Algorithm HW2

Order-Statistic Tree

2013-11381
강지원

● 구현할 내용

Red-Black Tree에서 각 노드 별로 자식 노드의 개수를 저장 할 수 있는 size 변수를 추가한 Order-Statistic Tree를 구현한다. 이때 size는 자기 자신을 포함해서 "왼쪽 자식 노드의 수 + 오른쪽 자식 노드의 수 + 1"이 된다. 그리고 구현한 Order-Statistic Tree를 이용해서 i 번째 작은 숫자를 찾는 select 기능과 Tree 안에 있는 숫자가 몇 번째 숫자인지를 찾아주는 rank 기능을 구현 한다. 이때 size를 이용하면 O(log n)의 시간 복잡도로 구현이 가능한다.

● 구현 방법

OS-Tree 전체와 Checker 프로그램 모두 파이썬 언어를 사용해서 ostree.py 파일과 checker.py 파일에 구현하였다. 트리의 각 노드는 'Node' 클래스로 구현하였고 OS-Tree 또한 'OS_Tree' 클래스로 구현하였다. 'Node' 클래스는 'parent'와 'left child', 'right child', 'size', 'value', 'color'를 변수로 가지고 있다. 'OS_Tree' 클래스는 root node를 가지고 있고, 'insert', 'delete', 'select', 'rank'를 내부에 메소드로 구현하였다.

1. Insert

기본적으로 삽입 되는 노드를 RED 컬러로 설정하였고, Red-Black Tree의 규칙에 맞춰서 5가지 케이스로 나누어서 구현하였다.

① Root가 없는 경우

② Parent 가 Black 경우

③ Parent 가 Red 이고, Parent의 Sibling이 Red 인 경우

④ Parent 가 Red 이면서, Parent의 Sibling이 Black, 삽입 되는 노드는 Parent 오른쪽 자식이고, Parent는 Parent의 Parent의 왼쪽 자식인 경우(여기서 오른쪽은 대칭)

⑤ Parent 가 Red 이면서, Parent의 Sibling이 Black, 삽입 되는 노드는 Parent의 왼쪽 자식이고, Parent는 Parent의 Parent의 왼쪽 자식인 경우(여기서 오른쪽은 대칭)

2. Delete

삭제 되는 노드의 값을 직전의 숫자나 직후의 숫자와 자리를 바꾼 후 해당 노드를 삭제하였다. 이때, 옮겨진 노드의 자식 노드는 leaf 노드(NIL node)뿐이거나 최대 하나의 child를 가질 수 밖에 없다. 옮겨진 노드를 삭제하고 자식 노드(또는 NIL node)를 삭제된 자리에 채우면 된다. 이때 채워질 노드를 n, n의 부모를 p, sibling을 s라고 하자.

① n 이 root가 되는 경우

② s가 Red 일 경우

③ p, s, s의 child 들이 Black 일 경우

④ p는 Red, s와 s의 child 들은 Black 인 경우

⑤ s는 Black s의 left child는 Red, s의 right child는 Black 이고 n이 부모의 왼쪽 자식인 경우

⑥ s가 Black, s의 right child는 Red, n이 p의 왼쪽 자식인 경우.

3. Select

Root node 부터 재귀적으로 내려가면서 사이즈를 이용해서 i번째 작은 숫자를 찾았다. 만약 node의 left child의 size + 1이 i와 같다면 해당 node가 i번째 작은 숫자가 된다.

4. Rank

찾고자 하는 숫자의 node를 찾고, 해당 노드부터 root까지 올라가면서 왼쪽에

있는 사이즈들을 더하면 찾고자 하는 숫자의 rank를 구할 수 있다.

● Checker Program

Checker 프로그램은 1부터 999까지의 0으로 채워진 배열(check_list)을 선언하여 구현하였다.

① Insert

해당 숫자 v가 입력 되면 check_list[v]의 값을 1로 변경해주었다. 만약 이미 1로 변경 되어 있다면 0을 return, 그렇지 않으면 v를 return.

② Delete

해당 숫자 v가 입력 되면 check_list[v]의 값을 0으로 변경해주었다. 만약 이미 0인 경우에는 0을 return, 그렇지 않으면 v를 return.

③ Select

check_list의 1번부터 확인해서 1의 개수를 세어서 해당하는 순서의 숫자를 return. 만약 1의 개수보다 큰 수가 들어오면 0을 return.

④ Rank

해당 숫자 v가 입력 되면, check_list[v]보다 앞의 1의 개수를 세었다. v 위치의 개수인 1을 더해서 return. 만약 check_list[v]가 0인 경우 0을 return.

ostree.py 프로그램을 실행하면 checker 프로그램을 위해 output을 output.txt에 저장한다. 따라서 어떠한 input file에 대해서 checker 프로그램을 실행하기 위해서는 ostree.py 프로그램을 실행한 후, 같은 input file로 checker 프로그램을 실행하면 된다. checker 프로그램을 실행하면, 명령어를 제외한 ostree.py의 결과를 리스트로 출력하고, checker 프로그램에서 배열을 이용한 결과를 리스트로 출력해준다. 그리고 두 리스트를 비교한 결과를 bool 값으로 출력한다. 만약 True가 출력된다면 ostree.py 프로그램이 올바르게 작동하는 것이다.

- **OS tree 및 Checker program 실행 방법**

```
python ostree.py input1.txt
python3 ostree.py input1.txt
```

위의 명령어 둘 중 하나를 실행 하면 된다(python의 버전이 여러 개인 경우 아랫 줄의 명령어, 그렇지 않은 경우 윗줄의 명령어). 다른 input file을 사용하고 싶으면, input1.txt 대신 다른 input file 이름을 입력하면 된다. 실행 시, 결과는 stdout으로 나오게 된다. 먼저 input이 차례대로 출력 되고 output이 input 순서대로 출력된다.

Checker 프로그램을 사용하기 위해서는 위의 명령어를 실행한 직후 아래와 같이 실행 하면 된다(ostree.py를 실행하면 output.txt 파일에 명령어를 제외한 결과를 저장하도록 하였다.).

```
python checker.py input1.txt
python3 checker.py input1.txt
```