자료 구조 2반 중간 고사 대체 과제 2020년 1학기

2020년 5월 10일 자정까지

이름 : 김은수

학번 : 20191568

선서:

본인은 이 과제를 수행함에 있어서 어떠한 부정행위도 없이 스스로의 힘으로 수행하였음을 선서합니다.

날짜: 5월 1일

서명: 김은수

문제 1 기본 개념 (20점)

간단한 재귀함수 하나를 제시하고, Time Complexity를 계산하시오. 단. 재귀함수는 슬라이드에 제시된 것은 사용하면 안됨.

팩토리얼을 구하는 함수

int factorial(int n) {

if (n == 0) {

return 1;

}

return n \* factorial(n-1);

}

(C는 상수이다.)

T(n) = T(n-1) + C

= T(n-2) + C + C = T(n-2) + 2C

= T(n-3) + 3C

. . .

= T(3) + (n-3)C

= T(2) + (n-2)C

= T(1) + (n-1)C

= T(0) + nC

≤ C + nC = (n+1)C

결국은 T(n) ≤ (n+1)C 이고 C는 상수이므로 점근표기법으로 나타내면 O(n)으로 나타낼 수 있다.

문제 2 Arrays (20점)

다항식의 덧셈이나 행렬의 덧셈 알고리즘과 비슷한 방식으로 실행될 수 있는 또 다른 문제를 하나 제시하고, 간단하게 알고리즘을 제시하시오.

다항식의 덧셈이나 행렬의 덧셈 알고리즘과 비슷한 방식으로 실행될 수 있는 것은 다항식의 곱셈도 비슷하게 할 수 있다. 과제로 주어졌던 다항식의 덧셈 뺄셈 코드에 곱셈을 추가해 보았다.

public Polynomial Multiply(Polynomial b) {

Polynomial c = new Polynomial(128);

int sz = termArray[0].exp + b.termArray[0].exp + 1;

double a[] = new double[sz];

for (int i=0; i < terms; i++) {

for (int j=0; j < b.terms; j++) {

double co = termArray[i].coef \* b.termArray[j].coef;

a[termArray[i].exp+b.termArray[j].exp] += co;

}

}

for (int i = sz-1; i>=0; i--) {

if (a[i] != 0) {

c.NewTerm(a[i], i);

}

}

return c;

}

다항식의 곱셈을 직접 손으로 해보면 결국 (식1) 와 (식2)의 모든 항을 한번씩 곱해야 한다. 그렇게 되면 지수가 같은 경우가 생길 수 있어서 지수의 숫자를 인덱스로 하는 배열을 만들어 그곳에 계수의 곱을 더해 나가는 것을 만들었다. Return은 Polynomial로 보내야 하므로 배열을 거꾸로 읽어 c를 만들었다.

문제 3 Stacks and Queues (20점)

Java의 Stack 클래스의 소스코드를 찾아서 분석하시오.

구체적으로 우선 찾은 소스코드의 URL을 제시하고, Stack 클래스에서 제공해주는 메소드가 어떤 것이 있는지 나열하고, 그 구현 방법을 간략하게 설명하시오.

<http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/687fd7c7986d/src/share/classes/java/util/Stack.java>

Stack 클래스에서 제공해주는 메소드에는 empty(), peek(), pop(), push(E item), search(Object 0) 가 있다.

Stack 은 vector 를 상속받아서 구현되었다.

Push 의 경우 vector의 메소드 addElemant를 사용해 구현한다.

public E push(E item) {

addElement(item);

return item;

}

Pop의 경우 동기화 처리를 했고 peek()을 이용해 리턴을 하고 vector의 메소드 removeElementAt()을 사용해 제일 마지막에 있는 요소를 지웠다.

public synchronized E pop() {

E obj;

int len = size();

obj = peek();

removeElementAt(len - 1);

return obj;

}

Peek의 경우 동기화 처리를 했고 만약에 스택의 사이즈가 0, 즉 아무것도 없다면 EmptyStackException()로 스택이 비어있는 것을 나타낸다. vector의 메소드 elementAt()을 사용해 top에 있는 요소를 리턴한다.

public synchronized E peek() {

int len = size();

if (len == 0)

throw new EmptyStackException();

return elementAt(len - 1);

}

Empty의 경우 vector의 메소드 size() 를 통해 사이즈가 0인지 아니지 true false로 리턴해준다.

public boolean empty() {

return size() == 0;

}

Search는 동기화 처리를 했고 vector의 메소드 lastIndexOf()를 사용해서 오브젝트의 마지막 위치를 반환하고 stack의 인덱스에 맞게 보내기 위해 (스택의 사이즈 – 반환된 값) 을 리턴한다. Stack의 인덱스는 마지막 저장한 값(top)부터 0이 아닌 1부터 시작한다.

public synchronized int search(Object o) {

int i = lastIndexOf(o);

if (i >= 0) {

return size() - i;

}

return -1;

}

문제 4 Linked Lists (20점)

Java의 LinkList 클래스의 소스코드를 찾아서 분석하시오.

구체적으로 우선 찾은 소스코드의 URL을 제시하고, 이 클래스 내부의 변수들을 나열하고 각 변수의 의미를 설명하시오.

또한 이 클래스에서 제공해주는 메소드 중 하나를 선택하여 그 코드의 내용을 설명하시오.

<http://hg.openjdk.java.net/jdk7/jdk7/jdk/file/9b8c96f96a0f/src/share/classes/java/util/LinkedList.java>

변수는 size first last가 있다

Size 말그대로 linkedlist의 크기이고, first는 첫번째 노드에 대한 포인터이고, last는 마지막 노드에 대한 포인터이다.

이 클래스에서 제공해주는 메소드는 getFirst() 가 있으며 노드 first를 f에 할당하고 f가 비어있으면 NoSuchElementException()을 반환하고 아니라면 노드에 있는 아이템을 리턴한다. 말 그대로 first에 있는 아이템을 가져오려고 하는 것이다.

public E getFirst() {

final Node<E> f = first;

if (f == null)

throw new NoSuchElementException();

return f.item;

}