2020●12468 김현서

1. 시간 복잡도 계산 (빅오 계산)

① tiling 이용 X (n×n 정방 행렬 곱셈)

$$A=\begin{bmatrix} A_{00} & A_{01} & A_{02} & \cdots & A_{0n} \\ A_{10} & A_{11} & A_{12} & \cdots & A_{1n} \\ A_{20} & A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & & & & \vdots \\ A_{n0} & A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \quad B=\begin{bmatrix} B_{00} & B_{01} & B_{02} & \cdots & B_{0n} \\ B_{10} & B_{11} & B_{12} & \cdots & B_{1n} \\ B_{20} & B_{21} & B_{22} & \cdots & B_{2n} \\ \vdots & & & & \vdots \\ B_{n0} & B_{n1} & B_{n2} & \cdots & B_{nn} \end{bmatrix}$$

n개 / n개 / n개 / n개

$$C(\text{multiple})=\begin{bmatrix} C_{00} & C_{01} & C_{02} & \cdots & C_{0n} \\ C_{10} & C_{11} & C_{12} & \cdots & C_{1n} \\ C_{20} & C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & & & & \vdots \\ C_{n0} & C_{n1} & C_{n2} & \cdots & C_{nn} \end{bmatrix}$$

$C_{00} = (A_{00}\times B_{00})+(A_{01}\times B_{10})+(A_{02}\times B_{20})+\cdots+(A_{0n}\times B_{n0})$

…> 총 n개의 항을 더함. (A에서 n개 × B에서 n개)

$C_{01} = (A_{00}\times B_{01})+(A_{01}\times B_{11})+(A_{02}\times B_{21})+\cdots+(A_{0n}\times B_{n1})$

…> 총 n개의 항을 더함 (A에서 n개 × B에서 n개)

$C_{nn} = (A_{n0}\times B_{0n})+(A_{n1}\times B_{1n})+(A_{n2}\times B_{2n})+\cdots+(A_{nn}\times B_{nn})$

…> 총 n개의 항을 더함 (A에서 n개 × B에서 n개)

$\therefore T(n) = T(\text{divide})+T(\text{conquer})+T(\text{merge})$

∴ 1×1, 2×2, 3×3, … n×n 이 총 n개 있을 …

{행렬 곱 n번, 행렬 합 n번} 인 행렬이 n^2개 존재

$\therefore T(n) = \{\Theta(n)+n\times n+\Theta(n)\}\times n^2$

$= n^4+2\cdot n^3 = \Theta(n^4)+\Theta(n^3) \Rightarrow \Theta(n^4)$

2020 12468 김현서

1. 시간 복잡도 계산 (빅오 계산, 중간과정 포함)

ⓐ tiling 사용 o (n×n 정방행렬. 단, n은 2의 제곱꼴)

$$
A=\begin{bmatrix}
A_{00} & A_{01} & A_{02} & \cdots & A_{0(\frac{n}{2}-1)} & A_{0(\frac{n}{2})} & \cdots & A_{0n} \\
A_{10} & A_{11} & A_{12} & \cdots & A_{1(\frac{n}{2}-1)} & A_{1(\frac{n}{2})} & \cdots & A_{1n} \\
A_{20} & A_{21} & A_{22} & \cdots & A_{2(\frac{n}{2}-1)} & A_{2(\frac{n}{2})} & \cdots & A_{2n} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
A_{(\frac{n}{2}-1)0} & A_{(\frac{n}{2}-1)1} & A_{(\frac{n}{2}-1)2} & \cdots & A_{(\frac{n}{2}-1)(\frac{n}{2}-1)} & A_{(\frac{n}{2}-1)(\frac{n}{2})} & \cdots & A_{(\frac{n}{2}-1)n} \\
A_{(\frac{n}{2})0} & A_{(\frac{n}{2})1} & A_{(\frac{n}{2})2} & \cdots & A_{(\frac{n}{2})(\frac{n}{2}-1)} & A_{(\frac{n}{2})(\frac{n}{2})} & \cdots & A_{(\frac{n}{2})n} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
A_{n0} & A_{n1} & A_{n2} & \cdots & A_{n(\frac{n}{2}-1)} & A_{n(\frac{n}{2})} & \cdots & A_{nn}
\end{bmatrix}
\begin{matrix}
\\ \\ \frac{n}{2}\text{개} \\ \\ \\ \\ \frac{n}{2}\text{개} \\ \\ \\
\end{matrix}
$$

(열 방향 $\frac{n}{2}$개 / $\frac{n}{2}$개)

$$
\times
$$

$$
B=\begin{bmatrix}
B_{00} & B_{01} & B_{02} & \cdots & B_{0(\frac{n}{2}-1)} & B_{0(\frac{n}{2})} & \cdots & B_{0n} \\
B_{10} & B_{11} & B_{12} & \cdots & B_{1(\frac{n}{2}-1)} & B_{1(\frac{n}{2})} & \cdots & B_{1n} \\
B_{20} & B_{21} & B_{22} & \cdots & B_{2(\frac{n}{2}-1)} & B_{2(\frac{n}{2})} & \cdots & B_{2n} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
B_{(\frac{n}{2}-1)0} & B_{(\frac{n}{2}-1)1} & B_{(\frac{n}{2}-1)2} & \cdots & B_{(\frac{n}{2}-1)(\frac{n}{2}-1)} & B_{(\frac{n}{2}-1)(\frac{n}{2})} & \cdots & B_{(\frac{n}{2}-1)n} \\
B_{(\frac{n}{2})0} & B_{(\frac{n}{2})1} & B_{(\frac{n}{2})2} & \cdots & B_{(\frac{n}{2})(\frac{n}{2}-1)} & B_{(\frac{n}{2})(\frac{n}{2})} & \cdots & B_{(\frac{n}{2})n} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
B_{n0} & B_{n1} & B_{n2} & \cdots & B_{n(\frac{n}{2}-1)} & B_{n(\frac{n}{2})} & \cdots & B_{nn}
\end{bmatrix}
\begin{matrix}
\\ \\ \frac{n}{2}\text{개} \\ \\ \\ \\ \frac{n}{2}\text{개} \\ \\ \\
\end{matrix}
$$

(열 방향 $\frac{n}{2}$개 / $\frac{n}{2}$개)

$$
=C=\begin{bmatrix}
C_{00} & C_{01} & C_{02} & \cdots & C_{0(\frac{n}{2}-1)} & C_{0(\frac{n}{2})} & \cdots & C_{0n} \\
C_{10} & C_{11} & C_{12} & \cdots & C_{1(\frac{n}{2}-1)} & C_{1(\frac{n}{2})} & \cdots & C_{1n} \\
C_{20} & C_{21} & C_{22} & \cdots & C_{2(\frac{n}{2}-1)} & C_{2(\frac{n}{2})} & \cdots & C_{2n} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
C_{(\frac{n}{2}-1)0} & C_{(\frac{n}{2}-1)1} & C_{(\frac{n}{2}-1)2} & \cdots & C_{(\frac{n}{2}-1)(\frac{n}{2}-1)} & C_{(\frac{n}{2}-1)(\frac{n}{2})} & \cdots & C_{(\frac{n}{2}-1)n} \\
C_{(\frac{n}{2})0} & C_{(\frac{n}{2})1} & C_{(\frac{n}{2})2} & \cdots & C_{(\frac{n}{2})(\frac{n}{2}-1)} & C_{(\frac{n}{2})(\frac{n}{2})} & \cdots & C_{(\frac{n}{2})n} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
C_{n0} & C_{n1} & C_{n2} & \cdots & C_{n(\frac{n}{2}-1)} & C_{n(\frac{n}{2})} & \cdots & C_{nn}
\end{bmatrix}
\begin{matrix}
\\ \\ \frac{n}{2}\text{개} \\ \\ \\ \\ \frac{n}{2}\text{개} \\ \\ \\
\end{matrix}
$$

(열 방향 $\frac{n}{2}$개 / $\frac{n}{2}$개)

20201246B 김현서 1-② tiling 사용 o.

Recursive Matrix Multiplication 에 적용하면,

nxn 정방 행렬을 $\frac{n}{2} \times \frac{n}{2}$ 정방 행렬 4개로 나눈 뒤

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{22} = A_{22} \cdot B_{12} + A_{22} \cdot B_{22}$$

행렬곱 8번
행렬합 4번

각각의 행렬을 내적해 공간 복잡도를 줄이는 것을 의미합니다.

nxn 정방 행렬을 내적하는 데 필요한 공간복잡도를 T(n) 이라 한다면

"T(n) = T(divide) + T(conquer) + T(merge)" 입니다.

T(divide)는 을 하나를 결정하는 공간 복잡도로 $\Theta(1)$ 을 가집니다.

T(conquer)은 $\frac{n}{2} \times \frac{n}{2}$ 내적을 8번 수행하므로 $8 \times T(\frac{n}{2})$과 같습니다.

T(merge)는 nxn 정방행렬 전체요소 n^2개의 1/4만큼 덧셈 연산을 수행합니다. 따라서 $\frac{1}{4}\Theta(n^2)$ 과 같습니다.

$$\therefore T(n) = \underbrace{\Theta(1)}_{\text{divide의 복잡도}} + \underbrace{8(T(\tfrac{n}{2}))}_{\text{conquer의 복잡도}} + \underbrace{\tfrac{1}{4}\Theta(n^2)}_{\text{merge의 복잡도}} = 8T(\tfrac{n}{2}) + \Theta(n^2)$$

* Master Method 기법 ($T(n) = aT(\frac{n}{b}) + f(n)$ 형태)

$a \geq 1$, $b > 1$, $f(n) > 0$ 일 때

$n^{\log_b a} > f(n) \rightarrow T(n) = \Theta(n^{\log_b a})$ 이용.

$T(n) = \Theta(n^{\log_2 8}) + \Theta(n^2) = \Theta(n^3) + \Theta(n^2) \Rightarrow \boxed{T(n) = \Theta(n^3)}$

($\Theta(n^3) > \Theta(n^2)$ 이므로 만족)

2020123468 김현서

2. 타일링을 사용하면 얻어지는 장점에 대한 설명. (공간복잡도를 어떻게 ↓?)

타일링을 사용하면 하나의 문제를 작은 문제로 나눠 순환적으로 푸는 하향 접근 방식으로 문제를 풀 수 있다.

이 쪼개진 작은 문제들은 원래의 문제와 동일한 성격과 특징을 가지고, 그렇기 때문에 재귀함수의 형태로 나타낼 수 있다.

해결한 작은 문제들을 이용해 윗단계의 문제를 해결한다.