”Generator Liczb Losowych”

Dawid Urbanik

1. Opis teoretyczny problemu

Projekt polegał na zaimplementowaniu generatora liczb pseudolosowych G o rozkładzie równomiernym, na którego podstawie stworzymy generatory mające następujące rozkłady:

- **J** (jednostajny)
- **B** (Bernoulliego)
- **D** (dwumianowy)
- **P** (Poissona)
- **W** (wykładniczy)
- **N** (normalny)

Następnie generatory zostaną poddane teście serii, który sprawdzi czy wartości liczbowe, które otrzymujemy spełniają postulaty losowości próby.

2. Implementacja rozwiązania

1) Generatory:

Wszystkie generatory znajdują się w pliku generators.py. Pierwsze linijki opisane etykietą DON’T TOUCH to parametry naszego generatora liniowego **G**, które odpowiadają za ‘losowość’ otrzymywanych liczb, a także gwarantują, że cykl, po którym liczby zaczną się powtarzać jest równy m-1.

Poniżej mamy wartość seed, którą możemy dowolnie zmieniać, nie wpływa ona na sposób działania generatora, a tylko definiuje początek i koniec cyklu.

Następnie mamy już same implementacje naszych generatorów w metodach, które teraz opiszę:

G(liczba) – jest to prosty generator addytywny LCG (ang. pseudorandom number linear congruential generator).

W argumencie podajemy ilość liczb, które chcemy otrzymać.

Zwracana jest tablica liczb z przedziału [0,m-1].

J(a, b, liczba) – generator liczb z wybranego przedziału. Szczególnie z przedziału [0,1].

W argumentach podajemy lewy oraz prawy przedział, a następnie ilość liczb, które chcemy otrzymać.

Zwracana jest tablica liczb z przedziału [a, b] .

B(p, liczba) – generator liczb 0 oraz 1.

W argumentach podajemy kolejno prawdopodobieństwo wystąpienia 1, a następnie ilość liczb, które chcemy otrzymać.

Zwracana jest tablica liczb zawierająca 0 oraz 1.

D(p, liczba prób) – generator wykonuje [liczbę prób] Bernoulliego używając generatora B, a następnie zlicza ile z nich zakończyło się sukcesem.

W argumentach podajemy kolejno prawdopodobieństwo wystąpienia 1 w pojedynczej próbie, a następnie ilość prób które wykonamy.

Zwracana jest liczba naturalna.

P(lambda) – generator zlicza ilość zdarzeń które występują do momentu lambda, który jest oczekiwaną liczbą zdarzeń w zadanym okresie czasu.

W argumencie podajemy parametr lambda.

Zwracana jest liczba naturalna.

W(liczba) – generator o rozkładzie wykładniczym. Polega na rzutowaniu liczb z przedziału [0,1] na przedział [0;+∞] za pomocą odwrotności funkcji dystrybuanty rozkładu wykładniczego.

W argumencie podajemy ilość liczb, które chcemy otrzymać.

Zwracana jest tablica liczb z przedziału [0;+∞]

N(liczba) – generator o rozkładzie normalnym. Stosując transformację Boxa-Mullera wyliczamy x'ową (bądź y'kową) z układu współrzędnych korzystając z jednego ze wzorów. Metoda ta dostarcza ciąg liczb o rozkładzie normalnym, wykorzystując dwie losowe wartości z rozkładu jednostajnego z przedziału [0,1], które **J** wygeneruje. Następnie otrzymane wartości możemy przeskalować mnożąc przez odchylenie standardowe, a następnie dodając wartość oczekiwaną otrzymując rozkład normalny N(μ; σ).

W argumencie podajemy ilość liczb, które chcemy otrzymać. Następnie odchylenie standardowe oraz wartość oczekiwaną.

Zwracana jest tablica liczb z otoczenia σ.

2) Test serii:

Test ten znajduje się w pliku tests.py pod funkcją **runs_test** przyjmując w argumencie ilość badanych elementów (liczb).

Wykonany test wyświetla wartość Z do własnego wglądu oraz wypisuje czy liczby, które generujemy używając **G** można uznać za losowe.

Opis działania:

Początkowo generujemy wybraną ilość liczb generatorem **G** (minimum 40).

Następnie określamy medianę z naszego ciągu liczb. Kolejnym krokiem jest zliczenie ilości przejść między kolejnymi liczbami ciągu zwracając uwagę na to czy są one mniejsze niż mediana, czy większe od niej. Dzięki wyliczonej liczbie przejść (serii) wyliczamy poprzez wyznaczenie średniej oraz wariancji liczbę Z ze standaryzowanego rozkładu normalnego. Ostatecznie przyjmując poziom ufności na poziomie 95% odcinamy 2,5% z lewej i prawej strony rozkładu normalnego sprawdzamy czy wartość Z zawiera się w nim. Jeśli tak, to test kończy się sukcesem.

3) Histogramy:

Funkcje do wywołania histogramu wybranego generatora znajdują się w pliku histograms.py. W pliku main.py podane zostały przykładowe wywołania wykresów wszystkich generatorów.

3. Eksperymenty

Eksperymenty polegające na przeprowadzeniu testu serii oraz wywołaniu wykresów wszystkich generatorów są możliwe do własnego wykonania uruchomiając plik main.py. Dodatkowo przykładowe histogramy wszystkich generatorów są dostępne w folderze plots.

4. Interpretacja wyników

Stworzone generatory z zadowalającym rezultatem spełniają kryteria losowości oraz własnych rozkładów przy prawie dowolnej ilości wylosowanych liczb. Jednakże atrakcyjność wyników przedstawiana na histogramach bardzo mocno zależy od wybranych parametrów generatora oraz funkcji rysującej wykres. Generator **G** zdając test serii gwarantuje, że pozostałe generatory również dostarczają losowe liczby.

Podsumowując generatory wykonują postawione im zadanie i działają w sposób losowy.

5. Źródła

https://pl.wikipedia.org/

generatory_16.pdf (agh.edu.pl)

Algorytmy i Struktury Danych - Liniowe generatory liczb pseudolosowych (eduinf.waw.pl)

Test serii – Wikipedia, wolna encyklopedia