Uppgift 10 del 1.

Jag tror k har större påverkan än x. Tar man ett stort x och ett litet k så blir det få beräkningar. Tar man däremot ett litet x och ett stort y så blir det många flera beräkningar som bör innebära längre tid i uträkningen.

Uppgift 10 del 2.

Nedanför visas en graf av n\_1(k). Funktionen rec\_raise();.

Den är linjär för dom första 50 värdena.

Nedan visas grafen n\_2(k) som är rec\_raise\_eff();. Den ser ut att vara logaritmisk i karraktären.

Uppgift 10 del 3.

Hypotes: Jag tror att indatat kommer att vara linjärt mot utdatat. D.V.S. X = Y. Anledningen är att för varje rekursiv funktion så minskar man exponenten med 1. Det medför att antalet rekursiva anrop är lika många som antalet k.

Detta är grafen för dom första 50000 värdena i funktionen N\_1(k).

I grafen ser vi att x = y. Det innebär att alla k kommer att vara linjära mot antalet rekursiva anrop. Detta kommer att ta riktigt lång tid att beräkna för stora k värden.

Uppgift 10 del 4.

Hypotes: Av grafen i uppgift 2 så tror jag att funktionen kommer att vara logaritmisk. Det innebär att N\_2(k) kommer att vara mera effektiv än N\_1(k).

Detta är grafen för dom första 50000 värdena i funktionen N\_1(k).

Denna graf ger funktionen som en naturlig logaritm, inte basen 2. För att kunna få basen två så testar jag värden i Wolfram Alpha och gör stickprov av värden som jag även kör i mitt program.

 

Vi ser att det finns en brytpunkt för alla tal som är 2n där n är positiva heltal.

Som ledtråden i uppgiften föreslår, så testade jag formeln log2(k), vilket var en mindre än det värde mitt program anger. Sedan avrundas värdet neråt så att det blir ett heltal. Det ser ut som att formeln blir 1 + log2( (Math.floor(k)). Man behöver använda Math.floor() för att formeln inte ska ge decimaler samt att man adderar 1 till formeln. Annars så får jag off-by-one i slutet. Denna funktion fungerar för alla stickprov som jag tagit. Stickproven var ett 50-tal siffror.