알고리즘 **1 / Algorithm 1 (COMP319)**

**Inuk Jung (inukjung@gmail.com)**

**Assignment #1 (17th March 2022)**

**“Insertion sort with binary search”**

Due Date: 31st March, 2022 (Thursday) Midnight (late submission not allowed)

Submission: Upload a single **.zip** file to LMS

Name:

Student ID#:

Email:

Score: \_\_\_\_\_\_ / 100 (total)

**What to submit:**

* The completed “insertion\_sort\_bsearch\_skel.py” code
* The output of the script, where n=1000 (line 77) with random.seed(1234) (line 79)
* The answers to questions below

**How to submit:**

* Upload a single **.zip** file to the new LMS (과제 및 평가), that includes 1) the code and 2) the PDF file with the answers (한글/영어로 작성 가능)

**Notifications:**

* C/C++ and python languages are allowed. The skeleton code is in python. C/C++ users, need to write their own code by referring to the skeleton code. Let me know if you have difficulties.
* Plagiarism will be checked and if detected **F** will be given without any notification.

**Tasks:**

삽입 정렬(insertion sort)는 **O(n2)** 임을 확인했다. 그러나 이는 n이 큰 수 일 때 수행속도가 매우 느리다. 주어진 코드는 n이 100,000 보다 클때 매우 느린, 삽입 정렬의 정석 방법을 구현한 것이다.

j(key) 왼쪽에 있는 subarray가 정렬이 되어 있는 상태이므로 이진 탐색을 적용하여 j가 삽입 되어야할 적절한 위치를 찾을 수 있다. 즉, 이진삽입정렬을 하는 것이 아닌, 기존 insertion sort 루틴 중 j가 삽입돼야 할 위치를 찾는데 이진탐색(binary search)을 활용 하도록 수정하는 것이 목표이다.

1. (20 points) 코드를 분석하고, 이진 탐색 함수를 완성하시오, ###으로 표시된 힌트들은 코드가 수정되거나 추가되어야 할 부분을 나타낸 것이다.23,27번째 줄에 비어있는함수 파라미터들도 작성하여야 한다.
2. (40 points) 이진 탐색을 완성 후, 48~58 줄을 주석처리 돼야합니다. 그 후 65번째 줄부터 주석을 해제하고, 이진탐색 함수를 이용하여 key가 배치 되어야 할 올바른 인덱스 i를 찾는다.

예를 들어 아래와 같은 배열 A의 경우 키(또는 j) = 5일 때 "binary\_search" 함수는 2를 반환한다.   
  
index=[0,1,**2**,3,4,5,6,7]  
A = [1,3,8,21,30,**5**,9,11]  
  
그러나 65번째 줄로는 충분하지 않으며, 이진 탐색 버전의 삽입 정렬 알고리즘을 완료하기 위해 추가 코드(약 5줄 정도)가 필요하다.

1. (20 points) Runtime cost T(n)은 주어진 코드에서 "tn" 변수로 계산되었는데, 이진 검색 버전 삽입 정령에서도 의 T(n)을 측정하여라.
2. (20 points) 코드를 완성 후 아래의 질문에 대해 답하시오.

**Questions**:

Q1. n=1000이고 random.seed(1234)일 때 기존 삽입 정렬 과 이진 탐색 버전 삽입정렬 의 런타임 T(n)은 어떻게 되는가?

A1. 기존 삽입 정렬의 경우 T(1000) = 995636, 이진탐색 버전 삽입정렬의 경우 T(1000) = 749117+52(binary search T(n))이 소요된다.

Q2. 이진 탐색 버전이 기존 삽입 정렬보다 점근적으로 빠른지 느린지에 대한 간단한 증명을 하십시오. 이진 검색이 원래 삽입 정렬보다 빠르거나 느리다고 생각하는 경우 자세한 근거를 기재해주십시오.

A2. 순차 검색에 의한 비교에서 이진 탐색을 통한 비교로 바뀌어 O(N^2)에서 O(N Log N)으로 개선되었다.

Q3. 이진탐색 버전의 the asymptotic runtime O(g(n)) 는 무엇인가?

A3. O(N Log N)