SPRAWOZDANIE
ALGORYTMY PRZYDZIAŁU CZASU PROCESORA

Repozytorium: https://github.com/MarArek/SOSimulations.git Branch: Scheduling

Sprawozdanie dotyczy symulacji algorytmów przydziału czasu procesora. W projekcie uwzględniono algorytmy FCFS (First-Come First-Served), LCFS (Last-Come First-Served), SJF (Shortest Job First), oraz Round-Robin (w wersji FCFS i LCFS).

Dokumentacja kodu została przygotowana za pomocą narzędzia **JAVADOC[1].**

Spis treści

Autor: Arkadiusz Maruszczak
Wydział: Elektroniki
Kierunek: Cyberbezpieczeństwo

[1] Strona narzędzia: https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html

1. Procedura testowania algorytmów.

Ogólne informacje

Symulacje opierają się na manipulowaniu „procesami” w pamięci procesora. Rolę kolejki procesów oczekujących pełni tablica o wielkości równej zadeklarowanej wcześniej ilości procesów (na przykład **100**).

Każdy proces jest zbudowany z:

- **id** – identyfikatora procesu
- **awaitingTime** – czasu oczekiwania procesu
- **processingTime** – czasu przetwarzania procesu
- **burstTime** – pozostały czas obliczeń procesu

Symulacje rozpoczynają się wygenerowaniem plików źródłowych w postaci dwóch kolumn, w których znajdują się wygenerowane numery procesów w zadanej wcześniej ilości, wraz z losowo wygenerowanym czasem przetwarzania z zadanego zakresu. Ilość plików źródłowych jest równa zadeklarowanej wcześniej ilości prób.

Przykład wygenerowanego pliku:

processes1.csv

id	burstTime
0	2
1	2
2	9
3	16
4	16
5	4
6	15
7	1
8	1

FCFS

Informacja

Algorytm polega na przydzielaniu czasu w kolejności, w której procesy pojawiły się w procesorze. Procesy są przetwarzane, a ich czas wykonywania jest dodawany do czasu oczekiwania każdego procesu, który jeszcze się nie wykonał.

Implementacja:

W pierwszym kroku lista pełniąca rolę kolejki oczekiwania zostaje wypełniona danymi z wygenerowanego pliku. Długość tej listy pełni rolę wyznacznika długości głównej pętli algorytmu:

```
czasOczekiwania := 0
czySkonczono := false
indeks := 0
while !czySkonczono do
        czasOczekiwania += kolejkaOczekiwania[indeks].czasObliczeń
        for i = indeks+1 to iloscProcesow do
                kolejkaOczekiwania[i].czasCzekania = czasOczekiwania
        endfor
        kolejkaOczekiwania[indeks].czasPrzetwarzania        =
                kolejkaOczekiwania[indeks].czasCzekania + kolejkaOczekiwania[indeks].czasObliczen

        kolejkaGotowychProcesow.dodaj( kolejkaOczekiwania[indeks] )

        indeks++

        if kolejkaGotowychProcesow.rozmiar == iloscProcesow do
                czySkonczono := true
        endif
endwhile
```

Po przejściu pętli, średnia wszystkich **czasów oczekiwania** jest szukanym przez nas wynikiem. W przypadku większej ilości prób **(n>1)**, wynik ostateczny jest średnią arytmetyczną wyników pośrednich. Pętla algorytmu jest wtedy zamknięta w pętli o długości równej ilości prób.

LCFS

Informacja

Algorytm polega na przydzielaniu czasu w odwrotnej kolejności niż, w której procesy pojawiły się w procesorze. Procesy są przetwarzane, a ich czas wykonywania jest dodawany do czasu oczekiwania każdego procesu, który jeszcze się nie wykonał.

Implementacja:

W pierwszym kroku lista pełniąca rolę kolejki oczekiwania zostaje wypełniona danymi z wygenerowanego pliku. Długość tej listy pełni rolę wyznacznika długości głównej pętli algorytmu:

```
czasOczekiwania := 0
czySkonczono := false
indeks := iloscProcesow - 1
while !czySkonczono do
        czasOczekiwania += kolejkaOczekiwania[indeks]. czasObliczeń
        for i = indeks-1 to 0 do
                kolejkaOczekiwania[i].czasCzekania = czasOczekiwania
        endfor
        kolejkaOczekiwania[indeks].czasPrzetwarzania        =
                kolejkaOczekiwania[indeks].czasCzekania + kolejkaOczekiwania[indeks]. czasObliczen

        kolejkaGotowychProcesow.dodaj( kolejkaOczekiwania[indeks] )

        indeks--

        if kolejkaGotowychProcesow.rozmiar == iloscProcesow do
                czySkonczono := true
        endif
endwhile
```

Po przejściu pętli, średnia wszystkich **czasów oczekiwania** jest szukanym przez nas wynikiem. W przypadku większej ilości prób **(n>1)**, wynik ostateczny jest średnią arytmetyczną wyników pośrednich. Pętla algorytmu jest wtedy zamknięta w pętli o długości równej ilości prób.

SJF

Informacja

Algorytm polega na przydzielaniu czasu w kolejności od procesu z najkrótszym czasem przetwarzania, do ostatniego. Procesy są przetwarzane, a ich czas wykonywania jest dodawany do czasu oczekiwania każdego procesu, który jeszcze się nie wykonał.

Implementacja:

W pierwszym kroku lista pełniąca rolę kolejki oczekiwania zostaje wypełniona danymi z wygenerowanego pliku. Lista ta zostaje posortowana rosnąco względem czasu obliczeń. Długość tej listy pełni rolę wyznacznika długości głównej pętli algorytmu:

```
czasOczekiwania := 0
czySkonczono := false
indeks := 0
while !czySkonczono do
        czasOczekiwania += kolejkaOczekiwania[indeks]. czasObliczeń
        for i = indeks+1 to iloscProcesow do
                kolejkaOczekiwania[i].czasCzekania = czasOczekiwania
        endfor
        kolejkaOczekiwania[indeks].czasPrzetwarzania        =
                kolejkaOczekiwania[indeks].czasCzekania + kolejkaOczekiwania[indeks].czasObliczen

        kolejkaGotowychProcesow.dodaj( kolejkaOczekiwania[indeks] )

        indeks--

        if kolejkaGotowychProcesow.rozmiar == iloscProcesow do
                czySkonczono := true
        endif
endwhile
```

Po przejściu pętli, średnia wszystkich **czasów oczekiwania** jest szukanym przez nas wynikiem. W przypadku większej ilości prób **(n>1)**, wynik ostateczny jest średnią arytmetyczną wyników pośrednich. Pętla algorytmu jest wtedy zamknięta w pętli o długości równej ilości prób.

Round-Robin FCFS

Informacja

Algorytm polega na przydzielaniu czasu w kolejności, w której procesy pojawiły się w procesorze, ale przetwarzane są tylko w zadanym okresie czasu (kwancie) do momentu, aż wszystkie procesy zostaną wykonane. Procesy są przetwarzane, a ich czas wykonywania jest dodawany do czasu oczekiwania każdego procesu, który jeszcze się nie wykonał.

Implementacja:

W pierwszym kroku lista pełniąca rolę kolejki oczekiwania zostaje wypełniona danymi z wygenerowanego pliku. Długość tej listy pełni rolę wyznacznika długości głównej pętli algorytmu:

```
czySkonczono := false
indeks := 0
while !czySkonczono do
        czasPrzetwarzania := kwantCzasu
        while kolejkaOczekiwania[indeks].czasObliczeń – czasPrzetwarzania < 0 do
                czasPrzetwarzania--
        endwhile
        for i = 0 to iloscProcesow do
                if i==indeks && kolejkaOczekiwania[i]. czasObliczeń > 0 do
                        kolejkaOczekiwania[i].czasPrzetwarzania =
                                kolejkaOczekiwania[i].czasPrzetwarzania + czasPrzetwarzania
                        kolejkaOczekiwania[i]. czasObliczen =
                                kolejkaOczekiwania[i].czasObliczen – czasPrzetwarzania
                        if kolejkaOczekiwania[i]. czasObliczen == 0 && !( ukonczono(kolejkaOczekiwania[i] ) do
                                kolejkaOczekiwania[i].czasPrzetwarzania =
                                        kolejkaOczekiwania[i].czasPrzetwarzania+kolejkaOczekiwania[i].czasCzekania
                                kolejkaGotowychProcesow.dodaj( kolejkaOczekiwania[i] )
                        endif
                else if i != indeks && !ukończono( kolejkaOczekiwania[i] ) do
                        kolejkaOczekiwania[i].czasCzekania =
                                kolejkaOczekiwania[i].czasCzekania + czasPrzetwarzania
                endif
        endfor

        indeks++

        if kolejkaGotowychProcesow.rozmiar == iloscProcesow do
                czySkonczono := true
        else if indeks == iloscProcesow do
                indeks := 0
        endif
endwhile
```

Po przejściu pętli, średnia wszystkich **czasów oczekiwania** jest szukanym przez nas wynikiem. W przypadku większej ilości prób **(n>1)**, wynik ostateczny jest średnią arytmetyczną wyników pośrednich. Pętla algorytmu jest wtedy zamknięta w pętli o długości równej ilości prób.

Round-Robin LCFS

Informacja

Algorytm polega na przydzielaniu czasu w odwrotnej kolejności niż, w której procesy pojawiły się w procesorze, ale przetwarzane są tylko w zadanym okresie czasu (kwancie) do momentu, aż wszystkie procesy zostaną wykonane. Procesy są przetwarzane, a ich czas wykonywania jest dodawany do czasu oczekiwania każdego procesu, który jeszcze się nie wykonał.

Implementacja:

W pierwszym kroku lista pełniąca rolę kolejki oczekiwania zostaje wypełniona danymi z wygenerowanego pliku. Długość tej listy pełni rolę wyznacznika długości głównej pętli algorytmu:

```
czySkonczono := false
indeks := iloscProcesow -1
while !czySkonczono do
        czasPrzetwarzania := kwantCzasu
        while kolejkaOczekiwania[indeks]. czasObliczeń – czasPrzetwarzania < 0 do
                czasPrzetwarzania--
        endwhile
        for i = iloscProcesow -1 to 0 do
                if i==indeks && kolejkaOczekiwania[i]. czasObliczeń > 0 do
                        kolejkaOczekiwania[i].czasPrzetwarzania =
                                kolejkaOczekiwania[i].czasPrzetwarzania + czasPrzetwarzania
                        kolejkaOczekiwania[i]. czasObliczen =
                                kolejkaOczekiwania[i].czasObliczen – czasPrzetwarzania
                        if kolejkaOczekiwania[i]. czasObliczen == 0 && !( ukonczono(kolejkaOczekiwania[i] ) do
                                kolejkaOczekiwania[i].czasPrzetwarzania =
                                        kolejkaOczekiwania[i].czasPrzetwarzania+kolejkaOczekiwania[i].czasCzekania
                                kolejkaGotowychProcesow.dodaj( kolejkaOczekiwania[i] )
                        endif
                else if i != indeks && !ukończono( kolejkaOczekiwania[i] ) do
                        kolejkaOczekiwania[i].czasCzekania =
                                kolejkaOczekiwania[i].czasCzekania + czasPrzetwarzania
                endif
        endfor

        indeks--

        if kolejkaGotowychProcesow.rozmiar == iloscProcesow do
                czySkonczono := true
        else if indeks < 0 do
                indeks := iloscProcesow - 1
        endif
endwhile
```

Po przejściu pętli, średnia wszystkich **czasów oczekiwania** jest szukanym przez nas wynikiem. W przypadku większej ilości prób **(n>1)**, wynik ostateczny jest średnią arytmetyczną wyników pośrednich. Pętla algorytmu jest wtedy zamknięta w pętli o długości równej ilości prób.

2. Opracowane wyniki eksperymentów

Symulacja generuje plik wynikowy w formacie csv. Symulację przeprowadzono dla danych:

- Ilość procesów: 100
- Kwanty czasu procesora: 0.5, 1, 1.5 (dodatkowo dodano kwant = 50)
- Ilość prób: 100
- Zakres czasu obliczeń: 1-20

Z wygenerowanego pliku csv zostaje stworzona tabela poniżej:

Time Quantum	Algorithm	Amount of processes	Average processing time [s]	Average waiting time [s]
0.5	FCFS	100	519,793	530,313
0.5	LCFS	100	521,697	532,217
0.5	SJF	100	356,407	366,927
0.5	RoundRobin FCFS	100	382,792	393,312
0.5	RoundRobin LCFS	100	382,994	393,514
1.0	FCFS	100	519,793	530,313
1.0	LCFS	100	521,697	532,217
1.0	SJF	100	356,407	366,927
1.0	RoundRobin FCFS	100	686,718	697,238
1.0	RoundRobin LCFS	100	686,920	697,440
1.5	FCFS	100	519,793	530,313
1.5	LCFS	100	521,697	532,217
1.5	SJF	100	356,407	366,927
1.5	RoundRobin FCFS	100	625,712	636,232
1.5	RoundRobin LCFS	100	626,110	636,630
50.0	FCFS	100	519,793	530,313
50.0	LCFS	100	521,697	532,217
50.0	SJF	100	356,407	366,927
50.0	RoundRobin FCFS	100	519,793	530,313
50.0	RoundRobin LCFS	100	521,697	532,217

Tabela wynikowa zawiera: **wartość kwantu czasu, nazwę algorytmu**, **ilość procesów w pamięci**, **średni czas przetwarzania procesu, średni czas oczekiwania procesu.**

3. Wnioski

Wnioski dla przeanalizowanych algorytmów są następujące:

1. Niezależnie od długości trwania kwantu czasu, algorytm SJF wypada najkorzystniej w porównaniu z innymi zasymulowanymi algorytmami.

2. W przypadku kwantu czasu równego 0.5 [s], czasy dla algorytmów Round-Robin są wyraźnie mniejsze, niż w porównaniu do większych wartości tej zmiennej.

3. Nie biorąc pod uwagę obliczeń dla kwantu 0.5 [s] to można zauważyć iż czasy dla algorytmów Round-Robin są funkcją malejącą. Przy kwancie dążącym do nieskończoności, czasy algorytmu Robin-Robin FCFS dążą do czasów algorytmu FCFS i analogicznie dla LCFS. Zauważyć to można dla kwantu równego 50. Jest on większy, niż maksymalny czas obliczeń dla procesu, więc czasy są identyczne dla algorytmów Round-Robin, FCFS i LCFS.

4. Symulacja została uruchomiona dla 100 powtórzeń, także ostateczny wynik jest średnią z nich wszystkich. Jest to duża liczba prób, więc można przyjąć, że wyniki z tabeli dobrze oddają poszczególne algorytmy dla danych procesów i ich czasów obliczeń z zakresu **1-20**.