ALGORITHM DESIGN HOMEWORK
-TECHINCAL REPORT-

1. HOMEWORK assignment problem description

We have the task of developing an algorithm for an advanced code editor that automatically corrects syntax errors in programming languages. It is assumed that we receive a clear specification of the valid large and very large data sets randomly generated syntax of the programming language in the form of a ”rule” and a code fragment that contains syntax errors, i.e., does not conform tothat rule.

Our objective is to build an algorithm that determines the minimum number of operations required to transform the code fragment into one that complies with the given rule. These operations may include character substitutions, insertions, or deletions.

Let’s take a concrete example to illustrate the problem: let’s assume we have the following syntax rule for function declarations in the programming language:

”Every function must start with the keyword ”func”, followed by the function name enclosed in parentheses.” An example of a valid function declaration would be ”func(myFunction)”.

Here’s how the situation looks:

Given code fragment: ”fnuc(myFuncion”

Our objective is to find the minimum number of operations required to correct the code fragment so that it matches the pattern

defined by the rule. These operations may include, for example, reversing the characters ”n” and ”u” to obtain ”func”, then inserting the missing characters ”t” and ”)”, so that we obtain ”func(myFunc)” according to the given rule.

2. Pseuodocode algorithm :

```
Function NrMinimOperatii(regula, fragment_cod, dp, n, m)
  // Initialize the dp array for base cases
  For i from 0 to n
    dp[i][0] = i  // Cost of deleting all characters up to i in regula

  For j from 0 to m
    dp[0][j] = j  // Cost of inserting all characters up to j in fragment_cod

  // Fill the dp array based on the operations: insertion, deletion, and substitution
  For i from 1 to n
    For j from 1 to m
      // Check if the current characters in regula and fragment_cod are the same
      If regula[i-1] == fragment_cod[j-1]
        // If the characters are the same, no operation is needed
        // Thus, the cost remains the same as it was for the previous characters
        dp[i][j] = dp[i-1][j-1]
      Else
        // If the characters are different, we need to consider the cost of three possible operations:

        // 1. Cost of Insertion:
        // Adding a character to regula to match fragment_cod.
        // This means we take the cost of transforming the first i characters of regula
        // to the first j-1 characters of fragment_cod (dp[i][j-1]),
        // and add 1 to account for the insertion operation.
        cost_inserare = dp[i][j-1] + 1

        // 2. Cost of Deletion:
        // Removing a character from regula to match fragment_cod.
```

```
            // This means we take the cost of transforming the first i-1 characters of regula
            // to the first j characters of fragment_cod (dp[i-1][j]),
            // and add 1 to account for the deletion operation.
            cost_stergere = dp[i-1][j] + 1

            // 3. Cost of Substitution:
            // Replacing a character in regula to match fragment_cod.
            // This means we take the cost of transforming the first i-1 characters of regula
            // to the first j-1 characters of fragment_cod (dp[i-1][j-1]),
            // and add 1 to account for the substitution operation.
            cost_inlocuire = dp[i-1][j-1] + 1

            // Find the minimum cost among insertion, deletion, and substitution.
            // This gives us the minimum number of operations needed to convert the first i characters of
regula
            // to the first j characters of fragment_cod.
            dp[i][j] = Minimum(Minimum(cost_inserare, cost_stergere), cost_inlocuire)
        End If
      End For
   End For

   // Return the minimum number of operations required to transform regula into fragment_cod
   Return dp[n][m]
End Function
```

3. EXPERIMENTAL DATA

This section presents the experimental data used in the analysis. The data is divided into 10 tests.

Number of tests : 10

***First row of each testcase is represented by : length of the rule and length of the fragment of code.**

***The following rows represents the rule and the fragment of code.**

```
3 8
wii
paejopbi

4 2
ulbx
qb

1 7
x
mtlvfzx

10 10
lqeznefeqr
geygwcjmtu
```

36 63

saybmtxyvqczyjlagogltmacvkdvhupnpnup

ilnyehbutrufyzdelphvzaiwpquazlxtgerfgunjbcpyjswmguktvoxcazphtug

150 234

sihjvvzyagqbgnkahqjzocmjmyihybqhnqknmszpwqiyblflqicogmsxxipkobxheuhwfbfzbmvgnisqo jacwyxcidvihuftldeszeipleyrzzaumjzahutcrxkcbzscwqzmzlfgilqayrhohojtpe

popelardkiydbhvsixqipusstycmnmkildbmglakygwoysmjdlndhzoybodbatfatgsptmoqozmrugen gqreojdzbzgbdlxvayfzbyypkczkyhicgtsarvrhehayvpijqnoaaszjvmycertvaruxchxkwjyezwdmxk bmgjpipwfeyzxridrqpiskztgdujlkcxvskgjwgnacxnjwoczvdpwuefxoffypjrdqrjoqzz

414 887

yrdowkiyfmvptdlvdelbdahkmlofwermzmhpepfnvyhxodepbsotrgnuqjohjhhxofndvpswpjkjtqksqi tqfqbfjxshldhoqfpqoqhtwicgvqfqvmerthyxelocrinqsrtuqqymdcseltbvntdkfievfvbawehiforgehla pgfxgrqetxsxyxdtrxxlrttvicgyxxixvkvghlqsfhvorullvrkusbdaypazlsrxdwcxrbtyqyxzwwswqdwad cnqsseffmqbeqolgyxixxajpyfanrnloytohchoyngarormvltuvofxfxcjodqvxivndzrrvlwcbuykbzlerr hzgngkvtghhzyhxmyujsejfnbinowecsdzbswnhtbfcigsnmqjyhdzqwtoqapllhkkhsqqumsxdb

hknlztjhrgrucstlahuisglrkizvbnwjlkuqmimbsvfztxlvheymavyfevntkorgwfbepqyruftdqzukcoksb yryicveflzoybajtejfughbcotyfcbznnfieyausujdaqszabcucqaaprwnzvnfbegbsghwapbscihbnjwd fbdxkdzragwvxijsngwdxixdlqiseimsfangrepnxekjjuagviivksawknbjbvdifmnjhmjyispiqwjqwiztip kudbblbklcldugwtztcfyxoofucvymfdosgalqubxmrbtrifbclerygsdkrkpvnhneviwwtwirleokxsmfw grsyctvjwlikfqtupnexqhutjijtcqztlzzhtriqtjxogsyeowtyjirumzwssmppxoutssivyhvuvehscmjljpxu xmfwdeaudxgingtyfmpzfhswdixdlipnpnlkaesbftjasubyqqwhwpepjrmlmjrunlnydzziljslivyomom notexfpayulhlxjwofwdzkhoyvakgjamtdcbkzyqelgcwdlsoavawqulqjqfibvstlbssfupygnpjgogyku qrewlnpkdhsyodublrarrqfatrqysgyggxwjlapjeoxgqrqzeggwosytsctzquchfregfrdtwbjepycaqdd btdouxongasocuflefwmhonwkjodnsfafnchnvsfhzpaehybpptphtyqjakttupaklklpuaiendrolyvdqy ajocznfluwnhjwucjllkadeoavnptdkxvelhgvrbbvytjquazmkypjzmfvxhbzoxsvwvinaeqpjfslvesqaw tmvdwdpqgknojksiegtmdxflfdcrkixexfszrjsqopylxr

211 1452

**eqoszstyoqdpgrrccnwccgbpgmhksphnfhodanemkpozmazrwmovxdzmsvxkfmppdqluigrlqxuui
gbiulzoycgmexwwjdeycyjnyueqmztnnslkifviiuwghsmqcdggsjtvhrmdykhrzuymjoxirkqbnqbdxjs
tomzskklbstyxnljbapvrncvghhrrsvrspzzrbfajettmupkuegx**

**syalhspabsihtksjwivhyouzoomzyzhwiyggxenskmhmfqrezabidsnpseihrboqhtouqgvzyobnnvynv
gxchxmlfchwgfllwlqgcpdnwxxycblcfnsxfqzhetoaxscizuwluvmyetfndxyvtoomrwlcgwnnwdmuc
afvnidbntwxokywzfslwlpyiypuqahjkwhypjzftucmonbyaykytrshvrpmyamtwxiuurxvvwlrvqcxeifc
flbqshtpbakawgochxdvcytmvbkxgekjjcyboydhcqnialfcqttbhjmdnyxfriuhwnvndizsfnvxnibacpzi
nccdwdpjcjunfsbasbhnnpkxdegxpxwjitrypwtaatgvyhqamhouzwwjlmwehxcnvkyelcuxmnxjoqm
iwuarmzwehusvmvliacfwyacrdkdszhxqdwubaiyzltskajvgomlwmxikdcxqqklgfqkibdsgdoqscxy
dfyjinhsgfuhmffqjtbhfzwqsfdsfhpbhgxlpcxqrqvxyuuiejpvpycxlslgnphakakrklfefnspwdvsmqhtd
vashxtuappkgfvdheaiwzdqbpbcnoojvgeknhzytlvbzehlmnlmsfaozqnsxcuivedkxyenaximtdfibuy
owcsnxgiugtegtymuvjprbmbbvnjokmkorpeafwuugwiuxrtehlolhjaqusvfthprljsthwtqacpsbmph
mvujvgydnvmzckmdaejvkjyiexoqjowhpcnujacbcrbwviccfjmzmgwsiynqlmzocuokdtogxxtwmjtk
kxbwptocyonybftehmeoipbtnajxxfydpvvlwlehbuveicoleuduchxdxmjxvmytzodipaedvtdwbfrcon
mltmhqoozkzixyeqbucpmkkyezrhdyrrxbipmcukshzwmlmlajugkoboxfgxetddnfjtijzdlqjqwmaua
xwuqzfokmgwgyjabzagwxdbazgfjdslgoauykixrcyvqveappauxmizaiunnobpbuwnmkijvhcutcoyy
tloneuiadnitbzvpriymlmxppccgrdgnpmcxkdswsnheuukhpwufqicswnvggshcmlfbkryxllphhvjhq
meejytggimjpfhwbqgiyksipoaxqstkksoqszvzozhltepyaywretrbcmucrgclgztlznntuodvslyuyfukc
lzcvxgelhlxkgkxwiqpblzatjtgldmyonzkarawidkguouhpxdrkwxixrnfblbnuhqgwhllilgsgbgoesplta
gaypnmiqbippjreuajlljnihnpcikbusulgjyqsalgtwpyiywhphitufccxlnkcictnbwmjpfehngblxwisbsk
myczsdviewklbawycfaycxbsfhxcasxhlbuichjxqvhanpnonsreaokyad**

1822 1394

**qvnzceyvaitzreroltfxeepamducautcmjzorkovpvvkhgelfogfhwwmihficvfldnfieubalwrffkogqcvw
dtumureoslqwphziocxlmpvzfpucsvtlvrfrrihdidiyywzqsgmbhpvrbdbarvlgyeaokywpdgdfszmjpy
kawdlqxqohpwpohbrqlnzoxmegrqkktbfrvxiibijjvovlkywmyjudglnsnfnqwlngmddtdloxkhoqvdmr
uunvjdeawrgafzihoocynnbdirajeffgcifvmuifwppylostbnjprirwrqntnclotghsmyfmfhejewfppchnlt
imxtktyyplnizwqlxixgklvlchgqynajiuywfrphiqitxvuwltusncpxuljtomncazbucecxicuacwzftyflcbq
kbajbqbfnhivcmajhrqzdkpkakizpgbfffuhcqoeatklvtauspjfotjkpoymycalhdcvcyqavkrsicidaseuq
lpnsfhpewqyybulpvbqqzsklvnwmrqmyxsdhumrzfsedbcospspymmptqgwmblzgaxfkvwyuwsey
uvpfptcztqvxquoekeotkckgeovtldbrysifltjoypxxlouykimzlmbtgbwfuxokjbotpjsouuaijalocyswpl
mcmrwdisuhookveglzqshslhirchwotbbomfspnddtxcidgnbikyyxgpzpyegngrwdmcwfxblmjtfpai
xyyrvtbfrpjypfwwnlyhpcdmvvqumnecqgcjvtddmkfwmsluelqhchqgufmmrcsuftdfvxtavsalasxrm
wfofqzisusmzvvzgfkukikrlwtzlyioyughomiigqsvpqtzfdndktrvahnbfilcfwraanbvlvcirywqovgndu
bllweeswvtligmobaqfbtsdvfyvjblnksjroapocewberhtsjujbfzgbsbpjgquqwoypbpxiwtvoxgwkem
ejmgnihojncagzxjzglhnqsygcnkvbbxcqzqxjrbrrucxqongpraaykjlwbaynwqtuxizkidtnmepvunye
ciqjeaocxpgzpbakexzkvxxdgqcjvuwbmqkzxacnlvivxtwooinroaqezdzilpukzhvwltnfopeofxuwvy
ykfjoyjwzemfrdmmisbkkvwjnpsgvjfxntrlximhbpezpklqrnpeowtoigczdvyewyodkbdwxzmpnlokk
htzfcikpxbjkxyjasfkwnaafqxuozkziywxozeilfsrptfiguehraahsehlbuhcykgoeazrfbapjqnayipwuhl
ulxyjkgpcdfjaxcnbxtaeqmkckhvendrvusfnztvvagsmctffaejevhlpjjqmxavkwbimihgivpquoeddfu
sunytiowwxxzfnsodjjuwgctwnhlfkydqouwcbistmxalwkphwiodkcvepdazmhzmthtkqsdadgfyala
ycgoxxieumfxgqtctslyqlmjcpzkgsodxysweqadmtiwldjejdopqxagxgvhrzfcmxjcxmmcorxnpaapil
cjmhdotzmawrxwbalhrvhizaugnqkcsskufuwdrqsqjvtstotjagtszimyeflnixhudbsspcgthexiuyliyg
kgqguvhcpnwgcruuthzafitnghvxjowwgtscgkniibqiakhhdqymzcuhedailskrybupbixswxdobxcva
rglldgmasltkwselqymrpzitzoptsumekltgvsfneltgoxkjzuydlpelkhinutvvkchqwxgzsszldlgmchmm
n**

**kkofifwdblcwixaohofiacqbonksyjkywkiqgdojfboshohokzddppqyfhefxybadvxyhbekxialxpnkfoa
wkxlwfbuggfmmmvpfdqdeljsvebyawybdoxgtsaibefuoqlhbqorfaznqobbflhwynvkgatvqbccwbp**

**qcxrzrochqeklxfgcnpqledxslhunyaneyztfbrfqvbzxwxdqwuqjvnfrcaklksmyvgqitctepxkseilutnrd
ibsmkvruyandpgzvpcxcowqaeywfcjscfkyycdnheikddfcyvpgabwadatreaqkneiivaaavczlfvuyqp
pdwgergqvzjrsfdjzfvfsznxrzbudhudqvzlpsstavktlbeuykebnoxvemavuriloeqkrirvnmsasruohcp
wuyvazftoinxbpvljicwykjgmxlgvkehyccxqwngbwmupjslsgauebfldksrswxspzhnltdqhouxwwiph
grnvkygpmjskrqtkvybxfkodydpkwjplxnqqqplmixddnknfscrsexkpbgmhftcvgkmkhxsbmhqkxbgf
vyxpilgejpzreouforrfrbecyctmkqlpwabuaafkazzgmnumejawnwwfjtsskcilvnkrydilqzgtzwbqphs
hizgctqvprqmnfcfcxhkbymtlfrjfoofwkhtpwwbnbzunyktbllwatnokvattemdnbomcwpzxwscnhgyi
omuszlkvzahczlrgslfzscqrdstdmredzdinqriadfmtrjkxrvdhmroxrrpbpoaoylructadgxruqresmtqy
yintcwrezcflhfyboczksqjirsiehozxyjupkumkqqrvloxkzboxzvqcbjjldlxmyrkfimsxbunkdevpufneo
uqkmbipbviwpasncnixwxiwzmnyqpifxkfycezczzjxchlzqsrwhmwfxhdnognfpltqbumytuhukjfuzur
wvyejjeyovrtaqewuimrvviopnqmozqbjyozgdstmesgteeahbuolowclhuhnebukqnhcegeiuocjiurz
yzdbpxnsfwtnndypoeputtcwpvgkmjtsictewmavztwkvrmznxgkutctzwbugklfmmbwvgdszxvqgo
efubkmrxkzstdunqqsygcmoamlsozrkjbdovlaeoywznytvjwouymilnkecjwyrtwjkioowdjydblzokc
mtjjkjjkxesmbnlrkbfcqsbydofudjedcvlsyxzedcyuulufbmrakwfiauofbnortenixvliriodiuahwalrssu
trbkyjylpbszobnanlfzuliopevghgieppiugvaevzsomrqwlmnwmhxbzltvqfpowpueazfskzrbweur**

187 2568

**fylumlghwbvyhwijtwsskmimstymbywzllinbozvhyoztsggmeaqfcyuqexeylikshlzrajhjeefrvvbsher
eddmzductbcyarsflgsheovxmxfjdyirpibpegbuujvumpheztjissvywbagphgpwqnscsuvtixnsecrnb
ycirnzqwxpgjxjdzfjqzy**

**yhwmezdtvbixmseakdmsgueqyyplsgfeutgtamkrvrsaoobhgnkfysahtnjkopeqweglwvashdhquh
afinqmzjfafgshvlymheuqvhgmcqesqynpvpicmprndnyeauxrqggevvrhrxjuopmqqfcbwdqmsjitv
wjhvyeqhrcojlospbsnspatiegbpvkgygiudmeepodgzijwgqqzckjitkuokscpcwdczwsmlbfiavmnbx
cmlgtmduyimqwvcigekftommuljqjrnfloyfbmbivcgdyjmpqlxnhxfnqmhbauqktfboonztglvgafkqmv
hiocmxtsdofzupqczkisrqvrcmhxxtoonrwxbxfcpsehzhuvrjhvybqegnjrphsaxtbqjyituhqazylokod
mztqjqqrmpfjisyzpyrkpnhjsjlfihqsgbbkpfmwdynwcsmmhxwrwzfkwhxzorbhrnroncaoxvyzgbcjx
pnchlioosgxotwzvtjzvjgrxdczxmcpwfllseeruhhumbektjgnlvwdseuwxiukjgrirpunmiodvmrxbbrq
rtnurdzgvdwjdrzdfkhpdtairhfqycdmvsynbedrkntwhbeakktxoweliffhgyjlvrernqpjlpkvglhrttaevz
yntheqjtvzdknwxlleudxgbhewytnkophvkqnxyrhslymbpqepbwrisyygpxdqztjbldkfpoqbflaaigdak
atpbqcuxapsxlxunhzqdtpopchbkunoppfexhkmckeimskgitexsgswxnheztaqqdeasvkfldyyyayzrf
snakmodhpigtewdaabkwhfmuvvazkdeynqiscuuvelpzdiwzqeqvoipnqjdjvobvbcgpamrmgijnhol
djsgecsdhitnmeinprroixbgbozetgodbtszhjoubrkmrtfinbtrcegvchlmkmvzpfjhoswiwasksgymua
chqdphcwgcjlhmaqdcaotjdwjuykdqtxsixzjojvuketjgjknhwpyuzvjfgodimuqhtziihivsrfppzvlsxijsz
uicckhayekavrktdgjnzukyjijdujkrrcokgrttsuamdrxqmkdsagefhefvljxpshhmmcvpxtitnilarjrkhqz
qqwrmmnomskkfytzrcvgzyvwhjhiueteiqzhgcndtmijirbligwojlfkhknizjxziirbyjhgzctaqelqxsrcrks
gyudgkfrtakgbyceimuhdthrimolzxfamfpwwwxrlyxpqnowdiijmzfvxpnppidnnqevlkhonqarsrixdr
dojuyulqljhvabwjrchsvpwexigmttaktzyhwzgqfxepvmlriuhiqcrpczxlpjmxhpcvhfyalnkfdcazrrjcy
kjeiltwvdngumvujznlcplulnevzystwmjnnuuabcticyiqplfpxvknkxwjgiuilatkcpbnczukdeyhxlwnyio
pfqsixjkuojgwexctgnfgadtimwbnadttbafucwbqkouzgqwvcseqorjobfwdhbebyxpzgasjghbhnxp
btsxwzduooaumacxlrqinqpusborzqoebezuwbzglpggixvcdlsebdypqhwmfowjyynjshaccmszzfw
bgvvpmucwkfwkriisnneqsnrswviybbeyfuvzigbbeujxyccffekjsycsintfzwkparwipiiocbjnixnswmh
nefmwdfebqvwmpnjbjedlltsaosbwcejumhoibpfsiexucnuivptahjwvwsbrbxlbpppcosmtlellikrfzol**

**rkaoyugikdeaicpziyaqctlhgrenmlqtyrkglrhhmzupunsiyelyodciwbindkgaveqkyzqahqqahqevitl
dxqfshbhdgjtuummolxbrftzdasrrlxicarpkluiijkqmmzafndbtyznfjvssnolnnjdfhcipwnshxthfrzerm
ofldtuptmwuvlltyfruxxopjheifabquwoskrodumhgqeuxkvynnwuruzfmgukkklhoxrxiiliozjmysjrba
ctyktyoozfgycjpojhjpeamspsccaxpkruqbeuthighwgxdbiqpqymbnworwnymhmaghtiqvtkbchiaq
mpwpkviwovgwlxxzeqjphmboevkityypmwpjvybjeewaufeqnudrznqmqtzkfmudwuhmglzdsmccy
bpfrlnroeixzyxadiprkwslmwfpcugqrivcwnzxnmamchngxmivdxsydinpvejbtavultglpyitrrxavrxql
uuvnnzedjctbkxmfptxxkxqclvduitomvvscyuahntbuoucsuoptaaxggdjzzvpdzyursrvtytulpluoxebj
nzetolkmkvddyjvfcfnplaksjgevlzqxnvdhrbajatepfqymrvcmdljzsggzteiulvyrqrxwbsiqojqofrdkxx
tbomzhveujazquflamgjgrerfbhhuvqgywgqoksvtzlawqlvycpckriodgtoyocmjmmnxcwyzjxhszfkj
dckjl**

4. Results & Conclusions

I provided in this section all the results and time complexities for each testcase.

Test case 1:

Numarul minim de operatii necesare: 7

Time Complexity: 0.000000

Test case 2:

Numarul minim de operatii necesare: 3

Time Complexity: 0.000000

Test case 3:

Numarul minim de operatii necesare: 6

Time Complexity: 0.000000

Test case 4:

Numarul minim de operatii necesare: 10

Time Complexity: 0.000000

Test case 5:

Numarul minim de operatii necesare: 50

Time Complexity: 0.000000

Test case 6:

Numarul minim de operatii necesare: 192

Time Complexity: 0.000000

Test case 7:

Numarul minim de operatii necesare: 731

Time Complexity: 0.003000

Test case 8:

Numarul minim de operatii necesare: 1307

Time Complexity: 0.000000

Test case 9:

Numarul minim de operatii necesare: 1490

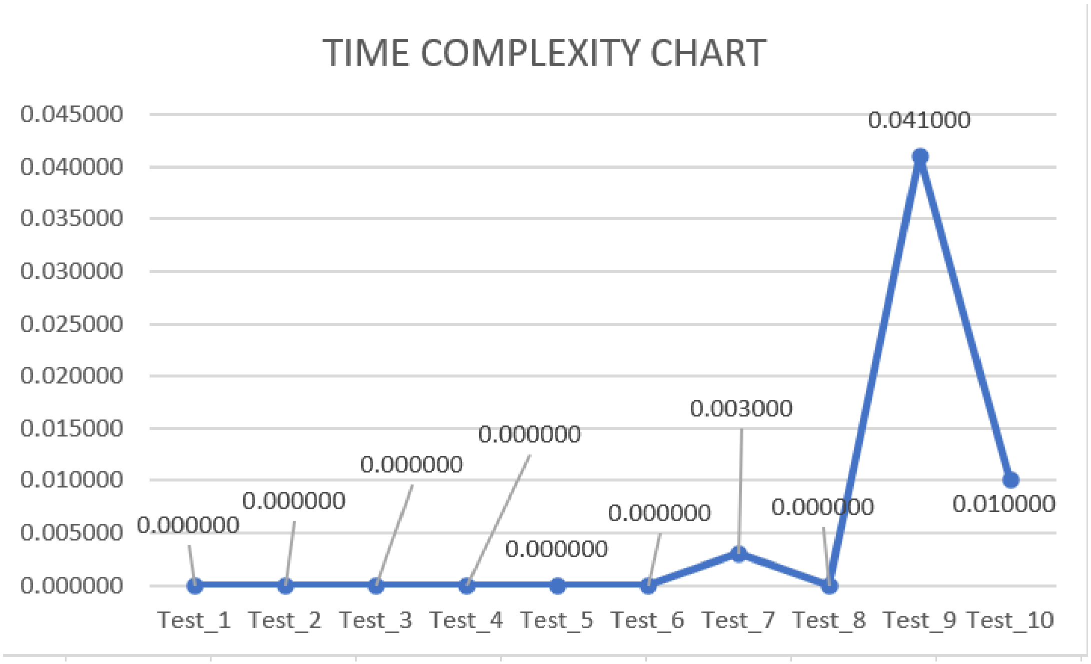
Time Complexity: 0.041000

Test case 10:

Numarul minim de operatii necesare: 2399

Time Complexity: 0.010000

The results presented here are the expected ones and the time complexities are also correct. Now, I am gonna present you the chart with the time complexities:



Conclusions :

- **For the first 6 testcases we have the cpu time complexity of 0 seconds because C has the power to compute the numbers up to 10.000 very quickly. After that bound, as you could see in the 10th case it take 0.04100 seconds which is also very well.**

- **So, as expected, the chart it is increasing only if length of the rule string is very big because we have to make more operations on a big rule string.**

Github link :

peca-ana-ace-ucv/HOMEWORK_ASSIGNMENT_AD (github.com)