자료구조 과제 5 – 트리 보고서

20222663 권기영

**1. 사전 탐색 트리 만들기**

**<문제 내용>**

주어진 사전 파일을 활용해 사전 탐색 트리를 만든다. 전체 높이 표시, 단어 개수 표시(트리 노드 수).

**<문제 해결 - python>**

**(1) 이진 탐색 트리 클래스**

1) 이진 탐색 트리 클래스를 만들고, 단어 삽입(insert), 높이 구하기(getHeight), 레벨 순회로 단어 수 세기(count\_by\_levelorder), 탐색 및 단어 레벨 구하기(search\_and\_getLevel) 메소드를 정의하였다.

2) insert 함수

노드는 단어와 뜻 2개로 이루어진 배열의 형태로 되어있다. 트리에 삽입을 할 때, 단어를 기준으로 비교하여 삽입한다.

3) getHeight 함수

재귀함수를 이용하여 높이를 구하는 함수이다. 왼쪽 오른쪽으로 계속 내려가면서, 왼쪽과 오른쪽 중 최대값을 반환한다.

4) count\_by\_levelorder 함수

레벨 순회를 통해 단어의 총 개수를 세는 함수이다. 큐에서 노드가 빠져나갈 때 마다 카운트 해준다. 무조건 left와 right 모두 푸시하도록 구현했으므로, pop 되는 노드가 None이 아닌 경우에만 카운트해준다.

5) search\_and\_getLevel 함수

레벨 순회를 활용하여 단어의 뜻과 레벨을 반환하는 함수이다. 단어를 기준으로 탐색을 진행해서 같은 단어인 노드가 나오면 뜻을 반환한다. 또한 큐의 길이만큼 반복해서 한 번 반복될 때 마다 레벨을 증가시킨다.

**(2) main 함수**

1) 텍스트 파일로부터 단어와 뜻 읽어오기

처음에는 :를 기준으로 split 하려고 했는데, :가 여러 개가 있는 문장들이 있어서 처음 나오는 공백을 기준으로 공백 전 까지는 단어, (공백 + 3)번째 인덱스 부터 끝까지는 뜻으로 저장하였다. [단어, 뜻]의 배열 형태로 트리에 삽입하였다.

2) 처음 단어 개수는 삽입을 하면서 카운트 해주었고, 두 번째 단어 개수는 레벨 순회를 통해 카운트했다. 높이는 재귀와 max 함수를 이용한 getHeight 함수를 활용해 출력하였다.

3) 단어를 입력받아서, 해당 단어와 맞는 단어의 뜻과 레벨을 출력하였고, 레벨 순회를 활용하였다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**2. 수식 트리 만들기**

**<문제 내용>**

연산자는 +-\*/(), 피연산자는 실수로 이루어진 수식 트리를 anytree 라이브러리를 활용해 구현한다. 중위표기식을 입력하면 후위표기식과 전위표기식과 식의 결과값을 출력하고, anytree에서 지원하는 트리의 모양을 출력한다.

**<문제 해결 - python>**

**(1) 중위표기식 입력 받기**

1) expression에 문자열 전체를 입력 받는다.

2) 문자열의 문자 하나하나에 대해서, 연산자면 배열에 바로 추가하고, 연산자가 아니라면(피연산자) digit이라는 새로운 배열에 추가하다가, 연산자가 나오면 digit 배열을 문자열로 바꾼 뒤, 실수로 바꾸어서 저장한다.

3) 괄호가 있는 경우 연산자가 연속하여 2번 나오는 경우가 있기 때문에, digit 배열에 숫자가 하나도 들어가지 않아서 오류가 나는 경우를 대비하여, 숫자가 추가될 시 count를 증가시켜, count가 0이 아닌 경우에만 피연산자를 저장하도록 한다.

**(2) 입력 받은 중위표기식 후위표기식으로 바꾸기**

1) 연산자 간의 우선순위를 반환하는 함수 Priority를 선언한다

2) 피연산자면 바로 postfix에 추가하고, 여는 괄호가 나오면 무조건 스택에 추가, 닫는 괄호가 나오면 여는 괄호가 나올 때 까지 스택에서 연산자를 pop하기, 괄호를 제외한 연산자가 나오면 우선순위가 더 작은 것이 나올 때 까지 스택에서 pop한다.

3) 중위표기식을 전부 돌고 나서 스택에 남은 것이 있다면 pop 해준다.

**(3) 후위표기식을 수식 트리로 변환하기**

1) 피연산자가 나오면 스택에 push한다. 연산자가 나오면 스택에서 2개를 꺼내 자식 노드로 만든 뒤(순서는 먼저 pop한 것이 right자식 노드, 나중에 pop한 것이 left자식 노드) 스택에 push한다.

2) 이 과정을 끝까지 반복하면 스택에는 하나의 완전한 트리만 남게 되고, 이를 반환한다.

**(4) 전위 순회와 후위 순회를 통해 전위표기식, 후위표기식 출력**

전위 순회는 노드 출력을 먼저 한 뒤, 자식 노드를 순회하고, 후위 순회는 자식 노드를 먼저 순회한 뒤, 출력을 한다. node.children은 [left, right]의 형태로 이루어진 배열이기 때문에 이 배열에 대해서 for문을 돌게 되면, 왼쪽을 간 뒤에 오른쪽을 가는 순서로 갈 수 있다.

**(5) 수식 결과값 출력**

1) is\_leaf 함수와 리프 노드는 무조건 피연산자임을 활용하여, is\_leaf가 True일 때 값을 return 한다.

2) 리프 노드가 아니면 연산자이므로, 각 연산자에 맞는 연산을 수행해서 return 한다.

**(6) 트리 모양 출력**

anytree 라이브러리에서 지원하는 RenderTree 함수를 활용하여 트리를 인자로 주면, 자동으로 모양을 출력해준다.

텍스트, 폰트, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명

**3. 이진 트리 시각화**

**<문제 내용>**

직접 트리 자료구조를 정의하고, 간단한 이진 트리를 만들고 이를 시각화 한다.

**<문제 해결 - python>**

**(1) 트리 자료구조 선정 – 배열**

트리를 배열의 형태로 저장한다. 그 이유는

1) 인덱스 자체가 레벨을 포함하는 개념이기 때문에 레벨을 따로 계산할 필요가 없다.

2) 배열을 활용하게 되면 삽입, 삭제가 힘들어지는데, 삽입 삭제를 하는 것이 아닌 시각화만이 목적이기 때문에 적합하다.

3) 왼쪽 자식 노드는 짝수, 오른쪽 자식 노드는 홀수로 쉽게 케이스 구분이 가능하다.

**(2) pre\_visual – 트리 시각화 함수**

1) 트리 배열과 인덱스를 인자로 하는 트리 시각화 함수이다.

2) 중요한 것은 전역변수 count인데, count는 아직 방문하지 않은 반대쪽 노드의 개수를 나타낸다. 왼쪽 노드 먼저 방문을 하면서, 만약 오른쪽 노드가 존재한다면 count를 1 증가시킨다. 나중에 방문해야 할 노드에 대한 선을 |로 표현한다. 오른쪽 노드를 방문할 때 마다, count를 1씩 감소시킨다.

3) 레벨에 대해서도 데이터의 위치가 달라진다. 같은 레벨의 데이터는 같은 위치에 있어야 하므로 log2 함수에 index를 넣어 레벨을 계산한다. Index는 2의 제곱수 형태로 증가하기 때문이다.

도표, 텍스트, 스크린샷, 평면도이(가) 표시된 사진

자동 생성된 설명도표, 스크린샷, 라인, 디자인이(가) 표시된 사진

자동 생성된 설명