자료구조 12주차 실습

Sorting

감성인공지능연구실
방윤석 임희수

인하대학교
INHA UNIVERSITY

Notice

인하대학교 INHA UNIVERSITY

Quiz #3

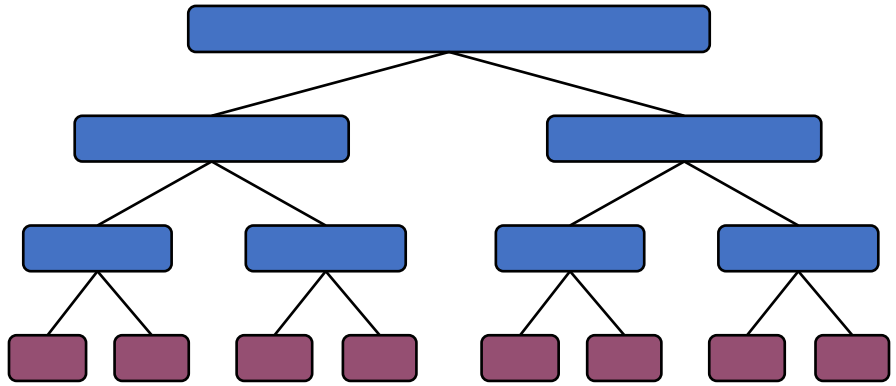
- 다음주 금요일(5/30) 마지막 퀴즈입니다.

Sorting

Analysis of Merge-Sort

- The height h of the merge-sort tree is $O(\log n)$
  - at each recursive call we divide in half the sequence,
- The overall amount or work done at the nodes of depth i is $O(n)$
  - we partition and merge 2^i sequences of size $n/2^i$
  - we make 2^{i+1} recursive calls
- Thus, the total running time of merge-sort is $O(n \log n)$

depth	#seqs	size
0	1	n
1	2	$n/2$
i	2^i	$n/2^i$
…	…	…



Exercise #1

인하대학교 INHA UNIVERSITY

MergeSort

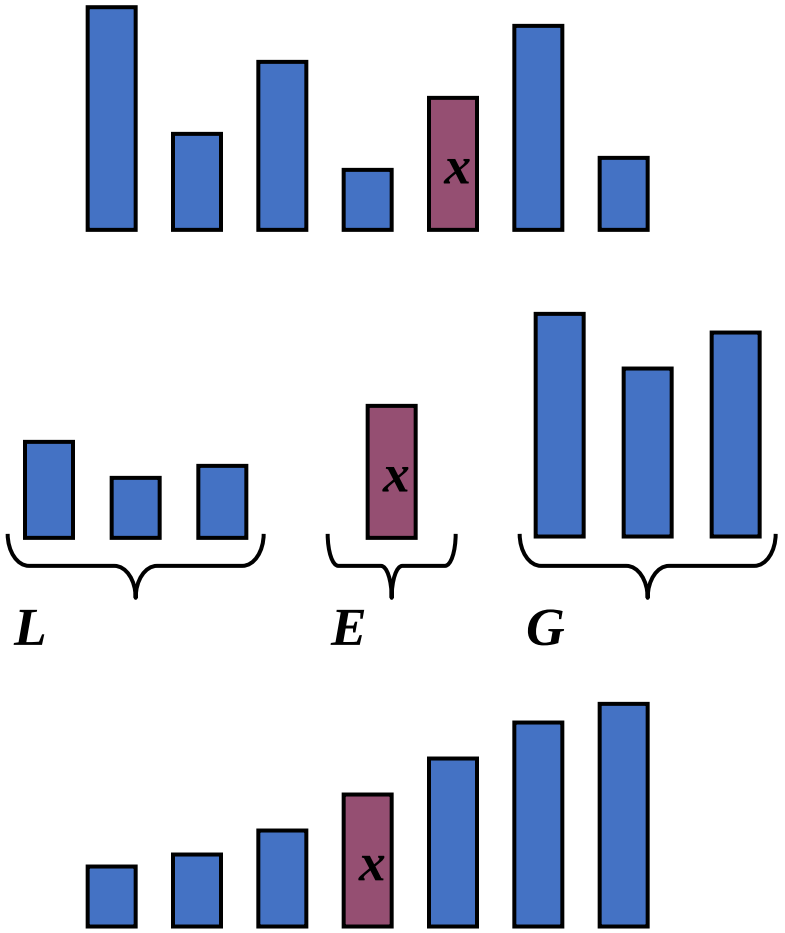
- 데이터를 절반씩 쪼갠 뒤, 하나로 합치면서 정렬하는 방식
- 시간 복잡도는 O(n log n)
- 공간 복잡도는 O(n) (추가적인 배열이 필요)

힌트

- 데이터가 하나만 있는 경우는 정렬이 되어 있다고 본다.

Quick-Sort

- Quick-sort is a randomized sorting algorithm based on the divide-and-conquer paradigm:
  - Divide: pick a random element x (called pivot) and partition S into
    - L elements less than x
    - E elements equal x
    - G elements greater than x
  - Conquer: sort L and G
  - Combine: join L, E and G



Exercise #2

인하대학교 INHA UNIVERSITY

QuickSort

- 데이터를 하나의 기준점을 중심으로 큰 값, 작은 값을 나눈다. 이후 나뉜 부분에 대해 다시 퀵 정렬을 진행해서 모두 정렬시킨다.
- 시간 복잡도는 O(n log n)
- 공간 복잡도는 O(n) (추가적인 배열이 필요)

- Pivot의 선택은 프로그래머의 선택에 따른다. 여러 방식이 존재

힌트

- 수업에서는 데이터의 가장 뒤의 값을 기준(pivot)으로 설정한다.

In-Place Quick-Sort

- Quick-sort can be implemented to run in-place
- In the partition step, we use replace operations to rearrange the elements of the input sequence such that
  - the elements less than the pivot have rank less than h
  - the elements equal to the pivot have rank between h and k
  - the elements greater than the pivot have rank greater than k
- The recursive calls consider
  - elements with rank less than h
  - elements with rank greater than k

Algorithm *inPlaceQuickSort*(S, l, r)
  Input sequence S, ranks l and r
  Output sequence S with the elements of rank between l and r rearranged in increasing order
  if $l \geq r$
    return
  $i \leftarrow$ a random integer between l and r
  $x \leftarrow S.elemAtRank(i)$
  $(h, k) \leftarrow inPlacePartition(x)$
  inPlaceQuickSort$(S, l, h-1)$
  inPlaceQuickSort$(S, k+1, r)$

Exercise #3

인하대학교 INHA UNIVERSITY

QuickSort (In place)

- 데이터를 하나의 기준점을 중심으로 큰 값, 작은 값을 나눈다. 이후 나뉜 부분에 대해 다시 퀵 정렬을 진행해서 모두 정렬시킨다.
- 이번에는 추가적인 배열을 생성하지 않고, 주어진 배열 1개만 사용한다.
- 시간 복잡도는 O(n log n)
- 공간 복잡도는 O(log n) (재귀 호출에 의한 공간 사용. 추가적인 배열은 없다.)

힌트

- Index를 활용하면 된다.

Exercise #4

인하대학교 INHA UNIVERSITY

TreeSort

- 입력 데이터를 이진 탐색 트리에 삽입한 뒤, 이진 탐색 트리의 특징을 이용해서 정렬한다.
- 시간 복잡도는 O(n log n)
- 공간 복잡도는 O(n) (tree 구성에 추가 공간 필요)

힌트

- 지금까지 배운 탐색 중 하나를 사용하면 된다.

Problem #1

인하대학교 INHA UNIVERSITY

정수 배열 nums 와 nums의 길이 n이 주어졌을 때, 배열에 있는 숫자들을 적절히 이어 붙여 **가장 큰 수**를 만들고, 그 결과를 반환하는 함수를 작성하시오.

- n은 1 이상, 100 이하
- 각 원소는 0 이상 100 이하
- 결과 값이 '0000' 처럼 0이 앞에 여러 개 붙을 경우, 0 으로 출력해야 한다.
- largestNumber는 O(n log n) 의 시간복잡도를 만족하여라.
- getDigits()는 O(1)로 실행된다고 가정한다.
- 정렬은 **In-place quick sort** 알고리즘을 사용하여라

메소드 설명

- largestNumber() : 주어진 수들을 이어 붙여 만들 수 있는 가장 큰 수를 문자열로 반환

Problem #1

인하대학교 INHA UNIVERSITY

예시입력

```
n = 5
arr = array.array('h', [3, 30, 34, 5, 9])
largestNumber = LargestNumber(arr, n)
print(largestNumber.largestNumber())

n = 3
arr = array.array('h', [0, 0, 0])
largestNumber = LargestNumber(arr, n)
print(largestNumber.largestNumber())
```

예시출력

```
9534330
0
```

Problem #2

인하대학교 INHA UNIVERSITY

여러개의 구간들이 주어진다. 이 구간들을 병합하여 출력하는 함수를 작성하시오

- 정수쌍은 10,000개 이하로 들어온다.
- 각 구간은 0이상 10,000 이하이다.
- print()에서 나오는 구간은 start를 기준으로 정렬되어 있어야 하며, 어떠한 겹침이 없어야 한다.
- add()는 O(1)의 시간 복잡도를 가진다.
- merge()는 O(n log n) 의 시간 복잡도를 가진다.
- 정렬은 **merge sort** 알고리즘을 이용하여라

메소드 설명

- add(start, end) : [start, end] 로 구성된 정수쌍을 추가한다.
- merge() : 겹치는 구간들이 병합된 새로운 구간 리스트를 만든다.
- print() : 객체가 가지고 있는 구간들을 출력한다.

Problem #2

인하대학교 INHA UNIVERSITY

예시입력

```
merger = IntervalMerger()
merger.add(1, 3)
merger.add(2, 6)
merger.add(8, 10)
merger.add(15, 18)
merger.intervalMerge()
merger.print()

merger2 = IntervalMerger()
merger2.add(1, 4)
merger2.add(4, 5)
merger2.intervalMerge()
merger2.print()
```

예시출력

```
[1, 6] [8, 10] [15, 18]
[1, 5]
```