FIFO 알고리즘 분석

2014154031 이정찬

import java.io.FileInputStream;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Scanner;

import java.util.StringTokenizer;

//이정찬 2014154031 FIFO 알고리즘

public class Fifo{

public static void main(String[] args) throws Exception{

int 요구페이지수=0; //적중률을 구하기위한 요구페이지수(for문인덱스활용)

int a=0;

int b=0;

int input\_index=0; //페이지프레임에 들어갈 인덱스 초기값 0

double hit=0;//적중횟수를 셀 변수

String[] pageframe = new String[5];//5개의 페이지프레임

ArrayList<Integer> store = new ArrayList<Integer>(); //메모장에서 불러와서 저장할 장치

FileInputStream fin = null; //파일 입출력을 위해 fin

Scanner sc = null;

try{

fin = new FileInputStream("C:\\input3.txt");//메모장 input.txt에서 읽어오기

sc = new Scanner(fin);

String str = sc.nextLine().trim(); // 공백삭제를 한 것을 string변수에 저장

StringTokenizer st = new StringTokenizer(str," ");//구분문자로 st.r과 공백

int count = st.countTokens(); //StringTokenizer가 분리한 토큰의 개수를 count변수에 저장

for(int i=0; i<count; i++) //count개수만큼 store 저장장치에 정수형으로 변환하여 저장

store.add(Integer.parseInt(st.nextToken()));

while(true){ //초기에 빈프레임 5개에 넣어주기

if(store.get(a)==99){ //99들어오면 종료

요구페이지수=a;//현재 store의 인덱스 a가 요구페이지개수

break;

}

pageframe[input\_index++]=Integer.toString(store.get(a));//input\_index에 해당하는 페이지프렘에 요구페이지 넣기

//그다음 input\_index 다음을 위해 1증가

for(int j=0; j<input\_index; j++){//요구페이지 적중 검사

if(pageframe[j].equals(Integer.toString(store.get(a+1)))){

hit++; //페이지 프레임의 인덱스개수만큼 보조기억장치에 들어있는 store 안에 다음 요구페이지가

a++; //현재 들어있는 페이지프레임에서 같을 경우 hit 증가시키고 해당 요구페이지 인덱스도 1증가

}

else

continue;

}

a++;//다음의 요구페이지 값 받기위해 a증가

if(input\_index>4){//만약에 인덱스크기가 4보다 클 경우 인덱스를 -1로 초기화 시켜주고 다음 for문으로 넘겨줌

input\_index=-1;//이때 다음 for문에서는 인덱스 값을 0으로 바꿔줌

break;

}

}

for(int i=a; i<store.size(); i++){ //

if(store.get(i)==99){ //만약에 99가 들어 올경우 break

요구페이지수=i; //현재 i가 store의 요구페이지 수

break;

}

if(pageframe[0]!=null && pageframe[1]!=null && pageframe[2]!=null&&

pageframe[3]!=null && pageframe[4]!=null){//빈프레임이 아닐 때

if(pageframe[0].equals(Integer.toString(store.get(i)))|| pageframe[1].equals(Integer.toString(store.get(i)))||

pageframe[2].equals(Integer.toString(store.get(i)))|| pageframe[3].equals(Integer.toString(store.get(i)))

||pageframe[4].equals(Integer.toString(store.get(i)))){//페이지프레임에 있는 수와 store에 있는 요구페이지가 같을 경우!!

hit++;//적중횟수 누적

continue; //여기서 페이지프렘 인덱스 상승안하기위해 바로 for문으로

}

else{//적중하지 않았다면

input\_index++;//초기0

if(input\_index>4) //index를 증가시키다가 4를 넘어 갈 경우 index를 0으로 초기화

input\_index=0; // 먼저들어온게 먼저 나가는 구조이기 때문에!!

pageframe[input\_index]=Integer.toString(store.get(i));//input\_index에 해당하는 페이지프렘에 요구페이지 넣기

continue;

}

}

}

System.out.println("<최종 프레임 출력값>");

for(b=0; b<pageframe.length; b++){//pa.geframe의 개수만큼 p.ageframe에있는 데이터 출력

System.out.println(Integer.parseInt(pageframe[b]));

}

System.out.println("적중횟수:"+hit);

System.out.println("적중률:"+(hit/((double)요구페이지수)\*100)+"%");

// 적중한 횟수가 count\_right에 저장 적중률은 요구페이지 갯수로 나눠주어야함. //store크기는 99를 제외

//끝!

}catch(IOException e){ //입출력 오류시

System.out.println("입출력 오류입니다.");

}finally{

if(fin != null){

try{

fin.close();

}catch(Exception e){}

}

}

}

}

자바(main)로 구현하였다. 적중률을 구하기위해서 들어온 요구 페이지 수를 저장 할 요구페이지 변수를 선언하였다. 페이지프레임에 들어 갈 인덱스를 위해 input\_index 변수 설정하였다.

Hit는 적중 횟수를 셀 변수이다. 페이지프레임 5개 String 변수 pageframe 배열 5개 선언하였다.

Input.txt에 있는 요구페이지를 저장 할 store 정수형 가변 배열을 선언하였다.

파일 입출력을 위해 FileInputStream을 통해 공백 삭제와 더불어서 StringTokenizer을 사용하여 구분 문자로 공백과 str을 이용하여 분리한 토큰의 개수만큼 input.txt에서 불러와 store에 저장하였다. 초기에 빈 프레임에 5개를 넣어 주기 위해서 무한반복문을 통해 99가 들어 올 경우 나가게 끔 하였다. 첫 빈 페이지프레임에 우선적으로 넣고 인덱스를 증가 시켜 준 후 for문을 통해서 적중 검사를 한다. 빈 페이지 프레임이어도 적중 일 경우 적중 횟수를 증가시켜줘야 하기 때문에 if문을 통해 적중일 경우 적중 횟수를 증가시켜주게 만들었다. 그 다음 적중일 경우 요구페이지의 다음 것이 들어와야 하기 때문에 store의 인덱스를 1증가 시켜준다. 만약 적중이 아니라면 적중 횟수는 증가하지 않고 다음의 요구페이지를 받기 위해 store인덱스를 증가시켜준다. 해당 페이지프레임 인덱스가 만약 꽉 찰 경우 해당 인덱스를 -1로 초기화 시켜준다. 다음 for문에서는 0으로 초기화를 시켜주게 끔 만들기 위함이다.

다음 for문에서 이제 빈 페이지프레임에 꽉 찬 상태로 페이지 부재시, 적중시를 위해 알고리즘을 짰다. 전 for문에서 돌았던 store 인덱스(a)를 초기값으로 하여 store의 크기만큼 돌려준다. 이때도 마찬가지로 99를 만나면 끝나게끔 만들었다. if문을 통해 빈 프레임이 아닐 경우 적중이라면 적중 횟수를 늘려준다. else(그렇지 않다면) 즉 적중하지 않았다면 input\_index=-1 값을 ++ 시켜주고 0으로 만든 다음 if문을 통해 해당 인덱스가 꽉 찼다면 다시 0으로 바꿔준다. 먼저 들어온게 나가는 구조 즉, 오래있었던게 나가는 것이기 때문에 알고리즘 상 꽉 찰 경우 인덱스를 0으로 초기화 시켜주게 만들었다. 마찬가지로 store에 있는 값을 페이지프레임에 넣는다. 그 후 마지막으로 최종 프레임에 있는 값들을 출력 하기 위해 pageframe의 크기 만큼 돌려서 출력해준다. 적중 횟수도 출력해주고 적중률 (hit/요구페이지수) 도 출력해줬다.

