2021 - 2학기 알고리즘

(심정섭 교수님)

<알고리즘 설계과제1>

학과 : 컴퓨터공학과

학번 : 12171661

이름 : 윤혁

이메일 : yhuk122@naver.com

**1. 개요**

1) 설계의 목적 : 레드블랙트리(red-black tree)를 이용하여 앱스토어에 등록된 애플리케이션들을 효율적으로 관리하는 프로그램을 설계한다. 이 프로그램은 판매하는 애플리케이션 관리를 위한 몇 가지 기능들을 수행한다. 각 기능에 대한 정보를 표준입력으로 입력받고 수행한 결과를 표준출력으로 출력하는 프로그램을 구현한다.

2) 요구사항 :

(1) 애플리케이션 등록

- 질의형식: “I A N S P”

I: 새로운 애플리케이션을 등록하는 질의를 나타내는 기호

A: ID

N: 이름

S: 용량

P: 가격

- 출력형식: “De”

De: 트리에서 애플리케이션 정보가 저장된 노드의 깊이 (root의 깊이는 0)

- 설명: 애플리케이션 정보를 입력받아 레드블랙트리에 노드를 삽입하고, 그 노드의 깊이를 출력 한다. 만약 해당 애플리케이션이 이미 등록되어 있다면, 그 노드의 깊이를 출력하고 등록은 거절된다.

(2) 애플리케이션 검색

- 질의형식: “F A”

F: 특정 애플리케이션을 검색하는 질의를 나타내는 기호

A: ID

- 출력형식: “De N S P” 또는 “NULL”

De: 트리에서 애플리케이션 정보가 저장된 노드의 깊이

N: 이름

S: 용량

P: 가격

- 설명: 입력으로 주어진 애플리케이션 ID를 탐색하여 존재하면, 애플리케이션의 정보를 출력한 다. 만약 애플리케이션이 존재하지 않으면, “NULL”(주의: 대•소문자 정확히 일치시킬 것)을 출력한 다.

(3) 애플리케이션 업데이트

- 질의형식: “R A N S P”

R: 애플리케이션을 업데이트 질의를 나타내는 기호

A: ID N: 업데이트할 이름

S: 업데이트할 용량

P: 업데이트할 가격

- 출력형식: “De” 또는 “NULL”

De: 트리에서 애플리케이션 정보가 저장된 노드의 깊이

- 설명: 입력으로 주어진 애플리케이션 ID를 탐색하여 존재하면, 입력받은 애플리케이션 정보를 업데이트하고, 애플리케이션에 대한 노드의 깊이를 출력한다. 만약 애플리케이션이 존재하지 않으면, “NULL”을 출력한다.

(4) 애플리케이션 할인

- 질의형식: “D x y P”

D: 애플리케이션을 할인하는 질의를 나타내는 기호

x: 범위의 시작을 나타내는 정수

y: 범위의 끝을 나타내는 정수

P: 할인율

- 출력형식: 없음

- 설명: 범위 [x, y] 내의 ID를 가진 애플리케이션을 모두 탐색하여 각 애플리케이션의 가격에 ‘P’%의 할인율을 적용한다.

할인가 = 가격 X (100-P)/100 (단, 할인가의 소수 부분은 버린다)

3)개발 환경

- 개발언어 : C++(Visual Studio 2019)

**2. 필요한 자료구조 및 기능**

1) 필요한 자료구조 :

\*이진탐색트리

- 노드들이 정렬된 순서로 위치된 이진 트리

- 노드의 키가 부모의 키보다 작으면 왼쪽 서브트리에 위치하고 크면 오른쪽 서브트리에 위치함

\*레드블랙트리 속성을 만족하여야 한다

- Root Property : 모든 root는 black이다

- External Property : 모든 leaf는 black이다

- Internal Property : red 노드의 자식은 black이다

- Depth Property : 모든 leaf들은 같은 black depth를 갖고 있다

2) 기능 : 트리에서의 삽입, 탐색, 업데이트, 할인

**3. 기능별 알고리즘 명세(복잡도 분석)**

1)삽입 :

트리가 비어있는 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

모두 단순 비교, 대입 연산이므로 O(1) time

트리가 비어있지 않은 경우

텍스트이(가) 표시된 사진

자동 생성된 설명

While 문에서 최대 트리의 높이만큼 탐색하므로 O(h) -> O(log n) time

나머지는 모두 단순 비교 및 대입연산 이므로 O(1) time 내에 수행된다

Check 함수 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

단순 비교 연산으로 삼촌노드 결정해주고, 비교연산의 결과에 따라 restruct, recolor 여부 결정한다. 총 O(1) time

Restruct 함수 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

단순 비교 연산 및 대입 연산이므로 O(1) time (updateDepth 함수는 restruct 기능 자체와는 무관하므로 제외)

Restruct 함수에는 위 사진과 같은 과정이 한번 더 일어나는데 변수만 바뀌고 형식은 같다. O(1) time

텍스트이(가) 표시된 사진

자동 생성된 설명

Recolor 자체는 단순히 해당 노드의 색깔만 바꿔주므로 O(1)

**Total : O(log n)**

2) 탐색 :

텍스트이(가) 표시된 사진

자동 생성된 설명

While 문에서 최대 트리의 높이만큼 탐색하므로 O(h) -> O(log n) time

나머지는 모두 단순 비교 및 대입연산 이므로 O(1) time 내에 수행된다

**Total : O(log n)**

3) 업데이트 :

텍스트이(가) 표시된 사진

자동 생성된 설명

트리에서 ID에 해당하는 노드를 탐색하고(O(log n)) 노드 클래스 안의 변수 값을 수정해주므로 O(1) 총 O(log n)

**Total : O(log n)**

4) 할인 :

텍스트이(가) 표시된 사진

자동 생성된 설명

트리에서 중위순회를 통해 범위의 값을 찾아서 할인한다.

중위순회는 최악의 경우 모든 노드를 탐색할 수도 있기 때문에 O(n) time

**Total : O(n)**

**4. 인터페이스 및 사용법**

텍스트이(가) 표시된 사진

자동 생성된 설명

1) 테스트케이스의 수 입력

2) 명령어 입력

(1) I : 애플리케이션 등록 명령어, 이후 id, 이름, 용량, 가격을 차례대로 입력하여 해당 정보를 가진 노드를 삽입하고 그 깊이를 출력한다. 기존의 있는 노드일 경우 삽입하지 않고 그 노드의 깊이만 출력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

(2) F : 애플리케이션 검색 명령어, 이후 노드의 id를 입력하여 노드를 찾는다. 찾으면 해당 노드 깊이와 정보 출력하고 노드가 존재하지 않으면 NULL을 출력한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

(3) R : 애플리케이션 업데이트 명령어, 이후 id, 이름, 용량, 가격을 차례대로 입력하여 해당 노드가 존재하면 입력받은대로 정보를 수정하고 애플리케이션이 존재하지 않으면 NULL을 출력한다.

텍스트, 장치, 측정기이(가) 표시된 사진

자동 생성된 설명

(4) D : 애플리케이션 할인 명령어, 이후 범위의 시작, 끝, 할인율을 차례대로 입력하여 범위 [x, y] 내의 id를 가진 애플리케이션을 모두 탐색하여 P%의 할인율을 적용한다.

텍스트, 장치, 측정기, 어두운이(가) 표시된 사진

자동 생성된 설명

**5. 평가 및 개선 방향**

1) 구현한 알고리즘의 장점

- 이 알고리즘은 구글링해서 얻은 정보들이 아닌 100% 알고리즘 과목 강의시간에 교수님이 강의하신 내용을 기반으로 학습해가며 만들었기 때문에 심정섭 교수님의 수업을 들은 학생이라면 본인의 코드가 아니더라도 이해하기가 어렵지 않을 것으로 예상된다. 또한 선형구조가 아닌 트리의 장점인 최악수행시간을 대부분 O(log n) 시간에 수행하므로 빠른 수행시간을 기대할 수 있고 실제 돔서버 테스트케이스에서 수행시간도 0.239초 이내로 나오는 것을 알 수 있다.

2) 구현한 알고리즘의 단점

- 요구사항에서 삽입할 때 restruct를 수행할 수도 있는데 이때 기능과는 관계없이 알맞은 깊이를 출력하기 위해 updateDepth함수를 추가해줬는데 여기서 수행시간이 조금 늘어날 수 있을 것으로 예상된다. 또한 restruct 함수에서 크게 2가지경우, 세부적으로 4가지 경우로 나누어 구현했는데 코드가 거의 유사하거나 변수이름만 다른 부분도 있어서 코드가 불필요하게 길어졌다.

3) 향후 개선 방향

- 노드들의 깊이를 부모노드보다 항상 +1인 상태를 자동적으로 유지하도록 하는 장치가 있으면 좋겠고 코드의 가독성과 효율성을 위해 restruct 부분에서 중복된 부분을 간략화 하면 좋을 것 같다. 또한 leaf 노드도 삽입되는 노드마다 left, right leaf를 추가하는 방식 보다는 leaf 노드를 하나만 만들고 모든 leaf의 부모노드들이 자식으로 한 leaf node를 가리킨다면 메모리 사용면에서 효율적일 수 있을 것 같다.