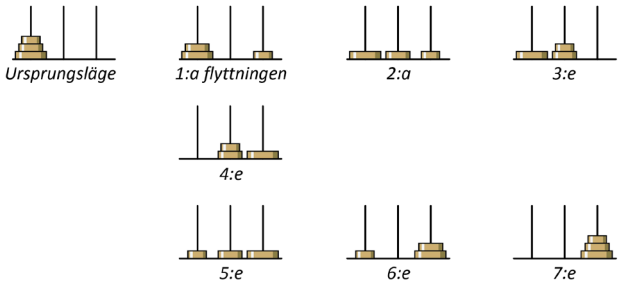
# Stora tal och STORA tal

Lite uttjatade är de kanske, skrönorna om tornet i Hanoi och schackbrädet med sina riskorn, men vi tar oss här ändå an dem då de genererar en del matematik som passar in på flera av våra högstadie- och gymnasiekurser.

## Vi låter tornet anropa sig själv

Tornet i Hanoi är väl närmast en slags leksak, och som består av tre stavar och därtill ett antal brickor. Dessa brickor, uppstaplade på en av stavarna, ska flyttas över till en av de andra stavarna. Ingen större bricka får läggas på en mindre. Frågan är hur många flyttningar det blir för ett visst antal brickor.

Med endast 1 bricka blir det så klart bara 1 flyttning och man ser rätt snart att med 2 brickor krävs 3 flyttningar. Vi provar med 3 brickor.




Med 3 brickor blev det 7 flyttningar. Bottenplattan flyttades bara en gång, och dess flyttning bildar en slags mitt: innan denna flyttning måste ett mindre torn av två brickor byggas upp. Sen flyttas bottenplattan. Därpå byggs det mindre tornet upp på nytt. De 7 flyttningarna kan därför skrivas: $7 = 2 \cdot 3 + 1$, där 1:an är bottenplattan och 3:an det antal flytt som krävdes då tornet bestod av enbart 2 brickor. Om vi inför symbolen $A$ för antalet flyttningar, och låter ett index-nummer få ange antalet brickor, kan detta samband skrivas:

$$A_3 = 2A_2 + 1$$

Säg att vi har 19 stycken brickor och att vi ingenting vet om antalet flyttningar. Men med det angreppssätt som vi här har påbörjat, inser vi att problemet med 19 brickor kan överföras till problemet med 18 brickor. Följande måste helt enkelt gälla: $A_{19} = 2A_{18} + 1$ trots att vi inte heller vet nånting om $A_{18}$. Detta att reducera ett problem till en mindre variant av samma problem, är kärnan i den kraftfulla idén om *rekursion*. Det allmänna fallet blir:

$$A_n = 2A_{n-1} + 1 \qquad A_1 = 1$$

Vi skriver nu några rader kod i programspråket Python, och vi låter koden få formen av just en rekursion, en kod som därmed blir både kort och i linje med våra resonemang.

Tornet i Hanoi rekursivt

```
n = int(input("Ange antal brickor: "))
def A(n):
    if n == 1:                          # Basfallet
        return 1
    else:
        return 2*A(n – 1) + 1           # Rekursionen
print A(n)
```

```
Ange antal brickor:  19
524 287
```

## En explicit formel skulle vara ett trevligt komplement

Skulle man bara ha en penna till hands, lockar det inte att såsom programmet behöva stega sig fram de 19 stegen för att till slut få fram 524 287. Så när vi nu har vårt program, skriver vi så klart ut ett antal värden för att från dem finna eventuella mönster. Några körningar till, och vi får lite råmaterial:

| $n$ | $A$ |
|---|---|
| 10 | $1023 = 1024 - 1 = 2^{10} - 1$ |
| 11 | $2047 = 2048 - 1 = 2^{11} - 1$ |
| 12 | $4095 = 4096 - 1 = 2^{12} - 1$ |

*Maskinens bidrag* — *Människans bidrag*

Ur strukturen som framträder infinner sig en hypotes: $A_n = 2^n - 1$.

Om den här formuleringen är korrekt, har vi nu Tornet i Hanoi på sluten form. Här hade varit platsen för ett induktionsbevis.

## Utkast till ett bevis

Då induktion bara förekommer sporadiskt på gymnasiet, redovisas här ett alternativ som explicit genomför den uttömningsprocess som det är frågan om. Vi låter $n = 5$:

$$A_5 = 1 + 2A_4 = 1 + 2[1 + 2A_3] = 1 + 2 + 2^2A_3 = 1 + 2 + 2^2[1 + 2A_2] =$$
$$1 + 2 + 2^2 + 2^3A_2 = 1 + 2 + 2^2 + 2^3[1 + 2A_1] = 1 + 2 + 2^2 + 2^3 + 2^4 \cdot 1$$

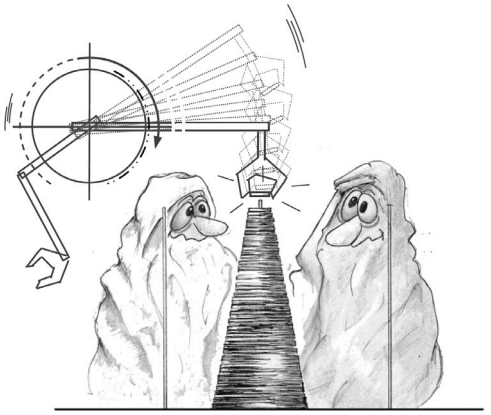
och vi ser att vi har att göra med en geometrisk summa med kvoten $k = 2$ och antal termer $n = 5$.

Den välbekanta summaformeln ger: $A_5 = 1 \cdot (2^5 - 1)/(2 - 1) = 2^5 - 1$. Detta kan den intresserade eleven sedan generalisera till $n$ stycken termer.

## Fler flyttningar än man kanske tror

Enligt myten – skapad av den franske matematikern Lucas i slutet av 1800-talet men här med ett litet tillägg av undertecknad – satt en gång två heliga män i ett tempel i Hanoi och framför sig hade de tre stavar av silver och 64 brickor av guld.

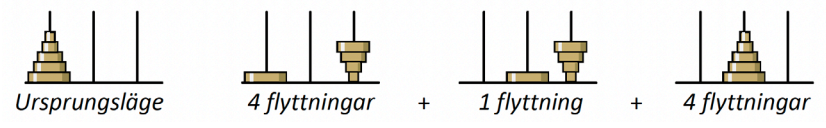
– Vi ska ej lämna detta tempel förrän alla brickorna är flyttade, lovade de, och satte igång att flytta runt dag som natt, men fann snart att de behövde ta hjälp av den robot som råkade stå där i ett hörn och som klarade av nio-tusen flytt i sekunden. Här om året var de äntligen klara. Vid vilken känd händelse i historien måste de då ha satt igång?



Här har jobbats på ordentligt.
9000 brickor har flyttats i sekunden.

## Hur blir det om vi stryker restriktionen?

Vi gör nu en förenkling av detta tornbygge genom att slopa kravet på att ingen bricka får läggas på en mindre. Hur blir det då? Helt klart måste det bli färre flyttningar, och matematiken bör också bli enklare. Kanske kan en sådan lightversion av hanoitornet passa som förövning i vissa klassrums-situationer. En första betraktelse med säg 5 brickor, ger enligt figuren nedan, att det räcker med 9 flyttningar då vi nu får placera brickorna hur vi vill.




Efter en del rekursiva tankar finner säkert några sambandet $a_n = a_{n-1} + 2$, medan andra i snarlika tankebanor finner: $a_n = 2n - 1$, även om formuleringarna troligen kommer att uttryckas mer vardagligt: ”Det ökar med två för varje ny bricka”.

Till klassen: Hur högt skulle ett sådant här torn bli om antalet flyttningar ska bli lika många som munkarna tvingades utföra? Blir det lika högt som Eiffeltornet?

## Riskornen – varför nöja sig med att fördubbla?

En talföljd av ett liknande slag är skrönan om riskornen på schackbrädet. Där fördubblas som bekant antalet riskorn för var ruta, 1, 2, 4, 8 ... och formuleringen för $a_n$ är snarlik den för tornet. Men om vi nu istället kvadrerar antalet allteftersom; hur många riskorn blir det då på sista rutan? Ja, en sak är säker: den väldiga mängd korn som ges av fördubblingsprincipen ovan, kommer vid en jämförelse med denna tillväxt att framstå som en halvfylld kaffekopp ungefär.

$a_n = a_{n-1}^2 \qquad a_1 = 2 \qquad$ Kvadratisk rekursion

Att utvecklingen blir trivial om man inte ändrar från 1 till 2 korn på första rutan i denna variant kan, liksom frågan om den slutna formen $a_n = f(n)$, inbjuda till diskussioner i klassrummet.

## Varning för svindel

Tycker man sen i klassen att denna mängd av ris inte heller räcker, kan man ju byta plats på symbolerna och fråga sig hur det blir om den kvadratiska rekursionen istället är exponentiell.

$a_n = 2^{a_{n-1}} \qquad a_1 = 2 \qquad$ Exponentiell rekursion

Här får samma elever möjlighet att bekanta sig med *Knuths pilnotation* då den slutna formen, i formell mening, blir:

$a_n = 2\uparrow\uparrow n$ med uttalet *n*:te tetraeringen av 2.

Den läsare som inte stött på detta skrivsätt, finner snart dess tolkning efter några insättningar i rekursionen ovan. (Den amerikanske matematikern Donald Knuth, född 1938, är upphovsman till denna pilnotation för stora tal. Se också nästa avsnitt.)

Kanske är det ändå bäst att låta Python göra jobbet. En ytterst enkel justering av koden på första sidan räcker. Några inledande körningar ger de beskedliga värdena 2, 4, 16. Men var försiktig. Antalet riskorn på fjärde rutan går väl också an, men redan på den femte rutan exploderar det. Där har vi $2\uparrow\uparrow 5$ vilket Python faktiskt klarar av med inledande 2003... och med alla 19 729 siffrorna utskrivna, fem täta A4-sidor. Hur det sen blir på den sjätte rutan, eller ännu hellre, den sista där uppe i vänstra hörnet av schackbrädet, $a_{64}$, får gärna någon hågad kollega reda ut.

## Då associativa lagen inte gäller

Räknesättet $a\hat{}a\hat{}a\ldots\hat{}a$, med $n$ stycken nivåer, kallas för *tetraering* och betecknas $a\uparrow\uparrow n$. Det handlar alltså om upprepad potensräkning. Med denna operation har vi ett exempel på att associativa lagen inte gäller överallt, vilket kan vara ett trevligt komplement till matrismultiplikationen som man brukar plocka fram för att ge ett exempel på att kommutativa lagen inte alltid gäller. Man ska här räkna uppifrån.

$$2\uparrow\uparrow 4 = 2^{2^{2^2}} = 2^{2^{(2^2)}} = 2^{2^4} = 2^{(2^4)} = 2^{16} = 65536$$

Tar man samma trappa nerifrån, blir värdet 256 som alltså är felaktigt. I en allmän betraktelse har vi här alltså följande: $(ab)c \neq a(bc)$, som på sitt sätt belyser hur viktigt det är att vi är noggranna i våra definitioner och begrepp när till exempel nya räkneregler införs.

## Summan av alla rishögar

En naturlig fundering är så klart frågan om det totala antalet korn på schackbrädet, det vill säga summan av alla rishögar. Lämpligt är då kanske att börja med något mindre dramatiskt än det vi precis har bevittnat, där inga tankeexperiment whatsoever finns för att på ett begripligt sätt skildra den hög som blev redan på femte rutan. Att summan formellt kan skrivas

$S_{64} = \sum_{k=1}^{64} 2\uparrow\uparrow k$ hjälper oss inte särskilt mycket.

Den *aritmetiska* och den *geometriska* summan är naturliga och enkla och kan i klassen erbjuda jämförelser om vilken summa som vinner under vissa betingelser et cetera. Det finns visserligen färdiga formler för dessa summor, men om vi fortsätter att anlägga ett rekursivt perspektiv finns inget behov av dem. De rader kod man får skriva blir inte mer omfattande än koden på första sidan. När klassen sen till slut tröttnar på riskornens framfart över brädet, kanske ett intresse för matematik ändå blir kvar.

## Färdigarbetat för munkarna

Den tidpunkt som jag tänker på vad gäller munkarna, är dinosauriernas utdöende, vilket skedde för 65 miljoner år sedan.