Uppgift: Hitta primtal

Rekommenderade språk	Alla språk, särskilt Octave/MATLAB eller Python3
Arbeta med kalkylblad	Rekommenderas inte
Svårighetsnivå ur programmeringssynpunkt	Kom igång med programmering
Innehåll från kurs	Matematik 1a, 1b och 1c
Centralt innehåll som berörs	Egenskaper hos mängden av heltal, olika talbaser samt begreppen primtal och delbarhet.

Den här uppgiften är skapad för att få en enkel ingång i att programmera. Till skillnad från andra uppgifter i workshopen utgår den här just från att olika saker ska programmeras – inte en mer allmän problemställning som kan lösas genom programmering.

Beskrivning

Nedan finns ett program som hittar och skriver ut alla primtal upp till 1000. Primtalen sparas i en array (lista med värden).

Kod i Python3

```
# Skapa en lista som ska innehålla primtal
primtal = [2]

# Räkna upp till 1000
n = 2
while (n <= 1000):
  n = n + 1

  # Testa om n är delbart med något primtal i listan
  nArPrimtal = True
  for p in primtal:
    if (n / p == round(n / p)):
      nArPrimtal = False

  # Lägg till n i listan om det inte var delbart
  # med något primtal
  if nArPrimtal:
    primtal.append(n)

# Skriv ut primtalen
print(primtal)
```

Deluppgift 1

Skriv av programmet ovan, och kontrollera att det fungerar. (Observera att Python3 *kräver* att du använder indrag i loopar och villkorssatser, medan det bara är till för ökad läsbarhet i de flesta andra språk.)

Deluppgift 2

Modifiera programmet så att det också skriver ut hur många primtal som hittats.

Förslag: Lägg till en variabel `antalPrimtal`, som börjar på 1 och ökar med 1 i värde på lämpligt ställe i kodstrukturen.

Deluppgift 3

Skriv ett nytt program som ska testa om ett visst tal är ett primtal. Återanvänd så mycket du vill från den förra koden. Börja med den här raden (för Python3):

```
talAttTesta = 1234567 # Välj vilket tal du vill
```

Samarbeta gärna med någon annan, och jämför gärna olika lösningar.

Uppgift: Avkastning vid varierande värdeutveckling

Rekommenderade språk	Alla språk
Arbeta med kalkylblad?	Kalkylblad kan användas med begräsningar
Svårighetsnivå ur programmeringssynpunkt	Nybörjare till medel
Innehåll från kurs	Matematik 1a, 1b och 1c
Centralt innehåll som berörs	Begreppen förändringsfaktor och index. Metoder för beräkningar av räntor och amorteringar för olika typer av lån med kalkylprogram.

Beskrivning

Alice köper en andel i en fond för 2000 kr. Anta att årliga värdeutvecklingen från början är 2 %, och att den varje månad kan röra sig upp eller ner högst 0,5 procentenheter (med lika stor sannolikhet för alla utfall däremellan).

1. Vad är det troliga värdet för Alices fondandel efter fem år, avrundat till hela tiotals kronor?
2. Hur stor sannolikhet är det värdet har minskat? Avrunda svaret till hela procent.

Förslag på upplägg för lösning

Vid vanliga ränta på ränta-effekter är det rimligt och effektivt att använda potensberäkningar, och att räkna ut värdet efter varje månad (eller år) är något som normalt undviks. I ett läge där värdeutvecklingen ändras, och dessutom på ett oförutsägbart sätt, är det däremot användbart och kanske nödvändigt att göra beräkningarna i många steg.

Ett program som utforskar det här problemet skulle kunna följa dessa principer.

- Låt ett datorprogram 12 × 5 gånger beräkna nytt värde på fondandelen, och därefter ändra värdeutvecklingen på ett slumpmässigt sätt inom givna gränser.
- Låt datorprogrammet simulera värdeutvecklingen över 5 år ett stort antal gånger. Spara varje resultat, och använd resultatet för att besvara frågorna i uppgiften.

Kommentarer

Om kalkylprogram i uppgiften: Uppgiften kan till viss del utforskas med hjälp av kalkylprogram – det går att simulera värdeutvecklingen över 5 år, men det är svårt att upprepa den beräkningen ett stort antal gånger. Ett möjligt sätt att använda uppgiften i helklass, med kalkylprogram istället för programmering, vore att låta varje elev göra var sin simulering och sedan samla in och undersöka resultaten.

Om att åskådliggöra resultaten: För att överblicka hur 5-årsresultaten ser ut kan det vara användbart att göra diagram. I vissa miljöer (framförallt Octave/MATLAB) finns verktyg för att göra detta direkt, medan det i andra fall kan vara rimligt att kopiera över värden till kalkylprogram.

Några möjliga diskussioner med elever med utgångspunkt i uppgiften

- Hur uppstår variationen i de olika utfallen?

- Värdeutvecklingen kan både öka och minska. Spelar det någon roll om den ökar i början och minskar i slutet, jämfört med tvärt om?
- Är det rimligt att det mest troliga värdet är samma som om utvecklingen hade varit oförändrad? Borde den bli mer? Mindre?
- Hur många 5-årsresultat är rimligt att samla ihop innan man kan dra slutsatser om hur fördelningen ser ut?
- I många lägen beräknas månadsränta genom att ta en tolftedel av årsräntan (vilket även de flesta banker gör i redovisningar). Vad är en mer korrekt metod för att bestämma den månadsvisa utvecklingen? På vilket sätt ger de två metoderna olika resultat i den här uppgiften?
- I uppgiften används en likformig sannolikhetsfördelning för alla möjliga förändringar. Vad händer om man istället har en klockformad sannolikhetsfördelning, utan någon bortre gräns för hur snabbt värdet kan ändras?
- Kan man lösa uppgiften exakt eller algebraiskt, istället för med simuleringar? Hur?

Exempel på kod som kan användas för att lösa problemet (Python3)

Koden nedan kan användas om du kör fast, eller vill jämföra din färdiga kod med ett annat förslag.

```
# Nödvändig import för att få slumptal
import random

maxforandring = 0.02
startvarde = 2000
ff_start = 1.02

# Upprepa simulering av 5-årsutveckling 1000 gånger
for n in range(1000):
  varde = startvarde
  ff_m = ff_start ** (1/12)

  # Simulera en 5-årsutveckling (60 månader)
  for m in range(60):
    varde = varde * ff_m
    # Beräkna ny månadsutveckling baserad på
    # ändrad årsutveckling
    ff_y = ff_m ** 12
    ff_y = ff_y + maxforandring * (2 * random.random() - 1)
    ff_m = ff_y ** (1/12)

  # Skriv ut slutvärdet efter 5 år, avrundat till 10-tal
  print(round(varde / 10) * 10)
```

Uppgift: Tillförlitlighet i testresultat

Rekommenderade språk	Alla språk
Arbeta med kalkylblad?	Kalkylblad kan användas med begränsningar
Svårighetsnivå ur programmeringssynpunkt	Nybörjare till medel
Innehåll från kurs	Matematik 1a, 1b och 1c
Centralt innehåll som berörs	granskning av hur statistiska metoder och resultat används i samhället och inom vetenskap begreppen beroende och oberoende händelser samt metoder för beräkning av sannolikheter vid slumpförsök i flera steg med exempel från spel och risk- och säkerhetsbedömningar.

Beskrivning

Bob genomför ett test på ett naturläkemedel som påstås lindra migrän. 100 personer med migrän är med i studien, och som slumpas till antingen en försöksgrupp eller en kontrollgrupp så att det är 50 personer i varje. Personerna i försöksgruppen får naturläkemedlet, medan de i kontrollgruppen får tabletter som ser ut och smakar likadant – men är bevisat verkningslösa mot migrän.

I försöksgruppen säger 30 personer att de mådde bättre av tabletten de åt. I kontrollgruppen är det bara 20 personer som tycker att de blivit hjälpta.

Hur stor sannolikhet är det att naturläkemedlet är verkningslöst, och att en fördelning 30/20 (eller mer skev) uppkommer endast av slump? Svara i hela procent.

Förslag på upplägg för lösning

Uppgiften, och försöket som beskrivs, är uppbyggt för att den enda parametern man behöver ta hänsyn till är att den givna fördelningen ska uppstå av en tillfällighet. Ett sätt att lösa problemet beskrivs nedan.

- Utgå från att 50 personer av 100 med migrän kommer att uppleva att en tablett kommer att hjälpa dem, även om tabletten är bevisat verkningslös.
- Slumpa var och av dessa 50 personer till antingen försöksgruppen eller kontrollgruppen.
- Kontrollera om det finns minst 30 personer slumpade till försöksgruppen.
- Upprepa slumpningen massor av gånger, och undersök hur vanligt det är att minst 30 personer hamnar i kontrollgruppen.

Kommentarer

Om kalkylprogram i uppgiften: Uppgiften kan till viss del utforskas med hjälp av kalkylprogram – det går att göra en slumpvis fördelning av 50 personer in i två grupper, men det är svårt att upprepa slumpningen många gånger. Ett möjligt sätt att använda uppgiften i helklass, med kalkylprogram istället för programmering, vore att låta varje elev göra var sin simulering och sedan samla in och undersöka resultaten.

Några möjliga diskussioner med elever med utgångspunkt i uppgiften

- Uppgiften kan vara ett bra tillfälle att diskutera hur dubbelblinda studier genomförs, vilka förutsättningar som är viktiga, och varför designen anses

vara bra för medicinska studier. Det kan också vara ett bra tillfälle att diskutera hur stort underlag man behöver ha för att kunna dra säkra slutsatser – och vad som räknas som säkra slutsatser – samt att diskutera styrkan i och värdet hos placeboeffekten.

- Hur många resultat är rimligt att samla ihop innan man kan dra slutsatser om hur vanlig en 20/30-fördelning ser ut?
- Är det en korrekt metod att slumpa ut 50 personer till försöks- eller kontrollgruppen med lika stor sannolikhet, som i förslaget på upplägg för lösning?
- Hur påverkas sannolikheten att en lika skev fördelning uppstår på slump om det istället var 500 personer i studien?
- Anta att det istället skulle vara totalt 75 personer som känner sig hjälpta, fördelade enligt samma proportioner (30/45). Är det mer eller mindre sannolikt att en sådan fördelning uppstår på slump, jämfört med 20/30?
- Kan man lösa uppgiften exakt eller algebraiskt, istället för med simuleringar? Hur?

Exempel på kod som kan användas för att lösa problemet (Python3)

Koden nedan kan användas om du kör fast, eller vill jämföra din färdiga kod med ett annat förslag.

```
# Nödvändig import för att få slumptal
import random

hjalpta = 50
antalSlumpningar = 1000
granskvot = 30 / 20
skevfordelningar = 0

# Upprepa slumpning många gånger
for n in range(antalSlumpningar):

  # Nollställ försöks- och kontrollgrupp
  forsoksgrupp = 0
  kontrollgrupp = 0

  # Slumpa de hjälpta personerna in i någon av grupperna
  for m in range(hjalpta):
    if (random.random() < 0.5): forsoksgrupp += 1
    else: kontrollgrupp += 1

  # Undersök om fördelningen blivit tillräckligt skev
  if (forsoksgrupp / kontrollgrupp >= granskvot): skevfordelningar +=
1

# Skriv ut resultat
print(str(skevfordelningar / antalSlumpningar) + " (" +
str(antalSlumpningar) + " slumpningar)")
```

Uppgift: Längden på en hängande kedja

Rekommenderade språk	Alla språk
Arbeta med kalkylblad?	Kalkylblad kan användas
Svårighetsnivå ur programmeringssynpunkt	Nybörjare till medel
Innehåll från kurs	Matematik 2a, 2b och 2c (se kommentar för 3c)
Centralt innehåll som berörs	begreppet kurva (2c), räta linjens och parabelns ekvation samt hur analytisk geometri binder ihop geometriska och algebraiska begrepp

Beskrivning

Bild från https://en.wikipedia.org/wiki/File:Kette_Kettenkurve_Catenary_2008_PD.JPG

En fritt hängande kedja har en höjd över marken som kan beskrivas av

$$y(x) = \frac{2^{ax} + 2^{-ax}}{2} + h$$

där x är avståndet i sidled från kedjans lägsta punkt, i meter. Anta att avståndet mellan stolparna på bilden är 2,00 m, att lägsta punkten ligger mitt emellan stolparna, och att värdena på a och h är 1,30 m^{-1} respektive –0,90 m. Hur lång är i så fall kedjan? Svara med två decimalers noggrannhet.

Förslag på upplägg för lösning

Denna uppgift går att lösa med förhållandevis avancerad integralkalkyl, som inte ingår i de nationella kurserna på gymnasial nivå. Ett alternativt sätt att lösa den är att dela in kedjan i små intervall och beräkna längden på varje intervall som om det vore en rak linje.

Kommentarer

Om kalkylprogram i uppgiften: Det är möjligt att i celler i kalkylprogram uttrycka både x- och y-värden för punkter på kedjan, för exempelvis varje centimeter i x-led. Detta kan sedan användas för att räkna ut kortaste avståndet mellan varje par av punkter, och summera till en approximering av kedjans längd.

Variationer på kurvor: Vilka kurvor som används kan förstås varieras stort. Det är även möjligt att införa parametriserade kurvor av typen x(t), y(t) istället för att bara använda funktionsgrafer. I **matematik 3c**, där enhetscirkeln introduceras, kan man exempelvis använda en variant av en här uppgiften för att beräkna omkretsen på en ellips – vilket är förvånansvärt svårt genom andra metoder. Den kurva som används då är lämpligtvis

$$x(t) = a \cdot cos(t), y(t) = b \cdot sin(t)$$

Exempel på kod som kan användas för att lösa problemet (Python3)

Koden nedan kan användas om du kör fast, eller vill jämföra din färdiga kod med ett annat förslag.

```
# math-klassen behövs för att beräkna kvadratrötter
import math

# Definiera funktionen y(x) för att underlätta läsbarhet
def y(x):
  return (2 ** (1.30 * x) + 2 ** (-1.30 * x)) / 2

# Sätt start- och parametervärden
xStart = -1
xSlut = 1
steg = 0.01

x = xStart
langd = 0

# Stega fram x från xStart till xSlut i små steg
while (x <= xSlut):
  x1 = x
  x2 = x + steg
  y1 = y(x1)
  y2 = y(x2)

  # Beräkna längden på detta korta segment,
  # lägg till på totala längden
  langd += math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)

  # Öka x-värdet ett steg
  x = x + steg

# Skriv ut totala längden
print(langd)
```

Uppgift: Approximera talet e

Rekommenderade språk	Alla språk
Arbeta med kalkylblad?	Kalkylblad kan användas
Svårighetsnivå ur programmeringssynpunkt	Nybörjare till medel
Innehåll från kurs	Matematik 3b och 3c
Centralt innehåll som berörs	introduktion av talet e och dess egenskaper

Beskrivning

Derivatan av varje exponentialfunktion $f(x) = a^x$ är en konstant gånger samma funktion: $f'(x) = k \cdot a^x$. Den matematiska konstanten e är det tal som uppfyller att derivatan av e^x är e^x.

Använd detta för att bestämma ett närmevärde till e med två decimalers noggrannhet.

Förslag på upplägg för lösning

Ett sätt att lösa det här problemet är att gissa värden på e, och undersöka derivatan i en punkt. Värdet på e ökas/minskas sedan successivt, tills funktionens och derivatans värde överensstämmer tillräckligt väl.

En viktig del i den här metoden är att derivata och funktionsvärde bara behöver stämma överens för *något* x, inte alla x-värden.

I de flesta programmeringsspråk saknas färdiga verktyg för att bestämma derivata, och då kan det vara lämpligt att använda en ändringskvot istället.

Kommentarer

Om man vill utveckla det här programmet kan man exempelvis använda följande frågor:

- Hur kan programmet ändras för att få större noggrannhet på e?
- Är det rimligt att använda en ändringskvot? Vad kan få ändringskvoten att skilja sig så mycket från den korrekta derivatan att programmet ger ett felaktigt svar?
- Hur kan algoritmen göras mer effektiv?
- Hur kan man göra en liknande algoritm i ett kalkylblad? Vilka fördelar och nackdelar har det?

Exempel på kod som kan användas för att lösa problemet (Python3)

Koden nedan kan användas om du kör fast, eller vill jämföra din färdiga kod med ett annat förslag.

```
import math

# Första gissningen på e-värdet
eGissning = 0.5
# Med hur stora steg e-värdet ska öka
steg = 1
# Steglängd för att beräkna ändringskvot
h = 0.0001
# x-värde att beräkna ändringskvot på
x = 5
# Hur stor skillnad mellan derivata och
# funktionsvärde som accepteras
felmarginal = 0.001
# Variabel som berättar om tillräcklig noggrannhet nåtts
klart = False

# Öka successivt gissade värdet på e
# tills skillnaden är tillräcklig liten
while (klart != True):
  e = eGissning
  andringskvot = (e ** (x+h) - e ** (x-h)) / (2*h)
  differens = e ** x - andringskvot

  # Kolla om differensen är tillräckligt liten
  if (abs(differens) <= felmarginal):
    klart = True

  # Om differensen är positiv behöver eGisning ökas
  if differens > 0: eGissning = eGissning + steg

  # Om differensen är negativ ökades eGissning för
  # mycket senast: backa ett steg och minska steglängden
  if differens < 0:
    eGissning = eGissning - steg
    steg = steg / 10

# Skriv ut approximeringen av e
print(e)
```

Uppgift: Komplex talserie

Rekommenderade språk	Octave/MATLAB eller Python3 (på grund av beräkningar med komplexa tal)
Arbeta med kalkylblad?	Kalkylblad rekommenderas inte
Svårighetsnivå ur programmeringssynpunkt	Medel
Innehåll från kurs	Matematik 4
Centralt innehåll som berörs	metoder för beräkningar med komplexa tal skrivna på olika former inklusive rektangulär och polär form komplexa talplanet, representation av komplext tal som punkt och vektor konjugat och absolutbelopp av ett komplext tal

Beskrivning

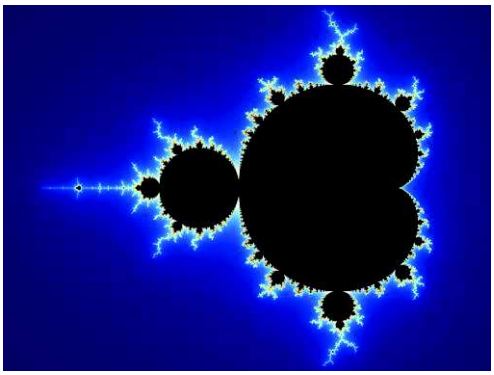

Bild hämtad från https://en.wikipedia.org/wiki/File:Mandel_zoom_00_mandelbrot_set.jpg

Talserien som beskrivs av $z_0 = 0$ och $z_{n+1} = z_n^2 + c$ beter sig på ett sätt som är svårt att förutsäga för olika värden på det komplexa talet c. För vissa värden på c divergerar talserien (går mot oändligheten), medan andra värden på c ger ett begränsat värde. Om något värde i talserien har ett absolutbelopp större än 2 kommer den dock alltid att divergera.

Undersök om serien divergerar eller inte för samtliga värden på c som har hela tiondelsvärden och där både real- och imaginärdelen ligger mellan –1 och 1. Markera de tal som inte divergerar i det komplexa talplanet.

Förslag på upplägg för lösning

Ett rimligt sätt att lösa den här uppgiften är att låta en dator helt enkelt beräkna talserien för de givna värdena på c. Eftersom det endast finns villkor för att avgöra om en serie divergerar, men inte konvergerar, är det rimligt (och förmodligen nödvändigt) att sätta en övre gräns på hur många värden i talserien som ska beräknas.

I programmeringsspråk som inte stödjer komplexa tal finns mycket att vinna på att definiera funktioner för att addera, multiplicera och bestämma absolutbelopp av komplexa tal.

Om programmeringsspråket inte stödjer grafisk utmatning på ett enkelt vis, kan man som ersättning använda textsträngar som pixlar/punkter i komplexa talplanet.

Kommentarer

Det här problemet ger en ingång i att diskutera fraktal geometri och kaotiska mönster. (Algoritmen i uppgiften är den som används för att skapa Mandelbrotfraktalen.)
Några sätt att fördjupa uppgiften:

- Ändra vilket område i det komplexa talplanet som undersöks, för att utforska områden där divergering verkar särskilt svår att förutspå.
- Färglägg punkter i komplexa talplanet med färger som avspeglar hur snabbt varje värde på c får talserien att divergera.
- Modifiera algoritmen för att istället motsvara Juliamängden.

Exempel på kod som kan användas för att lösa problemet (Python3)

I exemplet nedan används textsträngar som enkel grafisk representation av resultatet.

Koden nedan kan användas om du kör fast, eller vill jämföra din färdiga kod med ett annat förslag.

```
import cmath

# Funktion som avgör om talserien divergerar för givet c
def divergerar(c):
  z = 0 + 0j
  for n in range(maxIterationer):
    z = z * z + c
    if abs(z) > 2: return True
  return False

# Start- och parametervärden
xMin = -1
xMax = 1
yMin = -1
yMax = 1
xSteg = 0.1
ySteg = 0.1
maxIterationer = 10

# Börja längst upp till vänster i koordinatsystemet
x = xMin
y = yMax

while (y > yMin):
  # Återställ x-värdet och rad för utmaning av resultat
  x = xMin
  utmatningsrad = ""
  while (x < xMax):
    # Skapa komplext tal från x- och y-värden
    c = x + y * 1j
    # Stjärna markerar c-värden som inte divergerar,
    # värden som divergerar markeras med mellanslag
    if divergerar(c): utmatningsrad += " "
    else: utmatningsrad += "*"

    # Stega till nästa plats på samma rad
    x += xSteg

  # Skriv ut resultat för denna rad, stega till nästa rad
  print(utmatningsrad)
  y -= ySteg
```

Uppgift: ”3x + 1”-problemet

Rekommenderade språk	Alla språk
Arbeta med kalkylblad?	Kalkylblad rekommenderas inte
Svårighetsnivå ur programmeringssynpunkt	Nybörjare
Innehåll från kurs	Matematik 5
Centralt innehåll som berörs	Begreppen rekursion och talföljd. Matematiska problem med anknytning till matematikens kulturhistoria.

Beskrivning

Ett matematiskt påstående som ingen ännu lyckats bevisa är följande:

Ta ett positivt heltal n. Om det är jämnt, dela med två. Om det är udda, multiplicera med 3 och lägg till 1. Upprepa proceduren. Om du fortsätter tillräckligt länge, kommer du alltid att nå talet 1 oavsett vilket tal du startade med.

Visa att påståendet gäller för alla heltal mellan 1 och 100 000. Hitta också det eller de tal i intervallet som kräver flest steg innan talet 1 nås.

Kommentarer

Det här problemet kan vara en ingång för flera diskussioner med elever:

- Det finns matematiska frågeställningar som är obesvarade, även frågor som ger intryck av att vara ganska enkla.
- Att ”testa alla värden” är sällan en fungerande metod när värdena är oändligt många. Att låta en dator testa många värden kan däremot vara ett värdefullt sätt att utforska problemen.
- Vilka mönster kan man hitta i hur algoritmen fungerar för de 100 000 första talen?

Exempel på kod som kan användas för att lösa problemet (Python3)

Koden nedan kan användas om du kör fast, eller vill jämföra din färdiga kod med ett annat förslag.

```
# i räknar från 0 till 99 999
for i in range(100000):
  # Sätt startvärde för n, nollställ räknaren för steg
  n = i + 1
  steg = 0

  # Utför de steg som algoritmen föreskriver
  while (n != 1):
    steg += 1
    if (n / 2 == round(n / 2)):
      n = n / 2
    else:
      n = 3 * n + 1

  # Skriv ut resultat
  print(str(i + 1) + " (kräver " + str(steg) + " steg)")
```

Uppgift: Sannolikheten för Yatzy

Rekommenderade språk	Alla språk
Arbeta med kalkylblad?	Kalkylblad rekommenderas inte
Svårighetsnivå ur programmeringssynpunkt	Medel till svår
Innehåll från kurs	Matematik 5, matematik 1a, 1b och 1c
Centralt innehåll som berörs	Begreppen permutation och kombination (ma 5) Begreppen beroende och oberoende händelser samt metoder för beräkning av sannolikheter vid slumpförsök i flera steg med exempel från spel och risk- och säkerhetsbedömningar (ma 1abc)

Beskrivning

I spelet Yatzy har man fem tärningar. Under en spelares tur slås tre högst gånger, och inför slag 2 och 3 får man välja vilka tärningar som ska slås om och vilka som ska ligga kvar orörda. Man får poäng efter olika kombinationer av resultat, så som par, stege och kåk. Den högsta poängen får man för fem lika, kallat Yatzy.

Vad är sannolikheten för att få Yatzy på högst tre slag, förutsatt att man hela tiden sparar tärningar för att maximera den chansen? Svara i hela tiondels procent.

Förslag på upplägg för lösning

Det här problemet går att lösa med omfattande kombinatoriska beräkningar. En alternativ metod är att skriva ett program som simulerar ett stort antal turer, där man hela tiden sparar tärningar som det resultat som det finns flest av.

Det finns flera utmaningar i att skriva ett sådant program, särskilt om man vill slippa att skriva kod med mycket upprepningar. Det kan finnas anledning att använda ett par funktioner som inte finns i referensbladen – sök på nätet om du behöver.

Kommentarer

Det här problemet kan vara en ingång för exempelvis följande diskussioner:

- Vad är det som gör uppgiften svår att lösa med algebraiska eller kombinatoriska metoder?
- Hur kan man åtminstone börja lösa uppgiften på algebraisk väg/med hjälp av kombinatoriska beräkningar?
- Hur kan man göra en rimlighetsbedömning av resultat av beräkningar eller simuleringar?
- Hur många simuleringar behöver man göra för att kunna dra tillräckligt säkra slutsatser om resultatet?

Exempel på kod som kan användas för att lösa problemet (Python3)

Koden nedan kan användas om du kör fast, eller vill jämföra din färdiga kod med ett annat förslag.

```
import random

# Definiera ett par variabler som används i programmet
tarningar = [0, 0, 0, 0, 0]
antalYatzy = 0

# Simulera ett stort antal turer.
# Här har tärningarna värde 0–5 istället för 1–6
for forsok in range(1000):

  # Nollställ variabel som håller reda på vilket
  # tärningsresultat som är vanligast
  flestAv = -1

  for slag in range(3):
    # Slå om de tärningar som inte visar det som det finns mest av
    for t in range(5):
      if tarningar[t] != flestAv:
        tarningar[t] = int(random.random() * 6)

    flestAv = -1 # Variabel för vilket resultat som är vanligast
    flest = 0 # Variabel för antal tärningar med det värdet
    # Undersök hur många tärningar som visar varje värde 0–5
    for v in range(6):
      raknare = 0
      for t in tarningar:
        if t == v: raknare += 1
      if raknare > flest:
        flest = raknare
        flestAv = v

  if flest == 5: antalYatzy += 1

# Skriv ut resultatet
print(antalYatzy)
```

Uppgift: Sannolikheten för stege

Rekommenderade språk	Alla språk
Arbeta med kalkylblad?	Kalkylblad rekommenderas inte
Svårighetsnivå ur programmeringssynpunkt	Medel till svår
Innehåll från kurs	Matematik 5, matematik 1a, 1b och 1c
Centralt innehåll som berörs	Begreppen permutation och kombination (ma 5) Begreppen beroende och oberoende händelser samt metoder för beräkning av sannolikheter vid slumpförsök i flera steg med exempel från spel och risk- och säkerhetsbedömningar (ma 1abc)

Beskrivning

Den här uppgiften går förhållandevis enkelt att lösa utan programmering, men har ändå behållits som ett exempel.

En vanlig kortlek med 52 kort blandas och du får fem kort. Vad är sannolikheten att du har en stege? Avrunda till hela tiondels procent.

Kom ihåg att ess kan både räknas som 1 och 14 i en stege.

Förslag på upplägg för lösning

Det här problemet går att lösa utan särskilt omfattande beräkningar om rätt metoder används, men ett alternativ är att skriva ett program som simulerar ett stort antal händer med fem kort.

Det finns flera utmaningar i att skriva ett sådant program, särskilt om man vill slippa att skriva kod med mycket upprepningar. Man har också stor nytta av funktioner för att exempelvis slumpsortera eller sortera arrayer i storleksordning – sök på nätet efter hur det kan göras.

Kommentarer

Varianter på uppgiften: Uppgiften går enkelt att variera, ibland på sätt som ökar komplexiteten avsevärt men också för att minska komplexiteten. Några förslag:

- Hur mycket mer eller mindre sannolikt är det att få stege på fem kort om kortleken bara innehåller 13 kort i en och samma färg? Om det är en kortlek sammanslagen av ett stort antal vanliga kortlekar?
- Hur stor är sannolikheten för att få par, tvåpar, triss, stege, färg, kåk respektive fyrtal på fem kort?
- I vissa spel ges poäng för par (1), tvåpar (2), triss (3), stege (4), färg (5), kåk (6) och fyrtal (7). Hur förhåller sig poängen för respektive kombination till sannolikheten att få den kombinationen?
- I många spel får man slänga kort och dra nya en eller två gånger. Hur påverkas sannolikheterna för olika kombinationer av detta?

Som med andra uppgifter där utfall simuleras ett stort antal gånger finns anledning att undersöka hur många upprepningar som är rimligt att genomföra innan tillräckligt säkra slutsatser kan dras.

Exempel på kod som kan användas för att lösa problemet (Python3)

Koden nedan kan användas om du kör fast, eller vill jämföra din färdiga kod med ett annat förslag.

```
import random

# Bygg en kortlek med värdena 1–13 representerade 4 gånger var
valor = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
kortlek = []

for v in valor:
  for f in range(4):
    kortlek.append(v)

# Gör ett stort antal simulerade händer med 5 kort
antalStegar = 0
antalForsok = 1000
for n in range(antalForsok):
  # Blanda kortleken och ge de första fem
  random.shuffle(kortlek)
  hand = [kortlek[0], kortlek[1], kortlek[2], kortlek[3], kortlek[4]]

  # Sortera korten på hand. Om det finns ett ess,
  # och högsta är kung, räkna esset som 14 istället för 1
  hand = sorted(hand)
  if (hand[0] == 1 and hand[4] == 13):
    hand[0] = 14
    hand = sorted(hand)

  # Undersök om varje kort är 1 mer än föregående
  arStege = True
  forraKortet = hand[0] - 1
  for k in hand:
    if (k != forraKortet + 1):
      arStege = False
    forraKortet = k

  # Håll räkning på hur många stegar som dykt upp
  if (arStege): antalStegar += 1

# Skriv ut resultatet
print(antalStegar / antalForsok)
```