NTI Gymnasiet - Nacka Strand
Teknikprogrammet-Teknikvetenskap
Gymnasiearbete 100p
HT 2023 - VT 2024

En ny potentiell automatiskt kortleksblandare

Proof of Concept av en kortblandare

Namn: Eduards Abisevs
E-post: eduards.abisevs@elev.ga.ntig.se
Namn: Leo Altebro
E-post: leo.altebro@elev.ga.ntig.se
Handledare: Elias

Typsatt med LaTeX.
Kompilerad 10 december 2023.

Innehåll

1 Inledning

1.1 Introduktion till ämnet

Spelbranschen är en industri som omsätter miljardbelopp och precis som alla industrier utvecklas den med resten av världen. Industrier över hela världen har som mål att hitta nya sätt att effektivisera och sänka kostnad på det arbete som utförs för att bedriva vinst och spelbranschen har följt denna långvariga trend med till exempel online casinon. Inom spelbranschen är kortspel vanligt förekommande, att försöka automatisera dem är ett logiskt steg att ta, men för en så stor industri vill man vara extra säker på att allt utförs på bästa sätt. Alla sätt att blanda kort är nämligen inte lika bra, vilken sorteringsmetod som används kan ha påverkan på resultatet. Teknologins inflytande inom spelbranschen ökar kraftigt, de största delarna av branschen bedrivs mer och mer av maskiner och algoritmer som får stor potentiellt inflytande på spelresultat, därför är det bäst att börja tänka på möjliga problem så snart som möjligt. Redan nu finns det blandningsmaskiner för kortlekar som vi inte vet någonting om; varken hur de funkar eller hur bra de är. Allt som vi vet är de är certifierade av tredjepartsföretag. Faktumet att de kan kosta upp till 100 000 kr betyder att inte vem som helst kan ha tillgång till en kortblandningsmaskin.

1.2 Syfte

Syftet med denna undersökning är att ta fram den hypotetiskt bästa kortblandaren utifrån två faktorer: 1) hur slumpmässigt den algoritm som den byggs efter kan blanda kort; 2) till vilken grad den potentiella maskinen byggd efter algoritmen skulle fungera i verkligheten. Med resultaten förväntas variationen i slumpmässighet för olika blandningsmetoder kunna visas upp samt kunna använda de resultaten för att komma fram till den hypotetiskt definitiva maskinen, något som är viktigt då spelbranschen handlar mycket om chans. Det är därför viktigt att se till så allt funkar på bästa sätt. I den här vetenskapliga rapporten kommer en jämförelse av hur effektivt framtagna blandningsmetoder kan fungera i en hypotetisk blandningsmaskin att utföras. Samtidigt som man kan utforska hur en dator kan göra slumpmässiga sekvenser på ett effektivt sätt med tanken att den ska tillämpas till en potentiell kortblandare. Detta är viktigt för att spelbranschen är en stor industri som handlar mycket om tur, att se till att de slumpmässiga resultaten är framtagna på bästa möjliga sätt är en essentiell del. Vi söker med hjälp av data svaret på frågan:

Utifrån slumpmässighet och effektivitet, vilken kort blandningsmetod är bäst för en potentiell spelkortsblandare som uppfyller följande:

- Fysiskt tillämpning: Hur pass väl den kan framställas i verkligheten
- Effektivet: Utifrån mjukvaras och hårdvaras perspektiv
- Kortblandnings slumpmässighet

2 Teori

2.1 Beräkningsteori

Beräkningsteori är en del av matematik vars syfte är grundat i hur och om problem kan bli lösta på olika beräkningssätt. Beräkningsteori har flera olika grenar. En gren handlar om vad som går att bevisa inom matematik angående om nummer och funktioner är beräknelig eller inte. Med beräknelig menas något som kan beräknas, det vill säga värderas, uppfattas eller förutses, när det kommer till matematik syftar det på att bestämma något via matematiska modeller/processer, alltså att kalkylera.

Figur 1: Your caption here

2.2 Slumpmässighet

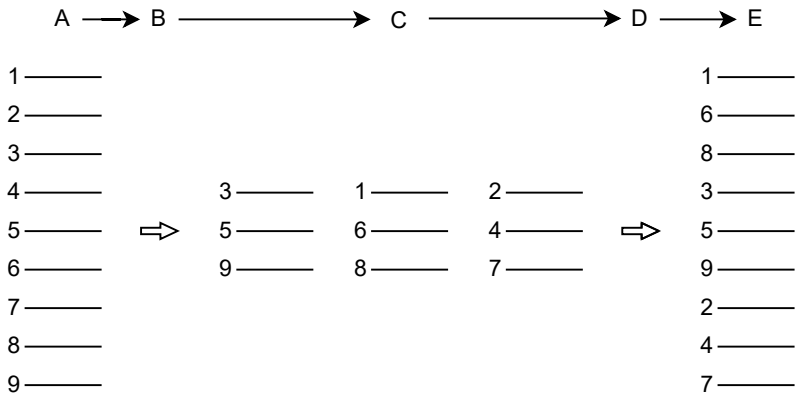
Slumpmässighet är en egenskap som anger att något sker utan klart mönster, när något är helt slumpmässigt är det i princip omöjligt att förutse. Slumpmässighet inom matematik är byggd beräkningsteori och den delas upp i två beroende på om det gäller slumpmässighet av en bestämd mängd eller en oändlig mängd av objekt (Terwijn, 2016).

2.3 algoritms

Theoretical Whats and whys angående algorithmer

2.3.1 Pile shuffle

Pile shuffle är en typ av kortleksblandings metod som genomförs med fysiska kort. Processen utgår att ett kort från kortleken läggs i en av högar. Processen försätts tills alla kort från kortleken är i sina högar. Sedan läggs ihop alla högar i ett nytt kortlek.



Figur 2: Steg i processen för Pile Shuffle. Illustrationen visar de iterativa stegen från den ursprungliga högen (A), genom uppdelning i högar (B), temporara högar (C), omarrangering av temporara högar (D), och tillbaka till en enda hög (E). Pilar indikerar riktningen för blandningsprocessen.

2.3.2 Riffle shuffle

Mest kända kortleksblandings metoden som har undersökets under många tiotals år.

2.4 Klassiskt poker test

Ett klassiskt poker test används att avgöra slumpmässighet i numeriska sekvenser, oftast för att testa slumptalsgeneratorer. Testet utförs genom att 3 till 5 nummer väljs ut ur en sekvens och placeras i en av sju kategorier beroende på mönstret som talen har. Mönsterna är baserade på händer i poker vilket är varför testet kallas poker test. Det olika mönster som letas efter i talen visas itabell 1.

Pokerhand	Mönster
Femtal	AAAAA
Fyrtal	AAAAB
Kåk	AAABB
Tretal	AAABC
Tvåpar	AABBC
Par	AABCD
Högt kort	ABCDE

Tabell 1: Vanliga kategorier till poker testet

2.4.1 Chi-två-test

Inom statistik används ett *goodness-of-fit* test för att mäta hur pass väl fördelningen för den data som observerats stämmer överens med fördelningen som data förväntas ha utifrån en model. Karl Pearsons chi-två-test kan användas som *goodness-of-fit* test, det använder chitvåfördeling. I testet jämnförs ett värde χ^2 med ett kritiskt värde för att bestäma om den observerade följer den förväntade fördelningen. χ^2 beräknas enligt formlen

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

Där O_i är observerade frekvensen för en kategori av data i och E_i är förväntade frekvensen för kategorin i (National Institute of Standards and Technology, 2023). Det kritiska värdet fås av antalet frihetsgrader(kategorier - parametrar) och signifikansnivån som bestämts för testet.

Chi-square, chi-två testing eller som den också betecknas χ^2 är statistikt test som oftas används till att definera och beräkna slumpmässighet av bestämd sannolikhet.

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

Vart O_i är observerade frekvensen för kategorin i och E_i är förväntade frekvensen för kategorin i (National Institute of Standards and Technology, 2023).

2.5 Pseudoslumptalsgenerator (PRNG)

Här berätta om ChaCha20 (Bernstein, 2008) den som är biobliotekes rand (Developers, 2022) default PRNG och mersenne twister som egentligen var första hands val (Matsumoto och Nishimura, 1998) men som vi sedan bytte till ChaCha för att i rust är det relativt svårt att implementera och använda mersenne twister. Berätta om dem skillnaderna som finns. Alltså nämn att för att det här är som et proof of concept, borde vid senare tillfälle användas en hårdvara PRNG och icke software PRNG. Men pga detta undersökning handlar mesta dels om stora mängder av data simulationer så måste vi använda och Sacrifice"verkliga till att lättare skapa simulationer!

3 Metod

Ett av målet med undersökningen var att avgöra hur slumpmässiga algoritmerna var. För att undersöka detta kriterium valdes det att simulera stora mängder av kortblandningar. Metoden kunde indelas i tre huvudområdena 1) framtagande av algoritmerna 2) rådata generering via simulation 3) normalisering av rådata och statistiska tester

3.1 Testmiljö

Datan insamlades från digital simulation på olika kortblandningsmetoder. De algoritmerna som undersöktes var skrivna i programspråk Rust i försök att göra dem så likt verkligheten som möjligt. För insamling av data användes programmeringsspråket Python 3.11 och utnyttjades bibliotek som Numpy till datamängd bearbetning, Scipy till statistisk analys (Chi-square) och Matplotlib till visualisering av resultat. Undersökningen kördes på en Linux dator med följande specifikationer:

Typ	Specification
Processor	AMD Ryzen 5 3600 6 cores / 12 threads 3.6 GHz
RAM	15.93 GB
Hårdisk	KINGSTON SA400S3, 447 GB
Operativsystem	`Ubuntu 22.04.3 (LTS) x86_64,` `Linux kernel 6.2.0-36-generic`

Tabell 2: Linux Dator

3.2 Kortblandings algoritmernas implemenation

Då tanken var att algoritmerna skulle potentiellt användas i ett fysiskt inbäddad system fanns det vissa kriterier som algoritmerna skulle följa: det skulle gå att bygga en maskin, den skulle vara slumpmässig, dvs. simulation skulle vara så lik verkligheten som möjligt. Med det sagt så måste vi bestämma oss för en pseudo-random number generator (PRNG) algoritm bestämmas. En väldigt känd och testad algoritm är Mersenne Twister (MT 19937) som har en väldigt långt period innan siffrorna börjar att upprepa sig, mer exakt 2^{19937} (Matsumoto och Nishimura, 1998).

3.2.1 Riffle shuffle

Det här är riffle shuffle, vi ska nu parata om detta närmare... så funkar det här då?? Yeeep det gör det $\binom{55}{2} = \theta$

3.2.2 Implementation av Pile shuffle

Inspiration att undersöka pile shuffle kom ifrån 3DprintedLife (2021), där pile shuffle implementerades i en 3d printat kortleksblandare.

3.3 Simulation för kortblandningar och implementation

Simulationen utvecklades med Rust version 1.73.0 och användes bibliotek (Rust Crates) som Rand version 0.8 för PRNG (Developers, 2022). Rayon för enkel användning av parallell multiprocessing (Matsakis och Stone, 2022). Simulation utgår från att man genererar matris med bestämd mängd av kortlekar. Vart varje kort, låt kalla dem till x som följer följande mängden $\{x \in \mathbb{N}, 0 \leq x \leq 51\}$ Resulterande datamängden är en matris D med 52 kolumner och m antal rader, vart m värde visar hur många kortlekar det finns i datamängden. rust is nice crazy

$$D = \begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & \cdots & x_{0,51} \\ x_{1,0} & x_{1,1} & x_{1,2} & \cdots & x_{1,51} \\ x_{2,0} & x_{2,1} & x_{2,2} & \cdots & x_{2,51} \\ \vdots & \vdots & \vdots & & \vdots \\ x_{m,0} & x_{m,1} & x_{m,2} & \cdots & x_{m,51} \end{bmatrix}$$

För att bestämma m värde följdes ett kriterium från ett av testerna som ska utföras, klassiskt poker test mer i detalj beskriven i underrubriken 2.4.1 . Till hjälps användes (National Institute of Standards and Technology, 2023) internätkälla, i denna nämns det att det är viktigt för chi-square approximationens trovärdighet att minsta kategorin ska inte vara mindre än fem teoretiska framkommande i den kategorin. Detta ska tillämpas till simulationen på följande sätt. Det är viktigt att hitta den minsta kategorin som kan förekomma. I poker är den mest sällsynta pokerhand kungliga färgstege (Royal Flush) som kan räknas ut på följande sätt antal av alla kortkombinationer med fem kort. $\binom{52}{5} = 2\,598\,960$ Kungliga färgstege finns det fyra kombinationer av i kortleken som kan räknas ut på följande sätt totala antal kombinationer av kungliga färgstege delat med totala antal femkorts kombinationer från standardkortlek.

$$P(\text{Kunglig färgstege}) = \frac{\binom{4}{1}}{\binom{52}{5}} = \frac{4}{2\,598\,960} \approx \frac{1}{649\,740}$$

$$m = P(\text{Kunglig färgstege}) \times 5 = 3\,248\,700$$

detta blir den minimala datamängden i simulationen .

3.3.1 Val av datatyp & rådatalagring

Efter teoretiska beräkningar av datamängden är det viktigt att välja lämplig datatyp att representera data i. Högsta talet som behövs är 51 definerad tidigare och betecknat med x. Detta betyder att minsta datatypen får användas som är 8 bit eller 1 byte långa som kan innehålla följande värden

$$\{b \in \mathbb{N}, 0 \leq b \leq 255\}$$

som övernsstämmer med datamängdens högsta talet x. Detta ger möjligheten att använda den fundermantala datatypen som används i minnes adressen. Fördelen med detta är det simplifierar lagring och inmatning av datamängder i t.ex Python med NumPy.

Andra aspekten till varför datatypen är viktigt är att spara minnen för att det kommer att simuleras och sparas tiotals rådata filer varje fil kommer att väga $161MB$, detta kan räknas ut på följande sätt

$$\frac{\text{totala bytes}}{\text{megabyte (MB)}} = \frac{52 \times m}{1024 \times 1024} = \frac{52 \times 3\,248\,700}{1024 \times 1024} = \frac{168\,932\,400}{1\,048\,576} \approx 161MB$$

som det är enkelt att spara data i binärt format för att 8 bits eller 1 byte är en fundamental och uniform mått i minnesadressen. Med det sagt ingen data bearbetning behövs och det är enkelt att ladda in denna i till exempel Python med numpy till statistiska testerna av rådata

Därför att sista teoretiska delen att bestämma är hur många iterationer per algoritm ska genomföras. Med iterationer menas hur många gånger kortleken blandas följande på varandra.

3.4 statistiska tester

Rådata som sparades efter simulationen är laddat in som en dimensionell serie med hjälp av Numpy efter det är rådata återställt tillbaka till tvådimensionell matris för vidare bearbetning och analys med följande metoder: Klassisk Poker Test och medelvärde av korta positioner i kortleken över hela datamängden kalkylerad såsom avvikelser av dessa positioner. Utförligt beskrivning i underrubriker.

3.4.1 implemention av Klassiskt poker test

Poker Test utgår ifrån att man karaktäriserar typer av mönster till pokerhänder och kalkylerar den absolutvärdet χ^2 i detalj beskrevs teorin bakom χ^2 testet i sektion 2.4.

I standardkortlek finns det 52 kort med fyra olika färger (Spader, Hjärter, Ruter och klöver) och 13 valörer (2, 3, 4, 5, 6, 7, 8, 9, 10, Knekt, Dam, Kung, Ess). Av skall att testerna utförs med standard kortleken,

Tabell 3: Alla pokerhänders namn, relativt mönster och antal av kombinationer

Pokerhand	Example på mönster	Antal kombinationer
Kunglig färgstege	$A\spadesuit\, K\spadesuit\, Q\spadesuit\, J\spadesuit\, 10\spadesuit$	4
Färgstege	$K\spadesuit\, Q\spadesuit\, J\spadesuit\, 10\spadesuit\, 9\spadesuit$	36
Fyrtal	$A\spadesuit\, A\heartsuit\, A\diamondsuit\, A\clubsuit\, K\spadesuit$	624
Kåk	$A\spadesuit\, A\heartsuit\, A\diamondsuit\, K\clubsuit\, K\spadesuit$	3 744
Färg	$K\spadesuit\, Q\spadesuit\, J\spadesuit, 10\spadesuit\, 8\spadesuit$	5 108
Stege	$K\spadesuit\, Q\heartsuit\, J\diamondsuit\, 10\clubsuit\, 9\spadesuit$	10 200
Triss	$A\spadesuit\, A\heartsuit\, A\diamondsuit\, K\clubsuit\, Q\spadesuit$	54 912
Två par	$A\spadesuit\, A\heartsuit\, K\diamondsuit\, K\clubsuit\, Q\spadesuit$	123 552
Ett par	$A\spadesuit\, A\heartsuit, K\diamondsuit\, Q\clubsuit\, J\spadesuit$	1 098 240
Högt kort	$A\spadesuit\, Q\heartsuit\, J\diamondsuit\, 5\clubsuit\, 4\spadesuit$	1 302 540
	Summan:	2 598 960

valdes det att anpassa χ^2 testet att använda alla poker händer. Matematiken speciellt kombinatoriken att räkna ut sannolikheter och frekvensen av alla poker händer är relativt svårt uppdrag därför addopterdes det Armstrong (2006) uträkningar se tabell 3. Detta medföljde med mer komplex karakterisering av pokerhänder än när man använder χ^2 testet att till example testa slumptalsgeneratorer. Valet log med att datamängderna innehåller kortlekar till att göra simulationer mer realistiska och anpassade till hur ett riktigt kortleksblandare skulle fungera i drift. Därför valet att anpassa statistiska tester till slutmål användnings sätt verkade vara ett logiskt steg att ta.

Poker Test kan anpassas till vilken som helst sekvens av nummer 3-5 stora mängder. Men för att vår simulation använder vi samma nummer som pokerkort därför kan vi anpassa klassiska poker test var man använder alla möjliga pokerhänder av sekvens av 5 kort. Resultatet från karakterisering av händer använder man i en chi-square testet. För att göra testet rätt följde jag Engineering Statistics handbok (NIST 2012). Dvs för att avgöra hur många händer uppkom med avseende till teoretiska värdena EXPECTED vs OBSERVED värdena. sen jämför man chi-square resultatet till en CRITICAL värde om den värden som vi fick är mindre kan vi med säkerhet säga att algoritmen är slumpmässigt och vice versa.

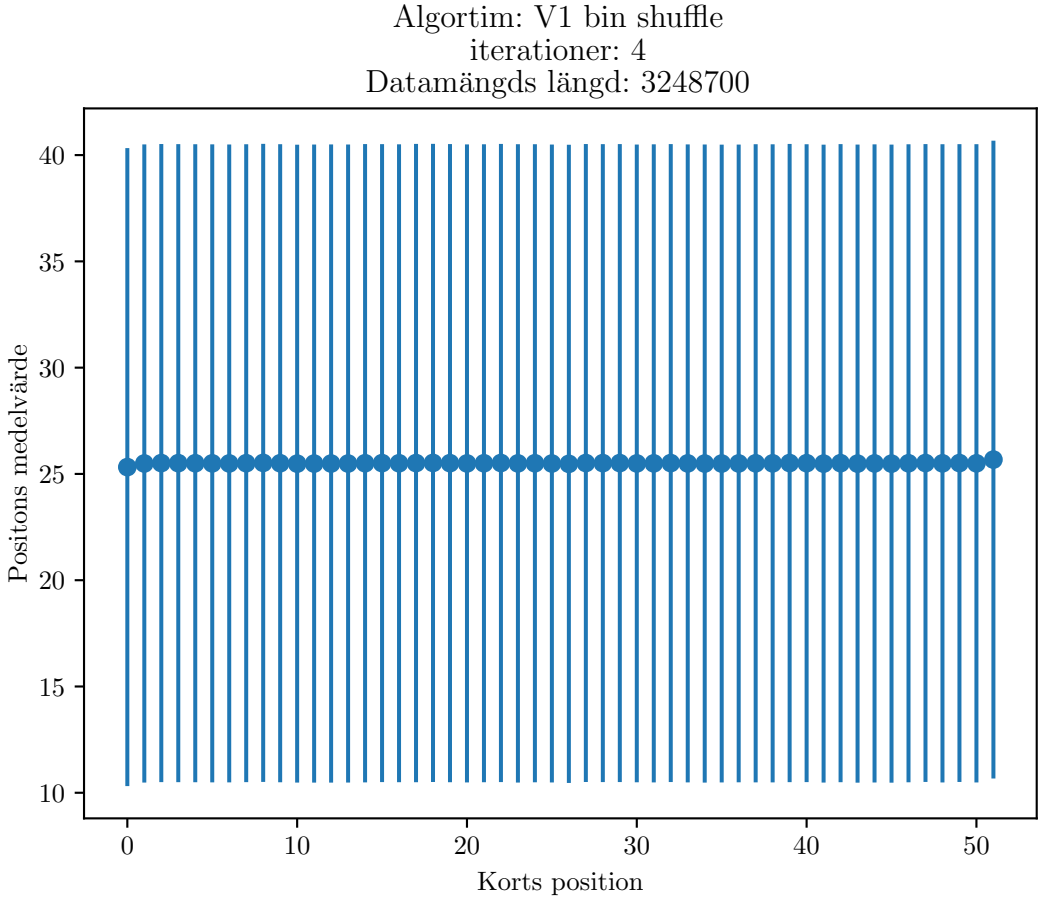
Testet utgår i två generella steg kategorisera fem kort och seden

3.4.2 Implemention av StdMean testet

Datamängden är utformat på sätt att kort värde är alltså dess position i kortleken. Första kort är 0 dvs position 0 i kortleken. Kalkylerar varje columns korts medelvärde och standaravvikelse ska man förustpå att medelvärde borde ligga nära $51/2 = 25.5$ detta betyder att i position 0 har alla kort varit i, och om standaravviklse i figuren är höga torn har position 0 stort variation av vilka kort som var där. Detta är då en indikator på en uniform distribution som betyder en slumpmässigt algoritm.

4 Resulat

Result är icke bestämt i vilken format kommer att skrivas Men dem kommer att innehålla figur 3. Antagligen mest intresanta figurer kommer att visas här och alla andra Bilagas i Appendix. För att det kommer att vara minst 30 figurer och chi två tabeller(dem kan göras i ett större tabel, antar jag)



Figur 3: Ett test resultat från StdMean testet

Och tabeller över resultat ifrån χ^2 testet.

Tabell 4: Test table resultat från v1 bin shuffle algoritmen

Pokerhand	Observerad frekvens	Förväntad frekvens	χ^2
Total	3 248 700	3 248 700	8.85

5 Diskussion

Och sista sektionen här, men den är relativt straightforward.

Bibliografi

3DprintedLife (2021). *Rigged Card Sorting Machine - ALWAYS Get The Hand You Want!* URL: `https://www.youtube.com/watch?v=eMTXyl7tPEk` (hämtad 2023-08-15).

Armstrong, Drew (2006). "Probability of Poker Hands". I: URL: `https://www-users.cse.umn.edu/~reiner/Classes/Poker.pdf` (hämtad 2023-12-09).

Bernstein, D. J. (jan. 2008). "ChaCha, a variant of Salsa20". I: URL: `https://cr.yp.to/papers.html#chacha` (hämtad 2023-12-06).

Developers, The Rand Project (2022). *Rand: A Rust library for random number generation*. Version 0.8. URL: https://crates.io/crates/rand.

Matsakis, Niko och Josh Stone (2022). *Rayon: Data-parallelism library for Rust*. Version 1.8.0. URL: https://crates.io/crates/rayon.

Matsumoto, Makoto och Takuji Nishimura (jan. 1998). "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator". I: *ACM Trans. Model. Comput. Simul.* 8.1, s. 3–30. ISSN: 1049-3301. DOI: 10.1145/272991.272995. URL: https://doi.org/10.1145/272991.272995.

National Institute of Standards and Technology (2023). *Engineering Statistics Handbook - Section 1.3.5.15: Chi-Square Goodness-of-Fit Test*. URL: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm (hämtad 2023-10-04).

Terwijn, Sebastiaan A. (2016). "The Mathematical Foundations of Randomness". I: *The Challenge of Chance: A Multidisciplinary Approach from Science and the Humanities*. Utg. av Klaas Landsman och Ellen van Wolde. Cham: Springer International Publishing, s. 49–66. URL: https://doi.org/10.1007/978-3-319-26300-7_3 (hämtad 2023-04-10).

Bilagor

A Github

Länk till Github repository vart Rust och Python källkod till simulationen respektive statistikst anlys är samlad, samt källkod till latex med vilken detta rapport hade skrivits länk: https://github.com/Abishevs/gymarbete

B Template att bifoga ett inline kod

Listing 1: Python example

```python
# Your source code here
print("Hello, World!")
```

C Template att bigoga källkod ifrån git repo