# Infectious Disease Simulator

## 1. Introducere

Scopul acestui proiect este analiza evoluției unei boli infecțioase într-un oraș mic, conform următoarelor reguli:

1. În oraș locuiesc $N$ oameni, dintre care, inițial, $K$ sunt infectați cu o boală.
2. În fiecare dimineață, oricare doi oameni se întâlnesc exact o dată și dacă unul este infectat și celălalt nu, acesta din urmă are o probabilitate de $p\%$ de a deveni infectat.
3. Oamenii se pot vindeca. În fiecare seară, o persoană bolnavă are o probabilitate de $q\%$ de a se vindeca.
4. Dorim să aflăm valoarea medie așteptată a numărului de persoane care sunt infectate după $Z$ zile.

## 2. Calculul teoretic al valorii

Algoritmul descris mai jos calculează cu exactitate valoarea medie așteptată a numărului de persoane care sunt infectate după $Z$ zile, folosind doar parametrii problemei.

Pentru început, algoritmul calculează care este probabilitatea unei persoane să devină infectată într-o dimineață în care $i$ dintre persoane sunt infectate cu următoarea formulă:

```
infection_chance[i]=1-(1-p/100)**i
```

Acum, algoritmul calculează valorile matricei $inf[i, j]$ care reprezintă probabilitatea ca dacă începem ziua cu $i$ persoane infectate, să avem $j$ persoane infectate la prânz cu următoarea formulă:

```
inf=np.zeros((n+1,n+1),dtype=float)
for i in range(1,n+1):
    for j in range(1,n+1):
        if j<i:
            inf[i,j]=0
        else:
            inf[i,j]=scipy.special.comb(n-i, j-i)*(
               infection_chance[i]**(j-i))*((1-infection_chance[i
               ])**(n-j))
```

Idea din spatele formulei este că noi căutăm o submulțime de $j - i$ persoane sănătoase și calculăm probabilitatea ca aceste persoane să se îmbolnăvească, și ca nimeni altcineva

să nu se îmbolnăvească. Coeficientul binomial $\binom{n-i}{j-i}$ apare deoarece putem alege această submulțime în $\binom{n-i}{j-i}$ moduri. Acum că am ale submulțimea, trebuie să înmulțim cu $infection\_chance[i]$ pentru fiecare din cele $j-i$ persoane din submulțime și să inmulțim cu $1-infection\_chance[i]$ pentru fiecare persoană sănătoasă care nu se află în mulțime.

Acum, algoritmul calculează valorile matricei $heal[i,j]$ care reprezintă probabilitatea ca dacă într-o zi, avem $i$ persoane infectate la prânz, să avem $j$ persoane infectate seara cu următoarea formulă:

```
healing_prob=q/100
heal=np.zeros((n+1,n+1),dtype=float)
for i in range(1,n+1):
    for j in range(1,n+1):
        if i<j:
            heal[i,j]=0
        else:
            heal[i,j]=scipy.special.comb(i, i-j)*(healing_prob**(
               i-j))*((1-healing_prob)**j)
```

Idea din spatele formulei este că noi căutăm o submulțime de $i-j$ persoane bolnave și calculăm probabilitatea ca aceste persoane să se vindece, și ca nimeni altcineva să nu se vindece. Coeficientul binomial $\binom{i}{i-j}$ apare deoarece putem alege această submulțime în $\binom{i}{i-j}$ moduri. Acum că am ales submulțimea, trebuie să înmulțim cu $healing\_prob$ pentru fiecare din cele $i-j$ persoane din submulțime și să inmulțim cu $1-healing\_prob$ pentru fiecare persoană bolnavă care nu se află în mulțime.

Acum că avem matricele $inf$ și $heal$, vrem să calculăm și matricea $dp[i][j]$ care reprezintă probabilitatea ca dacă începem ziua cu $i$ persoane infectate, să avem $j$ persoane infectate seara cu următoarea formulă:

```
dp=np.matmul(inf,heal)
```

Matricea $dp$ este pur și simplu produsul matriceal intre $inf$ și $heal$ deoarece formula

$$dp[i][j] = \sum_{k=1}^{n} \text{inf}[i][k] \cdot \text{heal}[k][j]$$

ia în considerare fiecare caz în care avem $k$ persoane infectate la prânz și această formulă este chiar formula produsului matriceal.

Acum, algoritmul calculează valorile matricei $final[i,j]$ care reprezintă probabilitatea ca dacă în dimineața primei zile, aveam $i$ persoane infectate, să avem $j$ persoane infectate în seara celei de a $Z$-a zi, după formula:

```
days=Z
final=np.identity(n+1,dtype=float)
for i in range(1,days+1):
    final=np.matmul(final,dp)
```

Matricea $final$ este pur și simplu matricea $dp$ ridicată la puterea $Z$, deoarece, analog ca și în cazul anterior, înmulțirea de matrice tratează fiecare caz intermediar de a avea $x_i$ persoane infectate după a $i$-a zi, pentru fiecare $i \in \{1, 2, 3, ..., Z-1\}$

În cele din urmă algorimul calculează și afișează valoarea medie căutată cu ajutorul formulei:

```
initial_cases=K
ans=0.0
for i in range(0,n+1):
    ans+=i*final[initial_cases][i]
print(ans)
```

## 3. Găsirea numărului de simulări necesare

Pentru găsirea numărului $R$ de simulări necesare pentru a avea o eroare de $\epsilon$ cu o acuratețe de $a\%$, vom folosi următoarea formulă dedusă din inegalitatea lui Cebâșev:

$$R = \left\lfloor \frac{\mathbb{V}ar[X]}{\epsilon^2} \cdot \frac{1}{1 - \frac{a}{100}} \right\rfloor + 1$$

unde $X$ este variabila aleatoare care are valoarea $i$ cu probabilitatea $final[K][i]$ pentru fiecare $i \in \{0, 1, 2, 3, ..., N\}$

De asemenea, pentru deducerea acestei formule am folosit următoarea formulă pentru varianța lui $S_n$:

$$\mathbb{V}\text{ar}(S_n) = \frac{\mathbb{V}\text{ar}(X)}{n}$$

Unde:

$$S_n = \frac{X_1 + X_2 + \cdots + X_n}{n}$$

Astfel aplicând inegalitatea lui Cebâșev, acuratețea dorită ne este garantată, dacă alegem $n \geq R$:

$$P\left(|S_n - \mathbb{E}[S_n]| \geq \epsilon\right) \leq \frac{\mathbb{V}ar(S_n)}{\epsilon^2} = \frac{\mathbb{V}ar(X)}{n\epsilon^2}$$

Mai jos, se găsește codul care afișează acest număr căutat, $R$:

```
ans=0.0
for i in range(0,n+1):
    ans+=i*final[initial_cases][i]

expected_value=ans
expected_value_of_square=0.0

for i in range(0,n+1):
    expected_value_of_square+=i*i*final[initial_cases][i]
```

```
variation=expected_value_of_square-(expected_value*expected_value
    )


accuracy_percentage=95

epsilon=0.25

sim_number=((variation/(epsilon*epsilon))/(1-accuracy_percentage
    /100))+1
sim_number=int(sim_number)

print(sim_number)
```

# 4. Teoria simulării

Cele $N$ persoane din oraș sunt reprezentate cu un vector cu $N$ celule. Numărul din a $i$-a celulă este egal cu 1 dacă a $i$-a persoană este bolnavă și este egal cu 0 în caz contrar.

## 4.1 O singură simulare

Într-o simulare individuală, algoritmul urmează acești pași pentru fiecare dintre cele $Z$ zile:

1. Se parcurg toate perechile de oameni. Dacă un infectat se întâlnește cu un sănătos, acesta din urmă se infectează cu o probabilitate de $p\%$.
2. La finalul zilei, se simulează procesul de vindecare. Fiecare persoană bolnavă are o probabilitate de $q\%$ de a se vindeca.

## 4.2 Monte Carlo: Algoritmul complet

Algoritmul Monte Carlo execută mai multe simulări pentru a calcula valoarea medie așteptată a numărului de persoane infectate. Pașii principali sunt:

1. Se execută $R$ simulări independente, unde $R$ este numărul de simulări necesare pentru a avea acuratețea dorită, care este dat de formula din algoritmul pentru estimarea teoretică descris mai sus.
2. Pentru fiecare simulare, se aplică algoritmul descris în secțiunea imediat anteroară.
3. După toate simulările, se calculează valoarea medie a numărului de persoane infectate la sfârșitul celor $Z$ zile.

După cele $R$ rulări, rezultatul final este media numărului de persoane infectate, ceea ce reprezintă o estimare destul de bună a valoarii medie așteptată.