Metody Probabilistyczne i Statystyka

Zadanie domowe 2

Rafał Włodarczyk 2024-11-24

Spis treści

1 Zadanie 1

1.1 Technologie wykorzystane w symulacji

Do realizacji zadania wykorzystałem język *rust*. Liczby pseudolosowe są generowane za pomocą biblioteki *rand_mt* (Mersenne Twister). Wykresy stworzyłem za pomocą narzędzia *matplotlib*.

1.1.1 Pojedyncza symulacja

Do uzyskania wyników wszystkich pomiarów, każdorazowo losowany jest urna do której zostanie wrzucona kula. W trakci symulacji mierzone są ilości urn z 0, 1 oraz 2 kulami, oraz $u(n)$ w tysięcznej iteracji. Po jej zakończeniu, czyli po tym, gdy w każdej z urn znajdują się co najmniej dwie kule zliczana jest różnica $d(n) - c(n)$.

```rust
let mut bins = simulation::create_bins(n);
let mut metrics = types::Metrics::new();
let mut count_0: u32 = n; // count occurrences of 0
let mut count_1: u32 = 0; // count occurrences of 1
let mut count_2: u32 = 0; // count occurrences of 2

let mut i = 0;
while count_2 != n || count_1 != n {
let selected = simulation::randomly_place_ball(&mut bins, n);

```

```rust
match bins[selected] {
    1 => {
        count_1 += 1;
        count_0 -= 1;
    }
    2 => count_2 += 1,
    _ => {}
}

// [...] all checks

}
```

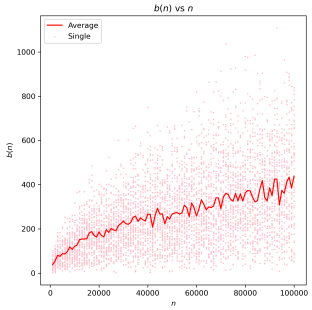
1.1.2 Grupa symulacji

Dla każdego $n \in \{1000, 2000, ..., 100000\}$ wykonuje $k = 50$ niezależnych powtórzeń a następnie uśredniam wyniki każdej zbieranej wartości. W celu usprawnienia prędkości jej wykonywania użyłem biblioteki *rayon*, która pozwoliła mi wykonywać symulację na wielu rdzeniach CPU.

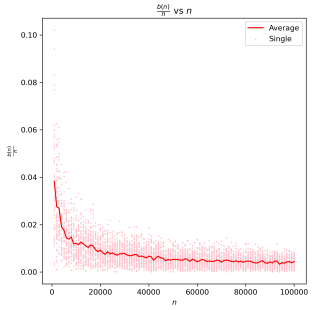
```rust
const SAME_SIM_REPS: usize = 50;

// [...]

let metrics_group: Vec<types::Metrics> = (0..SAME_SIM_REPS)
.into_par_iter()
.map(|_k| {
    // [...] single simulation
})
.collect();
```

1.2 Wykresy

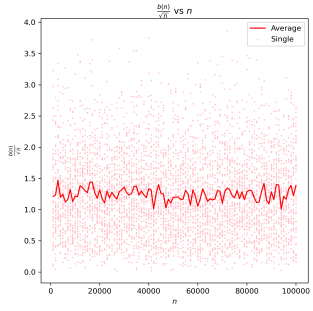
1.2.1 Paradoks urodzinowy - B_n, $\frac{b(n)}{n}$, $\frac{b(n)}{\sqrt{n}}$



Rysunek 1: $B(n)$

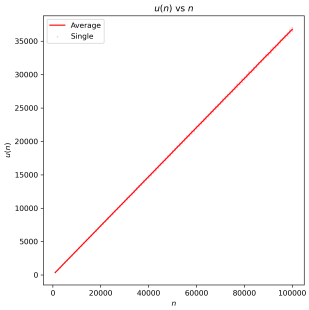


Rysunek 2: $\frac{b(n)}{n}$

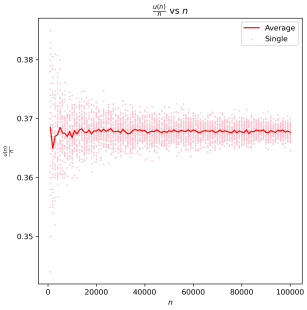


Rysunek 3: $\frac{b(n)}{\sqrt{n}}$

1.2.2 Puste urny po n rzutach - U_n, $\frac{u(n)}{n}$

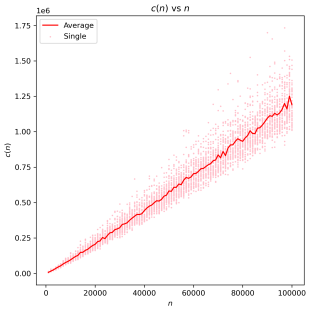


Rysunek 4: $u(n)$

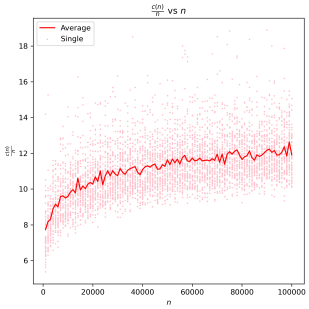


Rysunek 5: $\frac{u(n)}{n}$

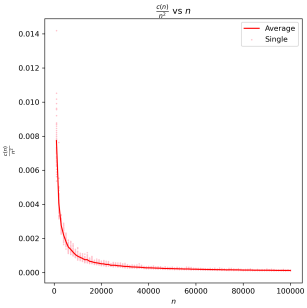
1.2.3 Problem Kolekcjonera Kuponów - C_n, $\frac{c(n)}{n}$, $\frac{c(n)}{n\ln(n)}$, $\frac{c(n)}{n^2}$



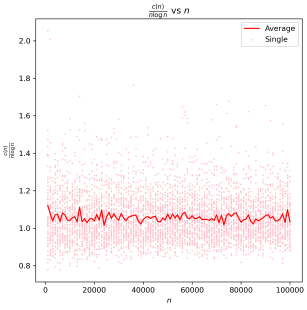
Rysunek 6: $C(n)$



Rysunek 7: $\frac{c(n)}{n}$

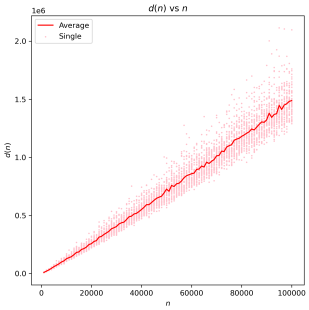


Rysunek 8: $\frac{c(n)}{n^2}$

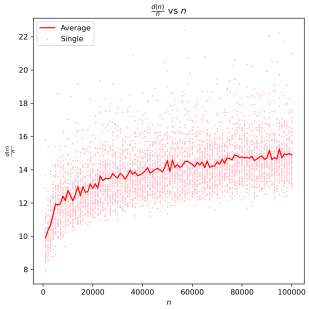


Rysunek 9: $\frac{c(n)}{n\log(n)}$

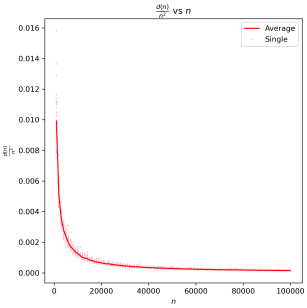
1.2.4 Problem Kolekcjonera Dwóch Kuponów - D_n, $\frac{d(n)}{n}$, $\frac{d(n)}{n\ln(n)}$, $\frac{d(n)}{n^2}$



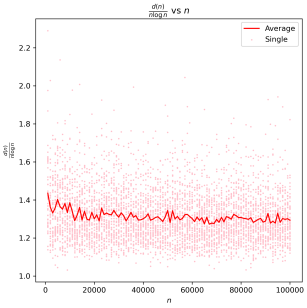
Rysunek 10: $D(n)$



Rysunek 11: $\frac{d(n)}{n}$

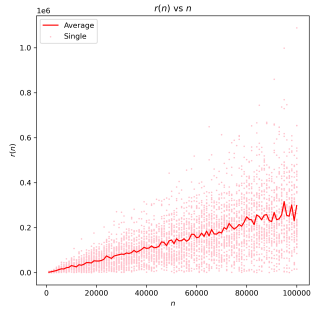


Rysunek 12: $\frac{d(n)}{n^2}$

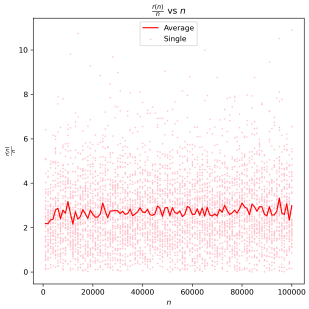


Rysunek 13: $\frac{d(n)}{n\log(n)}$

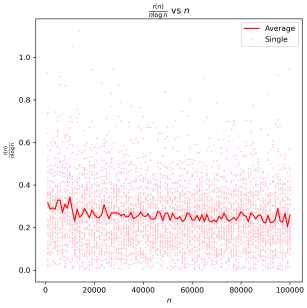
1.2.5 Różnica dwóch powyższych - $R_n = D(n) - C(n)$, $\frac{r(n)}{n}$, $\frac{r(n)}{n\ln(n)}$, $\frac{r(n)}{n\ln(\ln(n)))}$



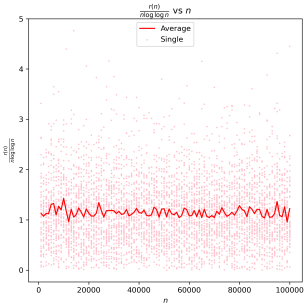
Rysunek 14: $R(n) = D(n) - C(n)$



Rysunek 15: $\frac{r(n)}{n}$



Rysunek 16: $\frac{r(n)}{n\log(n)}$



Rysunek 17: $\frac{r(n)}{n\log(\log(n))}$

2 Zadanie 2

2.1 Rezultaty

2.1.1 Omówienie rezultatów

W zadaniu pierwszym zostały zaprezentowane wykresy obrazujące wyniki poszczególnych symulacji oraz wartości średnich mierzonych parametrów. Wykonane dla nich symulacje oraz uśrednione wyniki dla każdego n pozwalają na wysnucie hipotez dotyczących asymptotyki wartości średnich badanych wielkości.

2.1.2 Koncentracja wyników wokół wartości średniej

Wraz ze wzrostem n widzimy wyraźne ekspansje przedziałów w jakich znajdują się wyniki poszczególnych wartości mierzonych na wykresach liniowych. Wskazać tu można na wartości parametów B, C, D, R. Jest to zdarzenie oczekiwane, związane ściśle z ilością możliwych wartości które może przyjąć zmienna losowa pojedyńczego rzutu kuli. Im większe n, tym bardziej rozrzuconych wyników należy się spodziewać. Dopiero przy wyznaczonych średnich ze wszystkich $k = 50$ przypadków możemy zaobserwować istotne tendencje.

W przypadku zdarzenia U nie widzimy jednak dużego rozstrzału wyników z poszczególnych prób.

2.1.3 Hipotezy odnośnie asymptotyki badanych wielkości

Znając dwa poniższe fakty:

Fakt. Big O Limit. Niech $f, g \in \mathbb{N} \to \mathbb{R}^+$. Zachodzi:

$$f(n) = O(g) \iff \limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

Fakt. Big Theta Limit. Niech $f, g \in \mathbb{N} \to \mathbb{R}^+$. Zachodzi:

$$f(n) = \Theta(g) \iff \left(\limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty\right) \wedge \left(\limsup_{n\to\infty} \frac{f(n)}{g(n)} > 0\right)$$

Możemy wysnuć hipotezy na temat zależności poszczególych mierzonych wartości od parametru n:

1. Moment pierwszej kolizji. Skoro wartości na wykresie z rysunku 3-go $\frac{b(n)}{\sqrt{n}}$ utrzymują się w niewielkim przedziale, to hipoteza: $b(n) = \Theta(\sqrt{n})$
2. Liczba pustych urn. Ze względu na wykres z rysunku 5-go możemy napisać hipoteze: $u(n) = \Theta(n)$
3. Minimalna liczba rzutów, po której w każdej z urn jest co najmniej jedna kula. Z rysunku 9-go możemy hipotetycznie określić asymptotykę na $c(n) = \Theta(n \ln n)$

4. Minimalna liczba rzutów, po której w każdej z urn są co najmniej dwie kule. Z rysunku 13-go możemy hipotetycznie ocenić asymptotykę $d(n) = \Theta(n \ln n)$

5. Różnica dwóch powyższych. Rysunki 15 oraz 17 są bardzo podobne, wobec tego w tej próbce danych możemy wysnuć dwie hipotezy $r(n) = d(n) - c(n) = \Theta(n)$ lub $r(n) = \Theta(n \log \log n)$. Wykresy są zbyt zbliżone do siebie, abym mógł z większą pewnością wybrać jeden z nich.

2.1.4 Intuicje stojące za nazwami *Birthday paradox* oraz *Coupon collector's problem*

Birthday Paradox - Opisuje zaskakująco wysokie prawdopodobieństwo, że w relatywnie niewielkiej grupie osób znajdą się dwie osoby, które mają urodziny w tym samym dniu. Matematyka wskazuje, że w grupie zaledwie 23 osób prawdopodobieństwo, że dwie z nich będą miały urodziny tego samego dnia, wynosi około 50%. Jest to wynik nieoczekiwany i sprzeczny z początkową intuicją.

Coupon collector's problem - Opisuje ile pojedyńczych kuponów musi losowo zebrać kolekcjoner aby móc cieszyć się pełną kolekcja. Aby mieć co najmniej jeden egzemplarz każdego typu, zbieracz musi podjąć znacznie więcej prób, niż wynosi liczba uniklanych kuponów.

2.1.5 Znaczenie *Birthday paradox* w kontekście funkcji hashujących

Birthday paradox znacząco zmniejsza ilość wymaganych prób do znalezienia kolizji (dwóch różnych argumentów dla których funkcja zwraca ten sam wynik) w funkcjach hashujących, wykorzystywanych w większości systemów informatycznych. Jak wiadomo mają one postać skończonych ciągów bitów, wobec tego dla funkcji zwracających krótkie ciągi możliwe jest relatywnie szybkie znalezienie dwóch różnych ciągów wejściowych generujących taki sam hash. Zadaniem kryptograficznych funkcji hashujących jest zatem wytworzenie wysokiej odporności na takie przypadki, które potencjalnie mogą prowadzić na przykład do przyznania nieautoryzowanych dostępów (hash hasła znajduje się w bazie danych pewnego serwisu). Rozwiązaniem jest stworzenie funkcji hashującej zwracającej większą liczbę bitów np $SHA256$, bądź $SHA512$.

2.2 Wnioski

Symulacja odpowiada na zadane pytania oraz prezentuje wystarczający zestaw danych na wysnucie hipotez dotyczących asymptotyki wartości średnich badanych wielkości z dokładnością do Θ. Symulacja prezentuje również przyczynę sprzężonego z nią problemu - istotnego znaczenia *Birthday Paradox* w łamaniu funkcji hashujących.