알고리즘(수12-신찬수) **중간고사** **26, October, 2022**

학번:_____ **이름:** _____

1. [12] 다음의 물음에 답하라.
   A. [1] print("-".join(["Al","gori","thm"]))의 결과는? _____
   B. [1.5] $f(n) = n/\log n + n^{0.5}$를 Big-O 기호로 나타내면 $f(n) = O($_____$)$이다.
   C. [1] 1보다 큰 양의 증가 함수 $f(n)$에 대해, $f(n) = O(\sqrt{f(n)})$이다. 맞으면 O, 틀리면 X (O / X)
   D. [1] 16개의 서로 다른 수 중에서 가장 작은 값과 두 번째로 작은 값을 찾기 위해서는 _____번의 비교면 충분하다. (가장 작은 비교회수를 적어야 한다)
   E. [1] 3개의 서로 다른 수가 있다. 오름차순으로 정렬하기 위해선 최소 3번의 비교가 반드시 필요한 경우가 있다. (O / X)
   F. [1] n개의 값들에 대한 quick select 알고리즘의 최악의 경우의 수행시간의 점화식 T(n)의 점화식은 T(n) = _____ 이다. (단, T(1) = 1이다.)
   G. [1] 위의 점화식을 전개한 후, Big-O로 표기하면 O(_____)이다.
   H. [1] 강의에서 설명한 selection 정렬 알고리즘은 (stable/unstable), (in-place/not in-place)이다. (맞는 성질에 동그라미. 부분점수 없음)
   I. [1.5] quick 정렬의 최악의 경우 수행시간을 f(n)이라 하고, 평균 수행시간을 g(n)이라 할 때, f(n)/g(n)을 Big-O로 표기하면? O(_____)
   J. [2] n개의 값이 이미 오름차순으로 정렬된 리스트 A에 다시 (오름차순으로 정렬하는) insertion sort 알고리즘을 적용했다. 이 정렬 알고리즘의 비교회수와 이동(교환)회수를 각각 Big-O로 표기하면?
      비교회수: O(_____), 이동(교환)회수: O(_____) (부분점수 없음)

2. [4] 오른쪽 python 코드에 대해 답하라.

```
def f2(A, n):  # n 개 값이 A 에 저장됨
    s = 0      # n 은 항상 짝수임
    while len(A):
        s += A.pop(0) - A.pop()
    return s
```

   A. [1.5] print(f2([5,4,3,2,1,0], 6))의 출력 결과는? _____
   B. [1] f2의 수행시간 T(n)을 Big-O로 표기하라. (pop 연산의 수행시간도 고려되어야 한다. Big-O와 괄호 모두 있어야 함) _____
   C. [1.5] B번의 수행시간이 되는 이유를 두 줄 이내로 설명하라.

3. [4] 오른쪽 코드에 대해, 다음 질문에 답하라.

```
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
```

   A. [1.5] f3([0,1,2,3,4,5,6,7],8))의 출력 결과는? _____
   B. [1] f3(A, n)의 수행시간 T(n)을 Big-O로 표기하라. T(n) = _____
   C. [1.5] B번의 수행시간이 되는 이유를 두 줄 이내로 설명하라.

4. [4.5] 오른쪽 코드를 보고 답하라.

```
def f4(n):
    if n == 1: return 1
    if n % 2 == 1:
        return 1 + f4(n//2)
    else:
        return f4(n//2)
```

   A. [1] print(f4(11))의 출력 값은 무엇인가?

      _____

   B. [1] f4(n)의 수행시간 T(n)을 점화식으로 표현하면?

      _____

   C. [2.5] 이 점화식을 **<u>전개한 후</u>**, Big-O로 표현하세요.
      (<u>가정이 필요하면 직접 언급할 것. T(1)도 언급할 것.</u>)

5. [9] 아래 stoogeSort는 리스트 A에 저장된 n개의 값을 오름차순으로 정렬하는 알고리즘 중 하나이다. 다음 물음에 답하라.

```
Algorithm stoogeSort(A):
    n = |A|
    if n == 2 and A[0] > A[1]:
        swap A[0] and A[1]
    m = 2*n//3        # 2n/3 의 올림을 의미함
    stoogeSort(A[:m])
    stoogeSort(A[n-m:])
    stoogeSort(A[:m])
```

   A. [2.5] stoogeSort가 A를 정렬하는 이유가 무엇인지 직관적으로 설명하라. (부분점수 있음)

   B. [1.5] 위의 코드의 수행시간 T(n)의 점화식을 세워보자. (T(1) = c라고 가정한다)

   C. [5] 점화식을 전개한 후, Big-O로 표기하라 ($n = (3/2)^k$이라 가정. 부분점수 있음)

6. [7.5] 아래 코드는 quickSort 함수이다. 다음 물음에 답하라.

```
def quickSort(A):
    if len(A) <= 1: return A
    print(A)              # (*) 현재 A 값들을 출력함
    S, M, L = [], [], [] # S, M, L은 각각 pivot보다 작거나, 같거나, 큰 값들을 저장
    p = A[0]
    for a in A:
        if _____________ : S.append(a)
        elif a == p: _______________
        else: ___________________
    return quicksort(L) + M + quicksort(S)   # 함수 호출 및 연산 순서 주의할 것!
```

A. [1.5] 위 코드의 빈 줄 3개를 채워라. (0.5 점씩)

B. [4] A = quickSort([4, 5, 1, 3, 10, 4, 7, 1])으로 호출한다. 그러면 (*) 부분이 총 4번 실행되는 데, 그 출력 값을 **출력 순서대로** 한 줄에 하나씩 쓰시오. (각 1점씩)

_____

_____

_____

_____

C. [1] A = quicksort(A) 후, print(A)의 결과는? _____

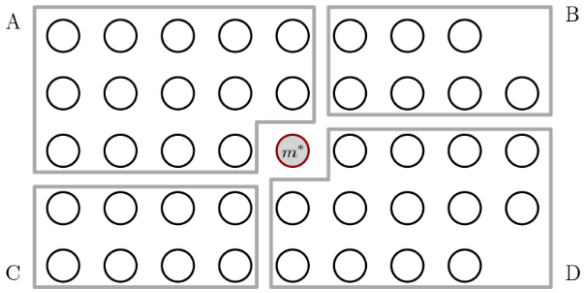
D. [1] 위의 알고리즘은 stable한가? Yes/No in-place인가? Yes/No (부분점수 없음)

7. [12.5] 아래 코드는 MoM(Median of Medians) 함수이다. 다음 페이지의 그림도 참고해서 각 물음에 답하라.

```
def MoM(A, k):
    # n개의 A의 값 중에서 k번째로 작은 수 찾아 리턴 (n개의 수는 모두 다르다고 가정)
    # 다음 페이지의 그림은 수업시간에 설명한 알고리즘의 중간 단계를 나타낸 것이다.
    # 여기서 m*는 median of medians이다.

    1. S = m*보다 작은 수 = A + {_____________ 값 중에서 m*보다 작은 수}
       L = m*보다 큰 수 = ___ + {_____________ 값 중에서 m*보다 큰 수}
    2. if _________________: return MoM(S, ______)
       elif _____________________: return MoM(L, _____________________)
       else: return _______
```

A. [3.5] 위 코드의 빈 칸 7개를 채워라. 단, A, B, C, D는 다음 페이지의 그림 참조. (0.5점씩)



B. [1] S의 값의 개수와 L의 값의 개수는 모두 최소 _____ 개 이상, 최대 _____ 개 이하다. (부분점수 없음)

C. [2] 그 이유는? (두 줄 이내로. 부분점수 있음)

D. [2] MoM 알고리즘의 최악의 경우의 비교횟수를 W(n)이 아래와 같다고 하자. 빈 두 칸을 채워라. (부분점수 없음)

W(n) <= W(_____) + W(_____) + n

E. [4] 그러면 $W(n) \le cn$ 이 성립하는 자연수 c가 존재함을 **수학적 귀납법**을 이용해 증명할 수 있다. 이 때, 이 부등식을 만족하는 **자연수 c 중에서 가장 작은 값**은 무엇인가? 왜 그런지 설명하라. 여기서, W(1)은 적당한 상수라고 가정해도 된다. (필요한 가정이 있다면, 직접 언급하면 됨. 부분점수 있음)

8. [10] 리스트 A에 저장된 서로 다른 n개의 정수와 정수 K가 입력으로 주어진다. A의 두 수의 합이 K가 되는 두 수의 쌍을 이진탐색을 호출해 **모두 구하고 싶다**. A[i] + A[j] == K라면, 튜플 (i, j)를 리스트 P에 저장한다. (단, i <= j인 경우만 리스트 P에 저장하면 된다.) 이진탐색 binary_search(A, left, right, x)는 **오름차순으로 정렬된 리스트** A의 A[left], ..., A[right] 중에서 특정 값 x가 있으면 x의 index를, 없으면 None을 리턴하는 함수이다.

A. [2] 아래 코드의 빈 칸을 채워라. (1점씩)

```
def find_pair(A, K):
    n = len(A)
    P = []
    A.sort() # 먼저 오름차순으로 정렬
    for i in range(n):
        j = binary_search(A, i, n-1, ____________)
        if j != None:
            P.append(_____________________) # append 연산은 O(1)이라 가정!
    return P
```

B. [2.5] find_pair 함수의 수행시간 T(n)을 **분석한 후** Big-O 표기법으로 나타내세요. (분석이 없으면 0점)

C. [5] A의 값들이 **이미 오름차순으로 정렬**되어 입력으로 주어진다면, find_pair보다 더 빠르게 계산할 수 있다. 어떻게 하면 되는지 논리 정연하게 (예: pseudo 코드) 설명하고, 수행시간을 분석해 Big-O로 표기하라. (부분점수 있음)

9. [3] n번째 피보나치 수열 값을 계산하는 재귀 함수 fibo(n)이 오른쪽과 같다. fibo(4)를 호출하면, fibo의 재귀호출 순서가 다음과 같다. 빈 칸을 채워라. (부분점수 없으니 신중하게 작성하라)

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

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

10. [1] 강의에 대해 좋았던 점과 개선할 점 등을 솔직히 써 주면 남은 강의에 반영하도록 노력하겠습니다. (쓰기만 하면 1점!)