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

Ett nytt potentiell automatiskt kortleksblandare till kortspel

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 8 december 2023.

Innehåll

1 Inledning **2**
1.1 Introduktion till ämnet . . . 2
1.2 Syfte . . . 2

2 Teori **2**
2.1 Beräkningsteori . . . 2
2.2 Slumpmässighet . . . 3
2.3 algoritms . . . 3
2.3.1 riffle shuffle . . . 3
2.4 Klassikt poker test . . . 3
2.4.1 Chi-två testet . . . 3
2.5 Pseudoslumptalsgenerator (PRNG) . . . 3

3 Metod **3**
3.1 Testmiljö . . . 4
3.2 Kortblandings algoritmernas implemenation . . . 4
3.2.1 Riffle shuffle . . . 4
3.3 Simulation för kortblandningar och implemenation . . . 4
3.3.1 Val av datatyp & rådatalagring . . . 5
3.4 statistiska tester . . . 5
3.4.1 implemention av Klasiskt poker test . . . 5
3.4.2 Implemention av StdMean testet . . . 5

Referenser **5**

Bilagor **6**

A Github **6**

B Källkod simulation **6**

C Källkod algorithm 1 implemention **6**

D Figure example **6**

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
- Tillräckligt snabbt med tillräckligt bra slumpmässighet- en bra kompromiss

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 angånde algorithmer

2.3.1 riffle shuffle

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

2.4 Klassikt poker test

Klassiskt poker test är ett test för att avgöra slumpmässighet i nummer sekvenserna oftast använd i att testa slumptalsgeneratorer. Testet utgår till att 3 till 5 nummer ur sekvensen väljs och dem är katigoriserad i poker händer. Vanligaste examplet är som i tabell 1.

poker hand	Mönster
Five of a kind	AAAAA
Four of a kind	AAAAB
Full house	AAABB
Three of a kind	AAABC
Two pairs	AABBC
One pair	AABCD
All diffrent	ABCDE

Tabell 1: Vanliga kategorier till poker testet

2.4.1 Chi-två 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.

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 implemntera och använda mersenne twister. Berätta om dem skillanderna 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 $\binom{55}{2} = \theta$

3.3 Simulation för kortblandningar och implemenation

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 datan 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 inehålla följande värden

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

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

Andra aspektet 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 datan 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 rådatan

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 rådatan 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 Klasiskt poker test

till denna användes Virtanen m. fl. (2020) Poker Test utgår till att man karaktäriserar typer av mönster till pokerhänder. 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.

3.4.2 Implemention av StdMean testet

Datanmängden 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 standarvvikelse 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 figurer är höga torn har position 0 stort varition av vilka kort som var där. Detta är då en indikator på en unifrom distribution som betyder en slumpmässigt algoritm.

Referenser

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).

Virtanen, Pauli m. fl. (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". I: *Nature Methods* 17, s. 261–272. DOI: 10.1038/s41592-019-0686-2.

Bilagor

A Github

github link: https://github.com/Abishevs/gymarbete

B Källkod simulation

Listing 1: Python example

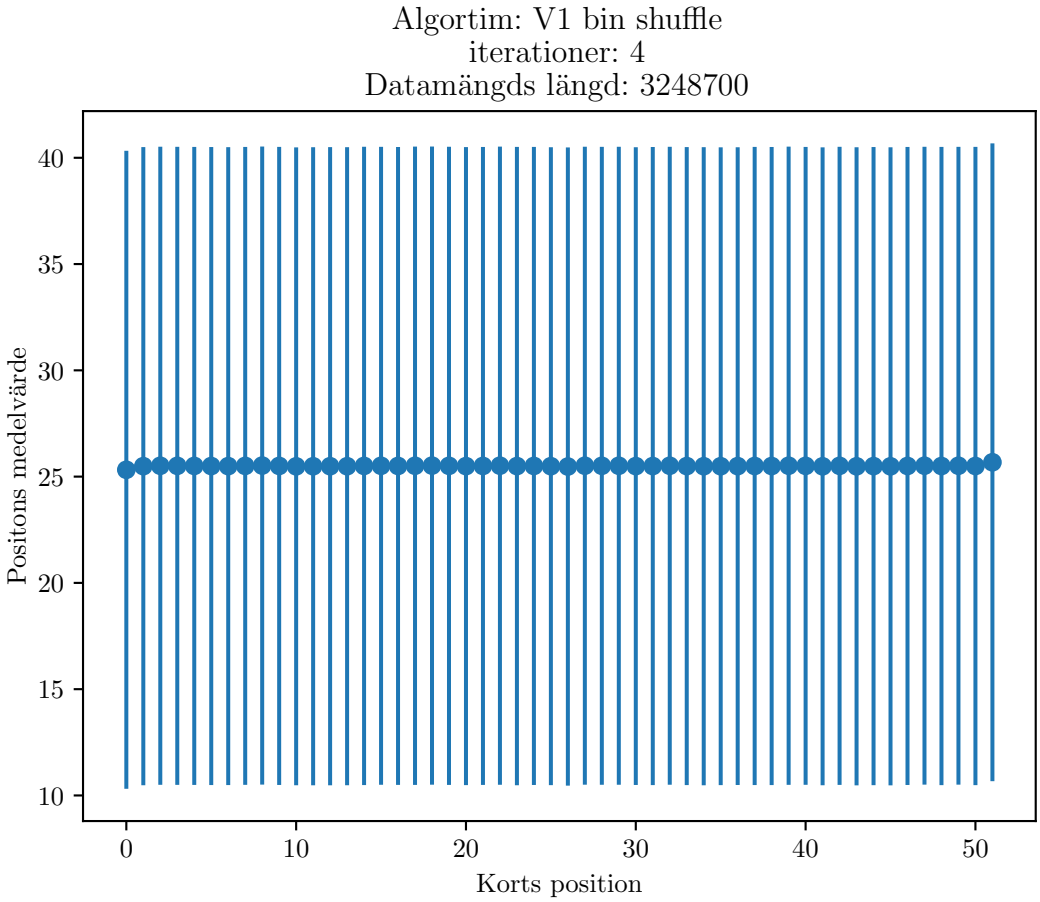
```python
# Your source code here
print("Hello, World!")
```

C Källkod algorithm 1 implemention

Listing 2: algorithm 1 implemention

```rust
impl  ShufflingAlgorithm for Algorithm1{
    fn name(&self) -> &str {
        "v1_bin_shuffle"
    }

    fn shuffle(&self, deck: &mut Deck){
        const BIN_COUNTS:usize = 6;
        let mut bins: Vec<Vec<Card>> = vec![Vec::new(); BIN_COUNTS];

        let mut rng = rand::thread_rng();
        for &card in deck.iter() {
            let random_bin = rng.gen_range(0..BIN_COUNTS);
            // put the cards in random bins
            bins[random_bin].push(card);
        }

        let mut deck_position = 0;
        // Reassemble the deck
        for bin in bins {
            for card in bin {
                deck[deck_position] = card;
                deck_position += 1;
            }
        }
    }

}
```

D Figure example



Figur 2: A PGF figure example från `matplotlib`.