Case1 ve Case2 Başarım Analizi Raporu

1. GİRİŞ

Bu çalışmada, işletim sistemlerinde yaygın olarak kullanılan CPU zamanlama algoritmaları Java dili ile simüle edilmiştir. Çalışmanın amacı, farklı zamanlama yöntemlerinin iki farklı işlem kümesi (Case1 ve Case2) üzerindeki davranışlarını incelemek ve performans ölçütleri açısından karşılaştırmaktır.

İncelenen algoritmalar:

- First Come First Served (FCFS)
- Shortest Job First (SJF – Preemptive)
- Shortest Job First (SJF – Non-Preemptive)
- Round Robin (Quantum = 2)
- Priority Scheduling (Preemptive)
- Priority Scheduling (Non-Preemptive)

2. CASE1 ANALİZİ

2.1 Case1 Zaman Tabloları

Case1 için her algoritmanın oluşturduğu zaman tabloları, program çıktısı olarak aşağıdaki dosyalarda ayrı ayrı sunulmuştur:

- FCFS_case1.txt
- SJF_Preemptive_case1.txt
- SJF_NonPreemptive_case1.txt
- RR_case1.txt
- Priority_Preemptive_case1.txt
- Priority_NonPreemptive_case1.txt

Bu dosyalarda her algoritmanın CPU'yu hangi zaman aralıklarında hangi işlemlere tahsis ettiği detaylı olarak gösterilmektedir.

2.2 Case1 Başarım Tablosu

Algoritma	Ortalama Bekleme	Maks. Bekleme	Ortalama Turnaround	Maks. Turnaround	CPU Kullanımı (%)	Context Switch
FCFS	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
SJF (P)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
SJF (NP)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
RR (q=2)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
Priority (P)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
Priority (NP)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı

2.3 Case1 Yorum

Case1 sonuçlarına bakıldığında, **SJF Preemptive** algoritmasının ortalama bekleme süresi açısından en başarılı yöntem olduğu görülmüştür. Bunun nedeni, CPU'nun her zaman en kısa kalan süreye sahip işleme tahsis edilmesidir.

FCFS algoritması ise işlemleri yalnızca geliş zamanına göre sıraladığı için uzun burst time'a sahip işlemler, sonraki işlemlerin bekleme süresini artırmıştır.

Round Robin algoritması, adil CPU paylaşımı sağlamasına rağmen context switch sayısının yüksek olması nedeniyle CPU kullanım oranını düşürmüştür.

Priority algoritmaları, yüksek öncelikli işlemler için avantaj sağlarken düşük öncelikli işlemlerin bekleme sürelerini artırmıştır.

3. CASE2 ANALİZİ

3.1 Case2 Zaman Tabloları

Case2 için zaman tabloları aşağıdaki dosyalarda ayrı ayrı sunulmuştur:

- FCFS_case2.txt
- SJF_Preemptive_case2.txt
- SJF_NonPreemptive_case2.txt
- RR_case2.txt
- Priority_Preemptive_case2.txt
- Priority_NonPreemptive_case2.txt

Bu tablolar, Case2'deki işlem kümesinin Case1'e göre farklı bir zamanlama davranışı sergilediğini göstermektedir.

3.2 Case2 Başarım Tablosu

Algoritma	Ortalama Bekleme	Maks. Bekleme	Ortalama Turnaround	Maks. Turnaround	CPU Kullanımı (%)	Context Switch
FCFS	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
SJF (P)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
SJF (NP)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
RR (q=2)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
Priority (P)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı
Priority (NP)	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı	Program Çıktısı

3.3 Case2 Yorum

Case2 senaryosunda, işlem geliş zamanları ve burst sürelerinin farklılığı nedeniyle algoritmaların performansları değişiklik göstermiştir.

Özellikle Round Robin algoritmasının adil paylaşım özelliği, Case2'de daha dengeli bir bekleme süresi dağılımı sağlamıştır. Ancak context switch sayısının fazla olması yine CPU verimliliğini düşürmüştür.

Priority algoritmalarında, düşük öncelikli işlemlerin starvation problemine daha yatkın olduğu gözlemlenmiştir.

4. GENEL KARŞILAŞTIRMA VE DEĞERLENDİRME

Ölçüt	En Başarılı Algoritma	Açıklama
Ortalama Bekleme	SJF (Preemptive)	Kısa işler önceliklendirildi
Turnaround Süresi	SJF (Preemptive)	İşlerin erken tamamlanması
Adillik	Round Robin	Tüm işlemlere eşit zaman
Kritik İşler	Priority	Yüksek öncelik avantajı
Düşük Context Switch	FCFS	Kesintisiz çalışma

5. SONUÇ

Bu çalışmada, CPU scheduling algoritmalarının farklı senaryolarda farklı performans sergilediği görülmüştür. Tek bir algoritmanın her durum için en iyi çözüm olmadığı, sistem gereksinimlerine göre uygun algoritmanın seçilmesi gerektiği sonucuna varılmıştır.

6. EKLER

- Tüm zaman tabloları ilgili .txt dosyalarında sunulmuştur.
- Program Java dili ile geliştirilmiştir.