Övningar

Slumpade gruppen

February 28, 2024

Övning 1

Vi har enligt formel

$$F(x) = \sum_{n=1}^{\infty} f_n x^n = x + \sum_{n=2}^{\infty} f_n x^n = x + \sum_{n=1}^{\infty} f_{2n} x^{2n} + \sum_{n=1}^{\infty} f_{2n+1} x^{2n+1} =$$

$$x + \sum_{n=1}^{\infty} f_n x^{2n} + x \sum_{n=1}^{\infty} (f_n + f_{n+1}) x^{2n} = x + F(x^2) + xF(x^2) + x \sum_{n=0}^{\infty} f_{n+1} x^{2n} =$$

$$x + F(x^2) + xF(x^2) + \frac{\sum_{n=1}^{\infty} f_{n+1} x^{2(n+1)}}{x} = x + F(x^2) + xF(x^2) + \frac{-x^2 + \sum_{n=1}^{\infty} f_n x^{2(n+1)}}{x} =$$

$$F(x^2) + xF(x^2) + \frac{F(x^2)}{x}$$

Delar vi med x, och noterar att $G(x^2) = \frac{F(x^2)}{x^2}$ får vi

$$G(x) = \frac{F(x^2)}{x} + F(x^2) + \frac{F(x^2)}{x^2} = (1 + x + x^2)G(x^2)$$

som önskat.

Vi visar nu att

$$G(x) = \prod_{j=0}^{\infty} (1 + x^{2^j} + x^{2^{j+1}})$$

genom att visa att högra ledet satisfierar funktionalekvationen

$$\frac{G(x)}{1 + x + x^2} = G(x^2)$$

Stoppar vi in $x = x^2$ i

$$G(x) = \prod_{j=0}^{\infty} (1 + x^{2^j} + x^{2^{j+1}})$$

får vi

$$G(x^2) = \prod_{j=0}^{\infty} (1 + x^{2^{j+1}} + x^{2^{j+2}}) = \prod_{j=0}^{\infty} (1 + x^{2^j} + x^{2^{j+1}})/(1 + x + x^2)$$

Alltså satifierar produkten samma funktionalekvation[1]

Division med x ger oss då att

$$F(x) = \frac{1}{x}\prod_{j=0}^{\infty}(1 + x^{2^j} + x^{2^{j+1}})$$

Det finns även ett annat tillvägagånssätt, men är osäker om det är ett giltigt sätt att resonera. Definiera $p_n(x) = (1 + x^{2^n} + x^{2^{n+1}})$. Notera att $p_n(x^2) = p_{n+1}(x)$. Vi har redan att $G(x) = p_0F(x^2)$. Utvärdering i x^2 ger oss att $G(x^2) = p_0(x^2)G(x^4) = p_1(x)G(x^4)$. Substituerar vi in detta i $G(x) = p_0G(x^2)$ får vi att $G(x) = p_0(x)p_1(x)G(x^4)$, och induktivt får vi att

$$G(x) = p_0(x)p_1(x)...p_n(x)G(x^{2n}) = G(x^{2n})\prod_{j=0}^{n}(1 + x^{2^j} + x^{2^{j+1}})$$

Vänstra ledet är oberoende av n, så vi kan ta gränsvärdet $n \to \infty$ av bägge led, d.v.s. fortsätta riva ut faktorer *ad infinitum*, och får

$$G(x) = \lim_{n\to\infty} G(x^{2n})\prod_{j=0}^{n}(1 + x^{2^j} + x^{2^{j+1}}) = (\lim_{n\to\infty} G(x^{2n}))(\lim_{n\to\infty}\prod_{j=0}^{n}(1 + x^{2^j} + x^{2^{j+1}}))$$

$$= (\lim_{n\to\infty} G(x^{2n}))\prod_{j=0}^{\infty}(1 + x^{2^j} + x^{2^{j+1}})$$

Eftersom $f_1 = 1$ är $F(x) = x + O(x^2)$, alltså är $G(x) = \frac{F(x)}{x} = 1 + O(x)$, så om $\lim_{n\to\infty} G(x^{2n})$ är en väldefinierad funktion är

$$\lim_{n\to\infty} G(x^{2n}) = 1 + \lim_{n\to\infty} O(x^{2n})$$

vilket, för $x \in (-1, 1)$, är konvergent och lika med 1, och divergent annars, alltså måste

$$\lim_{n\to\infty} G(x^{2n}) = 1$$

för $x \in (-1, 1)$. Intressant att anmärka är att den oändliga produkten endast är väldefinierad för samma x-värden.

Slår vi ihop allting får vi igen att

$$G(x) = \prod_{j=0}^{\infty}(1 + x^{2^j} + x^{2^{j+1}}))$$

vilket avslutar beviset.[2]

[1] Anmärkning till Vilhelm: Ingen aning varför detta är en giltig metod. Så vitt jag kan se har jag endast visat att de två uttrycken är lika upp till konstant, om ens det. Detta kan iofs lösas med ett begynnelsevilkor, förutom att det enda x-värde för vilket F är väldefinierad är $x = 0$, vilket ger oss $0 = 0$.

[2] En till anmärkning: Jag vet att detta inte exakt var reell analys

Övning 2

i)

Definiera alltså

$$P(x) = \sum_{n=0}^{\infty} p_n x^n$$

Enligt formel är väntevärdet

$$\mathbb{E}(X) = \sum_{n=0}^{\infty} np_n$$

Notera då att

$$P(x) = \sum_{n=0}^{\infty} p_n x^n = 0 + \sum_{n=0}^{\infty} p_{n+1} x^{n+1} \Rightarrow P'(x) = 0 + \sum_{n=0}^{\infty} (n+1)p_{n+1} x^{n+1} = \sum_{n=0}^{\infty} np_n x^n$$

Alltså är

$$P'(1) = \sum_{n=0}^{\infty} np_n 1^n = \sum_{n=0}^{\infty} np_n = \mathbb{E}(X)$$

Enligt formel anges standardavvikelsen av

$$\sigma(X) = \sqrt{\mathbb{E}(X^2) - \mathbb{E}(X)^2}$$

och vi vet att

$$\mathbb{E}(X^2) = \sum_{n=0}^{\infty} n^2 p_n$$

Definiera

$$B(x) = \sum_{n=0}^{\infty} n^2 p_n x^n$$

som då är generande funktionen för $n^2 p_n$. Då är, enligt formel,

$$B(x) = \sum_{n=0}^{\infty} n(np_n)x^n = \frac{P''(x)}{x} \Rightarrow B(1) = P''(1) = \mathbb{E}(X^2)$$

enligt formel, alltså är

$$\sigma(X) = \sqrt{P''(1) - P'(1)}$$

ii)

Antag att vi är givna $X = k$. Då är $X + Y = n$ omm $Y = n - k$, alltså är $\mathbb{P}(X + Y = n) = p_k p_{n-k}$. Eftersom k kan variera från 0 till n är

$$\mathbb{P}(X + Y = n) = \sum_{i=0}^{n} p_i p_{n-i}$$

vilket vi ser att är faltningen $(p)_n \star (p)_n$, alltså är $P_2(x) = P(x)P(x)$.

iii)

Eftersom summor av slumpvariabler är slumpvariabler, och vi i förra deluppgiften visade att $\mathbb{P}(X_1 + X_2 = n) = (p)_n \star (p)_n$ följer det induktivt att $\mathbb{P}(X_1 + X_2 + .. + X_n = n) = ((...(((p)_n \star (p)_n) \star (p)_n) \star ...)$, alltså faltningen av $(p)_n$ k gånger med sig själv. Alltså är den genererande funktionen för $p_k(x) = P(x)^k$

Programmeringsuppgifter

Övning 1

1,2,3

Funktionerna för uppgift 1,2 och 3 ges av följande kod:

```
import numpy as np
import random
import matplotlib.pyplot as plt

def diagonal_coordinates(n):
    diagonal_coordinates = []
    for i in range(n+1):
        diagonal_coordinates.append([i,i])
    x_values = [x[0] for x in diagonal_coordinates]
    y_values = [y[1] for y in diagonal_coordinates]
    return diagonal_coordinates , x_values , y_values

def coordinates_under_diagonal(n):
    coordinates = []
    for x in range(n + 1):
        for y in range(x + 1):
            if y < x:
                coordinates.append((x, y))
    x_values = [x[0] for x in coordinates]
    y_values = [y[1] for y in coordinates]
    return coordinates , x_values , y_values


def up_right():
    random_direction = random.randint(0,1)
    if random_direction == 1:
        return 'up'
    else:
        return 'right'
def random_gitter_stig(n):
    point = [0, 0]
    stig = []
    for i in range(2 * (n + 1)):
        step = up_right()
        if point[0] >= n and point[1] >= n:
            break
```

```python
        elif point[0] == n:
            point[1] +=1
            stig.append(tuple(point))
        elif point[1] == n:
            point[0] += 1
            stig.append(tuple(point))
        elif step == 'up':
            point[1] += 1
            stig.append(tuple(point))
        elif step == 'right':
            point[0] += 1
            stig.append(tuple(point))
    return stig

def gitter_stig_values(n):
    gitter_stig = random_gitter_stig(n)
    x_values = [x[0] for x in gitter_stig]
    y_values = [y[1] for y in gitter_stig]
    return x_values , y_values

def dyck_stig_check(n):
    coordinates = coordinates_under_diagonal(n)[0]
    random_stig = random_gitter_stig(n)

    if any(item in coordinates for item in random_stig):
        return False
    else:
        return True


def many_gitter_stigar(n,number_of_gitter_stigar):
    true = 0
    false = 0
    counter = 0
    for i in range(number_of_gitter_stigar):
        values = gitter_stig_values(n)
        stig_check = dyck_stig_check(n)
        if stig_check == True:
            true += 1
        elif stig_check == False:
            false += 1
        counter += 1
        print(counter)
        plt.plot(values[0],values[1])
    print(true,false)
    plt.plot(diagonal_coordinates(n)[1],diagonal_coordinates(n)[2])
    plt.show()
```

Övning 2

Vi kör våran many gitter stigar funktion med $n = 250$ och 10000 stigar. Då antalet Dyck-stigar d_n ges av $d_n = \frac{1}{n+1}\binom{2n}{n}$ och det totala antalet stigar ges av $\binom{2n}{n}$ kan vi förvänta oss att det kommer finnas $10000 * \frac{\frac{1}{250+1}\binom{2*250}{250}}{\binom{2*250}{250}} \approx 40$ stycken Dyck-stigar. Vi plottar 10000 olika stigar och får att utav dessa är 360 Dyck stigar och 9640 är inte det

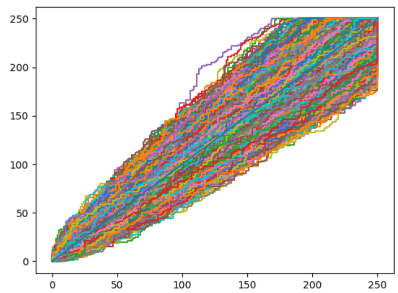

Figure 1: 10000 slumpande stigar där 360 är Dyck-stigar

Övning 3

Vi har att antalet sätt att skriva $2n$ matchande parenteser räknas ges av Catalantalen dvs $d_n = \frac{1}{n+1}\binom{2n}{n}$. Vi skriver en funktion som slumpar antalet paranteser med sannolikhet $\frac{1}{2}$ och håller koll på antalet öppnade och stängda paranteser. På detta sätt slipper vi jämföra med alla kordinater som ligger under diagonalen. Vi skriver funktionen på följande sätt:

```python
def generate_dyck_stig(n):
    stig = []
    point = [0, 0]
    stack = 0
    for i in range(2 * (n + 1)):
        random_parantes = np.random.randint(0, 2)
        if point[0] >= n and point[1] >= n:
            break
        elif point[0] == n:
            point[1] += 1
        elif point[1] == n:
            point[0] += 1
        elif random_parantes == 1:
            point[1] += 1
            stack += 1
        else:
            if stack > 0:
                point[0] += 1
                stack -= 1
            else:
                point[1] += 1
                stack += 1
        stig.append(point[:])

    return stig

def plot_dyck_stig(n,number_of_dyck_stigar):
    counter = 0
    for i in range(number_of_dyck_stigar):
        dyck_stig = generate_dyck_stig(n)
        x_values = [x[0] for x in dyck_stig]
        y_values = [y[1] for y in dyck_stig]
        plt.plot(x_values,y_values)
        counter += 1
        print(counter)
    plt.plot(diagonal_coordinates(n)[1],diagonal_coordinates(n)[2])
    plt.grid()
    plt.title('Dyck-stigar')
    plt.show()
```

Övning 5

Vi kan räkna ut givet en Dyck-stig den totala ytan under stigen med hjälp av funktionen nedan. Om vi plottar den genomsnittliga arean för dyck-stigar med n = 1 till n = 100 ser vi att arean ökar exponentiellt.

```
def area_under_dyck_stigar():
    areas = []
    all_area = []
    for i in range(1,100):
        dyck_stig = generate_dyck_stig(i)
        x_values = [x[0] for x in dyck_stig]
        y_values = [y[1] for y in dyck_stig]
        area = np.trapz(y_values, x_values)
        areas.append(area)
        all_area.append(sum(areas)- ((i**2)/2))
    plt.plot(all_area)
    plt.xlabel('Dyck-stig n')
    plt.ylabel('Areaenheter')
    plt.show()
    return areas, all_area
```

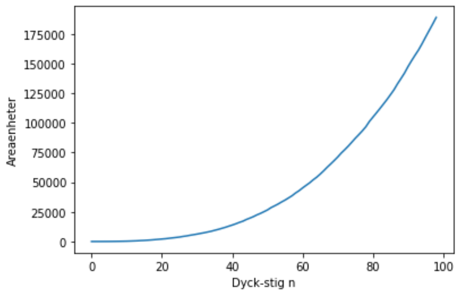

Figure 2: Arean under stig för varje n mellan 1 och 100