IE260 Midterm Homework Report

20150359 Chiyoon Park

1. Description

- 1. Class Node

Stack과 MakeFormulaIntoTree에서 사용되는 노드입니다. 먼저 __init__ 부분에서 노드 클래스는 parameter로 value 한 값을 받는데, 이 value값이 어떤지에 따라 value, type, priority가 자동으로 결정됩니다. 숫자의 경우 "digit", unary operator의 경우 "unaryoper", binary operator의 경우 "binaryoper", parentheses의 경우 "parentheses", x의 경우 "variable" 타입을 부여받게 되며, 각각에 맞는 priority가 결정됩니다. Head의 경우 스택에서 사용하기 위해서 만든 것으로 "head" type을 가집니다.

Node 클래스 내부의 메소드들을 보면, 크게 3가지 종류의 메소드가 있습니다. Get / Set / Is 입니다. Get 함수들의 경우 내부의 변수들을 return하는 역할을 합니다. 즉, 트리나 스택에서 자식 노드나 다음 노드를 반환하는 등의 역할을 수행합니다. Set 함수들은 내부의 변수들을 변경하여 트리의 자식 노드를 설정하거나, 스택의 다음 노드를 설정하는 역할을 합니다. 여기서 특별하게 binary tree의 구조에서 unary라는 특별한 자식 노드를 설정하였는데, 이는 unary operator의 연산을 위해서 사용됩니다. Is 함수들은 자식 노드가 있는지, 다음 노드가 있는지 확인하는 함수로, 그 유무에 따라서 True와 False를 반환합니다.

- 2. Class Stack

Node 클래스를 통해 스택 자료 구조를 만드는 클래스입니다. Head 노드와 length 두 가지 변수를 초기화시킵니다. 총 5가지 메소드가 있습니다. getTop 메소드는 Top 원소가 존재할 경우 이를 반환합니다 . push는 스택에 원소를 push하는 메소드입니다. Parameter로 입력받은 노드를 스택에 추가하고 length를 1 증가시킵니다. Pop의 경우 스택의 top을 pop하는 메소드입니다. 만약 원소가 하나도 없을 경우 IndexError를 발생시킵니다. getLength는 스택의 길이를 반환하는 메소드입니다. 마지막으로 returnStack의 경우 스택을 list에 담아서 반환하는 메소드입니다. for문을 통해서 스택을 처음부터 끝까지 탐색하면서, 스택의 원소들의 값을 리스트에 담아서 반환합니다. 이 메소드의 경우 Function의 solution을 구하는 데에 직접적으로 사용되지는 않고, 그 과정에서 스

택이 어떤 식으로 동작하고 있는지 확인하기 위한 용도로 구현하였습니다.

- 3. Class MakeFormulaIntoTree

Main 함수에서 입력받은 Equation을 list의 형태로 받아서 변형된 binarytree로 만들어주는 클래스입니다. Node 클래스와 Stack 클래스를 사용합니다. 총 네 가지 메소드가 있습니다. 가장 핵심이 되는 makeTree, 이 과정에서 사용되는 mergeOperand가 있으며, 트리가 만들어진 후 그 값을 계산하는 eval, 그리고 만들어진 tree의 root 노드를 반환하는 getTree가 있습니다.

먼저 mergeOperand의 경우 operator의 종류에 따라서 적합하게 operand를 결합하여 binary 트리를 만드는 메소드입니다. 여기서 operator가 unary인가 binary인가를 먼저 확인합니다. Unaryoperator의 경우 노드의 left나 right에 자식노드를 저장하지 않고, unary라는 변수에 저장합니다. 그리고 operand를 하나만 필요로 합니다. 따라서 operand 하나를 pop해서 부모 노드의 unary 변수에 저장합니다. Binaryoperator의 경우 두 자식 노드가 필요하므로 두 개의 operand를 pop하여 operator와 결합하여 stackoperand에 push합니다.

makeTree 메소드의 경우 operator를 담는 스택, operand를 담는 스택 두 개의 스택과 mergeroperand를 사용하여 입력받은 formula를 tree로 만드는 메소드입니다. List의 형태로 입력받은 formula를 for 문을 통해서 하나씩 탐색하여 노드에 담아 binarytree로 만듭니다. 여기서 가장 핵심적인 사항은 "-" 연산자가 binary로 사용되기도 하고, unary로 사용되기도 한다는 것입니다. Node 클래스에선 "-" 연산자가 들어올 경우 일단 "binary" type으로 설정하게 했습니다. 따라서 여기서 "-" 연산자가 unary로 사용될 경우 node의 type과 priority를 unaryoperator에 맞춰 바꿔주는 작업을 진행했습니다. 이를 위해서 lastappendnode라는 변수를 선언했는데, 이는 for 문에서 바로 직전 step에 사용된 노드를 저장합니다. 이 변수에 저장된 노드를 통해서 이전에 들어온 값을 확인하고 "-" 연산자의 type을 바꿉니다. "-" 연산자의 경우 식의 가장 처음에 사용되거나, 바로 앞의 항이 연산자인 경우 unaryoperator로 사용됩니다. 따라서 lastappendnode가 없거나, 혹은 lastappendnode가 숫자나 변수가 아닐 경우 unaryoperator로 변환하는 작업을 진행했습니다. 이후 tree를 만드는데, 만약 unaryoperator나 여는 괄호일 경우 바로 stackoperator에 push하고, 만약 닫는 괄호 ")"가 나올 경우 "(" 괄호가 나올 때까지 mergeOperand를 실행하며, 마지막으로 현재 stackoperator의 가장 마지막으로 들어있는 operator보다 낮은 priority를 가지는 operator가 들어올 경우 mergeOperator를 진행합니다. 그리고 For 문의 탐색을 전부 끝내면, 남은 operand와 operator들을 mergeOperator하여 root 하나로 합칩니다. 그리고 이 root를 변수 tree에 저장합니다.

Eval은 입력받은 노드의 값을 변수 x의 값과 함께 계산하는 메소드입니다. Parameter로 node와 x를 받는데, node는 입력받는 노드와 그 아래의 트리이고, x는 변수의 값입니다. 이 메소드는

recursion을 통해서 하위 노드를 계속 탐색하며 최종적으로 값을 구합니다. 여기서 총 4가지 경우로 갈래가 나뉘는데, 먼저 입력받은 노드의 type 변수가 variable인 경우(즉, 노드가 x인 경우) 이 경우는 자식 노드가 없고, 변수의 값을 반환해야 하므로, x값을 return합니다. 두 번째로 unaryoperator인 경우입니다. 이 경우 unary에 저장된 노드를 꺼내서 연산을 진행합니다. 여기서 ln의 경우 0의 값이나 음수 값이 들어가게 되면 문제가 생기는데, 이때는 ln에 엄청 작은 숫자를 넣게 하여 그 값을 반환합니다. 이 값은 – 무한대에 가까운 값입니다. 이렇게 반환할 경우 음수 부분은 solution을 구하는 findsolution2 메소드에서 자연스럽게 제거되므로, 해를 구하는데 문제가 생기지 않습니다. 나머지 연산자의 경우 자식노드의 값을 연산자에 넣어 연산한 값을 반환합니다. 세 번째로 binaryoperator인 경우입니다. 이 경우 왼쪽과 오른쪽 자식 노드가 모두 존재하고, 자식 노드의 값을 operator를 통해서 연산한 값을 반환합니다. 각 연산자가 어떤 것인지 getValue()를 통해 확인한 후 각 operator에 맞는 연산을 진행하였습니다. 마지막으로 노드가 숫자인 경우, recursion의 가장 마지막인 경우로, 이때는 그냥 node의 값을 float로 변환하여 반환합니다.

마지막 메소드는 getTree입니다. Tree 변수에 저장된 root 노드를 반환하는 함수입니다.

- 4. Method Check

Check 메소드는 중간값 정리에서 아이디어를 얻어 구현한 메소드입니다. Parameter로 equation의 좌항에 대해서 구간의 min과 max값을 대입한 값과 equation의 우항에 대해서 구간의 min과 max값을 대입한 값을 입력받습니다. 그리고 min 값에 대해서 좌항과 우항의 차의 부호를 계산하고 max값에 대해서 좌항과 우항의 차의 부호를 계산합니다. 이 부호가 바뀌었다면 이 안에 해가 최소한 하나가 존재한다는 것을 알 수 있으므로 True를 반환하고, 반대의 경우 False를 반환합니다. 이후 binary search와 findsolution2에 사용됩니다.

- 5. Method binarySearch

binarySeach 메소드는 binary search를 통해서 입력 받은 구간 사이에 존재하는 해를 찾는 함수입니다. Parameter로 equation의 좌항과 우항을 tree로 만든 것과 구간의 양 끝 값을 받습니다. 그리고 좌항과 우항의 중간 값을 mid 변수에 저장합니다. 그리고 구간의 왼쪽 값과 mid값을 check합니다. 여기서 False일 경우 구간의 왼쪽 값과 mid 사이에선 해가 없다는 것으로, 오른쪽 구간으로 넘어가서 과정을 반복합니다. 만약 True가 나온 경우 구간의 왼쪽과 mid에서 좌항과 우항의 대소가 바뀌었다는 것이므로 이 구간에서 다시 반복합니다. 이 과정을 mid 변수를 좌항과 우항에

집어넣은 값 사이의 차가 0.00001보다 작아질 때까지 반복합니다. 이렇게 최종적으로 구해진 mid 값을 반환합니다.

- 6. Method findsolution2

Findsolution2 메소드는 recursion과 Binary search를 사용하여 함수의 해를 구하고, 이 과정에서 걸린 시간 역시 계산하여 둘을 반환하는 메소드입니다. parameter로는 equation의 왼쪽 파트와 오른쪽파트, 그리고 range의 양 끝 값을 입력받습니다. 그리고 iter라는 값을 입력받아 이를 통해서 recursion의 횟수를 제어합니다. 먼저 iter값이 15가 되기 전까지 구간을 반으로 나눠 그 구간에 대해서 다시 findsolution2를 진행합니다. 그리고 iter값이 15인 경우 (구간의 크기가 우리가 구하고 싶은 구간의 크기의 1/2^15이 된 경우), 각 구간을 check합니다. 구간이 recursion을 통해서 충분히 작게 쪼개졌으므로, 이 작은 구간에 대해서 check값이 False가 나온 경우 해가 없다는 것을 예상할 수 있습니다. 따라서 여기서 check값이 true인 구간들에 대해서만 binary search를 진행합니다. 여기서 반환된 값들을 모아서 solutionList에 저장한 후 반환합니다. 총 걸린 시간은 datetime 모듈을 통해서 진행했는데, 메소드 실행 첫 부분에서 datetime.now()를 통해 시간을 측정하고 모든 과정이 끝난 후 반환하기 직전 datetime.now()를 통해 시간을 측정하여, 둘의 값을 빼서 걸린 시간을 측정했습니다.

- 7. Method overlap

이 메소드는 findsolution2에서 중복된 솔루션이 얻어진 경우 제거하는 메소드입니다. Parameter를 통해서 입력 받은 solution을 for문을 통해서 탐색합니다. 여기서 인접한 solution 두개의 차이가 0.01보다 작은 경우 sortsolution에서 추가하지 않음을 통해서 solution에서 제거합니다.

- Extra. Findsolution1

이 메소드는 klms Q&A 게시판의 게시글을 확인하다 발견한 댓글에서, 실제 해와 내가 구한 해 사이의 차이가 0.001 사이가 되게 하라는 것을 확인하고 구현한 메소드입니다. 이 메소드에선 입력 받은 구간을 0.001 간격으로 자릅니다. 그리고 이 구간의 양 끝 값에 대해서 계속 check를 진행하여서 만약 check값이 true가 나온다면 이 사이에 해가 있다는 것이므로, 이 구간의 양 끝 값을 기록하고, 양 끝 값을 2로 나눈 값을 저장한 후 반환합니다. 이렇게 찾은 mid값의 경우 항상 실제 해와 0.001 사이에 들어갑니다. 이후 findsolution2를 구현한 후 사용하지 않는 메소드입니다.

- 8. Method main

위에서 구현한 메소드와 클래스를 사용해서 입력 받은 Equation의 해를 구하는 메소드입니다. 먼저 Exit가 입력되기 전까지 계속 동작하게 while True를 통해서 루프를 구성했습니다. 이후 Exit가 입력될 경우 roop를 벗어나게 만들었습니다. 이후 입력한 Equation의 문제가 없는지 검사하는 과정을 진행합니다. 먼저 "="가 없는 경우 에러 메시지를 띄우고 다음 루프로 넘어갑니다. 만약 이런 문제가 없는 경우 입력 받은 line을 " "를 기준으로 나누어서 리스트로 만들어 트리를 만드는 데에 사용할 수 있도록 합니다. 여기서 만약 사용할 수 있는 연산자나 숫자, 변수가 아닌 다른 것이 들어올 경우 에러를 띄웁니다. 이를 위해서 err라는 변수를 선언했습니다. 그리고 여기까지 문제가 없는 경우 "="를 기준으로 leftside와 rightside를 나눕니다. 이때 만약 둘 중 하나라도 비어있다면, 좌항과 우항에 아무 것도 입력하지 않은 것이므로 에러를 띄웁니다. 여기까지 검사했다면 바른 양식으로 equation을 입력했다는 것이므로, Range를 입력 받습니다. 여기서 try 를 통해서 만약 숫자가 입력되지 않았다면 에러를 띄웁니다. 여기까지 전부 통과했다면 입력 값에는 문제가 없으므로 solution을 구하는 과정으로 넘어갑니다, 먼저 좌 우 항을 MakeFormulaIntoTree를 통해서 tree로 만들어줍니다. 이후 findsolution2를 통해 솔루션을 구합니다. 솔루션이 없는 경우 "No Solution"을 출력하고, 그렇지 않은 경우 overlap을 통해 솔루션의 중복을 제거합니다. 그리고 솔루션과 걸린 시간을 출력합니다.

2. Result

- samples in ppt

```
Write Exit to terminate the program.
Enter Equation : 3 * x + 1 = 10
Enter Min Range : 0
Enter Max Range : 100
Left Side : 3 * x + 1
Right Side : 10
Solution : 2.999997138977051
Left Side Evaluation : 9.999991416931152
Right Side Evaluation : 10.0
Elapsed sec : 0.928661

Write Exit to terminate the program.
Enter Equation : - 2 * x + 10 = x * 7 - 100
Enter Min Range : 0
Enter Max Range : 100
Left Side : - 2 * x + 10
Right Side : x * 7 - 100
Solution : 12.222221493721008
Left Side Evaluation : -14.444442987442017
Right Side Evaluation : -14.444449543952942
Elapsed sec : 1.556106

Write Exit to terminate the program.
Enter Equation : ln x - 10 = - 20 * x + 500
Enter Min Range : 0.0001
Enter Max Range : 100
Left Side : ln x - 10
Right Side : - 20 * x + 500
Solution : 25.338383919913326
Left Side Evaluation : -6.767679602992654
Right Side Evaluation : -6.767678398266526
Elapsed sec : 1.53509
```

```
Write Exit to terminate the program.
Enter Equation : ln x - 10 = - 20 * x + 500
Enter Min Range : 0
Enter Max Range : 100
Left Side : ln x - 10
Right Side : - 20 * x + 500
Solution : 25.33838376402855
Left Side Evaluation : -6.767679609144775
Right Side Evaluation : -6.767675280570984
Elapsed sec : 1.745242

Write Exit to terminate the program.
Enter Equation : ln x - 10 = - 20 * x + 500
Enter Min Range : 50
Enter Max Range : 100
Left Side : ln x - 10
Right Side : - 20 * x + 500
No Solution
Elapsed sec : 1.560107

Write Exit to terminate the program.
Enter Equation : Exit

Process finished with exit code 0
```

- samples not in ppt

```
Enter Equation : cos x = sin x
Enter Min Range : 0
Enter Max Range : 6
Left Side : cos x
Right Side : sin x
Solution : 0.7853965759277344
Left Side Evaluation : 0.7071079036962562
Right Side Evaluation : 0.7071056586750569
Solution : 3.9269943237304688
Left Side Evaluation : -0.7071043015402839
Right Side Evaluation : -0.7071092608241156
Elapsed sec : 0.77655

Write Exit to terminate the program.
Enter Equation : exp x = 2
Enter Min Range : 0
Enter Max Range : 10
Left Side : exp x
Right Side : 2
Solution : 0.6931495666503906
Left Side Evaluation : 2.000004772186584
Right Side Evaluation : 2.0
Elapsed sec : 0.668474
```

```
Write Exit to terminate the program.
Enter Equation : ln x = x
Enter Min Range : 0
Enter Max Range : 5
Left Side : ln x
Right Side : x
No Solution
Elapsed sec : 0.582413

Write Exit to terminate the program.
Enter Equation : ( x ^ 2 - 1 ) ^ 2 = 0
Enter Min Range : -2
Enter Max Range : 2
Left Side : ( x ^ 2 - 1 ) ^ 2
Right Side : 0
Solution : -0.99993896484375
Left Side Evaluation : 1.4900251713023671e-08
Right Side Evaluation : 0.0
Solution : 1.00006103515625
Left Side Evaluation : 1.4902070702427217e-08
Right Side Evaluation : 0.0
Elapsed sec : 1.364968

Write Exit to terminate the program.
Enter Equation : Exit
```

- error samples

```
Write Exit to terminate the program.
Enter Equation : =
Please input equation in correct order

Write Exit to terminate the program.
Enter Equation : 1
Please input equation in correct order

Write Exit to terminate the program.
Enter Equation :  = 0
Please input equation in correct order

Write Exit to terminate the program.
Enter Equation : ln  x = 1
Please input equation in correct order

Write Exit to terminate the program.
Enter Equation : ln x = 1
Enter Min Range : 1
Enter Max Range :
Please input range
```