Övning 6 - Tillämpad datalogi 2012

```
0 # coding: latin
```

Sammanfattning

Idag gick vi igenom **komprimering**, **kryptering** och **testning**. Specifikt löste vi komprimeringproblem med huffmankodning och Lempel-Ziv, funderade på testproblem till lab 5, samt krypteringsproblem rot13, transpositionschiffer och RSA.

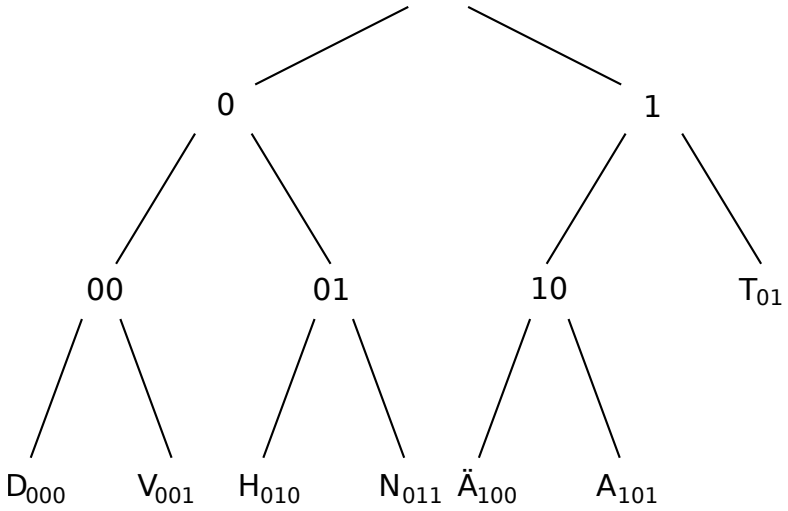
- Some imports, ignore

```
20 import sys, os
21 sys.path.append(os.getcwd())
```

Komprimering, kryptering och testning.

1. Smittskydd

Du har fått ett mail som innehåller tips mot spridning av virus. Informationen är komprimerad med ett Huffmanträd där nollor motsvarar vänster och ettor motsvarar höger (se figur, "T" kodas t.ex. som 11) ***Vad står det i meddelandet*** **0101010110001100110011111?**




```
010  101  011  000  11  001  100  11  11
H     A    N    D   T   V    Ä   T   T
```

Om Huffmankodning (från föreläsning 11)

Om vi vet hur vanliga olika tecken är i texten kan vi ställa upp en tabell där vi för varje tecken kan ange sannolikheten för att ett visst tecken ska dyka upp. I David A. Huffmans metod kodar man varje tecken med ett binärt tal, där vanligare tecken får kortare koder. Algoritmen som beräknar vilket tecken som ska få vilken binär kod går ut på att man ritar upp ett binärt träd, där varje tecken ses som ett löv. Sedan numrerar man t rädets grenar med 0 och 1 och följer trädet från roten ut till varje löv för att se koderna.

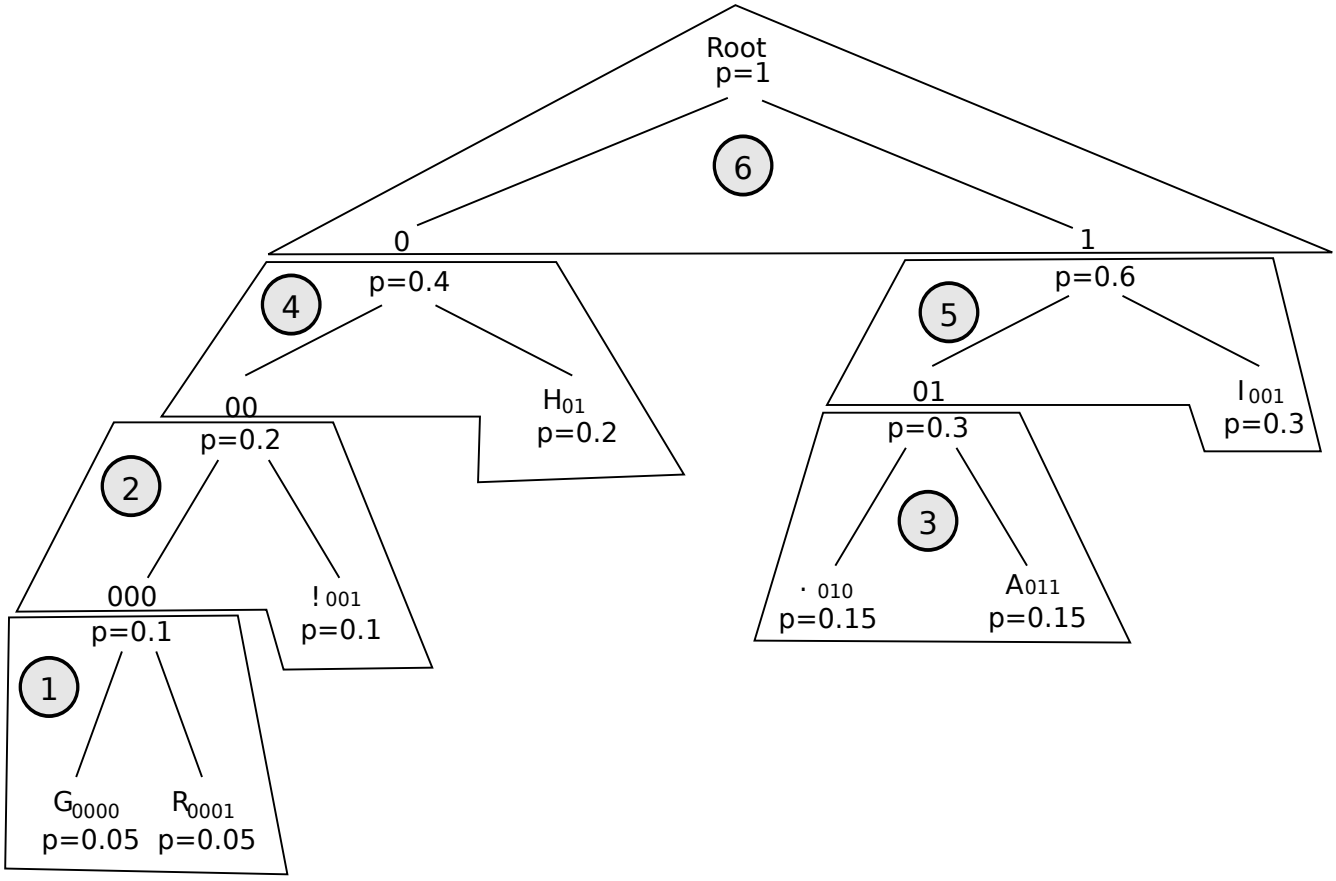
1. Sortera tecknen som ska kodas i stigande sannolikhetsordning.

2. Rita grenar från de två tecken som har lägst sannolikhet och låtsas att vi har ett nytt tecken med sannolikhet som är summan av deras sannolikheter. Numrera ena grenen med 0 och andra med 1.
3. Upprepa punkt 2 tills alla tecken kommit med. Roten bör få sannolikhet 1.
4. Börja från roten och följ grenarna ut till ett löv. Samla nollor och ettor på vägen - dessa ger koden för lövets tecken

Med följande frekvenstabell

Tecken	Sannolikhet
G	0.05
R	0.05
!	0.1
A	0.15
H	0.2
I	0.3

Får vi följande Huffmanträd, där vi skapade i ordning 1-6:

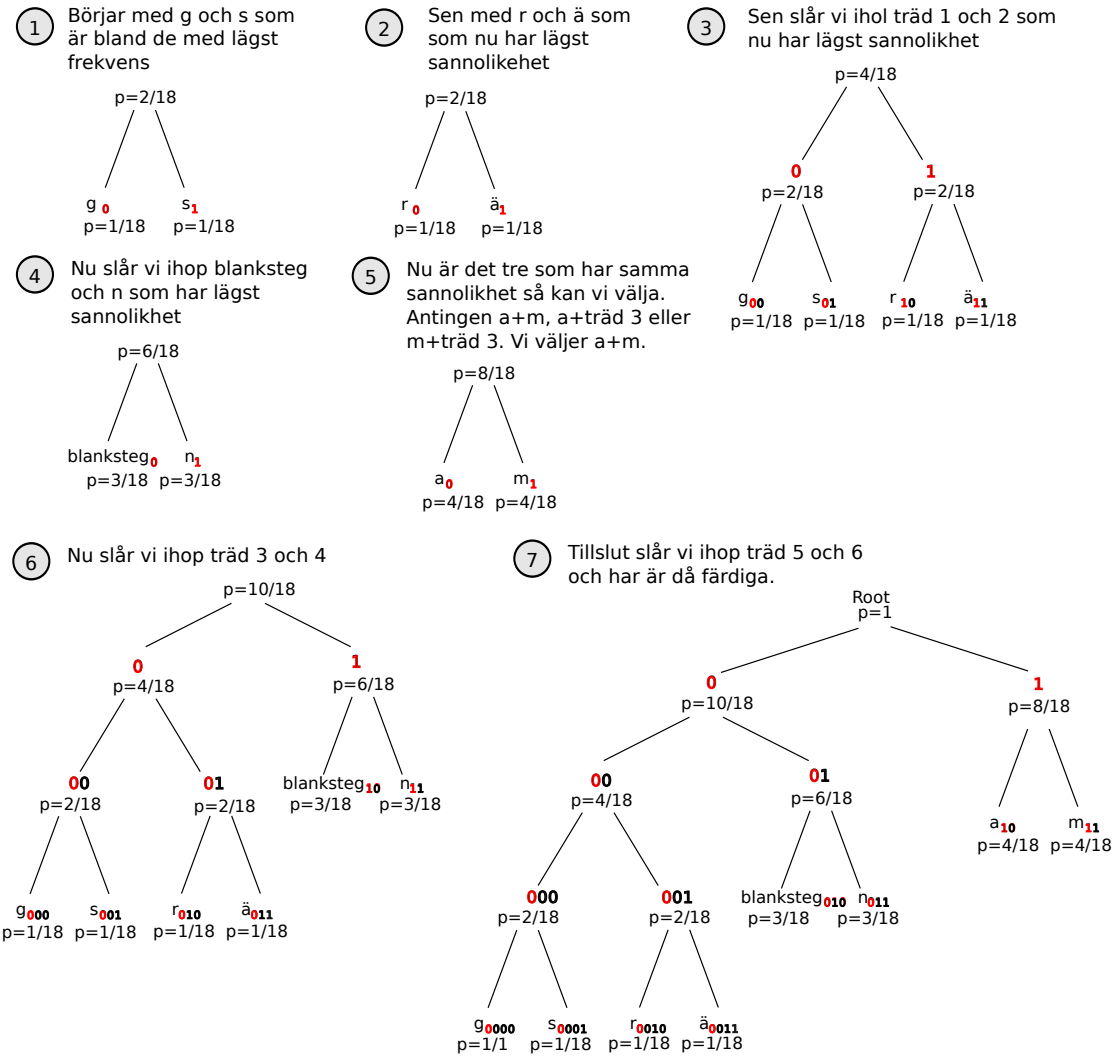


2. Huffman för Havamal

"man är mans gamman" kan man läsa i Havamal. Konstruera en huffman- kod för tecknen i detta uttryck (rita trädet) och skriv sedan upp det i kodad form.

Vi börjar med att räkna ut sannolikheten för varje bokstav i utrrycketvi utifrån att dess frekvensen dela på total och sorterar i stigande ordning. antal bokstäver.

Bokstav	g	s	r	ä	blanksteg	n	a	m
Frekvens	1	1	1	1	3	3	4	4
Sannolikhet	1/18	1/18	1/18	1/18	3/18	3/18	4/18	4/18



Vi får följande Huffmankod (OBS Huffmankoden hade sett lite annorlunda ut om vi valt en annan kombo i steg 5. Dock hade den varit lika optimal):

Bokstav	g	s	r	ä	blanksteg	n	a	m
Huffmankod	0000	0001	0010	0011	010	011	10	11

Koden för "man är mans gamman" blir:

```
01  11  011  010  0011  0011  010  01  11  011  0010  010  0000  10  11  11  01  011
m   a   n    _    ä     r     _    m   a   n    s     _    g     a   m   m   a   n
```

Totalt blir meddelandet 50 bit långt.

```
01110110  10001100  11010011  10110010  01000001  01111010  11
8         8         8         8         8         8         2   =50 bits
```

3. Enkel kryptering

Kryptera lösenordet SIMSALABIM med

1. rot13

2. Transpositionschiffer

1)
Med **rot13** skiffer förskjuter vi varje bokstav 13 steg, så a blir n, b blir o ,..., z blir m. Vi kan skriva följande kodtabell:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
N O P Q R S T U V W X Y Z A B C D E F G H I J K L M

Vi får då koden FVZFNYNOVZ

2)
För att skapa ett **transpositionschiffer** ska vi skriva meddelandet i n x m matris och sen bilda meddelandet utifrån kolumnerna. Meddelandet är 10 långt. Låt oss sätta n=3 och m=4. Matrisen blir då:

S	I	M	S
A	L	A	B
I	M		

och vi får meddelandet SAIILMMASB från kolumnerna (andra varianter är också möjliga med andra värden på n och m)

4. Testning

I labb 6 ska ni göra ett program som kontrollerar syntaxen för en molekylfor- mel. Skriv upp en samling indata att testa programmet med!

- Giltiga formler
- Ogiltiga formler
- Tomma strängen?
- både en- och tvåteckens siffror
- Tänk på att ej korrekta kemiska formler är tillåtna.

5. RSA-kryptering (ur kursboken, uppg 2 s. 125)

Du vill hålla ditt turnummer (18) hemligt, och har därför bestämt dig för att kryptera det med RSA. Välj två primtal större än 500 och använd dom för att kryptera turnumret.

RSA (från Ron Rivest, Adi Shamir och Leonard Adleman som först publiceraden) kräver att vi skapar en **offentlig nyckel** och en **privat nyckel**. Den offenliga nyckeln delas ut till alla som ska kunna skicka kryptera meddelanden. medans den privata nyckeln ges till den som ska kunna dekruptera meddelandena. e och d beräknas på fäljande sätt:

- Välj två stora primtal, p och q.
- Beräkna **n** = p*q.
- Beräkna f = (p-1)*(q-1).
- Välj **e** så att det inte har några gemensamma faktorer med f (dvs gcd(e,f) ska bli 1).
- Beräkna **d** som den multiplikativa inversen till e modulo f, dvs så att e*d modulo f = 1.

För att **kryptera** ett meddelande konverteras det till ett tal (t.ex. konkatenera ASCII-koderna). som sedan delas upp i lagom stora delar m_i. Delarna krypteras sen genom att ta m_i^e modulo n vilket resulterar i de kodade delarna c_i.

För att **dekryptera** meddelandet tar vi c_i^d modulo n.

Kryptering:

Vi börjar med att bräkna **e**

$$\left.\begin{array}{l} p = 587 \\ q = 719 \end{array}\right\} = p * q = 422053$$

$f(n) = (p-1) * (q-1) = 420748 = 2 * 2 * 293 * 359$

$e = 17$ väljs så att $1 < e < f$, och e och f är relativa primtal, dvs de har inga faktorer gemensamma, eller $gcd(e, f) = 1$.

Nu är vi egentligen klara med uppgiften. Krypteringen blir: 18^{17} **modulus** $422053 = 161344$

Dekryptering:

Om vi nu även vill dekryptera meddelandet behöver vi skapa nyckeln **d**

$d = 222749$ så att $d*e \equiv 1 mod(f(n))$, dvs $d*17 \equiv 1 mod(420748)$ sp, även går att skriva som $d*17 = 1+n*420748$. Detta är en ekvation med två obekanta. Vi vill hitta det n för vilken d är ett heltal. Med lite testande finner vi att n=9 ger den minsta heltalslösningen $d = 222749$.

Dekrypteringen blir: 161344^{222749} modlus $422053 = 18$

6. Lempel-Ziv

I denna uppgift ska du avkoda ett meddelande som komprimerats med Lempel-Zivs metod. Komprimeringen går till så att komprimeraren lagrar en lista med strängar som från början enbart innehåller tomma strängen.

Komprimeraren läser tecken för tecken från intexten den längsta sträng som ligger i strängtabellen. Sedan skrivs index för denna sträng i tabellen ut, följt av nästa tecken i intexten.

Strängtabellen utökas med strängen plus nästa tecken. Därefter läses åter tecken för tecken från intexten.

Så fort en sträng inte finns i strängtabellen läggs den alltså till.

Metoden kan i (icke-optimerad) kod beskrivas så här:

```
281 def lzw(text):
282     table = Table()
283     q=Queue()
284     for c in text: # spara texten tecken för tecken i en kö
285 
286         q.put(c)
287     s=""
288     table.add(s)
289     kodtext="" # här sparas den kodade texten
290 
291     while not q.isempty():
292         c=q.get()
293         if table.exists(s+c):
294             s=s+c
295         else:
296             kodtext+=str(table.code(s))+c
297             table.add(s+c)
298             s=""
299     if not s=="":
300         kodtext+=str(table.code(s))
301     return kodtext, table
```

Klassen Table stöder inläggning av strängar, kontroll av om en sträng finns i tabellen och möjlighet att ta reda på index hos en speciell sträng.
Index bestäms av ordningen strängarna lades in i, där den första strängen har index 1.

Uppgiften är att avkoda följande text, där alfabetet består av versaler och mellanslag kodat som _ .

1S1T1O1R2T4C1K1H4L1M2 _ 6O1L3A1 _ 9O14M1A5

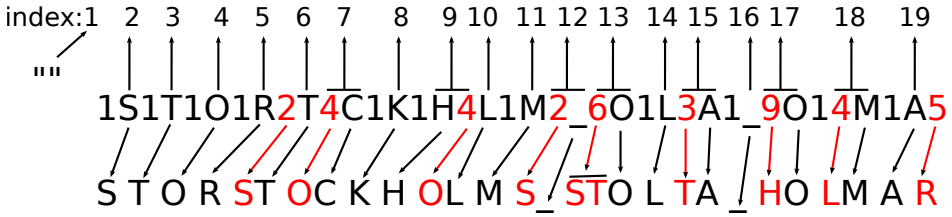
Det vill säga: **textitvad är t om print lzw(t) ger ovanstående som utskrift?**

Ge också en tabell med strängar och index som motsvarar tabellen i koden!

Algoritm
Lempel-Zivs algoritmen ovan i ord:
0) Vi börjar med s="".
1) Ifall s+c existerar gör vi inget med tabellen och låter istället s bir s+c och kollar igenom
2) Om s+c inte finns i tabellen lägger vi till det i tabellen, nollställer s, (s="") samt ökar på kodtext med kod för s plus c.
3) Gå till 1 och ta ut ett nytt tecken ur c kön tills kön är slut.
4) Om inte s är tom lägg till koden för s.

Avkodning

För att avkoda ett meddelande bygger vi upp tabellen utifrån meddelandet på följande sätt:



Den resulterande tabellen blir då:

index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
code	""	S	T	O	R	ST	OC	K	H	OL	M	S _	STO	L	TA	_	HO	LM	A

Med lzw får vi:

```
325 from my_utils import Table, Queue
326
327 text="STORSTOCKHOLMS STOLTA HOLMAR"
328 kodtext, table = lzw(text)
329 print table
```

```
{'': 1, 'M': 11, 'OL': 10, ' ': 16, 'H': 9, 'K': 8, 'STO': 13, 'OC': 7, 'L
    ': 14, 'O': 4, 'ST': 6, 'HO': 17, 'S': 2, 'R': 5, 'A': 19, 'T': 3, 'S '
    : 12, 'LM': 18, 'TA': 15}
```

```
330 print kodtext
```

```
1S1T1O1R2T4C1K1H4L1M2 6O1L3A1 9O14M1A5
```