

1.1-1

“정렬” 또는 “볼록 덮개 찾기” 계산 문제 중 하나를 골라 이런 문제가 발생하는 현실의 예를 제시하라.

정렬의 문제는 일자별로 데이터가 취향될 때,
일자별 순서대로 데이터 확인이 하고 싶을 때

1.1-2

현실에서 속도 외에 효율성을 평가할 만한 다른 척도로 무엇이 있는지 제시하라.

속도 이외에도 얼마나 메모리를 효율적으로 사용하는지,
방법론지를 통해 속도를 더 높일 수 있는지 등

1.1-3

예전에 본 적이 있는 자료구조 중 하나를 골라 그것의 장점과 단점을 각각 논하라.

- Linked List
- 장점 : 데이터를 List의 중간에 넣고 뺄 때 빠르게 처리 가능
- 단점 : 인덱스를 통해 List 중간 데이터에 바로 접근은 불가능 하며, 항상 link된 데이터를 통해 접근 가능하다

1.1-4

앞서 설명한 최단 경로 문제와 순회 판매원 문제는 어떻게 비슷한가, 그리고 어떻게 다른가?

비슷한 점 : 둘 모두 최적의 순서를 찾고자 하는 문제이다.

다른 점 : 최단 경로 문제는 출발지와 도착지만 존재하지만, 순회 판매원 문제는 중간에 경유지로 존재 할수 있으며 최종적으로 다시 출발지로 돌아와야 한다.

1.1-5

현실의 문제 중 최적해만 의미가 있는 문제를 제시하라. 반대로 최적은 아니지만 “근접한” 해를 구해도 충분한 문제를 제시하라.

최적해만 의미 있는 문제: 정렬 문제.

근접한 해를 구해도 충분한 문제: π 의 값을 구하는 문제

1.2-1

응용 프로그램 수준에서 알고리즘이 필요한 프로그램의 예를 들라. 그리고 이용된 알고리즘의 기능에 대해 논하라.

응용 프로그램에서 정색이 축전이 필요할 때 tree 통해서 search 한다.
해당 알고리즘의 기능은 단어 전체를 한번에 비교하지 않고
차례 차례 비교한다.

1.2-2

동일한 기계에서 삽입 정렬과 병합 정렬의 구현 결과를 비교한다고 가정하자. n 개의 입력에 대해 삽입 정렬은 $8n^2$ 번을, 병합 정렬은 $64n \lg n$ 번을 계산하고 각각 종료한다. n 값이 얼마일 때 삽입 정렬이 병합 정렬보다 빠를까?

1.2-3

동일한 기계에서 수행시간이 $100n^2$ 인 알고리즘이 수행시간이 2^n 인 알고리즘보다 빨라지는 n 의 최솟값은 얼마인가?

4.1-1

A 의 모든 원소가 음수일 때 FIND-MAXIMUM-SUBARRAY가 반환하는 값은 무엇인가?

모든 원소 중 0에 가장 가까운 원소 하나만 선택되고.
해당 인덱스가 선택된다.

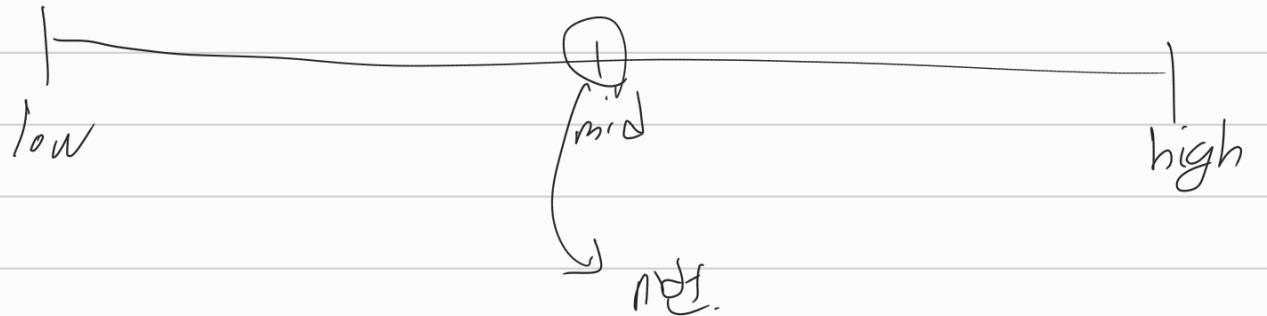
4.1-2

최대 부분 배열 문제를 푸는 주먹구구식 방법에 대한 의사코드를 작성하라. 작성한
프로시저는 $\Theta(n^2)$ 시간에 수행되어야 한다.

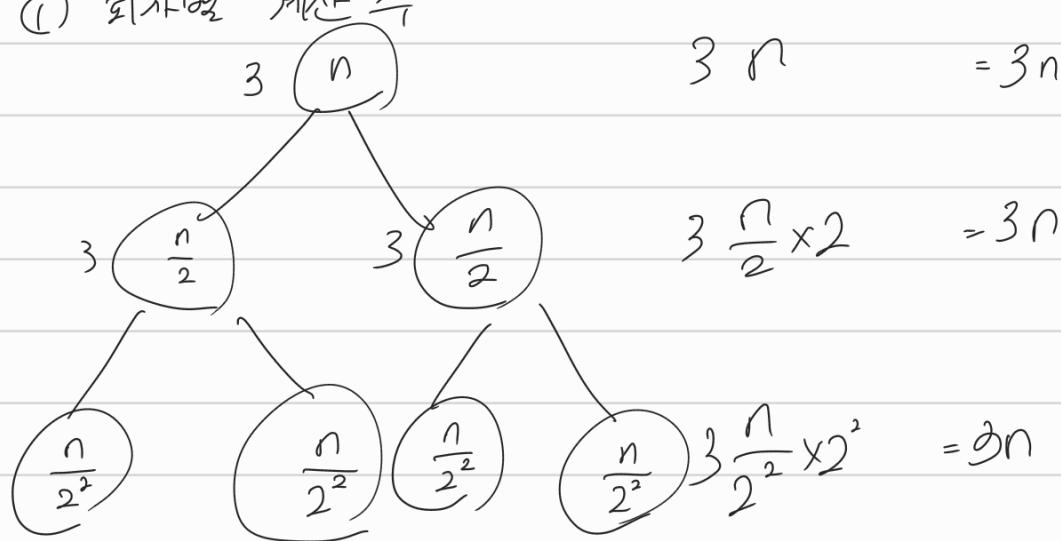
```
def find_max_subarray (in-arr) :  
    n := length (in-arr) :  
        left_index = -1  
        right_index = -1  
        max = -∞  
        for i=1 to n :  
            sum = in-arr[i]  
            for j=i+1 to n :  
                sum = sum + in-arr[j]  
                if max < sum :  
                    max = sum  
                    left_index = i  
                    right_index = j  
  
    return (left_index, right_index, max )
```

4.1-3

당신의 컴퓨터를 사용해 최대 부분 배열 문제를 푸는 주먹구구식 알고리즘과 재귀 알고리즘을 둘 다 구현하라. 재귀 알고리즘이 주먹구구식 알고리즘을 넘어설 때의 교차 지점이 되는 문제의 크기 n_0 은 무엇인가? 그리고 문제의 크기가 n_0 보다 작을 때 주먹구구식 알고리즘을 사용하도록 재귀 알고리즘의 베이스 케이스를 수정하라. 이로 인해 교차 지점이 달라지는가?



① 회차별 계산 수



② 최소 깊이 계산

$$\frac{n}{2^k} \leq 1, \quad k \geq \log_2 n$$

$$\Rightarrow ① \times ② \geq 3n \log_2 n$$

주역구구식 알고리즘.

$$n^2$$

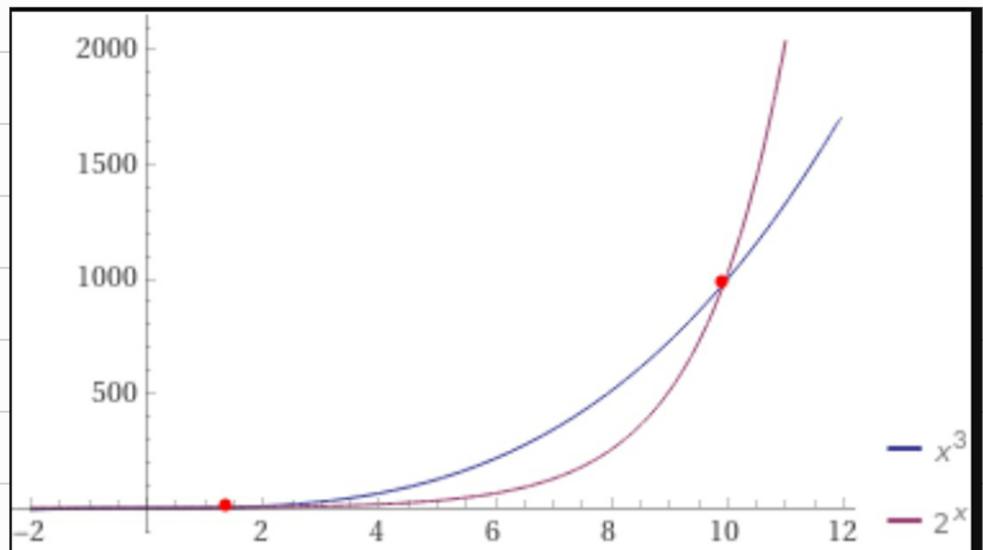
주역구구식 > 재귀.

$$n^2 > 3n \log_2 n$$

$$n > 3 \log_2 n$$

$$2^n > n^3$$

$n \geq 10$ 일 때.



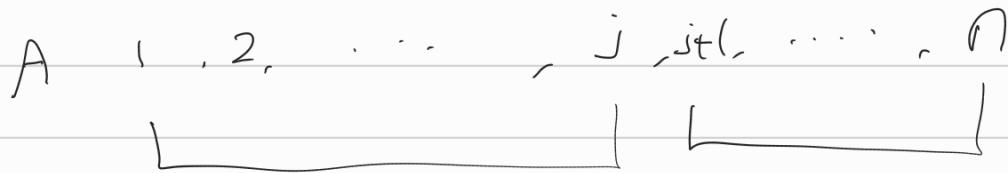
4.1-4

최대 부분 배열 문제를 수정해 빈 부분 수열을 결과로 허용한다고 가정하자. 빈 부분 배열의 합은 0이다. 빈 부분 수열을 허용하지 않는 알고리즘이 빈 부분 배열을 결과로 허용하도록 하려면 어떻게 수정해야 할까?

빈 부분 배열이 허용이 된다는 이야기는 최대값이 0보다 작게 나오는 경우에 빈 부분 배열을 결과값으로 주려는 것으로 보인다. 그렇다면 maximum-subarray를 찾을 때 `left`, `mid`, `right`의 값을 비교하여 0보다 작은 경우엔 빈 배열을 `return`하도록 짜야 한다.

4.1-5

다음 아이디어를 이용해 최대 부분 배열 문제를 푸는 재귀적이지 않은 선형 시간 알고리즘을 개발하라. 배열의 왼쪽 끝에서 시작해 오른쪽으로 진행하면서 지금까지의 최대 부분 배열을 기록한다. $A[1..j]$ 의 최대 부분 배열을 알고 있으면 다음 관찰을 이용해 인덱스 $j+1$ 에서 끝나는 최대 부분 배열을 찾을 수 있도록 확장할 수 있다. $A[1..j+1]$ 의 최대 부분 배열은 $1 \leq i \leq j+1$ 에 대해 $A[1..j]$ 의 최대 부분 배열이거나 부분 배열 $A[i..j+1]$ 이다. 인덱스 j 로 끝나는 최대 부분 배열을 알고 있다는 사실에 기반해서 $A[i..j+1]$ 형태의 최대 부분 배열을 상수 시간에 구하라.



최대 부분 배열의 시작 인덱스 : a ($1 \leq a < j$)
 끝 인덱스 : b ($1 < b \leq j$)

이제 //

$$\left[\begin{smallmatrix} 1001 & 1002 & \dots & 1000 \end{smallmatrix} \right]$$

4.2-1

스트라센 알고리즘을 이용해 다음 행렬 곱을 계산하는 과정을 보여라.

$$\begin{pmatrix} 13 \\ 75 \end{pmatrix} \begin{pmatrix} 68 \\ 42 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad B_1 = \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$S_1 = B_{12} - B_{22} = 6$$

$$P_1 = A_{11} \cdot S_1 = 6$$

$$S_2 = A_{11} + A_{12} = 4$$

$$P_2 = S_2 \cdot B_{22} = 8$$

$$S_3 = A_{21} + A_{22} = 12$$

$$P_3 = S_3 \cdot B_{11} = 72$$

$$S_4 = B_{21} - B_{11} = -2$$

$$P_4 = A_{22} \cdot S_4 = -10$$

$$S_5 = A_{11} + A_{22} = 6$$

$$P_5 = S_5 \cdot S_6 = 48$$

$$S_6 = B_{11} + B_{22} = 8$$

$$P_6 = S_7 \cdot S_8 = -12$$

$$S_7 = A_{12} - A_{22} = -2$$

$$P_7 = S_9 \cdot S_{10} = -84$$

$$S_8 = B_{21} + B_{22} = 6$$

$$S_9 = A_{11} - A_{21} = -6$$

$$S_{10} = B_{11} + B_{12} = 14$$

$$C_{11} = P_5 + P_4 - P_2 + P_6 = 48 - 10 - 8 - 12 = 18$$

$$C_{12} = P_1 + P_2 = 6 + 8 = 14$$

$$C_{21} = P_3 + P_4 = 62$$

$$C_{22} = P_5 + P_1 - P_3 - P_7 = 48 + 6 - 72 + 84 = 68$$

$$C = \begin{pmatrix} 18 & 14 \\ 62 & 68 \end{pmatrix}$$

4.2-2

스트라센 알고리즘의 의사코드를 작성하라.

Matrix-multiply (A, B)

1 if size(A) == (1, 1)
2 return A(1,1) × B(1×1)
3 else
4 $S_1 = B_{12} - B_{22}$

$$S_{10} = B_{11} + B_{12}$$

5 $P_1 = \text{matrix-multiply}(A_{11}, B_{12})$
 - $\text{matrix-multiply}(A_{11}, B_{22})$

P_7 까지.

6 $C_1 = P_5 + P_4 - P_2 + P_6$

C_4 까지 |

7 return $\begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix}$.

4.3-1

$T(n) = T(n-1) + n$ 의 해가 $O(n^2)$ 임을 보여라.

$$T(1) = T(0) + 1 = 1$$

$$T(2) = T(1) + 2 = 3$$

$$T(3) = T(2) + 3 = 6$$

⋮

$$T(n) = T(n-1) + n = T(n-2) + (n-1) + n$$

⋮

$$= T(0) + 1 + \dots + n$$

$$= 0 + \frac{n(n+1)}{2}$$

$$\leq n^2$$

$$\Rightarrow T(n) \in O(n^2)$$

4.3-2

$T(n) = T(\lceil n/2 \rceil) + 1$ 의 해가 $O(\lg n)$ 임을 보여라.

$$T(1) = T(0) + 1 = 1$$

$$T(2) = T(1) + 1 = 2$$

$$T(3) = T(2) + 1 = 3$$

:

:

$\text{0} \leq \frac{n}{2} \leq \frac{n}{2}$ 가정.

$$T(n) \leq c \lg n \quad \text{for some } c.$$

Let's show $T(n+1) \leq c \lg(n+1)$

$$\begin{aligned} T(n+1) &= T(\lceil \frac{n+1}{2} \rceil) + 1 \\ &\leq c \lg \left(\frac{n+1}{2} \right) + 1 \\ &= c \lg(n+1) - c \lg 2 + 1 \\ &= c \lg(n+1) - c + 1 \\ &\leq c \lg(n+1) \end{aligned}$$

c 를 100을 선택하면 된다.

4.3-3

$T(n) = 2T(\lfloor n/2 \rfloor) + n$ 의 해가 $O(n \lg n)$ 임을 보았다. 이 점화식의 해가 $\Omega(n \lg n)$ 도 됨을 보여라. 그리고 해가 $\Theta(n \lg n)$ 이 된다고 결론지어라.

$$T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$$

$$T(1) = 2T(0) + 1 = 1$$

$$T(2) = 2T(1) + 2 = 4$$

$$T(3) = 2T(1) + 3 = 5$$

⋮

$$T(n) = \Omega(n \lg n) \text{이라고 해보자.}$$

그럼 어떤 C_0 가 존재하여

$$T(n) \geq C_0 n \lg n \text{ 을 만족시킨다.}$$

$$\begin{aligned} \text{이제 } T(n+1) &= 2T\left(\lfloor \frac{n+1}{2} \rfloor\right) + (n+1) \\ &\geq 2C_0 \lfloor \frac{n+1}{2} \rfloor \lg \lfloor \frac{n+1}{2} \rfloor + (n+1) \\ &\geq 2C_0 \frac{n}{2} \lg \frac{n}{2} + (n+1) \\ &= C_0 n \lg \frac{n}{2} + (n+1) \\ &= C_0 n \lg \left(\frac{n+1}{n+1} \cdot \frac{n}{2}\right) + (n+1) \\ &= C_0 n (\lg(n+1) - 1 + \lg n - \lg(n+1)) + (n+1) \\ &= C_0 n \lg(n+1) - C_0 n + C_0 n \lg \frac{n}{n+1} + n+1 \\ &= C_0 (n+1) \lg(n+1) - C_0 \lg(n+1) - C_0 n \\ &\quad + C_0 n \lg \frac{n}{n+1} + n+1 \end{aligned}$$

$$\geq C_0 (n+1) \lg(n+1) \quad \text{where } C_0 = \frac{1}{n}$$

4.3-4

점화식 (4.19)에 대해 다른 귀납적 가정을 함으로써 $(T) = 1$ 이라는 한계 조건을 그대로 두고서도 어려움을 해결할 수 있음을 보여라.

$$T(n) = 2T\left(\lfloor \frac{n}{2} \rfloor\right) + n$$

$$T\left(\frac{n}{2}\right) \leq C_0 \frac{n}{2} \lg\left(\frac{n}{2}\right) + 1 \quad \text{이 성립한다고 하자.}$$

$$T(n) \leq 2$$

4.3-5

$\Theta(n \lg n)$ 이 병합 정렬의 점화식 (4.3)의 “정확한” 해임을 보여라.

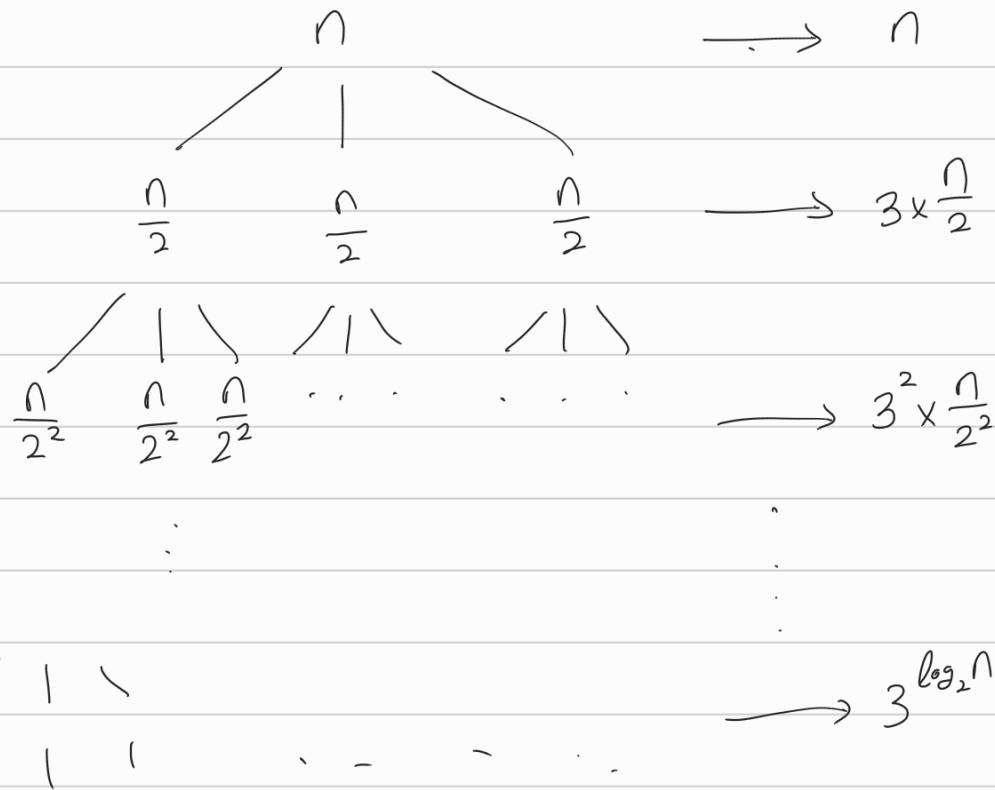
크기가 실제로는 $n/2$ 이 아니다. MERGE-SORT의 최악의 경우 수행시간을 나타내는 실제 점화식은 기술적을 다음과 같다.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{if } n > 1 \end{cases} \quad (4.3)$$

$T(n)$ 이 $\mathcal{O}(n \lg n)$, $\Omega(n \lg n)$ 임을 보이면 된다.

4.4-1

점화식 $T(n) = 3T(\lfloor n/2 \rfloor) + n$ 에 대해 재귀 트리를 사용해 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.



n 을 2의 제곱수라고 가정하자.

재귀 트리의 깊이는 $K = \log_2 n$ 이다.

$$\begin{aligned}
 \text{전체 } T(n) &= n + \frac{3}{2}n + \left(\frac{3}{2}\right)^2 n + \dots + \left(\frac{3}{2}\right)^{\log_2 n - 1} n + 3^{\log_2 n} \\
 &= \frac{\left(\frac{3}{2}\right)^{\log_2 n} - 1}{\frac{3}{2} - 1} n + 3^{\log_2 n} \\
 &= 2n \times \left(\left(\frac{3}{2}\right)^{\log_2 n} - 1\right) + 3^{\log_2 n}
 \end{aligned}$$

$$= 2n \times \left(\frac{3^{\log_2 n}}{n} - 1\right) + 3^{\log_2 n}$$

$$= 3 \cdot 3^{\log_2 n} - 2n$$

$$= 3 \cdot n^{\log_2 3} - 2n$$

$$\Rightarrow O(n^{\log_2 3})$$

$$\begin{aligned}
 3^{\log_2 n} &= x \\
 \log_2 3^{\log_2 n} &= \log_2 x \\
 (\log_2 3)^{\log_2 n} &= \log_2 x
 \end{aligned}$$

$$\log_2 n^{\log_2 3} > \log_2 x$$

$$n^{\log_2 3} = O$$

4.4-2

점화식 $T(n) = T(n/2) + n^2$ 에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

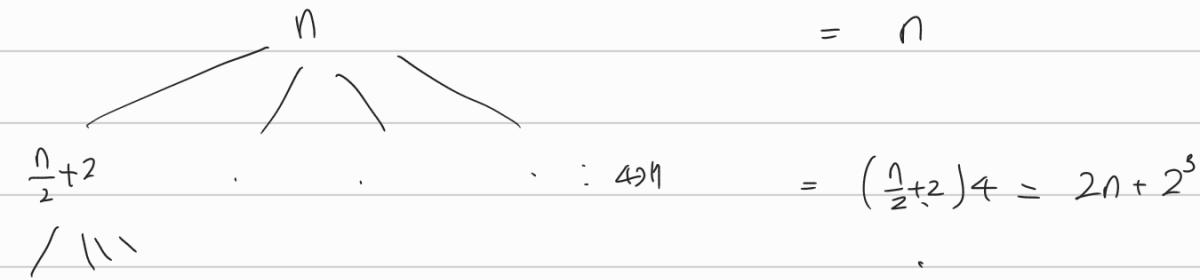
n 을 2^k 의 제곱수의 형태, $n = 2^k$ 라고 가정해보자.

$$\begin{aligned}
 & n^2 = n^2 \\
 & \left(\frac{n}{2}\right)^2 = \frac{n^2}{2^2} \\
 & \left(\frac{n}{2^2}\right)^2 = \frac{n^2}{2^4} \\
 & \vdots \\
 & \left(1\right)^2 = 1
 \end{aligned}
 \quad \left| \quad \begin{aligned}
 & = n^2 + \frac{n^2}{4} + \frac{n^2}{4^2} + \cdots + \frac{n^2}{4^{\log_2 n - 1}} + 1 \\
 & = n^2 \left(\frac{1 - \left(\frac{1}{4}\right)^{\log_2 n}}{1 - \frac{1}{4}} \right) + 1 \\
 & = \frac{4n^2}{3} - \frac{n^2}{3} \frac{4^k}{2^k} + 1 \\
 & = \frac{n^2}{3} - \frac{2n}{3} + 1
 \end{aligned} \right.$$

$\mathcal{O}(n^2)$

4.4-3

점화식 $T(n) = 4T(n/2 + 2) + n$ 에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.



$$\begin{aligned} & \text{Level 3: } \frac{\frac{\frac{n}{2} + 2}{2} + 2}{2} + 2 = \frac{n}{2^3} + \frac{2}{2^2} + \frac{2}{2} + 2 \quad : 4^3 \text{개} \\ & \vdots \quad : 4^{i-1} \text{개} \end{aligned}$$

$$\frac{1 - \left(\frac{1}{2}\right)^{i-2}}{1 - \frac{1}{2}}$$

$$\frac{n}{2^i} + 2 \sum_{k=1}^i \frac{1}{2^{k-1}} : 4^i \text{개} = 4^i \left(\frac{n}{2^i} + 2 \sum_{k=1}^i \frac{1}{2^{k-1}} \right)$$

$$= 2^i n + 4^i \cdot 4 \cdot \left(1 - \frac{4}{2^i} \right)$$

$$= 2^i n + \underbrace{4^{i+1}}_{= 2^{2i+2}} - 2^{i+2}$$

$$= 2^i n + 2^{i+2} (2^i - 1)$$

$$\sum_{i=0}^{\log_2 n} 2^i n + 2^{i+2} (2^i - 1) = n \sum_{i=0}^{\log_2 n} 2^i + 4 \sum_{i=0}^{\log_2 n} 4^i - 4 \sum_{i=0}^{\log_2 n} 2^i$$

$$= n \frac{2 \cdot 2^{\log_2 n} - 1}{2-1} + 4 \frac{2 \cdot 4^{\log_2 n} - 1}{4-1} - 4 \frac{2 \cdot 2^{\log_2 n} - 1}{2-1}$$

$$= 2n^2 - n + \frac{8}{3}n^2 - \frac{4}{3} - 8n + 4$$

$$= \frac{14}{3}n^2 - 9n + \frac{8}{3}$$

$$\boxed{T(n) = O(n^2)}$$

4.5-1

마스터 방법을 사용해 다음 점화식들의 정확한 점근적 한계를 구하라.

- a. $T(n) = 2T(n/4) + 1$
- b. $T(n) = 2T(n/4) + \sqrt{n}$
- c. $T(n) = 2T(n/4) + n$
- d. $T(n) = 2T(n/4) + n^2$

a. $a=2, b=4, f(n)=1$

$$n^{\log_4 2 - \varepsilon} = n^{\frac{1}{2} - \varepsilon} \rightarrow \varepsilon = 0 \text{ 일 때 } f(n) = O(n^{\log_4 2 - \varepsilon})$$

따라서 마스터 방법에 의해 $T(n) = \Theta(n^{\frac{1}{2}})$

b. $a=2, b=4, f(n)=\sqrt{n} = n^{\frac{1}{2}}$

$$n^{\log_4 4 - \varepsilon} = n^{\frac{1}{2} - \varepsilon} . \quad \varepsilon = 0 \text{ 일 때 } n^{\frac{1}{2} - \varepsilon} = f(n) \text{ 이므로 마스터 방법 } 2\frac{1}{2} \text{ 이용해 } T(n) = \Theta(n^{\frac{1}{2}} \log n)$$

c. $a=2, b=4, f(n)=n$

$$n^{\log_4 4} = n^{\frac{1}{2}} \text{ 이 } f(n) \text{ 과 같아지려면 } \varepsilon = \frac{1}{2} \text{ 이면 된다.}$$

마스터 방법 3을 사용할 수 있는지 확인하자.

$$2f\left(\frac{n}{4}\right) = 2 \cdot \frac{n}{4} = \frac{n}{2} = \frac{1}{2} \cdot n = \frac{1}{2}f(n) . \quad C = \frac{1}{2} < 1 \text{ 이면 된다. (마스터 방법 3)}$$

$$T(n) = \Theta(n)$$

d. $a=2, b=4, f(n)=n^2$

$$\varepsilon = \frac{3}{2} \text{ 이면 } f(n) = n^{\log_4 4 + \varepsilon} \text{ 이다. } \exists c < 1 \text{ 이어서 } a f\left(\frac{n}{b}\right) \leq c f(n) \text{ 인지 보자.}$$

$$2f\left(\frac{n}{4}\right) = \frac{n^2}{8} \leq \frac{1}{8}n^2 = \frac{1}{8}f(n) . \quad C = \frac{1}{8} < 1 \text{ 이면 성립한다. (마스터 방법 3)}$$

$$T(n) = \Theta(n^2)$$

4.5-2

Caesar 교수는 스트라센 알고리즘보다 점근적으로 더 빠른 행렬 곱셈 알고리즘을 개발하기 원한다. 그의 알고리즘은 각 행렬을 $n/4 \times n/4$ 크기로 나누고, 분할과 결합에 $\Theta(n^2)$ 시간이 걸리는 분할정복 알고리즘을 사용한다. 그가 스트라센 알고리즘을 이기려면 얼마나 많은 부분 문제를 만들어야 하는지 알아야 한다. 그의 알고리즘이 a 개의 부분 문제를 만든다면 수행시간 $T(n)$ 에 대한 점화식은 $T(n) = aT(n/4) + \Theta(n^2)$ 이 된다. Caesar 교수의 알고리즘이 스트라센 알고리즘보다 빠르기 위한 가장 큰 정수 a 는 무엇인가?
 $\Rightarrow \Theta(n^{\log_2 7})$

스트라센 알고리즘은 $\Theta(n^{\log_2 7}) = \Theta(n^{\log_4 49})$

$n^{\log_4 \alpha - \epsilon}$ 이 n^2 이상 같아질수 있는 $\epsilon > 0$ 은
 α 가 16보다 클때 항상 선택할수 있으므로
 $\alpha > 16$ 이면 $\Theta(n^{\log_4 \alpha})$ 이다.
 $\Theta(n^{\log_4 \alpha}) < \Theta(n^{\log_4 49})$ 이면 $\alpha = 48$ 이면 된다.

5.1-1

HIRE-ASSISTANT 프로시저의 4행에서 어떤 지원자가 가장 적합한지를 결정할 수 있다면 지원자들의 전체 순위를 알 수 있게 됨을 보여라.
implies

↳ 영어로 된 문제 확인

5.2-1

HIRE-ASSISTANT에서 지원자가 무작위 순서로 온다고 할 때 한 번만 고용할 확률은 얼마인가? 그리고 정확히 n 번 고용할 확률은 얼마인가?

$$\textcircled{1} \text{ 한 번만 고용할 확률} = \frac{1}{n}$$

$$\textcircled{2} \text{ 정확히 } n \text{ 번 고용할 확률} = \frac{1}{n} \times \frac{1}{n-1} \times \cdots \times 1 = \frac{1}{n!}$$

5.2-2

HIRE-ASSISTANT에서 지원자가 무작위 순서로 온다고 할 때 정확하게 두 번만 고용할 확률은 얼마인가? 지원자를 $1 \sim n$ 이라 하자 (n 이 가장 높은 경우)

① 첫번째 지원자는 항상 고용됨

E_i 를 i 번째 랭크 사람이 첫번째 지원자로 오는 경우라 하자.
아래 처음이 i 일때 두 번만 고용될 확률을 구해 보자.

$$= P_i$$

$P_i = \frac{1}{n-i}$. 처음이 i 일때 $1, \dots, i-1$ 까지는 어떤 순서로 와도 상관 없고, 오직 $i+1, \dots, n$ 까지가 2번만 고용될 확률에 영향을 주며
그때 n 은 $i+1, \dots, n$ 중 가장 먼저 나와야 하고 이 확률이 $n-i$ 이다.

모든 i 에 대한 확률을 더하면

$$\sum_{i=1}^{n-1} \underbrace{\frac{1}{n} \times \frac{1}{n-i}}_{\downarrow \rightarrow P_i} \quad \begin{array}{l} \text{↑} \\ \text{n은 처음 오면} \\ \text{안됨} \end{array}$$

i 가 처음 올 확률

5.3-1

Marceau 교수는 보조정리 5.5의 증명에 사용된 루프 불변성에 반대한다. 그는 첫 반복 이전에 루프 불변성 조건을 만족하는지 의문을 품고 있다. 그의 논리는 빈 부분 배열이 어떠한 0 순열들도 포함하지 않는다고 쉽사리 선언할 수 있다는 것이다. 그러므로 빈 부분 배열이 0 순열을 포함할 확률은 0이어야 한다. 이는 첫 반복 이전에 루프 불변성을 깬다. 함수 RANDOMIZE-IN-PLACE를 첫 반복 이전에 관련된 루프 불변성이 비지 않은 부분 배열에 적용할 수 있도록 다시 작성하고 작성한 함수에 대한 것으로 보조정리 5.5의 증명을 바꿔라.

1 $n = A.length$
2 if $n \leq 1$,
then A
3 for $i = 2$ to n
 $A[i] \leftarrow A[\text{Random}(i, n)]$ 교환

5.3-2

Kelp 교수는 동일한 순열을 제외하고는 모든 순열을 임의로 만들 수 있도록 하는 함수를 만들기로 결심했다. 그는 다음 프로시저를 제안했다. 이 코드는 Kelp 교수가 의도한대로 동작하는가?

PERMUTE-WITHOUT-IDENTITY(A)

- 1 $n = A.length$
- 2 **for** $i = 1$ **to** $n - 1$
- 3 swap $A[i]$ with $A[\text{RANDOM}(i + 1, n)]$

한 원소 $[1, 2, 3, 4, 5]$ 순열이 있을 때
 $[1 3 2 4 5]$ 순열을 만들려고 하지만
무조건 첫 번째 1은 2~5 번째 자리에 오기 때문에
만들 수 없다.

6.1-1

높이가 h 인 힙의 최대, 최소 원소 수는 각각 얼마인가?

<p>0 높이 $\begin{array}{c} 0 \\ \diagup \quad \diagdown \\ 1 \quad 1 \end{array} \quad - 2^0$</p> <p>1 높이 $\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ \diagup & \diagdown & \diagup & \diagdown & \diagup \\ 1 & 1 & 1 & 1 & 1 \end{array} \quad - 2^1$</p> <p>$\vdots$</p> <p>$h$ 높이: $\underbrace{\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \diagup & \diagdown & \diagup & \diagdown & \diagup & \cdots & \diagup \\ 1 & 1 & 1 & 1 & 1 & \cdots & 1 \end{array}}_{2^{h-1} \text{ 개}} \quad - 2^h$</p>	<p>최대 원소 수 = $2^0 + 2^1 + \dots + 2^h$ = $2^{h+1} - 1$</p> <p>최소 원소 수는 최대 원소 수에서 $2^h - 1$ 개 만큼 원소 수가 적아진 경우 = 2^h</p>
--	--