Nombre: Juan Manuel Varela

Entrega Métodos de Monte Carlo

Unidad 1 – Sesión 02 - Ejercicio 2.1

Descripción del problema:

Supongamos que para construir una casa debemos efectuar la siguiente lista de tareas:

• T1 - cimientos - tiempo aleatorio uniforme entre 40 y 56 hs.

• T2 - contrapiso - tiempo aleatorio uniforme entre 24 y 32 hs.

• T3 - paredes - tiempo aleatorio uniforme entre 20 y 40 hs.

• T4 - techo - tiempo aleatorio uniforme entre 16 y 48 hs.

• T5 - instalación sanitaria - tiempo aleatorio uniforme entre 10 y 30 hs.

• T6 - instalación eléctrica - tiempo aleatorio uniforme entre 15 y 30 hs.

• T7 - cerramientos - tiempo aleatorio uniforme entre 20 y 25 hs.

• T8 - pintura - tiempo aleatorio uniforme entre 30 y 50 hs.

• T9 - revestimientos sanitarios - tiempo aleatorio uniforme entre 40 y 60 hs.

• T10 - limpieza final - tiempo aleatorio uniforme entre 8 y 16 hs.

Hay ciertas dependencias que implican que una tarea no puede comenzar hasta haberse terminado otra previa:

• T2, T3 dependen de 1.

• T4 depende de 2 y 3

• T5 depende de 2 y 3

• T6 depende de 3

• T7 depende de 3

• T8 depende de 4, 5, 6 y 7

• T9 depende de 5

• T10 depende de 7, 8 y 9

Tareas:

1. implementar un programa que reciba como parámetros de línea de comando (o pregunte en pantalla) la cantidad de replicaciones *n* a realizar, y emplee Monte Carlo para calcular (e imprimir) la estimación del tiempo total promedio desde que se comienza la obra hasta que se finaliza la misma, y la desviación estándar de este estimador.

2. Incluir código para calcular el tiempo de cálculo empleado por el programa.

3. Utilizar el programa con n = 10, 10^2, 10^3, 10^4, 10^5, 10^6, y mostrar en una tabla las estimaciones de media y desviación estándar, así como los tiempos de cálculo. Discutir estos

resultados. (en caso de que el tiempo de ejecución para 10^6 replicaciones sea menor que 60 segundos, agregar experimentos con mayor número de iteraciones, siempre multiplicando por 10, hasta que uno de los experimentos supere esa duración).

Solución a aplicar en Python:

1. Importar random #Esta biblioteca incluye funciones para generar números seudoaleatorios
2. Importar numpy #Esta biblioteca incluye funciones para calcular media, suma, raiz cuadrada
3. Importar time #Esta biblioteca incluye funciones que permiten calcular el tiempo de ejecución
4. Definir semilla
5. Definir función: tiempo_de_obra(): """Hace una simulación del tiempo total de construcción de una casa"""
   5.1. Tarea1=Uniforme entre 40 y 56 #Sorteo de la duración de cada tarea con una distribución uniforme
   5.2. Tarea2, Tarea3,…,Tarea10.
   5.3. Tarea1final=Tarea1 #Cálculo del tiempo de finalización de cada tarea en base a las dependencias
   5.4. Tarea2final, Tarea3final,…,Tarea10final
   5.5. Tiempo_total= máximo(Tarea1final,…,Tarea10final) #Cálculo del tiempo total de construcción como el tiempo en el que se finalizaron todas las tareas
   5.6. Salida: Tiempo_total
6. Definir función: montecarlo(n): """Ejecuta la función 'tiempo_de_obra' n veces y calcula la estimación del tiempo total promedio de obra y la desviación estándar del estimador"""
   6.1. Tiempos= array vacío #Inicialización
   6.2. For i entre 0 y n-1:
      6.2.1. Agregar a Tiempos: tiempo_de_obra()
   6.3. X=promedio(Tiempos)
   6.4. V=suma(Tiempos^2)/(n*(n-1))+X^2/(n-1) #Tiempos^2 eleva al cuadrado todos los elementos del array
   6.5. Desv=raíz(V)
   6.6. Salida: X, Desv

7. Escribir “Ingrese el tamaño de la muestra”
8. Leer n
9. Inicio = tiempo de inicio
10. X_hat, Desv_hat = montecarlo(n)
11. Tiempo_ejecución=Tiempo final-inicio #Cálculo del tiempo de ejecución
12. Escribir “El tiempo estimado de obra es “X_hat” Hs. con una desviación estándar de “Desv_hat” Hs.”
13. Escribir “Tiempo de ejecución: “Tiempo_ejecución” segundos”

Resultados:

Plataforma de cómputo: PC de escritorio, procesador Intel I5-12400, RAM: 32 Gb, Windows 10
Semilla: 6 para todos los tamaños de muestra

n	X_h (Hs.)	σ_h (Hs.)	t (s)
10	171,6694482	2,26555152	0,00100708
10^2	168,4240678	1,01897607	0,0
10^3	168,484499	0,32112910	0,002989053
10^4	168,578447	0,10266999	0,022921562
10^5	168,5583011	0,03254217	0,188367366
10^6	168,5523761	0,01028096	1,877717494
10^7	168,5696021	0,00325222	18,72835373
10^8	168,5669953	0,00102839	187,4122297

Se puede apreciar en la tabla que la media rápidamente (a partir de un tamaño de muestra de 100) se acerca a un valor determinado y oscila entorno a ese valor alejandose cada vez menos, este valor es la esperanza de la distribución real y según la simulación con 10^8 muestras, que es el valor más preciso que disponemos, es aproximadamente 168,567 Hs.

La desviación estándar, como es de esperar, se va reduciendo con el aumento del tamaño de la muestra. Esto se debe a que la varianza se estima como Var(X)/n, y la desviación estándar es la raíz de esto, entonces al multiplicar n por 10, la varianza se reduce 10 veces y la desviación estándar $\sqrt{10}$ veces.

El tiempo de ejecución aumenta linealmente con el tamaño de la muestra, por lo que se puede concluir que, a partir de cierto punto, al alcanzarse estimaciones para la media que varían en el orden de las centésimas de hora, con desviaciones estándar menores a 5 minutos, en un tiempo razonable, no vale la pena gastar recursos en correr la simulación con tamaños de muestra más grandes, a no ser que se quiera tener un gran nivel de precisión por alguna razón. Esto se puede ver en este caso a partir del tamaño de muestra de 10^5.

Anexos:

Log de ejecuciones:

```
runcell(0, 'G:/…/Ej 2.1.py')

Ingrese el tamaño de la muestra, n:
10
El tiempo estimado de obra es 171.66944823519938 Hs. con una desviación
estándar de 2.265551515999461Hs.
Tiempo de ejecución: 0.001007080078125 segundos

runcell(0, 'G:/…/Ej 2.1.py')

Ingrese el tamaño de la muestra, n:
100
El tiempo estimado de obra es 168.42406778721295 Hs. con una desviación
estándar de 1.0189760702634334Hs.
Tiempo de ejecución: 0.0 segundos

runcell(0, 'G:/…/Ej 2.1.py')

Ingrese el tamaño de la muestra, n:
1000
El tiempo estimado de obra es 168.48449902704579 Hs. con una desviación
estándar de 0.32112910040689663Hs.
Tiempo de ejecución: 0.002989053726196289 segundos

runcell(0, 'G:/…/Ej 2.1.py')

Ingrese el tamaño de la muestra, n:
10000
El tiempo estimado de obra es 168.57844698910634 Hs. con una desviación
estándar de 0.10266998891635286Hs.
Tiempo de ejecución: 0.02292156219482422 segundos

runcell(0, 'G:/…/Ej 2.1.py')

Ingrese el tamaño de la muestra, n:
100000
El tiempo estimado de obra es 168.55830106863485 Hs. con una desviación
estándar de 0.03254216619686187Hs.
Tiempo de ejecución: 0.18836736679077148 segundos

runcell(0, 'G:/…/Ej 2.1.py')

Ingrese el tamaño de la muestra, n:
1000000
El tiempo estimado de obra es 168.55237614378177 Hs. con una desviación
estándar de 0.01028095792029827Hs.
Tiempo de ejecución: 1.8777174949645996 segundos

runcell(0, 'G:/…/Ej 2.1.py')

Ingrese el tamaño de la muestra, n:
10000000
El tiempo estimado de obra es 168.56960208740992 Hs. con una desviación
estándar de 0.003252222721471177Hs.
Tiempo de ejecución: 18.72835373878479 segundos
```

```
runcell(0, 'G:/…/Ej 2.1.py')

Ingrese el tamaño de la muestra, n:
100000000
El tiempo estimado de obra es 168.5669953045704 Hs. con una desviación
estándar de 0.0010283970190714 36Hs.
Tiempo de ejecución: 187.41222977638245 segundos
```

Código fuente:

```
import random #Esta biblioteca incluye funciones para generar números
seudoaleatorios
import numpy as np #Esta biblioteca incluye funciones para calcular media,
suma, raiz cuadrada
import time #Esta biblioteca incluye funciones que permiten calcular el tiempo
de ejecución

random.seed(6) #Inicialización de la secuencia de números seudoaleatorios

def tiempo_de_obra():
    """Hace una simulación del tiempo total de construcción de una casa"""
    t1=random.uniform(40,56) #Sorteo de la duración de cada tarea
    t2=random.uniform(24,32)
    t3=random.uniform(20,40)
    t4=random.uniform(16,48)
    t5=random.uniform(10,30)
    t6=random.uniform(15,30)
    t7=random.uniform(20,25)
    t8=random.uniform(30,50)
    t9=random.uniform(40,60)
    t10=random.uniform(8,16)

    t1f=t1 #Cálculo del tiempo de finalización de cada tarea en base a las
dependencias
    t2f=t1f+t2
    t3f=t1f+t3
    t4f=max(t2f,t3f)+t4
    t5f=max(t2f,t3f)+t5
    t6f=t3f+t6
    t7f=t3f+t7
    t8f=max(t4f,t5f,t6f,t7f)+t8
    t9f=t5f+t9
    t10f=max(t7f,t8f,t9f)+t10
    #Cálculo del tiempo total de construcción como el tiempo en el que se
finalizaron todas las tareas
    t_tot=max(t1f,t2f,t3f,t4f,t5f,t6f,t7f,t8f,t9f,t10f)
    return t_tot

def montecarlo(n):
    """Ejecuta la función 'tiempo_de_obra' n veces y calcula la estimación del
tiempo total promedio de obra y la desviación estándar del estimador"""
    tiempos=[]
    for i in range(n):
        tiempos.append(tiempo_de_obra())
    tiempos_arr=np.array(tiempos)
    X_est=np.mean(tiempos_arr)
    V_est=np.sum(tiempos_arr**2)/(n*(n-1))-X_est**2/(n-1)
    Desv_est=np.sqrt(V_est)
```

```
    return X_est, Desv_est

entrada=input('Ingrese el tamaño de la muestra, n:\n')

inicio=time.time()
X_h, Desv_h = montecarlo(int(entrada))
tiempo_ej=time.time()-inicio #Cálculo del tiempo de ejecución
print('El tiempo estimado de obra es '+str(X_h)+' Hs. con una desviación
estándar de '+str(Desv_h)+'Hs.')
print('Tiempo de ejecución: '+str(tiempo_ej)+' segundos')
```