알고리즘-융합-AI **중간고사** **April, 20, 2022**

학번:_____ **이름:** _____

1. [12.5] 다음의 물음에 답하라. 특별한 언급이 없으면 부분 점수 없음.
   A. [0.5] A = "Algorithm"에 대해, print(A[::2])를 한 결과는? _____
   B. [1] $f(n) = 4^{\log_2 n} + 6n^2/\log n + 9n$를 Big-O 기호로 나타내면 $f(n) =$ O(_____)이다.
   C. [1] 1보다 큰 양의 증가 함수 $f(n)$과 $g(n)$에 대해, $f(n) + g(n) = O(\min(f(n), g(n)))$이다. 맞으면 O, 틀리면 X (O / X)
   D. [1] T(n) = T(n/2) + cn을 전개했다. T(1) = c이다. Big-O로 표기하면 O(_____)이다.
   E. [1] T(n) = 2T(n/2) + cn을 전개했다. T(1) = c이다. Big-O로 표기하면 O(_____)이다
   F. [1] 16개의 서로 다른 수 중에서 최소 값과 최대 값을 찾기 위해 _____번의 비교면 충분하다. (비교 횟수는 최대한 작아야 한다.)
   G. [0.5] n개의 값들에 대한 quick select와 quick sort의 최악의 경우의 수행시간의 점화식은 서로 같다 (O / X)
   H. [0.5] 강의 시간에 배운 insertion, heap, merge, radix 정렬 알고리즘 중에서 in-place 알고리즘은 무엇인가? 모두 골라, 동그라미로 표시하라.
   I. [1] 임의의 n개의 값이 저장된 리스트 A를 오름차순으로 정렬했다. 단, 비교기반 정렬 알고리즘을 사용했다. 이 정렬 시간을 M(n)이라 하자. 정렬된 A에서 특정 값이 있는지 알고 싶어, 탐색을 B(n) 시간에 수행했다. 두 가지 모두 가장 빠른 알고리즘을 사용한다면, M(n)/B(n)을 Big-O로 표기하면 O(_____)이다.
   J. [0.5] 강의시간에 배운 insertion, quick, merge, bubble 정렬 알고리즘 중에서 최악의 경우와 최선의 경우의 수행시간이 다른 알고리즘을 모두 골라, 동그라미로 표시하라. Quick 정렬의 경우에는 첫 번째 값을 피봇으로 정한다고 가정한다.
   K. [1] n개의 값이 이미 **내림차순**으로 정렬된 리스트 A를 오름차순으로 정렬하는 insertion 정렬 알고리즘을 적용했다. 단, 각 A[i]의 위치를 A[0], … , A[i-1]에 대한 이진탐색으로 찾는 방식으로 처리한다면 이 알고리즘의 비교회수와 이동(교환)회수를 각각 Big-O로 표기하면? (단, 횟수가 0이라면 O(1)으로 표기한다.)
      비교회수: O(_____), 이동(교환)회수: O(_____)
   L. [1] n개의 값이 이미 **오름차순**으로 정렬된 리스트 A에 다시 (오름차순으로 정렬하는) bubble 정렬 알고리즘을 적용했다. 이 정렬 알고리즘의 비교회수와 이동(교환)회수를 각각 Big-O로 표기하면? (단, 횟수가 0이라면 O(1)으로 표기한다.)
      비교회수: O(_____), 이동(교환)회수: O(_____)
   M. [0.5] A = [2, 6, 5, 1, 3, 0]은 힙(heap)이다. (O / X)
   N. [1] n개의 값으로 구성된 heap의 높이의 하한이 Ω(_____)이고, n개의 값을 비교 기반으로 정렬하는 문제의 비교 횟수의 하한이 Ω(_____)이다.
   O. [1] n개의 값이 리스트 A에 오름차순으로 정렬되어 있고, m개의 값이 리스트 B에 내림차순으로 정렬되어 있다. 두 리스트에 저장된 값을 새로운 리스트 C에 오름차순으로 저장하려고 한다. 가장 빠르게 한다면 O(_____) 시간에 가능하다.

2. [4.5] 오른쪽 파이썬 코드에 대한 질문에 답하라.

A. [1] (1)의 print문의 출력 결과는? _____

B. [1] (2)의 print문의 출력 결과는? (대괄호 필요함) _____

C. [1] f2의 수행시간 T(n) = O(_____)이다.

D. [1.5] C번의 수행시간이 되는 이유를 두 줄 이내로 설명하라.

```
def f2(A, n):  # n 개 값이 A 에 저장됨
    s = 0      # n 은 항상 짝수임
    for _ in range(len(A)//2):
        s += A.pop(0)
        A.append(s)
    return s
A = [5,4,3,2,1,0]
print(f2(A,6))        # (1)
print(A)              # (2)
```

3. [5] 오른쪽 코드에 대해, 다음 질문에 답하라.

A. [2] f3([7,6,5,4,3,2,1,0],8))의 출력 결과는? _____

B. [1] f3(A, n)의 수행시간 T(n)을 Big-O로 표기하라. T(n) = _____

C. [2] B번의 수행시간이 되는 이유를 두 줄 이내로 설명하라.

```
def f3(A, n): # n 개 값이 A 에 저장됨
    s, j = 0, 1
    while j < n:
        s += A[j-1]
        j *= 2
    return s
```

4. [4.5] 아래 코드를 보고 물음에 답하라.

A. [1] f4(2, 5)는 어떤 값을 리턴하는가?

B. [1] f4(a, n)의 수행시간 T(n)을 **점화식**으로 표현하면?

C. [2.5] 이 점화식을 **전개한 후**, Big-O 기호로 표기하라. (가정이 필요하면 직접 언급할 것. T(1)도 언급할 것. 점화식 전개 없으면 0점.)

```
def f4(a, n):
    if n == 1: return a
    if n%2 == 0:
        return f4(a*a, n//2)
    else:
        return f4(a*a, n//2)*a
```

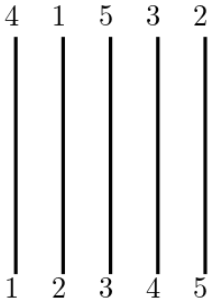
5. [4] n자리 두 정수를 곱셈하는 Karatsuba 알고리즘의 점화식은 아래와 같다고 하자.

$$T(n) = aT\left(\frac{n}{b}\right) + n, \qquad T(1) = 1$$

   A. [1] a 값과 b 값은 얼마인가? a = _____, b = _____ (부분점수 없음)

   B. [4] 점화식을 전개한 후, Big-O 기호로 표기하라. (필요한 가정이 있다면 직접 언급할 것. 점화식 전개 없으면 0점. 부분점수 있음.)

6. [5] 오른쪽 그림처럼 사다리타기 게임을 해서 다섯 개의 값을 정렬하려고 한다.



   A. [1.5] 인접한 두 세로 줄을 연결하는 가로 선분을 그려 정렬하려고 한다. **최소** 몇 개의 가로 줄이 필요한가? _____

   B. [2] A번에서 답한 개수만큼 가로 선분을 왼쪽 그림에 직접 그려라.

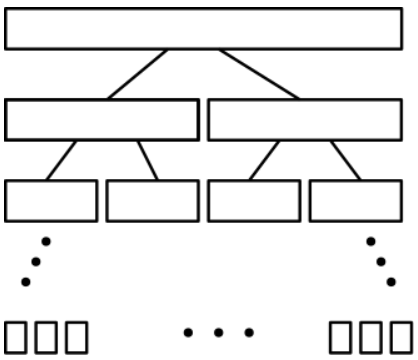
   C. [1.5] 만약, 인접하지 않은 두 개의 세로 줄도 가로 선분으로 연결할 수 있다면, 최소 몇 개의 가로 선분이 필요한가? _____

7. [6] n개의 값을 merge 정렬하려고 한다. 다음 질문에 답하라.

   A. [1.5] 2-way merge 정렬 알고리즘은 절반씩 재귀적으로 나눠 정렬하는 알고리즘이다. 만약, 분할 과정에서 $\sqrt{n}$ 개 이하가 되면 재귀를 멈추고 insertion 정렬을 한다고 하자. 나머지 합병 과정은 동일하다. 이 경우의 수행 시간 T(n)의 점화식은?

      T(n) = _____, T($\sqrt{n}$) = _____

   B. [3] 이 점화식의 재귀 과정을 트리 모양으로 그려보자. 아래 그림에 점화식 전개 과정을 자유롭게 적어보자. (단, n = 2^k이라 가정한다. 부분 점수 있음.)



   C. [1.5] 이 점화식의 전개한 값을 Big-O로 표기하면? T(n) = O(_____)

8. [10] 다음은 quick 정렬 알고리즘의 python 코드이다.

```
def quickSort(A):
    if len(A) == 1: return A
    else:
        pivot = A[0]                                          # (1)
        S, M, L = [], [], []  # S, M, L은 각각 pivot보다 작거나 같거나 큰 수로 구성
        for x in A:
            if x < pivot: _____________________________       # (2)
            elif x > pivot: ___________________________       # (3)
            else: _____________________________               # (4)
        return ____________________________________________   # (5)
```

A. [2] 위 코드의 빈 칸 (2) ~ (4)를 채우세요. (2),(3),(4)는 모두 맞아야 1점. (5)는 1점

B. [1] 위 알고리즘은 stable/unstable 하며, in-place/not-in-place하다. 알맞은 단어에 각각 동그라미! (0.5점씩)

C. [2] 위의 pivot을 정하는 단계 (1)을 pivot = MoM(A, len(A)//3)으로 정했다고 하자. MoM(A, k)는 A의 값 중에서 k번째로 작은 값을 찾는 median of medians 알고리즘이다. 그러면 이 quick sort 알고리즘의 최악의 경우의 비교횟수 W(n)의 점화식은 다음과 같이 표기할 수 있다. 두 빈 칸을 채워라. (부분점수 없음)

W(n) = W(_____________) + W(_____________) + _____________

D. [5] C번에서 세운 점화식을 풀어 Big-O로 표기하라. (**힌트**: 귀납법!)

9. [6.5] n개의 **서로 다른 양의 정수**가 저장된 리스트 A에서 가장 큰 수와 두 번째로 큰 수를 분할정복 (Divide and Conquer) 방법으로 구하고 싶다. 이 두 수를 구하는 함수를 find_two_max이라 하자. find_two_max(A, a, b)는 리스트 A에서 가장 큰 수와 두 번째로 큰 수를 [first max, second max]처럼 리스트로 묶어 리턴한다. (n >= 2이라고 가정)

```
def find_two_max(A, a, b):
    if a == b: return [A[____], -1] # 두 번째로 큰 수는 없으므로 -1로 처리
    L = find_two_max(A, a, (a+b)//2)
    R = find_two_max(A, (a+b)//2+1, b)
    M = [] # 정답을 저장하기 위한 리스트
    if L[0] > R[0]:
        ____________________________________________________
        ____________________________________________________
    else:
        ____________________________________________________
        ____________________________________________________
    return M
```

A. [3] 빈 칸과 빈 줄을 채워 코드를 완성하라. (두 수에 대한 max 사용 가능. 부분점수 있음.)

B. [1] 위 코드의 수행시간 T(n)의 점화식은? T(n) = _____

C. [2.5] 이 점화식을 전개한 후, Big-O 기호로 표시하세요. (가정이 필요하면 언급하면 됨. 전개과정 없으면 0점)

10. [3.5] n번째 피보나치 수열 값을 계산하는 재귀 함수 fibo(n)이 오른쪽과 같다. 단, fibo(0) = 0, fibo(1) = 1이다.

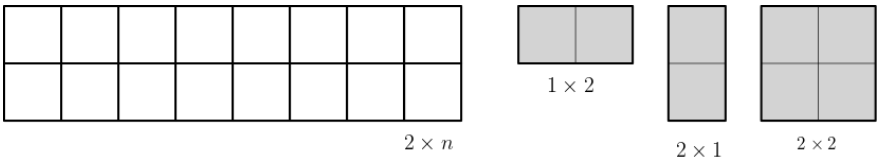
```
def fibo(n):
    if n <= 1: return n
    return fibo(n-2) + fibo(n-1)
```

A. [2] fibo(4)를 호출하면, fibo의 재귀호출 순서가 다음과 같다. 빈 칸을 채워라. (부분점수 없으니 신중하게 작성하라)

fibo(4) -> fibo(2) -> _____ -> fibo(1) -> _____ -> _____ ->
fibo(2) -> fibo(0) -> _____

B. [1.5] fibo(5)를 호출하면 fibo(2)가 몇 번 호출되는가? _____ 번

11. [7] 아래 그림처럼 2xn (이 예에서는 n = 8임) 빈 칸이 있고, 이를 그림에 나타난 세 가지 기본 타일로 채우고 싶다. 임의의 n >= 1에 대해, 2xn 빈 칸을 채울 수 있는 경우의 수를 A[n]이라 하자. A[n]의 점화식은 어떻게 되나? 초기 값도 함께 언급해야 한다.



A. [1] A[1] = _____, A[2] = _____

B. [2] A[n] = _____

C. [3] A[n]을 계산하는 python-like 코드를 아래에 작성하라.

```
def compute_A(n): # return A[n]
```

D. [1] C번의 코드의 수행 시간 T(n)을 Big-O로 표기하라. (C번에서 제안한 코드의 효율성과 관계 없음) T(n) = O(_____)

12. [1] 지금까지의 알고리즘 강의에 대해 느낀 점, 아쉬운 점, 개선할 점 등을 솔직히 써 주면 남은 강의에 반영하도록 노력하겠습니다. (쓰기만 하면 1점!)