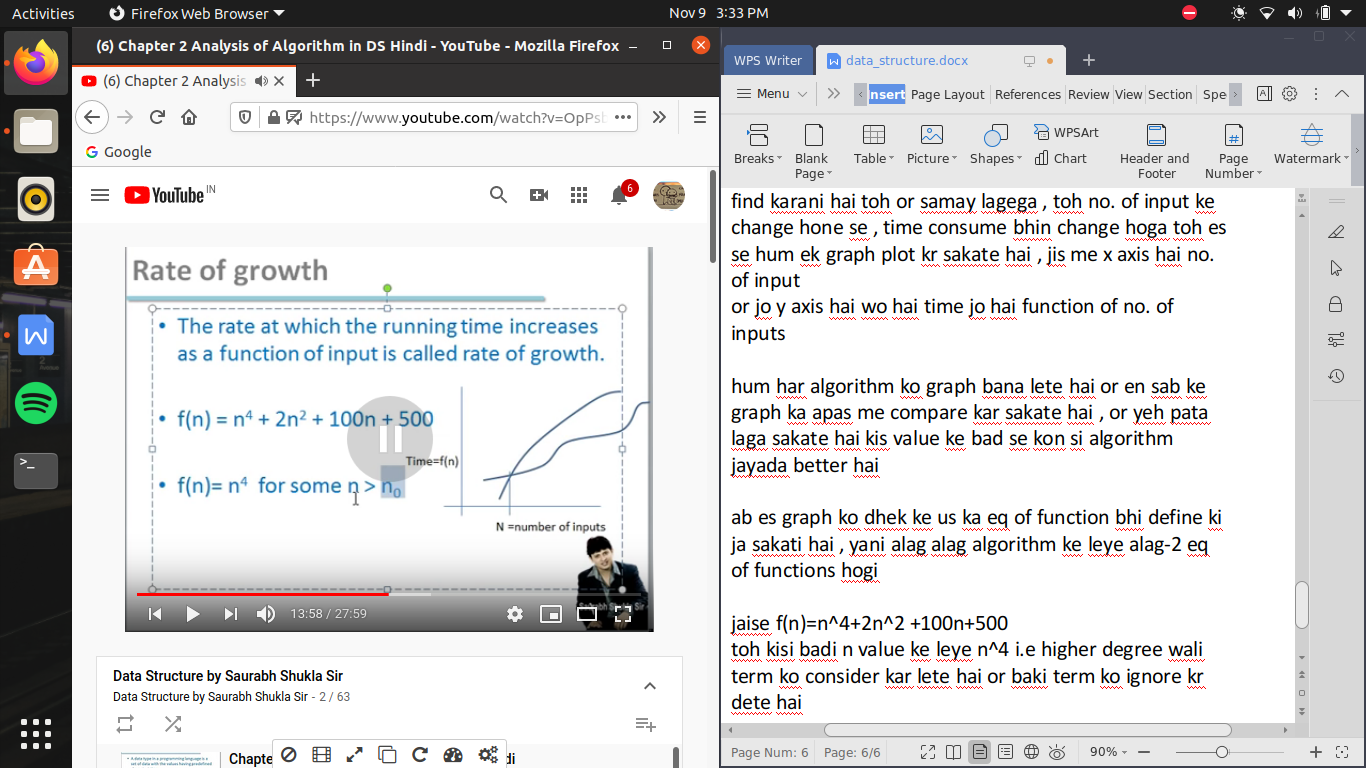
or jo y axis hai wo hai time jo hai function of no. of inputs

hum har algorithm ko graph bana lete hai or en sab ke graph ka apas me compare kar sakate hai , or yeh pata laga sakate hai kis value ke bad se kon si algorithm jayada better hai

ab es graph ko dhek ke us ka eq of function bhi define ki ja sakati hai , yani alag alag algorithm ke leye alag-2 eq of functions hogi

jaise f(n)=n^4+2n^2 +100n+500

toh kisi badi n value ke leye n^4 i.e higher degree wali term ko consider kar lete hai or baki term ko ignore kr dete hai



types of ananlysis ->

kisi bhi algorithm me 3 case ho sakate hai

1. worst case -> jis me sab se jayada operations karane ko pade
2. average case
3. best case ->jis me sab se kam operation lage

algorithm ke analysis me hum asymptotic notation ka bhi use karate hai

asymptotic notation ,me hum 3 chijo ka analysis karte hai

1. big -O notation
2. omega notation
3. theta notations

yeh sab me big notation jayada kam ka hai

big- O notation ->

jiase ki hum ananlyis karate time hum higher degree value hi consider karate hai , i.e ek approx value find kar lete hai toh , making it safe we make another curve which having upper value of initial curve and now we analyse this upper bound curve , as taking a safer limit so that program will never exceed this value

toh agar ek function of big -O notation hai

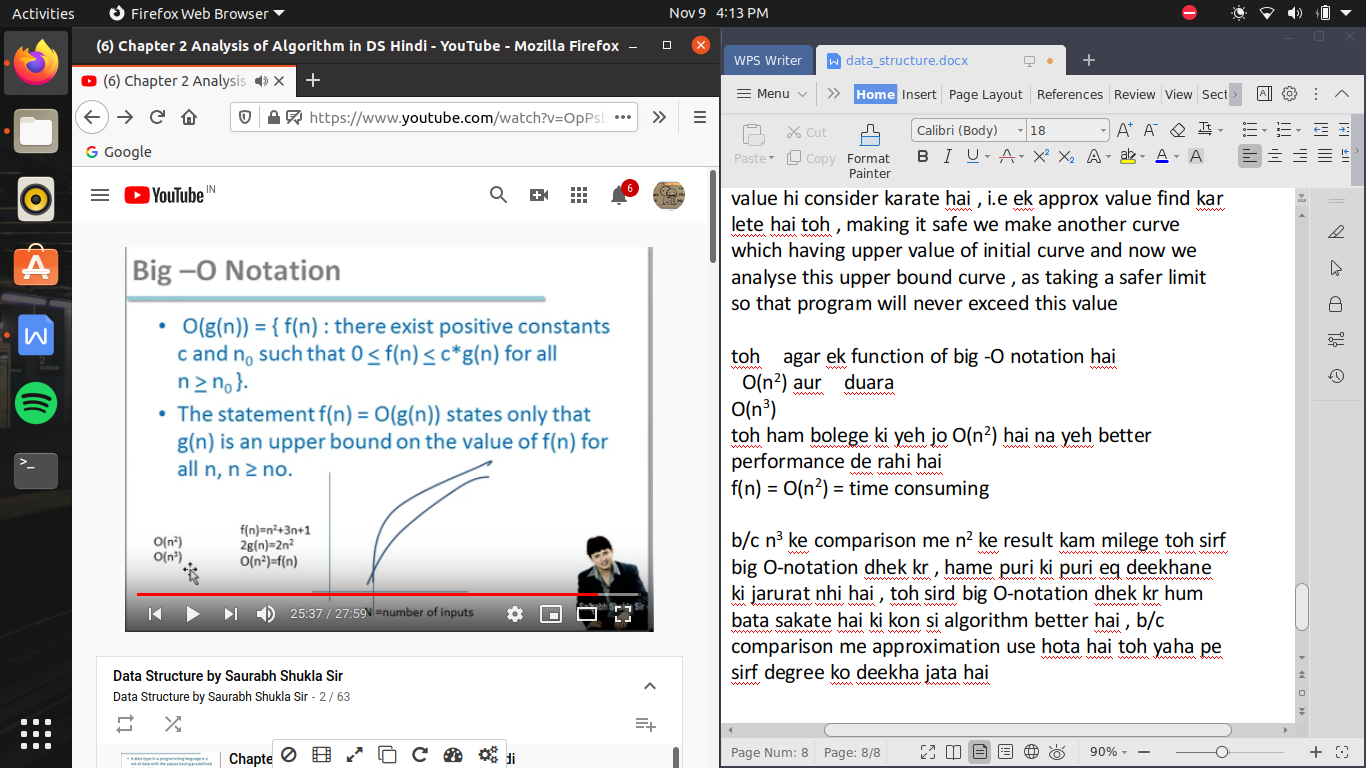
O(n2) aur duara

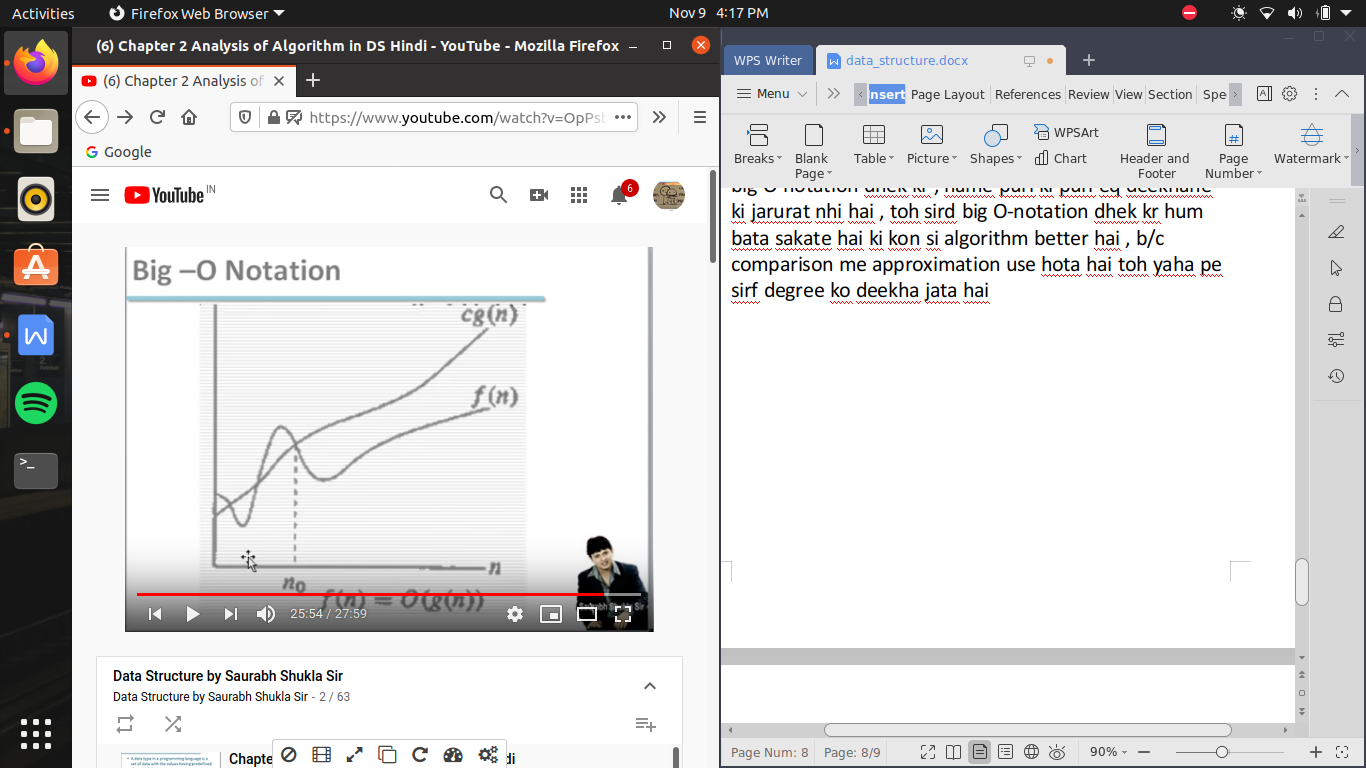
O(n3)

toh ham bolege ki yeh jo O(n2) hai na yeh better performance de rahi hai

f(n) = O(n2) = time consuming

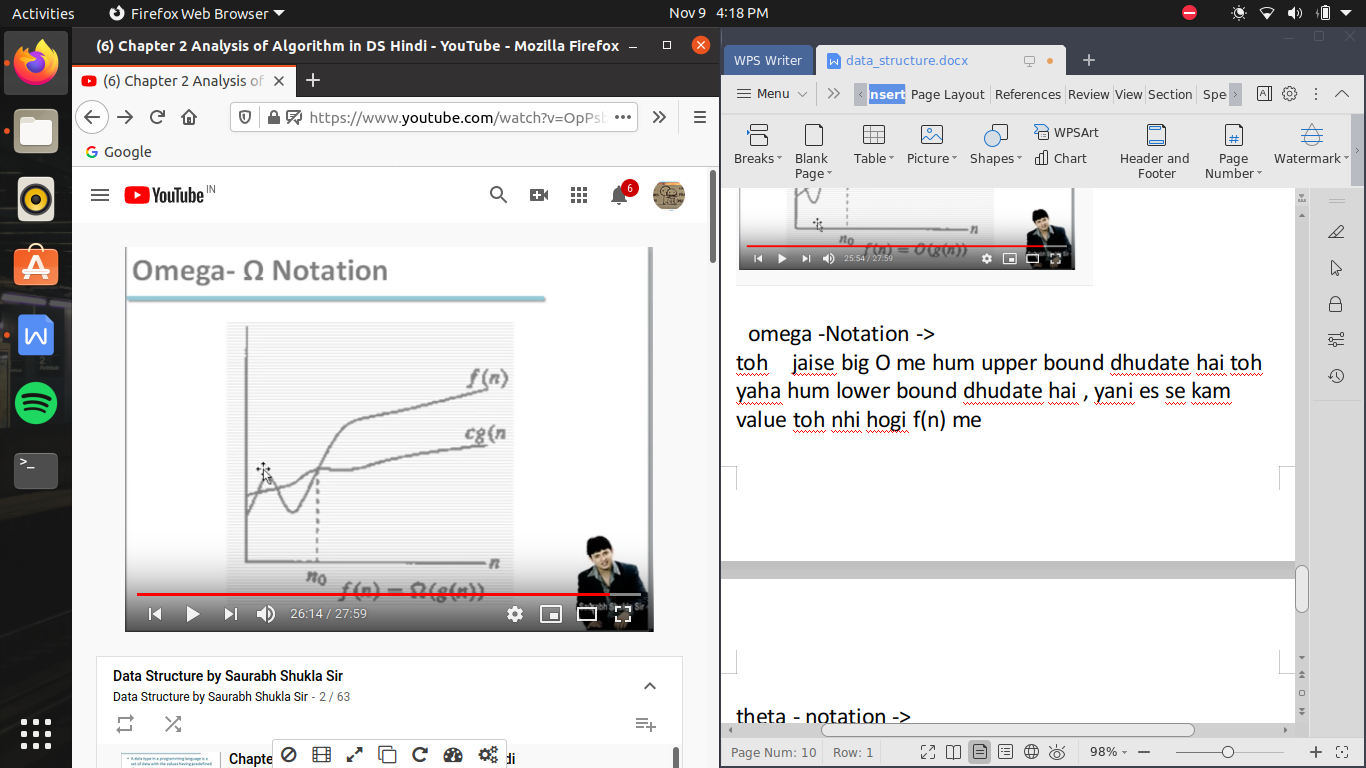
b/c n3 ke comparison me n2 ke result kam milege toh sirf big O-notation dhek kr , hame puri ki puri eq deekhane ki jarurat nhi hai , toh sird big O-notation dhek kr hum bata sakate hai ki kon si algorithm better hai , b/c comparison me approximation use hota hai toh yaha pe sirf degree ko deekha jata hai



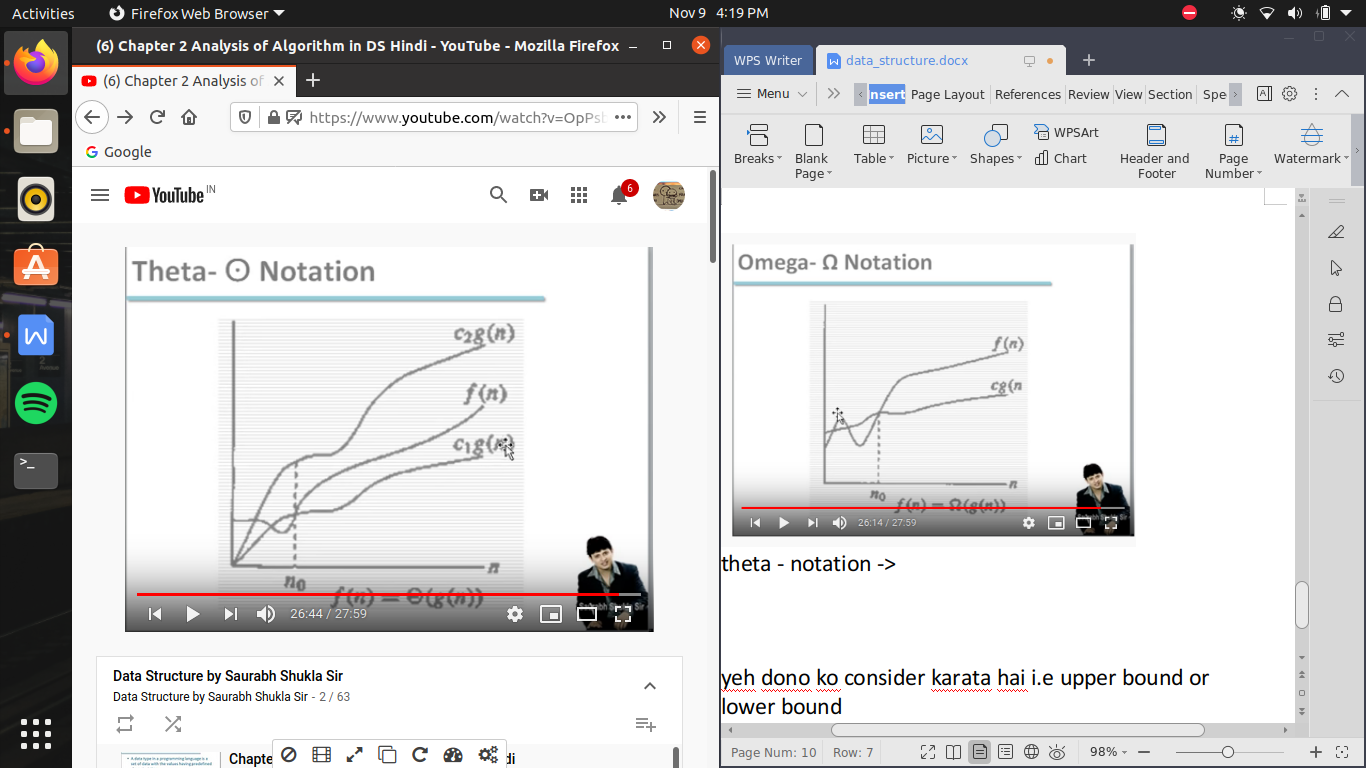


omega -Notation ->

toh jaise big O me hum upper bound dhudate hai toh yaha hum lower bound dhudate hai , yani es se kam value toh nhi hogi f(n) me



theta - notation ->



yeh dono ko consider karata hai i.e upper bound or lower bound

or theta notation tabhi likha ja sakata ho jab upper bound or lower bound ki degree same ho , yani g(n) upper hai jo or g(n) neeche hai jo , wo same hona chahiye

haan but constant me antar hai

ek

c2g(n) -> upper bound

c1g(n) -> lower bound

here g(n) is same but c2 > c1

so we can write it as

f(n)=theta(g(n))