재귀와 반복 - 정렬 (실습)

Recursion & Iteration - Sorting

삽입정렬 Insertion Sort

실습 1

이제 알고리즘을 완전히 이해했으면 insert0 재귀 함수를 다음 틀에 맞추어 구현해보자.

```
1  def insert0(x,ss):
2      if ss != []:
3          if x <= ss[0]:
4              return
5          else:
6              return
7      else:
8          return
```

실습 2

위의 재귀함수 insert0은 꼬리 재귀가 아니다. 꼬리재귀 형태로 다음 틀에 맞추어 변환하자.

```
1  def insert1(x,ss):
2      def loop(ss,left):
3          if ss != []:
4              if x <= ss[0]:
5                  return
6              else:
7                  return
8          else:
9              return
10     return loop(ss,[])
```

귀띔 1 : 여기서 loop의 둘째 파라미터 left는 x의 왼쪽에 붙일 리스트를 하나씩 아래 실행추적 사례의 밑줄 친 부분과 같이 모아간다.

귀띔 2 : x를 끼워 넣을 지점을 찾으면 left, x, ss 순으로 나열되게 리스트를 붙여서 내준다.

```
insert1(1,[2,4,5,7,8])
=> loop([2,4,5,7,8],[])
=> [] + [1] + [2,4,5,7,8] == [1, 2, 4, 5, 7, 8]

insert1(6,[2,4,5,7,8])
=> loop([2,4,5,7,8],[])
```

```
=> loop([4,5,7,8],[2])
=> loop([5,7,8],[2,4])
=> loop([7,8],[2,4,5])
=> [2,4,5] + [6] + [7,8] == [2, 4, 5, 6, 7, 8]

insert1(9,[])
=> loop([],[])
=> [9]
```

실습 3

꼬리 재귀함수 insert1을 다음 틀에 맞추어 while 반복문 버전으로 바꾸어 짜보자.

```
1  def insert(x,ss):
2      left = []
3      while ss != []:
4          if x <= ss[0]:
5              return
6          else:
7              ss, left =
8      return
```

삽입정렬 구현

이제 insert 함수를 완성하였으므로 앞에서 작성해 둔 다음 삽입정렬 재귀함수 isort0이 작동한다.

```
1  def isort0(s):
2      if s != []:
3          return insert(s[0],isort0(s[1:])
4      else:
5          return []
```

간단한 예제를 가지고 실행추적해보면 다음과 같았다.

```
isort0([3,5,4,2])
=> insert(3,isort0([5,4,2]))
=> insert(3,insert(5,isort0([4,2])))
=> insert(3,insert(5,insert(4,isort0([2]))))
=> insert(3,insert(5,insert(4,insert(2,isort0([])))))
=> insert(3,insert(5,insert(4,insert(2,[]))))
=> insert(3,insert(5,insert(4,[2])))
=> insert(3,insert(5,[2,4]))
=> insert(3,[2,4,5])
=> [2,3,4,5]
```

실습 4

이 isort0 재귀함수를 아래 틀에 맞추어 꼬리재귀 함수로 만들자. 꼬리 재귀 형태로 바꾸기 위해서는 재귀 호출 함수에 추가 파라미터를 두어 재귀호출하면서 바로 삽입해서 가지고 다니는 방법을 쓰면 된다.

```
1 def isort1(s):
2     def loop(s,ss):
3         if s != []:
4             return
5         else:
6             return
7     return loop(s,[])
```

같은 예제를 가지고 실행추적하면 다음과 같이 실행될 것이다.

```
isort1([3,5,4,2])
=> loop([3,5,4,2],[])
=> loop([5,4,2],insert(3,[])) => loop([5,4,2],[3])
=> loop([4,2],insert(5,[3])) => loop([4,2],[3,5])
=> loop([2],insert(4,[3,5])) => loop([2],[3,4,5])
=> loop([],insert(2,[3,4,5])) => loop([],[2,3,4,5])
=> [2,3,4,5]
```

실습 5

꼬리 재귀함수 isort1을 while 반복문 버전으로 짜보자.

```
1 def isort(s) :
2     ss = []
3     while s != []:
4         s, ss =
5     return
```

실습 6

isort 함수를 for 반복문 버전으로 바꾸어 짜보자.

합병정렬 Merge Sort

실습 7

위의 재귀함수를 꼬리 재귀로 다음 틀에 맞추어 변환하자.

```
1  def merge1(left,right):
2      def loop(left,right,ss):
3          if not (left == [] or right == []):
4              if left[0] <= right[0]:
5                  ss.append(  )
6                  return
7              else:
8                  ss.append(  )
9                  return
10         else:
11             return
12     return loop(left,right,[])
```

실습 8

위의 꼬리 재귀 함수를 다음 틀에 맞추어 while 반복문으로변환하자.

```
1  def merge(left,right):
2      ss = []
3      while not (left == [] or right == []):
4          if left[0] <= right[0]:
5              ss.append(  )
6
7          else:
8              ss.append(  )
9
10     return
```

버블정렬 Bubble sort

실습 9

for 반복문만 사용하여 추가공간을 전혀 사용하지 않고 수의 교환만으로 정렬을 완성하는 버블정렬 함수를 다음 틀에 맞추어 완성하시오.

```
1 def bsort(s):
2     for k in range(    ):
3         for i in range(     ):
4             if s[i] > s[i+1]:
5                 s[i], s[i+1] = s[i+1], s[i]
6     return s
```

이 실습문제를 풀기 위해서 정수범위 range를 이해해야 한다. 정수범위를 먼저 공부하고 이해한 다음 위 코드의 빈칸을 채워서 프로그램을 완성하자.

정수범위

정수범위range는 정수가 일정 간격으로 나열된 순서열이며 다음과 같이 표현한다. 순서열에 공통적으로 사용하는 연산을 모두 사용할 수 있다 (강의노트 4~5쪽의 표 참조).

range(n)은 정수 0부터 n-1까지의 간격 1의 정수범위 순서열을 나타낸다. 즉, range(5)은 0, 1, 2, 3, 4 순서열을 나타낸다.

```
>>> r = range(5)
>>> r
range(0, 5)
>>> r[1]
1
>>> r[-1]
4
>>> r[1:4]
range(1, 4)
>>> r[:]
range(0, 5)
```

다음 코드를 이해하고 실행하여 이해한 대로 결과가 나타나는지 확인하자.

```
1 for i in range(5):
2     print(i)
```

range(m,n)은 정수 m부터 n-1까지의 간격 1의 정수범위 순서열을 나타낸다. 즉, range(3,10)은 3, 4, 5, 6, 7, 8, 9 순서열을 나타낸다.

```
>>> r = range(3,10)
>>> r
range(3, 10)
>>> r[1]
4
>>> r[1:4]
```

```
range(4, 7)
```

다음 코드를 이해하고 실행하여 이해한 대로 결과가 나타나는지 확인하자.

```
1 for i in range(3,10):
2     print(i)
```

정수범위 시퀀스의 간격을 1이 아닌 정수로 지정하려면 셋째 인수를 추가한다. 즉, range(m,n,k)는 정수 m부터 n-1까지의 간격 k의 정수범위 순서열을 나타낸다. range(3,11,2)은 3, 5, 7, 9를 나타내고, range(10,3,-1)은 10, 9, 8, 7, 6, 5, 4를 나타낸다.

```
>>> r = range(3,11,2)
>>> r[2]
7
>>> r[2:5]
range(7, 11, 2)
>>> s = range(10,3,-1)
>>> s[4]
6
>>> s[4:6]
range(6, 4, -1)
```

다음 코드를 이해하고 실행하여 이해한 대로 결과가 나타나는지 확인하자.

```
1 for i in range(10,3,-3):
2     for j in range(3,11,3):
3         print(i,j)
```