DESCRIEREA INFORMALA A SOLUTIEI

Dupa citirea datelor de intrare: p, sirul clar si sirul criptat, respectiv retinerea lor in memorie, abordez 3 cazuri : p este mai mic strict ca 2 (si negativ), p=2 rezulta generatorul este 1, si p nu este numar prim . In aceste 3 cazuri se vor afisa mesaje corespunzatoare.

Daca p este numar prim, incep sa caut generatorul, plecand de la 2 pana la p-1 . Pentru fiecare generator retin in vectorul v valoarea puterii g la 0,1,2….p-1.Apoi, verific daca primele p-1 elemente ale vectorului sunt distincte, acesta incepand si terminandu-se cu 1 . In cazul in care, elementele vectorului sunt distincte, ies din generare astfel, gasind cel mai mic generator care va fi afisat .

Avand generatorul si vectorul cu puteri, pot sa trec la criptarea sirului clar, folosind parcurgerea caracter cu caracter a sirului clar.Fiecarui caracter ii caut pozitia in sirul cu literele ordonate alfabetic, pozitie care ii va corespunde lui v[pozitie\*4], care va reprezenta pozitia in sirul alfabet astfel, gasind corespondenta si trecand intuitiv prin transformarea in cifre .

Dupa criptare voi afisa sirul rezultat si voi trece la decriptare. Aici, ma folosesc de codul ascii al caracterelor (facand diferenta cu litera ‘A’ care are codul ascii 65), astfel afland mult mai usor pozitia cracterelor in sirul ordonat alfabetic (alfabet “ABCDF….”) .Valoarea rezultata o caut in vectorul v cu puteri si pozitia pe care o gasesc, va reprezenta ordinul din sirul alfabet al caracterului decriptat.

Exemple de date intrare in urma carora, algoritmul prezentat mai jos afiseaza:

1. Obtinerea generatorului :

p=-1=>Nu s-a dat p numar prim

p=2=>Generatorul g este 1

p=3=> Generatorul g este 2 (v: 1, 2, 1)

p=11=> Generatorul g este 2 (v: 1, 2, 4, 8, 5, 10, 9, 7, 3, 6, 1)

2.3. CODARE⬄DECODARE:

\*Pentru p nu este numar prim =>nu se afiseaza nimic decat mesajul de inceput: “Nu s-a dat p numar prim”, iar pentru un input corect de exemplu p=7 avem:

AAAA <=>BBBB

DAHI <=> GBBD

AFGHU<=>BFBBB

CODUL SURSA

.data

p: .space 4

g: .space 4

mg: .asciiz "Generatorul g este "

mesaj: .asciiz "Nu s-a dat p numar prim"

clar: .space 100

criptat: .space 100

v: .space 400

endl: .asciiz "\n"

chr: .byte 'A'

alfabet: .asciiz "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

mg2: .asciiz "Generatorul este 1"

.text

main:

#citre p

li $v0,5

syscall

sw $v0,p #salvare p in memorie

#citire sir clar

la $a0,clar

li $a1,99

li $v0,8

syscall

#citire sir criptat

la $a0,criptat

li $a1,99

li $v0,8

syscall

#verific daca p<2

lw $t0,p

li $t1,2

blt $t0,$t1,nueprim

#verific daca p este numar prim

li $t1,2 #$t1 divizorii

lw $t2,p #numarul p de verificat

div $t3,$t2,2 #conditia de oprire pentru diviozori

prim:

bgt $t1,$t3, exitprim #div>p/2

rem $t4,$t2,$t1 #$t4=p%div

beq $t4,$0,nueprim #p%div=0

addi $t1,1

j prim

nueprim:

#p%div=0 sau p<2

li $v0,4

la $a0,mesaj

syscall

li $v0,10

syscall

exitprim:

#daca p este numar prim

li $t0,2 #generator g

lw $t1,p #limita superioara a generatorului

loopgen:

beq $t0, $t1, exitgen #g merge pana la p-1

li $t3,1 #$t3= g^0 =1

li $t2, 0 # accesul la vector

sw $t3, v($t2) #adaug pe g^0 in vector

li $t4, 1 #$t4=puterea la care este ridicat g

loopputere:

beq $t4,$t1,exitputere #parcurg puterile

mul $t3,$t3, $t0

rem $t3,$t3, $t1 # calcul putere modulo p

addi $t2,4 # trec la urmatoarea locatie de mem. in v

sw $t3, v($t2) # salvez valoarea ridicarii la putere %p

addi $t4,1 #cresc puterea

j loopputere

exitgen:

#pt cazul p=2

#afisare mesaj=> generatorul este 1

li $v0,4

la $a0,mg2

syscall

li $v0,10

syscall

exitputere:

#cautare secventiala

sw $t0,g #mut in memorie generatorul curent g

li $t2,2 #sar peste primul element

#care va fi mereu egal cu ultimul =1

lw $t3,p

li $t4,4 #salturi in memorie

loop: beq $t2,$t3, exit # primele n-1 numere

addi $t5,$t2,1

addi $t6,$t4,4 #salturi in memorie elementul urmator

lw $t7,v($t4) #retin elementul curent de comparat

loop1: bgt $t5, $t3,exit1

lw $t8 , v($t6) #elementul urmator de comparat

beq $t7, $t8, exv #daca cel putin o pereche are numere egale

addi $t5,1

addi $t6,4

j loop1

exit1:

addi $t2,1

addi $t4,4

j loop

exv:

#trec la generatorul urmator pentru ca cel curent nu e bun

addi $t0,1

j loopgen

exit:

#afisare mesaj

li $v0,4

la $a0,mg

syscall

#afisare g

li $v0,1

lw $a0,g

syscall

#afisare endline

li $v0,4

la $a0,endl

syscall

#transformare clar criptat

li $t0,0

lb $t1, clar($t0)

caut : beqz $t1, excaut #parcurg sirul clar

li $t2,0

lb $t3,alfabet($t2)

alf: beqz $t3,exalf #caut in sirul alfabet

beq $t3,$t1,gasit #comparare caractere

addi $t2,1

lb $t3,alfabet($t2)

j alf

exalf:

addi $t0,1

lb $t1, clar($t0)

j caut

gasit:

lw $t4, p

rem $t2,$t2,$t4 #am gasit la pozitia $t2%p

#accesez in v locatia de memorie t2\*4

mul $t2, $t2,4

lw $t5, v($t2) #puterea g^$t2/4 ii corespunde

#pozitiei literei din alfabet

lb $t6,alfabet($t5)

#print byte

move $a0,$t6

li $v0,11

syscall

j exalf

excaut:

#afisare endline

li $v0,4

la $a0,endl

syscall

#lucrez cu codul ascii diferenta cu caracterul 'A'

#parcurgere sir criptat

li $t0,0

lb $t1,criptat($t0) #caracter curent sir 1

lb $t2,chr #caracterul 'A' cu codul ascii 65

lw $t5,p

lp: beqz $t1, exitlp #parcurg sir criptat

sub $t3,$t1,$t2 #diferenta coduri ascii

rem $t3,$t3,$t5 #poz %p

#il caut in v pe t3

li $t4,0

li $t7,0 #sar in locatii de memorie

lp1: beq $t4,$t5,exitlp1

lw $t6,v($t7)

beq $t6,$t3,af

addi $t4,1

addi $t7,4

j lp1

exitlp1:

addi $t0,1

lb $t1,criptat($t0)

j lp

af:

#afisez litera de la pozitia $t4

lb $a0,alfabet($t4)

li $v0,11

syscall

j exitlp1

exitlp:

li $v0,10

syscall