SPRAWOZDANIE
ALGORYTMY ZASTĘPOWANIA STRON

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

Sprawozdanie dotyczy symulacji algorytmów zastępowania stron. W projekcie uwzględniono algorytmy LRU (Least Recently Used), oraz LFU (Least Frequently Used).

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 „ramkami” w pamięci. Rolę pamięci pełni tablica ramek o wielkości zadeklarowanej wcześniej (na przykład **3**).

Każda ramka jest zbudowana z:

- **id** – identyfikatora ramki/strony
- **ageOfFrame –** „wieku” ramki
- **amountOfUses –** ilości użyć danej ramki.

Symulacje rozpoczynają się wygenerowaniem plików źródłowych w postaci jednej kolumny, w której znajdują się losowo wygenerowane numery stron w zadanej wcześniej ilości z zadanego wcześniej zakresu (w tym przypadku **1-20**). Ilość plików źródłowych jest równa zadeklarowanej wcześniej ilości prób.

Przykład wygenerowanego pliku:

pages1.csv

id
20
8
4
14
19
15
13
3
1

LFU

Informacja

Algorytm polega na zastępowaniu strony, do której ramki było najmniej odwołań. Każde zastąpienie strony której ramka nie istniała w pamięci jest liczone jako brakująca strona. Licznik użycia strony jest zerowany po każdym usunięciu strony z pamięci fizycznej.

Implementacja:

W następnym kroku lista pełniąca rolę tablicy odniesień stron zostaje wypełniona danymi z wygenerowanego pliku. Długość tej listy pełni rolę wyznacznika długości głównej pętli algorytmu:

```
iloscBrakujacychStron := 0
indeksTablicyRamek := 0
for i = 0 to dlugoscTablicyOdniesien do
        if tablicaOdniesien[i] nie istnieje w tablicaRamek do
                iloscBrakujacychStron++
                indeksTablicyRamek := pobierzIndeksNajmniejUzywanejRamki()
                tablicaRamek[indeksTablicyRamek] := tablicaOdniesien[i]
                tablicaRamek[indeksTablicyRamek].wiek := 1
                tablicaRamek[indeksTablicyRamek].iloscUzyc := 1
                indeksTablicyRamek++
                podniesWiekInnychRamekWTablicyOJeden()
        else
                indeksTablicyRamek := pobierzIndeksPodanejRamki( tablicaOdniesien[i] )
                tablicaRamek[indeksTablicyRamek].iloscUzyc++
                tablicaRamek[indeksTablicyRamek].wiek := 1
                indeksTablicyRamek++
                podniesWiekInnychRamekWTablicyOJeden()
        endif
        if indeksTablicyRamek == wielkoscTablicyRamek do
                indeksTablicyRamek := 0
        endif
endfor
```

Po przejściu pętli, ilość brakujących stron jest oczekiwanym 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.

LRU

Informacja:

Algorytm polega na zastępowaniu strony, do której ramki odwołanie było najdawniej. Każde zastąpienie strony której ramka nie istniała w pamięci jest liczone jako brakująca strona.

Implementacja:

W następnym kroku lista pełniąca rolę tablicy odniesień stron zostaje wypełniona danymi z wygenerowanego pliku. Długość tej listy pełni rolę wyznacznika długości głównej pętli algorytmu:

```
iloscBrakujacychStron := 0
indeksTablicyRamek := 0
for i = 0 to dlugoscTablicyOdniesien do
        if tablicaOdniesien[i] nie istnieje w tablicaRamek do
                iloscBrakujacychStron++
                indeksTablicyRamek := pobierzIndeksNajstarszejRamki()
                tablicaRamek[indeksTablicyRamek] := tablicaOdniesien[i]
                tablicaRamek[indeksTablicyRamek].wiek := 1
                indeksTablicyRamek++
                podniesWiekInnychRamekWTablicyOJeden()
        else
                indeksTablicyRamek := pobierzIndeksPodanejRamki( tablicaOdniesien[i] )
                tablicaRamek[indeksTablicyRamek].wiek := 1
                indeksTablicyRamek++
                podniesWiekInnychRamekWTablicyOJeden()
        endif
        if indeksTablicyRamek == wielkoscTablicyRamek do
                indeksTablicyRamek := 0
        endif
endfor
```

Po przejściu pętli, ilość brakujących stron jest oczekiwanym 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 stron: 20
- Dostępne ramki pamięci: 3,5,7
- Ilość prób: 100
- Długość listy odniesień: 100

Z wygenerowanego pliku csv zostaje stworzona tabela poniżej:

Available frames in physical memory	***Algorithm***	***Amount of pages***	***Average amount of page faults***
3	**LRU**	**20**	**85.49**
3	**LFU**	**20**	**85.30**
5	**LRU**	**20**	**76.03**
5	**LFU**	**20**	**75.99**
7	**LRU**	**20**	**67.10**
7	**LFU**	**20**	**66.92**

Tabela wynikowa zawiera: **ilość dostępnych ramek w pamięci**, **nazwę algorytmu**, **ilość stron w pamięci**, **średnią ilość brakujących stron pamięci.**

3. Wnioski

Wnioski dla obu przeanalizowanych algorytmów są następujące:

1. Wraz ze wzrostem ilości dostępnych ramek w pamięci, zmniejsza się ilość brakujących stron.

2. Średnie ilości brakujących stron dla obu algorytmów dla tej samej ilości ramek są porównywalne. Różnice są nieznaczne.

3. 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 stron w ilości od **1** do **20**.