Assignment 03

[2023-2] 데이터사이언스를 위한 컴퓨팅의 기초 (M3239.005600 001)

Due: 2023 년 ***11*** 월 ***3*** 일 금요일 ***23*** 시 ***59*** 분

1. Naive Word Processor [30pts]

아래 지시문에 따라 워드프로세서의 문장 입력 과정을 간단하게 구현하는 코드를 완성하시오.

Instruction:

1. 해당 코드를 통하여 구현하고자 하는 것은 워드 등의 일반적인 키보드 타이핑 환경에서의 cursor 의 움직임과 문자를 입력하고 삭제하는 과정을 가능하게 하는 class 이다.
2. 문제에서 주어진 stack 과 queue 를 활용하여 문제를 푸는 것만을 허용한다. 세부적인 함수에 대한 설명은 다음과 같다.
3. addText: cursor 기준으로 오른쪽부터 순서대로 문자를 입력한다.
4. deleteText: cursor 기준으로 왼쪽부터 순서대로 문자를 삭제한다.
5. cursorLeft: cursor 를 왼쪽으로 원하는 칸 수만큼 이동할 수 있다.
6. cursorRight: cursor 를 오른쪽으로 원하는 칸 수만큼 이동할 수 있다.
7. "deque"에 대한 설명은 다음 링크를 참조하여 활용할 수 있다. : https://docs.python.org/ko/3/library/collections.html#collections.deque

Quiz:

Q1.1 addText 함수를 완성하시오.

Q1.2 deleteText 함수를 완성하시오.

Q1.3 cursorLeft 함수를 완성하시오.

Q1.4 cursorRight 함수를 완성하시오.

Implementation:

```
# Given Example 1
TE = Naive_WordProcessor()
print(TE.addText('I love python!'))
print(TE.deleteText(5))
print(TE.addText('practice'))
print(TE.cursorLeft(8))
print(TE.cursorRight(10))
print(TE.deleteText(10))
```

```
I love python!|
I love py|
I love pypractice|
I love py|practice
I love pypractice|
I love |
```

```
# Given Example 2
TE = Naive_WordProcessor()
print(TE.addText('Do you know BTS?'))
print(TE.deleteText(4))
print(TE.addText('IU?'))
print(TE.cursorLeft(4))
print(TE.deleteText(4))
print(TE.addText('love'))
print(TE.cursorRight(6))
```

```
Do you know BTS?|
Do you know |
Do you know IU?|
Do you know| IU?
Do you | IU?
Do you love| IU?
Do you love IU?|
```

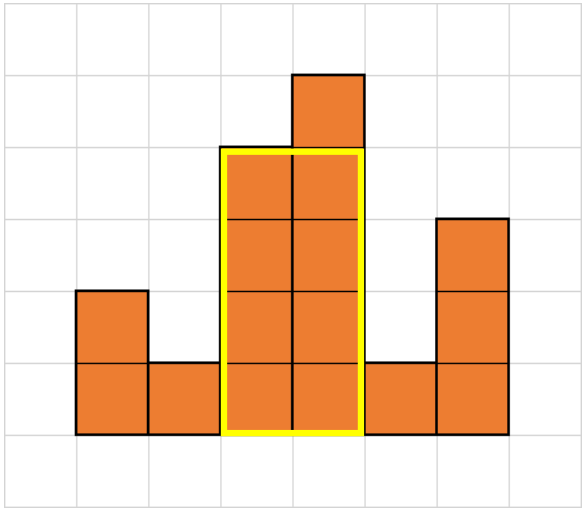
```
# Given Example 3
TE = Naive_WordProcessor()
print(TE.addText('Today is Monday!'))
print(TE.cursorLeft(11))
print(TE.deleteText(8))
print(TE.addText('Tomorrow'))
print(TE.cursorRight(20))
print(TE.addText('Terrible!'))
```

```
Today is Monday!|
Today| is Monday!
| is Monday!
Tomorrow| is Monday!
Tomorrow is Monday!|
Tomorrow is Monday!Terrible!|
```

2. 가장 넓은 박스 면적 구하기 [30pts]

아래 지시문에 따라 가장 넓은 박스 면적을 구하는 코드를 완성하시오.

김컴기는 아르바이트로 박스를 옮기는 일을 하고 있다. 모든 박스의 너비와 높이는 같다. 하지만 박스가 쌓여 있는 개수는 정해져 있지 않다. 예를 들어, 오늘 컴기가 옮겨야 하는 박스가 2 개, 1 개, 4 개, 5 개, 1 개, 3 개, 3 개로 나누어서 쌓여 있다. 쌓여 있는 더미의 박스 개수는 변하지 않는다. 또한 박스가 놓여있는 순서대로 옮겨야 한다. 박스를 옮기는 것이 지루했던 컴기는 박스를 모두 옮기고 나서 한 곳에 모아 놓았을 때 가장 면적이 큰 직사각형의 넓이가 궁금해졌다. 요즘 파이썬 공부를 하고 있는 컴기는 이것을 파이썬 프로그램으로 구현할 수 있을 것 같아 직접 코드로 구현해보고자 한다. 컴기가 옮기는 박스의 대략적인 모양의 예시는 아래의 그림과 같다.



Instruction:

1. Stack 을 활용하여 문제를 푼 경우에만 정답 처리를 할 예정이오니 주의하시기 바랍니다.
2. get_size 라는 함수는 num(box 더미의 개수)와 box(box 더미들의 높이를 옮길 순서대로 담은 list)를 input 으로 받으며, 면적의 최댓값을 저장하는 max_result 를 return 하게 된다.
3. num 과 box list 에 들어가게 될 값은 음수가 아니다.
4. Hint: stack 에 box list 의 값을 순서대로 특정 조건에 따라 pop 하고 append 하면 최대 면적을 구할 수 있습니다. 작성되어 있는 코드 부분을 참고하며 해당 코드

전과 후로 코드로 구현해야 하는 부분은 Quiz instruction 을 참고하여 작성하시면 됩니다.

Quiz:

Q2.1 stack 의 맨 위에 저장된 box list 의 값보다 현재의 height 가 더 작으면 stack 에서 box list 값을 빼내며 최대 면적을 계산한다. 그리고 stack 에 들어있는 값 중에 height 보다 큰 것들은 꺼내서 다시 한 번 면적을 계산한다.

Q2.2 모든 stack 에 대해서 Q2.1 을 반복한 후에 stack 에 여전히 남아있는 값이 있으면 면적을 계산한다.

Implementation:

```
# Given Example 1
get_size(7, [2,1,4,5,1,3,3])
```

```
8
```

```
# Given Example 2
get_size(10, [3,7,1,9,8,7,4,5,9,10])
```

```
28
```

```
# Given Example 3
get_size(0, [])
```

```
0
```

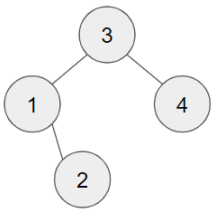
3. Binary Search Tree [20pts]

아래 지시문에 따라 Binary Search Tree 관련 함수들을 구현하시오.

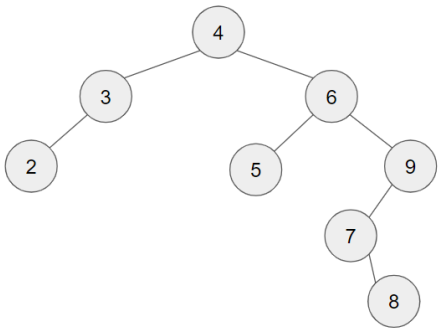
Instruction:

1. BST class 는 초기화될 때 정수로 이루어진 list 를 받아 그 list 에 들어있는 값들을 앞에서부터 순서대로 binary search tree 에 insert 하는 방식으로 tree 를 만든다.

예 1) BST([3, 1, 4, 2])



예 2) BST([4, 6, 9, 5, 7, 3, 8, 2])



2. Q3.1 과 Q3.2 모두에서 트리에 들어있지 않은 target value 등의 invalid 한 input 과 비어있는 트리는 고려하지 않는다.

3. BST class 가 구현되어 있는 코드는 수정하지 않는다.

Quiz:

Q3.1 BST object(tree)와 그 안에 들어있는 두 개의 값(targetVal1, targetVal2)을 받아, 두 값이 각각 들어있는 노드들을 모두 포함하는 subtree 중 가장 작은 subtree 의 root node 의 value 를 반환하는 branchingPoint 함수를 완성하시오. 다양한 input 케이스에 따라 반환해야 하는 값은 아래 **Implementation** 중 branchingPoint 부분 참고. (Q3.2 에서와는 달리 tree 에 연속적인 정수만 포함되어 있지 않을 수 있다.) [10pts]

Q3.2 BST object(tree)와 그 안에 들어있는 값 하나(targetVal)를 받아, 그 값이 들어있는 노드를 root node 로 하는 subtree 에 포함되어 있는 노드의 개수(descendent node 의 개수)를 반환하는 numberOfDescendents 함수를 완성하시오.
단, tree 에는 연속적인 정수만 포함되어 있고, target node 자기 자신은 descendent node 에 포함하지 않는다. [10pts]

Implementation:

```
# Given Example 1
tree1 = BST([4, 6, 9, 5, 7, 3, 8, 2])

# Input 1
print("Input 1: ", branchingPoint(tree1, 3, 9))
# Input 2
print("Input 2: ", branchingPoint(tree1, 7, 5))
# Input 3
print("Input 3: ", branchingPoint(tree1, 2, 4))
# Input 4
print("Input 4: ", branchingPoint(tree1, 6, 6))
```

```
Input 1:  4
Input 2:  6
Input 3:  4
Input 4:  6
```

```
# Given Example 2
tree2 = BST([11, 15, 13, 7, 17, 20, 18, 5, 6, 8, 10, 2, 1, 12, 4, 19, 3, 9, 14, 16])

# Input 1
print("Input 1: ", branchingPoint(tree2, 10, 17))
# Input 2
print("Input 2: ", branchingPoint(tree2, 9, 4))
# Input 3
print("Input 3: ", branchingPoint(tree2, 15, 19))
# Input 4
print("Input 4: ", branchingPoint(tree2, 5, 5))
```

```
Input 1:  11
Input 2:  7
Input 3:  15
Input 4:  5
```

```
# Given Example 3
tree3 = BST([10, 7, 15, 26, 48, 3, 50, 1])

# Input 1
print("Input 1: ", branchingPoint(tree3, 7, 50))
# Input 2
print("Input 2: ", branchingPoint(tree3, 26, 1))
# Input 3
print("Input 3: ", branchingPoint(tree3, 48, 15))
# Input 4
print("Input 4: ", branchingPoint(tree3, 10, 10))
```

```
Input 1:  10
Input 2:  10
Input 3:  15
Input 4:  10
```

```
# Given Example 1
tree1 = BST([4, 6, 9, 5, 7, 3, 8, 2])

target1, target2 = 6, 7
print("Input target node value: ", target1, "-> # of descendants: ", numberOfDescendents(tree1, target1))
print("Input target node value: ", target2, "-> # of descendants: ", numberOfDescendents(tree1, target2))
```

```
Input target node value:  6 -> # of descendants:  4
Input target node value:  7 -> # of descendants:  1
```

```
# Given Example 2
tree2 = BST([11, 15, 13, 7, 17, 20, 18, 5, 6, 8, 10, 2, 1, 12, 4, 19, 3, 9, 14, 16])

target1, target2 = 1, 15
print("Input target node value: ", target1, "-> # of descendants: ", numberOfDescendents(tree2, target1))
print("Input target node value: ", target2, "-> # of descendants: ", numberOfDescendents(tree2, target2))
```

```
Input target node value:  1 -> # of descendants:  0
Input target node value:  15 -> # of descendants:  8
```

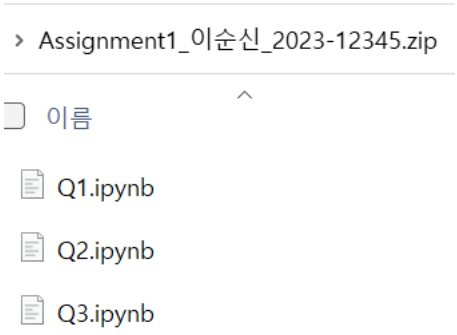
```
# Given Example 3
tree3 = BST([1, 2, 3, 4, 5, 6, 7, 8, 9])

target1, target2 = 1, 9
print("Input target node value: ", target1, "-> # of descendants: ", numberOfDescendents(tree3, target1))
print("Input target node value: ", target2, "-> # of descendants: ", numberOfDescendents(tree3, target2))
```

```
Input target node value:  1 -> # of descendants:  8
Input target node value:  9 -> # of descendants:  0
```

Cautions

1. 코드 채점 시에는, 예시로 주어진 test case 외에 추가적인 test case 들을 활용하여 채점할 예정으로 edge case 들에 대한 검증을 꼼꼼히 하고 제출하기 바랍니다.

2. 작성한 ipynb 파일을 Assignment3_학생이름_학번.zip 로 압축하여 제출하기 바랍니다. (아래 예시 참고)



3. 제출 양식 미 준수 시 채점 간 오류가 발생할 수 있습니다. 제출 양식 미 준수로 인한 오류 교정은 없습니다. 수강생 분들은 제출 양식을 반드시 확인 바랍니다.