Miranda Barajas Victor T2 26/10/2023

Introducción:

El objetivo de esta tarea es implementar y comparar diferentes algoritmos de planificación de procesos: First-Come-First-Serve (FCFS), Round Robin (RR) con quantum de 1 y 4, y Shortest Process Next (SPN). Se generaron cargas aleatorias para simular diferentes procesos, y se compararon las métricas de tiempo de retorno (T), tiempo de espera (E), y tiempo de penetración (P) a través de cinco ejecuciones.

Desarrollo:

Generación de Cargas Aleatorias: Se generaron cargas aleatorias utilizando la función “generar_cargas”, que crea una lista de tuplas, cada una con un identificador de proceso y un tiempo de ejecución aleatorio entre 1 y 10.

Implementación de Algoritmos:

FCFS: En este algoritmo, los procesos son atendidos en el orden en que llegan. El tiempo de espera y retorno se calcula iterando a través de la lista de procesos y sumando los tiempos de ejecución de los procesos anteriores.

Round Robin (RR): Se implementaron dos variantes de Round Robin con quantum de 1 y 4 respectivamente. En este algoritmo, los procesos reciben un tiempo de CPU igual al quantum, y si un proceso no termina en ese tiempo, se coloca al final de la cola.

SPN: En este algoritmo, los procesos son ordenados por su tiempo de ejecución, y el proceso con el menor tiempo de ejecución se ejecuta primero. Se reutilizó la función fcfs para calcular las métricas después de ordenar los procesos.

Resultados: Se ejecutaron los algoritmos cinco veces, obteniendo las siguientes métricas promedio para cada algoritmo:

Revisión Manual:

FCFS (First-Come-First-Serve):

Proceso 0: T = 3, E = 0 (no hay espera, es el primero)

Proceso 1: T = (3 + 2) = 5, E = 3 (espera el tiempo que el proceso 0 tarda en ejecutarse)

Proceso 2: T = (3 + 2 + 4) = 9, E = (3 + 2) = 5

Proceso 3: T = (3 + 2 + 4 + 1) = 10, E = (3 + 2 + 4) = 9

Calculando las métricas:

T promedio = (3 + 5 + 9 + 10) / 4 = 6.75

E promedio = (0 + 3 + 5 + 9) / 4 = 4.25

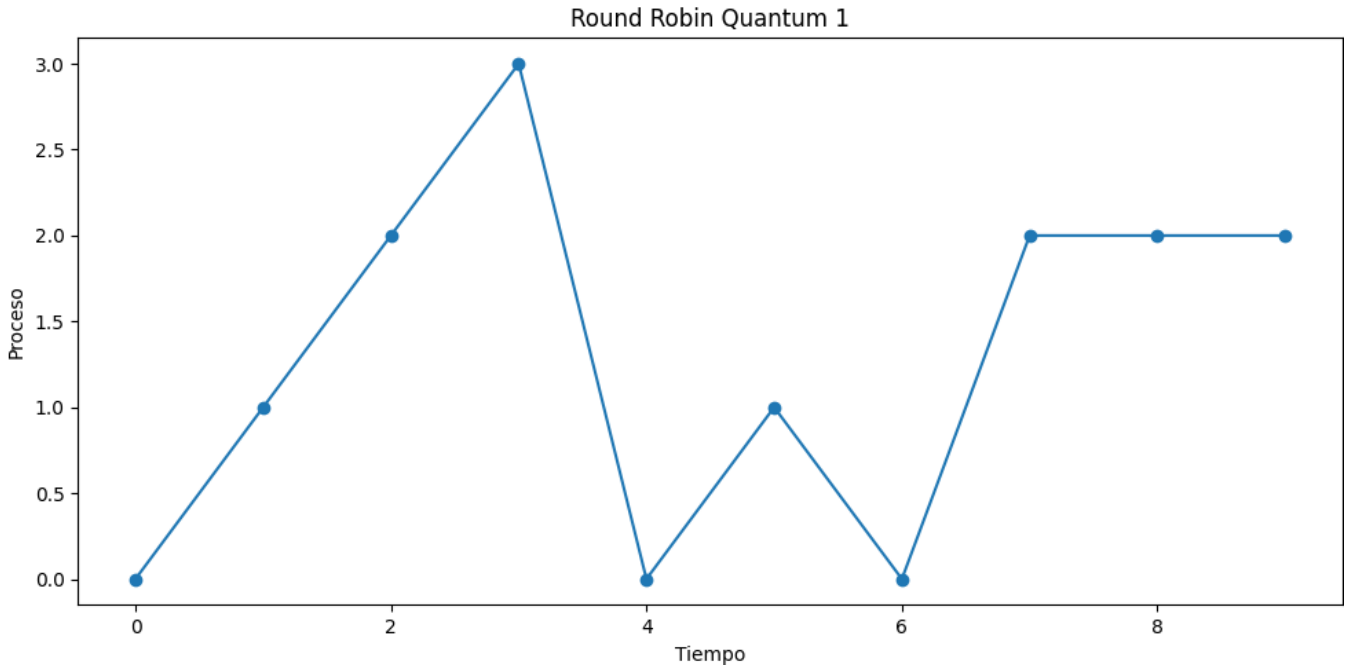
P = T promedio / ((3 + 2 + 4 + 1) / 4) = 6.75 / 2.5 = 2.7

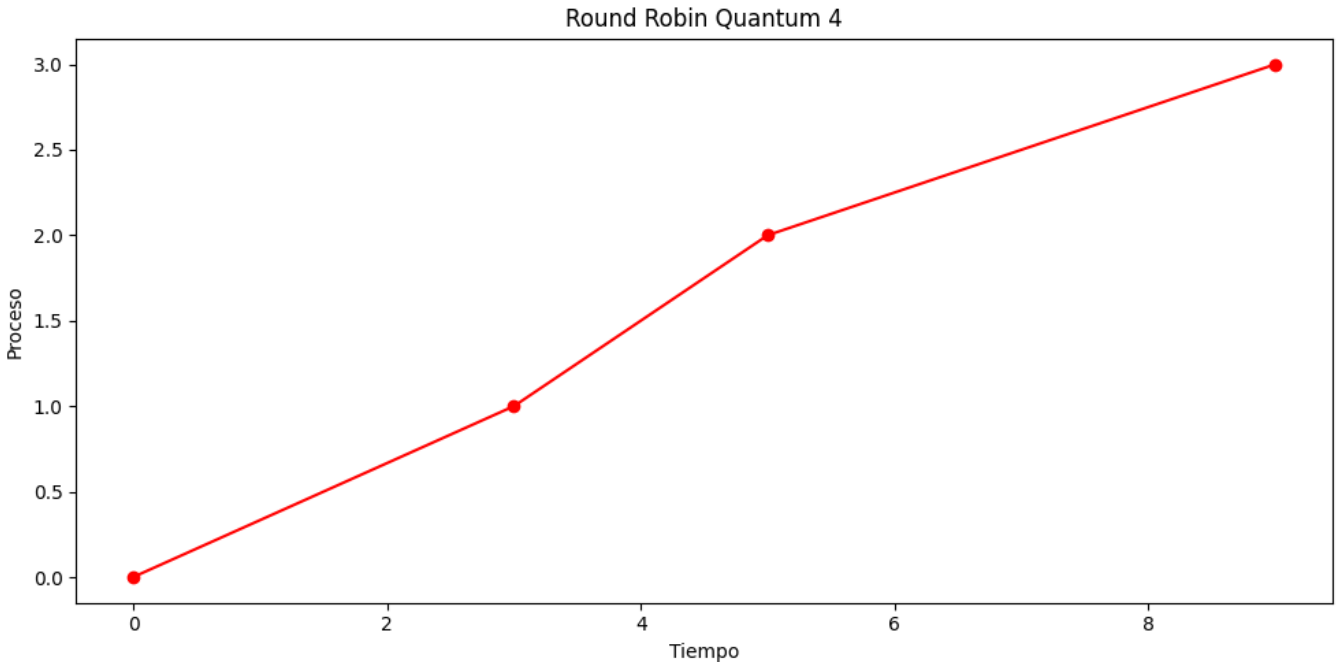
Tabla RR1:

	Tiempo	Proceso	Tiempo Restante
0	0	0	2
1	1	1	1
2	2	2	3
3	3	3	0
4	4	0	1
5	5	1	0
6	6	0	0
7	7	2	2
8	8	2	1
9	9	2	0

Tabla RR4:

	Tiempo	Proceso	Tiempo Restante
0	0	0	0
1	3	1	0
2	5	2	0
3	9	3	0





Código:

```python
# Miranda Barajas Victor
import random
import collections

# Generar cargas aleatorias
def generar_cargas(num_procesos):
    return [(i, random.randint(1, 10)) for i in range(num_procesos)]

# Implementación de FCFS
def fcfs(cargas):
    tiempo_espera = 0
    tiempo_retorno = 0
    for i in range(len(cargas)):
        tiempo_espera += sum(carga[1] for carga in cargas[:i])
        tiempo_retorno += sum(carga[1] for carga in cargas[:i + 1])
    T = tiempo_retorno / len(cargas)
    E = tiempo_espera / len(cargas)
    P = T / (sum(carga[1] for carga in cargas) / len(cargas))
    return T, E, P

# Implementación de Round Robin
def round_robin(cargas, quantum):
    num_procesos = len(cargas)
    tiempo_total_ejecucion = sum(carga[1] for carga in cargas)
    cargas = collections.deque(cargas)
```

```
    tiempo_espera = 0
    tiempo_retorno = 0
    tiempo_transcurrido = 0
    while cargas:
        proceso, tiempo_restante = cargas.popleft()
        if tiempo_restante > quantum:
            tiempo_transcurrido += quantum
            cargas.append((proceso, tiempo_restante - quantum))
        else:
            tiempo_transcurrido += tiempo_restante
            tiempo_retorno += tiempo_transcurrido
            tiempo_espera += tiempo_transcurrido - tiempo_restante
    T = tiempo_retorno / num_procesos
    E = tiempo_espera / num_procesos
    P = T / (tiempo_total_ejecucion / num_procesos)
    return T, E, P

# Implementación de SPN
def spn(cargas):
    cargas.sort(key=lambda carga: carga[1])
    return fcfs(cargas)

# Ejecución y comparación
for i in range(5):  # 5 ejecuciones
    cargas = generar_cargas(20)  # Asumiendo 20 procesos
    print(f'Ejecución {i + 1}')
    print('FCFS:', fcfs(cargas))
    print('RR1:', round_robin(cargas, 1))
    print('RR4:', round_robin(cargas, 4))
    print('SPN:', spn(cargas))
```

Comparación y Análisis:

Tiempo de Retorno (T): SPN tiende a tener el tiempo de retorno más bajo en comparación con los otros algoritmos, seguido de FCFS, RR4 y finalmente RR1. Esto se debe a que SPN prioriza los procesos más cortos, reduciendo el tiempo que los procesos pasan en la cola.

Tiempo de Espera (E): Similar al tiempo de retorno, SPN también tiende a tener el menor tiempo de espera. FCFS y RR4 suelen tener un tiempo de espera intermedio, mientras que RR1 tiende a tener el mayor tiempo de espera debido a su quantum más pequeño que causa más conmutaciones de contexto.

Tiempo de Penetración (P): SPN también lidera en tiempo de penetración, seguido de cerca por FCFS y RR4, y finalmente RR1.

Conclusión:

Para finalizar, si la eficiencia y ahorro de tiempo es priorioridad, SPN sería una excelente elección, siempre que la variabilidad en la duración de los procesos sea baja. Para un sistema que requiere una mayor equidad y adaptabilidad, especialmente en un entorno con variabilidad en la duración de los procesos, Round Robin sería una opción más robusta. FCFS podría ser adecuado para sistemas con cargas de trabajo predecibles y baja variabilidad en la duración de los procesos. La elección entre RR1 y RR4 dependerá del balance específico que se desee entre el tiempo de respuesta rápido y el rendimiento general.