Project 2

Group 8: Måns Lindeberg & Ludvig Lifting

30 november 2021

Home Exercises

For **Home Exercise 1-3** please see the hand-written solutions in appendix.

Lab Exercise

1

See solution in *Figure 1*.

```
R.<x> = PolynomialRing(GF(2))

p_1 = x^23 + x^5 + 1
p_2 = x^23 + x^6 + 1
p_3 = x^18 + x^3 + 1



print(f'{p_1} is irreducible: {p_1.is_irreducible()}')
print(f'{p_1} is primitive: {p_1.is_primitive()}')
print(f'{p_2} is irreducible: {p_2.is_irreducible()}')
print(f'{p_2} is primitive: {p_2.is_primitive()}')
print(f'{p_3} is irreducible: {p_3.is_irreducible()}')
print(f'{p_3} is primitive: {p_3.is_primitive()}')

T.<y> = PolynomialRing(GF(7))

p_4 = x^8 + x^6 + 1
print(f'{p_4} is irreducible: {p_4.is_irreducible()}')
```

```
x^23 + x^5 + 1 is irreducible: True
x^23 + x^5 + 1 is primitive: True
x^23 + x^6 + 1 is irreducible: False
x^23 + x^6 + 1 is primitive: False
x^18 + x^3 + 1 is irreducible: True
x^18 + x^3 + 1 is primitive: False
x^8 + x^6 + 1 is irreducible: False
```

Figur 1: SageMath solution for LE1.1 LE1.2 and LE1.3.

2

See solution in *Figure 2*.

3.1

The cycle sets of

$$p(x) = x^{23} + x^5 + 1$$

over *GF(2)*can be seen in *Figure 3*.

```
R.<a> = GF(2^18, name='a', modulus=x^18 + x^3+ 1)

print("a¹ has period:")
print((a^1).multiplicative_order())
print("")
print("a² has period:")
print((a^2).multiplicative_order())
print("")
print("a³ has period:")
print((a^3).multiplicative_order())
print("")
print("a³ + a has period:")
print((a^3 + a).multiplicative_order())
```

```
a¹ has period:
189

a² has period:
189

a³ has period:
63

a³ + a has period:
262143
```

Figur 2: SageMath period calculation. Provides solutions for LE2.1, LE2.2, LE2.3 and LE2.4.

3.2

As can be seen in *Figure 4*, the first two irreducible factors of

$$x^{23} + x^6 + 1$$

over *GF(2)* is primitive and the last is not, but can quickly be calculated by finding out the period. The cycle sets of the individual factors can be seen in *Equation 1,2 & 3*.

$$S_1 = [1(1) \oplus 1(7)] \tag{1}$$

$$S_2 = [1(1) \oplus 1(15)] \tag{2}$$

$$S_3 = [1(1) \oplus 3(21845)] \tag{3}$$

$$\begin{gathered} S = S_1 \times S_2 \times S_3 = \\ = [1(1) \oplus 2(7) \oplus 2(15)] \times [1(1) \oplus 3(21845)] = \\ = [1(1) \oplus 2(7) \oplus 2(15)] \times 1(1) \oplus [1(1) \oplus 2(7) \oplus 2(15)] \times 3(21845)] = \\ = 1(1) \oplus 2(7) \oplus 2(15) \oplus 1(1) \times 3(21845) \oplus 2(7) \times 3(21845) \oplus 2(15) \times 3(21845) = \\ = 1(1) \oplus 2(7) \oplus 2(15) \oplus 3(21845) \oplus 6(152915) \oplus 30(65535) = \end{gathered}$$

The final cycle set is given in *Equation 4*.

$$S = 1(1) \oplus 2(7) \oplus 2(15) \oplus 3(21845) \oplus 6(152915) \oplus 30(65535) \tag{4}$$

```
R.<x> = PolynomialRing(GF(2))
p_1 = x^23 + x^5 + 1
print("We know p_1 is primitive so the cycle set becomes")
print(f"Cycle set for p_1 = 1(1) + 1({2**23 - 1})")
```

```
We know p_1 is primitive so the cycle set becomes
Cycle set for p_1 = 1(1) + 1(8388607)
```

Figur 3: SageMath cycle set calculation for LE3.1.

```
R.<x> = PolynomialRing(GF(2))

p_2 = x^23 + x^6 + 1
F = p_2.factor()

factors = [f for f in F]
cycle_sets = []

for factor in factors:
    if factor[0].is_primitive():
        print(f"Primitive factor: {factors[0]}")
        cycle_set = ((1,1), (1,2**factor[0].degree() - 1))
        cycle_sets.append(cycle_set)
    else:
        print(f"Not primitive factor: {factor[0]}")
        T = factor[0].degree()
        while True:
            if (factor[0]).divides(1 + x^T):
                print(f"Period of {factor[0]}: {T}")
                cycle_set =((1,1), ((2**factor[0].degree() -1)/T, T))
                cycle_sets.append(cycle_set)
                break
            else:
                T += 1

for index, cycle_set in enumerate(cycle_sets):
    print(f"Cycle set for polynomial {factors[index]}: {cycle_set}")


    ### Calculate the cycle set
```

```
Primitive factor: (x^3 + x + 1, 1)
Primitive factor: (x^3 + x + 1, 1)
Not primitive factor: x^16 + x^15 + x^13 + x^12 + x^8 + x^6 + x^4 + x^3 + x^2 + x + 1
Period of x^16 + x^15 + x^13 + x^12 + x^8 + x^6 + x^4 + x^3 + x^2 + x + 1: 21845
Cycle set for polynomial (x^3 + x + 1, 1): ((1, 1), (1, 7))
Cycle set for polynomial (x^4 + x^3 + 1, 1): ((1, 1), (1, 15))
Cycle set for polynomial (x^16 + x^15 + x^13 + x^12 + x^8 + x^6 + x^4 + x^3 + x^2 + x + 1, 1): ((1, 1), (3, 21845))
```

Figur 4: SageMath cycle set calculation for LE3.2

4

Using the SageMath command *is_primitive* we find that a suitable connection polynomial of degree 4 in GF(5) is as in *Equation 5.*

$$2x^4 + 2x^2 + x + 1 \tag{5}$$

De Bruijn Sequence

The 10'003 digit De Bruijn sequence produces by the code in *Listing 1*:

0006685352771452414568847166217143086852947912803248779463870060223775548177236120189
5519198428322275872617842940207759351970190304986506356317110187891815900533409566182562
3703447688363682454340677628195417411278991508511911007986164881084200878747199116104168
8647177015110295871537540724305895393683084142898525059425203389653646542713449898462574

2531136987073673361422855908378305124388794809710541417687283880283443768615358306204185
9607066245042097984909732642406555271560494230987807168427242478651715933932415698194882
1640025996490581294412795718057328134169994519922913438595490860284023565836255215212477
5719187125300056680807721902419568392661262143586807497417303298774918820510228775093672
2811206895064693473322775827167347440257754806920640309986051851362110687846365405033459
5616375128203497683818632904345677173690462411778946058016411057981619831534205878292694
1611046688192672060110795826087045224355890848633534147898070554470203889608196047213499
8939175247031186982528623811427855453873350124888749359215041467682738830733448768160853
3512046859152561290042597939459237142456550726510944235987352663472242978606265438432465
6936498326140075991945531744417795263552373134669949069427413488590945810734028565381750
2602129775264682170300556635357226402469563847611712148586352992462303798729468325010278
7705486227311256890519643923327775372662392440757709356425140359981506801812115687391860
4500339595161870173203997638368137404395672628640912416778491553061411557936169336034255
8737476446111096683647622510115795371582090224855845398138034197893525504920208889153691
0922139998484670292031686937078128311477850908823800129888294854260041967637288335233498
7636158038012096854607511740047597484954282142956505276015444285982807613922247978151760
4834329656481993371140575946495036244467790718502823139669494564472413988545495315234078
5608367007102179770719632620305556180852271402969518397116212198581807942912308798274963
3700107787250981272311756845069148423377770827612842445757254851470140859936056306312165
6828468109000389590616820623208997183863182404895627178145412466773946503511416557481664
3810347558282971491111596638197127010165790826532540229855390893183034697848075009420258
8846086415422189993939620742036686482573173311977805458328300179883749804710046967182783
3802339987181653083012596809157016240097592939904732147956050771060444785937357118422297
9736067109334379651936943821145575491990081244967745268007323189664949514922418988090990
3602345785153862052102679725269137120355551635802721407969063892161212698536357447412358
7937299138200157782705931722316756390564193423877725377117342495752709801920145859481551
3513126656373963154000889545166325123258992638813632409895172673190412966728496008011466
5529366148310397553737921941116596183692172010665745376037040279850845843633039697393570
0544207588391581460422689948489125242086681937523623316977350953373300679838299309210096
9626377338302389982636603533017596354652061240597547489409232197951505721510449785482852
1634227979281562154334879606486448321195570946940531249967290763052323689619499019422468
9835459408102395780608812502107679270764182120855506185307221457964518842611217698081852
4924128587482794183200657737255436222366751845514643428877270872162342995707259306420195
8549365018013176651828913604005889090661370123758947188318132459890627623640417966273991
0530119665074861193310897508287426441166591638642622015665290871082040779805395348133089
6928485205044257583846531910427689493984170242586636487028123366972805903823305679383794
3542105969171872383302889937186108033067591809602511245597092984454232697906055226010499
7809378026134277974736512604339879151981493321695525496445031299962745713502328689164994
0644229689380954453102895735158317002157674725714632125855051680352221957919068347111267
6935368029424178582937744633205657282750481222866706395019143478872725822612347995252754
3514206958094860063013676606378418104055884545611820128758492683363132959845177128140467
9617289415030169660529811643315897053782471441666546188147122065660745821532045779350890
3931335896473980250044757538396036410477684948934620247586181982073123866927355408323355
6748387448042155964626822833307889482681153033567546359107011295592547934904237697451550
2710109997354873071134777929286017104389874606931943326695070991490031799917295218002378
6846199445144279684835904903107895280653362002657629275219132175850506630802226957464563
3921117676480863074424678537487249133255652737700931227866251890064143978827275327112397
9907077048014256953549810513018676151873463104555839095116320178753947633813137959390672
1731409679162784460030669615079316143365892508732921446666091683192122565615295326032095
7748058408431385891928930700049757083891081410977639498439120297581636932523128866472850
4533238556293882493042655919176327333357884937631603038567091854152011795547097439404287
6929065007210159992809823521139777474781062104889829156436443376690525941940036799462790
2630028786391694490144779639385409403157890735603812007657174770264132675805056135302276
9529195138421167671935813524429678082982294133755607287205431277861706840514148978372770
3721128979452572093014756908099315013068671606823913109555384590161320678708497138313187
9548456226231459674617734910035669160574361143865847058237421496661546633642127565160790

3710325957293553453431885846478435200099752538841531415977184993484120797536186437023178
8619278009033288551748832943047655464671372333857839487136103088562546804602016795092592
4844047876474560052210659947359328021189772929731512109889374651481443876645075446440086
7949177407130078781846644940149779184880454403657845285108312057652629720714137675350551
1803027769074690183421667626485318024479673537932744138755152782250431777816256345014198
9738277208221178974907522543019756453594360013568626156328413159550839540611325678253992
1833136879093951271231959629167239410085664615524811148865392553282421996616096138142177
5606157408210375952748503903436885391973480200599707088346031465972639943934125797081681
4820236788164773054033788506298337443097650919621822338857384982181103588517096309102066
7905475429344097871929510502215659492854373021689727479236012159884829601931448876190570
4914405867494672452130578736396149440199774639830904408657390780153312557607179225214187
6708055016303077764529640633426667171980363024979628087437244188750607732700436777361751
3900146989283772253221678929457027043069751908544810018568171651373413659505389045111375
6737089426333186874548901721236959174662284410585619165029311198860847503732426996161591
1831426775151652453210875907298008403486880846923930205599252583391031965927189448434175
7925366319320286783619723504038788051793382443597605469126322388852839932631108588062591
3541025667450970474344597826479015002265654947804823026689272974281012659839379106431498
8716455209414455862949622902135578281891194440699729189335404458652845730603317557152674
2702146876253550061303577719079145133476662626930813029979173582482244688705157237200486
7728167018400196984738722703226678474952072043569706458049310068563626601823418659050884
0901118756282584471333686829098406221286954629612734415585164660074311698815397008232476
9916165416331476770606602903215875452793053403986835396428430255594707533841036965472684
4934346757470861364320786738169228004088783506743832448597150964171322888807389437131158
5835175418041075662905920924349597371974060002765609497309323076684727924731017659384874
1514319988261950254414955817499127402185573736841644445699274684380404958607395235103367
5526076247202196871708500511308577264574190133976617176435313079974628532932249688250652
2822009867273662063400696939288227203276673929902522048569251953094310568518176106323468
6545058345401168751737534921338686374593451221786909179117234465580619610524316698360892
0532329769461660461331976725156107403265870907743503408986380891473430755549257038341086
9609276349434396752925811814325786283664273004588738056248332498592605914621327888352884
4821316585380670463041575617455425424399592826924510007765154992354323576639277429231067
6548398246014369983716900704419955362994172402685528286346144495694729634830409958152890
2801038675071571292202696826258005011358572719524640138976162671480313579929178037432299
6837056027322059862728612513405696484783272203776628479407022098564706903544315568063671
1513239686090553390401668706287039421388681829543901226786454674162234965535169115024366
6938158425032379764916610911336976270651152403765825457248003458981835841923435755094752
0833415869154771394434896707475316314375781738614723009588283551293332998547155419121377
8838078349321366580835620913046575162950470424899547376429010057760609942804328576184772
4742315676093893291014869938266405204469950817944622407685073781391144995649279139330459
9536078407301088670526521742207696371753050011858527269029140188971617621930318579474673
0824327996382551072322559817278117013455691939733722208776173974452022598519256408044365
5635186216013289681545503840406668251782084421888636379048401276781909624612239965080664
1600248666483653470032879719466115411386971725601602408765370952293003958936385346423485
7505497025333465864609721844439896252970361314875736288119223059583738501743337998092650
4641218778383573394321866535385125413096570617900920429899092871474010557715159447304378
5716397229242365671548843741019869483761450204969905367449122457680528731841149995194774
1843309599081573452301588625076026242257691826703500016858072764074140688926167126430368
5749296235324377991837501522327559362773162013955646489238222258771628924902027598064751
4530448655180681261013789636095008340456663706732534426888181874093401776736459129112289
9605356146100298661938603920037879264961160411886926275106102458760825902743008958481880
3914239857050992070333965819159226344489891707920811319875281783164223559538288006243387
9935476009141268773838523844326866080880170413596525167405420479894547821924015557260654
4923048785261892274242865626098348241069864938711900209969450862494122957635078236341199
990649724634335959453652390230658817057107124275764637620800000

Listing 1: debruijn$_{g}$*en.py*

```
import time

def LFSR5(gf: int = 5):
    STATE = [0,0,0,1]
    INIT_STATE = STATE.copy()
    SEQUENCE = []
    counter = 0

    while True:
        if STATE == INIT_STATE and counter != 0:
            STATE = [0,0,0,0]
            SEQUENCE.append(STATE.pop(0))
            break
        else:
            counter +=1
            new_val = (2*STATE[0] + 2*STATE[2] + STATE[3]) % gf
            SEQUENCE.append(STATE.pop(0))
            STATE.append(new_val)
    return SEQUENCE

def LFSR2(gf: int = 2):
    STATE = [0,0,0,1]
    INIT_STATE = STATE.copy()
    SEQUENCE = []
    counter = 0
    while True:
        if STATE == INIT_STATE and counter != 0:
            STATE = [0,0,0,0]
            SEQUENCE.append(STATE.pop(0))
            break
        else:
            counter +=1
            new_val = (STATE[0] + STATE[3]) % gf
            SEQUENCE.append(STATE.pop(0))
            STATE.append(new_val)
    return SEQUENCE

def is_unique(sequence):
    unique_sequences = []
    for i in range(0, len(sequence) - 3):
        seq = []
        for j in range(0, 4):
            seq.append(sequence[i+j])
        if seq in unique_sequences:
            return False
        else:
            unique_sequences.append(seq)
    return True

if __name__ == '__main__':
    lfsr5 = LFSR5()
    lfsr2 = LFSR2()

    lfsrseq2 = []
    lfsrseq5 = []

    seq = ""
    for i in range(0,626):
        lfsrseq2 += list(lfsr2)
    for i in range(0,17):
        lfsrseq5 += list(lfsr5)
    for i in range(0,10003):
        if lfsrseq2[i] == 1:
            seq +=(str(lfsrseq5[i] + 5))
        else:
            seq+=(str(lfsrseq5[i]))
        f = open("seq.txt", "w")
        f.write(str(seq))
        f.close()
```