HW1 풀이 2020203090 한용옥

1. Give the pseudo-code of a logarithmic-time ($\theta(logn)$-time) algorithm for computing the n-th Fibonacci number.

피보나치 수열 F_n에 대해 다음이 성립한다 ($F_0 = 0, F_1 = 1$)

$$\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{n-1} \\ F_{n-2} \end{bmatrix} \to \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} F_1 \\ F_0 \end{bmatrix} \to \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

행렬을 반씩 쪼개어 곱하면 $\theta(logn)$ 시간이 걸린다 따라서 의사코드는 다음과 같다

```
exp(A : Matrix, n : int):
    if n == 1:
        return A
    A = exp(A, n/2)
    return A * A

fibo(n : int):
    A = exp([[1, 1], [1, 0]], n - 1)
    A = A * [1, 0]
    return A[0]
```

2. Prove that $n^2 \in o(2^n)$ using the formal definition of small-o notation.

엡실론 델타 논법

$$\lim_{x\to\infty} f(x) = L \Leftrightarrow \forall\epsilon > 0, \exists\delta > 0, x > \delta \to |f(x) - L| < \epsilon$$

$small - o$의 정의

$$o(f(x)) = g(x) \mid \forall c > 0, \exists N > 0, \forall x > N, g(x) \le cf(x)$$

로피탈의 정리

$$\lim_{x\to\infty} f(x) = \infty, \lim_{x\to\infty} g(x) = \infty, \text{ f, g는 미분 가능}, \exists \lim_{x\to\infty} \frac{f'(x)}{g'(x)} \text{ 이면}$$

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = \lim_{x\to\infty} \frac{f'(x)}{g'(x)}$$

보조정리 1 임의 x에 대해 $x > a \to p(x) \Leftrightarrow \forall x > a \ \ p(x)$: 전칭 일반화

보조정리 2 : $g(x) > 0, f(x) > 0$ 일 때 $\lim_{x\to\infty} \frac{g(x)}{f(x)} = 0 \to g(x) \in o(f(x))$

보조정리 2 증명 :

$$\lim_{x\to\infty} \frac{g(x)}{f(x)} = 0$$

$$\to \forall\epsilon > 0, \exists\delta > 0, x > \delta \to \frac{g(x)}{f(x)} < \epsilon \text{ (엡실론 델타 논법)}$$

$$\rightarrow \forall\epsilon > 0, \exists\delta > 0, \forall x > \delta, g(x) < \epsilon f(x) \text{ (보조정리 1)}$$

$$\rightarrow \forall\epsilon > 0, \exists\delta > 0, \forall x > \delta, g(x) \le \epsilon f(x)$$

$$\rightarrow g(x) \in o(f(x)) \text{ (small-o의 정의)}$$

본 문제 증명:

$$\lim_{n\to\infty} \frac{n^2}{2^n} = \lim_{n\to\infty} \frac{2n}{ln2\ 2^n} = \lim_{n\to\infty} \frac{2}{(ln2)^2 2^n} = 0 \text{ (로피탈의 정리)}$$

$$\lim_{n\to\infty} \frac{n^2}{2^n} = 0 \text{ 이므로 보조정리 2에 의해 } n^2 \in o(2^n)$$

3. Give the closed forms for the following recurrence relations. Solve them using their characteristic equations.

(3a) $f_{n+2} = f_{n+1} + f_n \ (n \ge 0), f_0 = 0, f_1 = 1$

위 재귀 관계의 특성 방정식과 해는 아래와 같다

$$r^2 - r - 1 = 0, r = \frac{1 \pm \sqrt{5}}{2}$$

재귀관계의 해는 아래와 같은 구조이고 초기값 대입시 다음과 같다

$$f_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$f_0 = 0, f_1 = 1 \text{ 이므로, } \begin{cases} \alpha_1 + \alpha_2 = 0 \\ \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right) = 1 \end{cases}$$

$$\alpha_1 = \frac{1}{\sqrt{5}}, \alpha_2 = -\frac{1}{\sqrt{5}}$$

따라서 재귀관계의 해는 아래와 같다

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2}\right)^n$$

(3b) $f_{n+2} = f_{n+1} + f_n + 1 \ (n \ge 0), f_0 = 0, f_1 = 1$

비동차항이 상수항이므로 특수해는 상수형태 가정하고 재귀관계에 대입시 다음과 같다

$$\alpha = \alpha + \alpha + 1 \rightarrow -1 = \alpha$$

따라서 특수해는 -1이므로 일반해의 구조는 동차해와 특수해를 더한 구조이다 초기값 대입시 다음과 같다

$$f_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right)^n - 1$$

$$f_0 = 0, f_1 = 1 \text{ 이므로, } \begin{cases} \alpha_1 + \alpha_2 = 1 \\ \alpha_1 \left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2 \left(\frac{1-\sqrt{5}}{2}\right) = 2 \end{cases}$$

$$\alpha_1 = \frac{3+\sqrt{5}}{2\sqrt{5}}, \alpha_2 = \frac{\sqrt{5}-3}{2\sqrt{5}}$$

따라서 재귀관계의 해는 아래와 같다

$$f_n = \frac{3+\sqrt{5}}{2\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{\sqrt{5}-3}{2\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n - 1$$

3c, 3d 공통 풀이

3c, 3d의 동차해를 구하기 위해 아래 문제를 풀어야 한다

$$T(n) = 3T(\frac{n}{3}) \text{ (You may assume that } n = 3^k, k \geq 0)$$

가정에 의해 $n = 3^k, T(3^k) = 3T(3^{k-1}), t(k) = T(3^k)$ 라 하자 그럼 $t(k) = 3t(k-1)$ 가 성립한다
특성 방정식은 $r^k = 3r^{k-1} \to r = 3$ 이다 따라서 k에 대한 재귀관계의 동차해는 아래와 같다

$$t(k) = \alpha 3^k$$

(3c) $T(n) = 3T(\frac{n}{3}) + 2\frac{n}{3}, T(1) = 0 \text{ (You may assume that } n = 3^k, k \geq 0)$

가정에 의해 $n = 3^k \to k = log_3 n$ 이다 $T(n) = T(3^k) = t(k)$ 라 하면 아래가 성립한다

$$T(n) = 3T(\frac{n}{3}) + 2\frac{n}{3} \to t(k) = 3t(k-1) + 2 \cdot 3^{k-1}, T(1) = t(0) = 0$$

동차해가 지수함수이므로 특수해 $t_p(k) = akb^k$ 가정하여 재귀관계에 대입시 아래와 같다

$$akb^k = 3a(k-1)b^{k-1} + 2 \cdot 3^{k-1} \to (abk - 3ak + 3a)b^{k-1} = 2 \cdot 3^{k-1}$$

$$b = 3, a = \frac{2}{3} \to t_p(k) = \frac{2}{3}k3^k$$

동차해와 특수해를 구했으므로 둘을 더하면 일반해의 구조이고 초기값 대입시 아래와 같다

$$t(k) = \frac{2}{3}k3^k + \alpha 3^k \xrightarrow{t(0)=0} 0 = \alpha, t(k) = \frac{2}{3}k3^k$$

변수를 n 으로 변환시 아래와 같다

$$t(k) = \frac{2}{3}k3^k \xrightarrow{k=log_3 n, t(k)=T(3^k)} T(n) = \frac{2}{3}log_3 n 3^{log_3 n}$$

따라서 위 재귀관계의 초기 해는 아래와 같다

$$T(n) = \frac{2}{3}n log_3 n$$

(3d) $T(n) = 3T(\frac{n}{3}) + 2n, T(1) = 0 \text{ (You may assume that } n = 3^k, k \geq 0)$

가정에 의해 $n = 3^k \to k = log_3 n$ 이다 $T(n) = T(3^k) = t(k)$ 라 하면 아래가 성립한다

$$T(n) = 3T(\frac{n}{3}) + 2n \to t(k) = 3t(k-1) + 2 \cdot 3^k, T(1) = t(0) = 0$$

동차해가 지수함수이므로 특수해 $t_p(k) = akb^k$ 가정하여 재귀관계에 대입시 아래와 같다

$$akb^k = 3a(k-1)b^{k-1} + 2 \cdot 3^k \rightarrow (abk - 3ak + 3a)b^{k-1} = 6 \cdot 3^{k-1}$$

$$b = 3, a = 2 \rightarrow t_p(k) = 2k3^k$$

따라서 동차해와 특수해를 구했으므로 둘을 더하면 일반해의 구조이고 초기값 대입시 아래와 같다

$$t(k) = 2k3^k + \alpha 3^k \xrightarrow{t(0)=0} 0 = \alpha, t(k) = 2k3^k$$

변수를 n 으로 변환시 아래와 같다

$$t(k) = 2k3^k \xrightarrow{k=log_3 n, t(k)=T(3^k)} T(n) = 2log_3 n 3^{log_3 n}$$

따라서 위 재귀관계의 초기 해는 아래와 같다

$$T(n) = 2nlog_3 n$$

4. Consider MergeSort algorithm in the textbook.

(4a) Given array size n, find a recurrence relation for the best-case time complexity for MergeSort.

교재의 병합정렬 알고리즘은 아래와 같다

1. 1개의 원소가 될 때까지 배열 반으로 쪼개기
2. 1의 것들을 정렬하며 병합
   2.1. 병합시 두 배열의 첫 부터 시작해서 각각 비교, 순회하며 정렬 순서로 채워넣음

1은 배열이 무엇이든 반으로 나누므로 길이가 같으면 수행 횟수는 고정이다
2의 경우는 두 배열의 상태에 따라 수행시간이 달라진다
A, B를 이어 붙였을때 정렬된 한 배열이 나올 때 A, B를 병합하는 것이 최선의 경우이다
이 경우는 $min(|A|, |B|)$ 의 시간이 소요된다

최선의 경우 입력 배열이 반씩 쪼개지고 병합되므로 $min(|A|, |B|) = \frac{n}{2}$
배열의 길이가 2인 경우, 반으로 쪼개진 다음 병합 되므로 시간은 1 소요된다
따라서 최선 시간복잡도 $B(n)$ 의 재귀관계는 아래와 같다

$$B(n) = 2B(\frac{n}{2}) + \frac{n}{2}, B(2) = 1$$

(4b) Solve the recurrence relation for (1), given n (= 2^k for some integer $k > 0$).

가정에 의해 $n = 2^k, k = log_2 n$ 이다 $b(k) = B(2^k) = B(n)$ 라 하면 아래가 성립한다

$$B(n) = 2B(\frac{n}{2}) + \frac{n}{2} \xrightarrow{b(k)=B(2^k)} b(k) = 2b(k-1) + 2^{k-1}, B(2) = b(1) = 1$$

선형 재귀관계 $b(k)$ 의 동차식의 풀이는 아래와 같다

$$b(k) = 2b(k-1) \text{ 의 특성방정식 : } r^k = 2r^{k-1} \rightarrow r = 2$$

따라서 재귀관계 $b(k)$ 의 동차해 $b_c(k) = \alpha 2^k$ 이다

동차해가 지수함수이므로 특수해 $b_p(k) = akb^k$ 가정하여 재귀관계에 대입시 아래와 같다

$$akb^k = 2a(k-1)b^{k-1} + 2^{k-1} \rightarrow (abk - 2ak + 2a)b^{k-1} = 2^{k-1}$$

$$b = 2, a = \frac{1}{2} \rightarrow b_p(k) = \frac{1}{2}k2^k$$

따라서 동차해와 특수해를 구했으므로 둘을 더하면 일반해의 구조이고 초기값 대입시 아래와 같다

$$b(k) = \frac{1}{2}k2^k + \alpha 2^k \xrightarrow{b(1)=1} 1 = 1 + 2\alpha \rightarrow \alpha = 0, b(k) = \frac{1}{2}k2^k$$

변수를 n 으로 변환시 아래와 같다

$$b(k) = \frac{1}{2}k2^k \xrightarrow{k=log_2 n, b(k)=B(2^k)} B(n) = \frac{1}{2}log_2 n 2^{log_2 n}$$

따라서 위 재귀관계의 초기 해는 아래와 같다

$$B(n) = \frac{1}{2}nlog_2 n$$

5. (Theoretically Fast QuickSort) There exists a linear-time ($O(n)$-time) algorithm that computes a median value among given n values. Assume that we have already known this algorithm. Using this algorithm

(5a) design a variant QuickSort algorithm of which the worse-case time complexity is $O(nlogn)$ (just give its pseudo-code)

선형시간에 중앙값의 위치를 구할 수 있는 알고리즘을 Median이라 하자 의사코드는 아래와 같다

```
Partition(A : array, low : int, high : int):
    pivot, p_i = A[low], low
    for i in range(low, high):
        if A[i] < pivot:
            p_i += 1
            swap(A[i], A[p_i])
    swap(A[p_i], A[low])
    return p_i

FastQuickSort(A : array, low : int, high : int):
    i = Median(A, low, high)
    swap(A[low], A[i]) // 중앙값을 피벗으로
    i = Partition(A, low, high)
    FastQuickSort(A, low, i)
    FastQuickSort(A, i, high)
```

(5b) prove that its time complexity is $O(nlogn)$

$O(n)$으로 중앙값을 구하고 분할해 반으로 쪼개진다 분할 시간은 배열의 크기로 고정이므로
배열 크기 n에 대한 시간복잡도 $B(n)$의 재귀관계는 아래와 같다

$$B(n) = 2B(\frac{n}{2}) + \frac{n}{2} + f(n), f(n) \in O(n)$$

양의 정수 k 에 대해 $n = 2^k, k = log_2 n$ 라고 가정, $b(k) = B(2^k) = B(n)$ 라 하면 아래가 성립한다

$$B(n) = 2B(\frac{n}{2}) + \frac{n}{2} + f(n) \xrightarrow{b(k)=B(2^k)} b(k) = 2b(k-1) + 2^{k-1} + f(2^k)$$

$$f(2^k) \in O(2^k)$$

$b(k)$ 는 선형 재귀관계이며 동차해는 **4번** 에서 구한 $b_c(k) = \alpha 2^k$ 이다

동차해가 지수함수이므로 특수해 $b_p(k) = akb^k$ 가정하여 재귀관계에 대입시 아래와 같다

$$akb^k = 2a(k-1)b^{k-1} + 2^{k-1} + f(2^k)$$

$$\rightarrow (abk - 2ak + 2a)b^{k-1} - 2^{k-1} = f(2^k) \in O(2^k) \tag{1}$$

빅-오의 정의에 의해 **식(1)** 을 만족시키려면 아래를 만족해야한다

$$\exists c > 0, N > 0, \forall k > N, (abk - 2ak + 2a)b^{k-1} - 2^{k-1} \leq c2^k$$

뒤의 부등식을 고쳐쓰면

$$((b-2)ak + 2a)b^{k-1} \leq (c + \frac{1}{2})2^k$$

a 는 상수이므로 $c + \frac{1}{2} > a$ 인 c 를 항상 잡을 수 있다
$b = 2$ 일때 $c_0 + \frac{1}{2} > a$ 인 c_0 에 대해 $N = 1$이면

$$\forall k > N, a2^k \leq (c_0 + \frac{1}{2})2^k$$

따라서 $b = 2$ 일 때 **식(1)** 을 만족하므로 재귀관계의 특수해 $b_p(k) = ak2^k$ 다

따라서 동차해와 특수해를 구했으므로 둘을 더하면 일반해이고 아래와 같다

$$b(k) = ak2^k + \alpha 2^k$$

변수를 n 으로 변환시 아래와 같다

$$b(k) = ak2^k + \alpha 2^k \xrightarrow{k=log_2 n, b(k)=B(2^k)} B(n) = alog_2 n 2^{log_2 n} + \alpha 2^{log_2 n}$$

따라서 재귀관계 $B(n)$ 의 해는 아래와 같다

$$B(n) = anlog_2 n + \alpha n$$

$n > 2$ 인 n 에 대해 다음이 성립한다 $B(n)$은 $n > 0$에서 정의되고 양수이므로 $a, \alpha > 0$ 이다

$$B(n) = anlog_2 n + \alpha n = n(alog_2 n + \alpha) \leq n(alog_2 n + \alpha log_2 n)$$

$$n(alog_2 n + \alpha log_2 n) = (a + \alpha)nlog_2 n$$

$(a + \alpha)$ 는 상수이므로 항상 그보다 더 큰 c_0을 잡을 수 있다 그러한 c_0 에 대해 다음이 성립한다

$$B(n) = anlog_2 n + \alpha n \leq (a + \alpha)nlog_2 n \leq c_0 nlog_2 n$$

따라서 $c = c_0, N = 2, \forall n > N, B(n) \leq c_0 nlog_2 n$ 이므로

$$\exists c > 0, N > 0, \forall n > N, B(n) \leq cnlog_2 n \text{ 이며 , 이는 } B(n) \in O(nlog_2 n)\text{이다}$$