Homework #1

Due: 10/4

HW1 instructions

이번 HW1 은 이론 6 문항과, 하나의 coding 및 analysis 실습 문항으로 이루어져 있습니다. 이론 풀이 및 analysis 를 담은 하나의 pdf 파일, 마지막 문제의 Python 파일 2 개를 .zip 으로 압축하여 제출하여 주세요. 모든 코드는 Python(.py extension)으로 작성하시기 바랍니다.

파일 1: 학번_이름.pdf
파일 2: 학번_이름_3way.py → **HW1_학번_이름.zip** 압축 후 제출 (예 : HW1_202200000_홍길동.zip)
파일 3: 학번_이름_kway.py

(Total 100 points)

1. **(10 points)** 아래는 selection sort 에 대한 코드이다. 관련하여 질문에 답하시오.

```
1  def selection_sort(A):
2      for i in range(len(A)-1):
3          min_idx = i
4          for j in range(i+1, len(A)):
5              if A[j] < A[min_idx]:
6                  min_idx = j
7          # Swaps elements occupying min_idx and i in place
8          swap(min_idx, i)
```

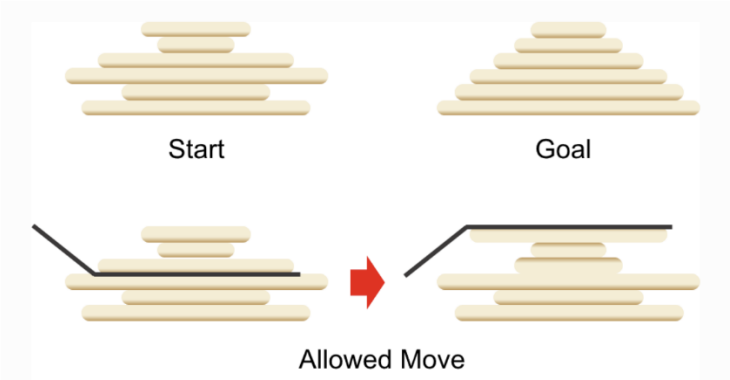
   a. What is the loop invariant associated with the **outer for** loop?

   b. What is the loop invariant associated with the **inner for** loop?

   c. Fill in the following information for the **outer for** loop.
      i. Inductive hypothesis
      ii. Base case
      iii. Inductive step
      iv. Conclusion

   d. Fill in the following information for the **inner for** loop.
      i. Inductive hypothesis
      ii. Base case
      iii. Inductive step

2. **(15 points)** 현풍건설은 디지스트에 새로운 기숙사를 건설하기로 하였다. 설계도에 따르면, 아래 Goal 과 같이 작은 블록이 위에 위치해야 한다. 하지만 간밤에 다녀간 태풍으로, 블록이 뒤죽박죽 섞여버렸다. 디지스트 공학도들은 힘을 합쳐 기존의 형태로 복구하려 한다. 이 때, 사용할 장비는 인접한 두 블록 사이에 집어넣어 뒤집는 동작 밖에 수행할 수 없다는 제약이 존재한다. (지면과 블록사이도 가능)
   컴퓨터 알고리즘 재수강생인 훈승이는 이번 복구 작업의 시간 복잡도를 분석하려고 한다.
   n 개의 Block 에 대해 최악의 경우 (Worst-Case), 몇 번의 동작 수행이 필요한가? 그 풀이 과정과 답을 적으시오.



   - **Worst-Case Count**:
   - **Why?**

3. **(12 points)** 아래 재귀 관계식들을 각각 $T(n) = \Theta(n)$ 형태로 나타내시오.

   a. $T(n) = 3T(\frac{n}{2}) + 1$

   b. $T(n) = 3T(\frac{n}{2}) + n$

   c. $T(n) = 3T(\frac{n}{2}) + 3^n$

   d. $T(n) = 8T\left(\frac{n}{2}\right) + n^3$

   e. $T(n) = 3T(n-1) + 1$

4. **(10 points)** 아래 재귀식으로 표현된 시간복잡도 T(n)에 대해 substitution method 를 활용해 big-O notation 으로 표현하시오. (*example form* : T(n) = $O(n)$)

$$T(n) = T(n/2) + T(n/4) + T(n/8) + T(n/16) + n$$

Where $T(1) = 1$

5. **(15 points)** 배열 A에는 1~n 까지의 정수가 주어져 있다. 이때 i < j 에 대해 A[i] > A[j] 인 쌍의 개수를 찾는 알고리즘을 설계하고자 한다. 가령 A={2,3,6,4,1,5}일 때, 해당 조건을 만족하는 쌍은 (2,1), (3,1), (6,4), (6,1), (6,5), (4,1)으로 총6개이다. 해당 알고리즘은 $O(n^2)$에 완성할수 있는 직관적인 알고리즘이 존재하지만 $O(n \log n)$의 시간복잡도를 가지는 알고리즘으로 구현할 수 있다. $O(n \log n)$ 의 pseudo code를 완성하고 알고리즘의 시간복잡도를 재귀관계식(Recurrence relation)으로 표현하시오. [Hint : Divide and Conquer]

```
Function Find-Inversion-Pairs(A, low, high)
//////////////////////
/// design algorithm!///
//////////////////////
Return [Number of Inversion Pairs]
```

a. 위 알고리즘의 pseudo code 를 작성하시오.

b. 위 알고리즘의 재귀관계식을 표현하시오.

6. **(8 points)** Radix sort 는 대표적인 선형시간 정렬알고리즘(Linear Time Sorting) 중 하나이다. Radix sort 의 subroutine 인 bucket sort 에 대해, 배열 A = [byte, pits, bits, pins] 가 주어졌을 때 아래의 질문에 답하시오. (a to z ascending order, 풀이과정 필요 X)

   a. Bucket Sort 의 첫번째 호출 후 배열의 상태는?

   b. Bucket Sort 의 두번째 호출 후 배열의 상태는?

   c. Bucket Sort 의 세번째 호출 후 배열의 상태는?

   d. Bucket Sort 의 네번째 호출 후 배열의 상태는?

Coding assignment instructions

This assignment requires you to submit programs in **Python** that you have written yourself. You cannot use existing library and methods that might solve the problem (e.g., NewArr = sorted(Arr)) .

Your programs will be tested on randomly generated input files, some of which will be very large. Make sure your output is EXACTLY the right format as it will be compared with a text comparison program.

The line below is an example of the terminal input that will be used to test your code.

```
python3 202200000_홍길동_kway.py < input.txt > output.txt
```

I highly recommend you use sys.stdin to read input lines
ex) for line in sys.stdin:

7. **(30 points)** (Merge Sort) 첨부된 **2way.py** 파일은 lecture3 시간에 배운 merge sort 를 python 으로 구현한 예시 코드이다. 해당 코드의 핵심 logic 은 아래의 3 가지 step 으로 abstract 할 수 있다.

   1) **분할(Divide)**: 배열을 1/2 로 분할한다. 분할한 배열의 원소의 개수는 각각 n/2 이다.

   2) **정복(Conquer)**: 분할된 배열을 각각 따로 sorting 한다. 분할한 배열에 원소의 크기가 2 개 이상이면 재귀적으로 divide and conquer 알고리즘을 적용한다.

   3) **결합(Combine)**: 정렬된 각 배열들을 다시 하나의 배열로 합병하여 정렬한다.

a. **(5 points)** 기존의 Merge Sort 를 응용하여, 3 등분 후 다시 merge 하는 3-Way Merge Sort 를 구현하시오. 아래 instruction 을 따라 3way.py 파일의 mergesort3 함수의 part1, part2, part3, main 함수 부분을 각각 채워 python 파일을 완성하시오. (파일 형식: 학번_이름_3way.py)

Input format:
첫번째 줄: Input array 의 원소 개수, n (1 이상의 정수)
그 밑의 n 개의 줄: Input array 의 순서대로 각 원소들 (정수)

sample input
8
36
12
54
97
94
56
86
9
(이 경우, Input Array = [36, 12, 54, 97, 94, 56, 86, 9])

Output format:
첫번째 줄: 파일 안의 isSorted() 함수를 이용하여, 결과를 출력하시오.

sample output
Well Sorted!

유의사항: input & ouput parsing 을 위한 추가적인 구현이 필요하기에, 그에 대한 추가적인 함수 구현은 허용한다. 또한, mergesort3 함수의 구현 순서는 굳이 part 1,2,3 을 따르지 않아도 된다. 하지만, **if unit == 0:** 까지의 코드는 수정할 수 없고, 필수적으로 unit 을 함수 구현에 사용해야 한다. merge 또한 따로 구현할 필요 없이 파일 안의 함수를 사용한다.

b. **(15 points)** 좀 더 확장하여, k 등분 후 다시 merge 하는 k-Way Merge Sort 를 구현하시오. 아래 instruction 을 따라 kway.py 파일의 mergesortk 함수의 part1, part2, part3, main 함수 부분을 각각 채워 python 파일을 완성하시오.
(파일 형식: 학번_이름_kway.py)

Input format:
첫번째 줄: n k (k 는 2 이상의 정수)
그 밑의 n 개의 줄: Input array 의 순서대로 각 원소들 (정수)

sample input
8 4
36
12
54
97
94
56
86
9
(이 경우, Input Array = [36, 12, 54, 97, 94, 56, 86, 9]에 대해 4-way merge sort 수행)

Output format:
첫번째 줄: 파일 안의 isSorted() 함수를 이용하여, 결과를 출력하시오.

sample output
Well Sorted!

유의사항: input & ouput parsing 을 위한 추가적인 구현이 필요하기에, 그에 대한 추가적인 함수 구현은 허용한다. 또한, mergesortK 함수의 구현 순서는 굳이 part 1,2,3 을 따르지 않아도 된다. 하지만, **if unit == 0:** 까지의 코드는 수정할 수 없고, 필수적으로 unit 을 함수 구현에 사용해야 한다. merge 또한 따로 구현할 필요 없이 파일 안의 함수를 사용한다.

c. **(10 points)** 2-way, 3-way, k-way 각각의 time complexity 를 Big-O notation 으로 표현하시오. 풀이 과정과 함께 3 가지 알고리즘의 time complexity 를 비교하시오.