|  |  |
| --- | --- |
| Nama Mahasiswa/NIM | 1. Helma Nuriyah (301220042) |
| Judul Tugas | Buatlah pemodelan dan simulasi menggunakan python (gunakan google colab). Studi kasus yang  digunakan bebas tentang RNG, dan setiap mahasiswa berbeda studikasusnya |
| Tahun | 2024 |

|  |  |
| --- | --- |
| SIMULASI PENGACAKAN NOMOR ANTRIAN RUMAH SAKIT DENGAN RNG | |
|  | **Deskripsi Studi Kasus** |
| Di rumah sakit, nomor antrian digunakan untuk mengatur urutan pasien. Untuk memastikan distribusi nomor antrian yang adil dan acak, algoritma Linear Congruential Generator (LCG) digunakan. Dengan parameter tertentu, LCG menghasilkan bilangan acak yang digunakan sebagai nomor antrian. Studi ini bertujuan untuk:   * Mengimplementasikan algoritma LCG untuk simulasi pengacakan nomor antrian. * Menampilkan hasil dalam bentuk tabel dan grafik untuk menganalisis distribusi nomor antrian. | |
|  | **Persamaan atau Rumus yang Digunakan** |
| 1. **Rumus dasar LCG:**      *  Zi​: Bilangan sebelumnya. * aaa: Konstanta pengali. * mmm: Modulus. * Z0Z\_0Z0​: Seed (bilangan awal).  1. **Bilangan acak seragam (UUU)**:     Nilai UiU\_iUi​ berada dalam rentang 0≤Ui<10 \leq U\_i < 10≤Ui​<1. | |
|  | **Pemodelan Menggunakan Python** |
| import matplotlib.pyplot as plt  # 1. PARAMETER  a = 5 # Konstanta pengali  m = 100 # Modulus (rentang nomor antrian 1 sampai 100)  Z0 = 42 # Seed (bilangan awal)  n = 20 # Jumlah nomor antrian yang akan dihasilkan  # 2. INISIALISASI DAFTAR  results = [] # Menyimpan hasil pengacakan  Z = Z0 # Inisialisasi bilangan awal  # 3. ALGORITMA LCG  for i in range(1, n + 1):  Z\_prev = Z # Menyimpan bilangan sebelumnya  Z = (a \* Z) % m # Menghitung nomor antrian selanjutnya  U = Z / m # Bilangan acak seragam  rumus = f"({a} \* {Z\_prev}) mod {m}" # Menyimpan rumus perhitungan  results.append([i, Z\_prev, rumus, Z, round(U, 3)])  # 4. OUTPUT TABEL  print(f"{'No. Antrian':<12} | {'Nomor Sebelumnya':<18} | {'Rumus RNG':<30} | {'Nomor Antrian':<15} | {'U (Acak Seragam)':<10}")  print("-" \* 120)  for row in results:  print(f"{row[0]:<12} | {row[1]:<18} | {row[2]:<30} | {row[3]:<15} | {row[4]:<10}")  # 5. PEMODELAN GRAFIK  nomor\_antrian = [row[3] for row in results] # Nomor antrian  U\_values = [row[4] for row in results] # Nilai U (bilangan acak seragam)  # Plot grafik  plt.figure(figsize=(10, 5))  plt.plot(range(1, n + 1), nomor\_antrian, marker='o', label="Nomor Antrian")  plt.bar(range(1, n + 1), U\_values, alpha=0.5, color='orange', label="U (Acak Seragam)")  plt.title("Simulasi Pengacakan Nomor Antrian Rumah Sakit")  plt.xlabel("Iterasi")  plt.ylabel("Nomor / Bilangan Acak")  plt.legend()  plt.grid()  plt.show() | |
|  | **Penjelasam Coding** |
| 1. **Parameter**:  * Parameter seperti aaa, mmm, dan Z0Z\_0Z0​ menentukan pola bilangan acak.  1. **Algoritma**:  * Rumus LCG digunakan untuk menghasilkan bilangan acak.  1. **Hasil dalam Tabel**:  * Nomor antrian, bilangan sebelumnya, rumus perhitungan, nomor baru, dan nilai UUU dicetak.  1. **Pemodelan Grafik**:  * **Nomor antrian** diplot sebagai garis. * **Nilai UUU** diplot sebagai histogram untuk melihat distribusinya. | |
|  | **Hasil Output** |
| **Analisis Hasil:**   * Pada **iterasi pertama**, nomor antrian dihitung dari rumus (5×42)mod  100(5 \times 42) \mod 100(5×42)mod100, yang menghasilkan 10 dengan U=0.1U = 0.1U=0.1. * Pada **iterasi kedua**, nomor antrian dihitung dengan rumus (5×10)mod  100(5 \times 10) \mod 100(5×10)mod100, menghasilkan 50 dan U=0.5U = 0.5U=0.5. * Pada **iterasi ketiga dan seterusnya**, perhitungan rumus yang sama menghasilkan nomor antrian yang tetap yaitu 50, karena setelah beberapa iterasi nilai dari ZZZ stabil pada 50, yang membuat semua iterasi berikutnya menghasilkan angka yang sama (50). Ini adalah sifat dari algoritma Linear Congruential Generator yang kadang menghasilkan nilai tetap setelah beberapa iterasi. | |
|  | **Referensi** |
| <https://www.youtube.com/watch?v=CgW4E4DRudk&pp=ygUYcGVuZ2d1bmFhbiBhbGdvcml0bWEgUk5H>  <https://www.youtube.com/watch?v=phag4UJEHVo&pp=ygUYcGVuZ2d1bmFhbiBhbGdvcml0bWEgUk5H>  <https://www.youtube.com/watch?v=nDv3yXdD0rk&pp=ygUuc2ltdWxhc2kgbGVtcGFyIGRhZHUgbWVuZ2d1bmFrYW4gYWxnb3JpdG1hIFJORw%3D%3D>  <https://www.youtube.com/watch?v=8z8QuttAy6s&pp=ygUuc2ltdWxhc2kgbGVtcGFyIGRhZHUgbWVuZ2d1bmFrYW4gYWxnb3JpdG1hIFJORw%3D%3D>  <https://www.youtube.com/watch?v=3mnvcG8os_s&list=PLovW1TU0Q47wLb9ltauKLDxJ3e3SrYyc_>  <https://towardsdatascience.com/markov-chain-analysis-and-simulation-using-python-4507cee0b06e> | |