Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

Strassen's Matrix Multipication

Suppose that $A_{n \times n}$ and $B_{n \times n}$ are two matrices and we want to compute $C_{n \times n} = A_{n \times n} \times B_{n \times n}$.

Strassen's Matrix Multipication

Suppose that $A_{n \times n}$ and $B_{n \times n}$ are two matrices and we want to compute $C_{n \times n} = A_{n \times n} \times B_{n \times n}$.

- Direct approach: has $O(n^3)$ time complexity.

Strassen's Matrix Multipication

Suppose that $A_{n \times n}$ and $B_{n \times n}$ are two matrices and we want to compute $C_{n \times n} = A_{n \times n} \times B_{n \times n}$.

- Direct approach: has $O(n^3)$ time complexity.
- Divide and Conquer:

Strassen's Matrix Multipication

Suppose that $A_{n\times n}$ and $B_{n\times n}$ are two matrices and we want to compute $C_{n\times n} = A_{n\times n} \times B_{n\times n}$.

- Direct approach: has $O(n^3)$ time complexity.
- Divide and Conquer:

$$\underset{C_{n\times n}}{\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}} = \underset{A_{n\times n}}{\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}} \times \underset{B_{n\times n}}{\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}} \quad n/2,\ n/2$$

Now we have:

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

In this case we have $T(n) = 8T(n/2) + O(n^2) = \Theta(n^3)$! How we can reduce the time complexity?

Strassen's Matrix Multipication

In order to improve the algorithm we have to reduce the number of multiplication by introducing the new variables as follows:

- $P = (A_{11} + A_{22}) \times (B_{11} + B_{22})$
- $Q = (A_{21} + A_{22}) \times B_{11}$
- $R = A_{11} \times (B_{12} - B_{22})$
- $S = A_{22} \times (B_{21} - B_{11})$
- $T = (A_{11} + A_{12}) \times B_{22}$
- $U = (A_{21} - A_{11}) \times (B_{11} + B_{12})$
- $V = (A_{12} - A_{22}) \times (B_{21} + B_{22})$

Strassen's Matrix Multipication

In order to improve the algorithm we have to reduce the number of multiplication by introducing the new variables as follows:

- $P = (A_{11} + A_{22}) \times (B_{11} + B_{22})$
- $Q = (A_{21} + A_{22}) \times B_{11}$
- $R = A_{11} \times (B_{12} - B_{22})$
- $S = A_{22} \times (B_{21} - B_{11})$
- $T = (A_{11} + A_{12}) \times B_{22}$
- $U = (A_{21} - A_{11}) \times (B_{11} + B_{12})$
- $V = (A_{12} - A_{22}) \times (B_{21} + B_{22})$

Now, we have:

- $C_{11} = P + S - T + V$
- $C_{12} = R + T$
- $C_{21} = Q + S$
- $C_{22} = P + R - Q + U$

Strassen's Matrix Multipication

In order to improve the algorithm we have to reduce the number of multiplication by introducing the new variables as follows:

- $P = (A_{11} + A_{22}) \times (B_{11} + B_{22})$
- $Q = (A_{21} + A_{22}) \times B_{11}$
- $R = A_{11} \times (B_{12} - B_{22})$
- $S = A_{22} \times (B_{21} - B_{11})$
- $T = (A_{11} + A_{12}) \times B_{22}$
- $U = (A_{21} - A_{11}) \times (B_{11} + B_{12})$
- $V = (A_{12} - A_{22}) \times (B_{21} + B_{22})$

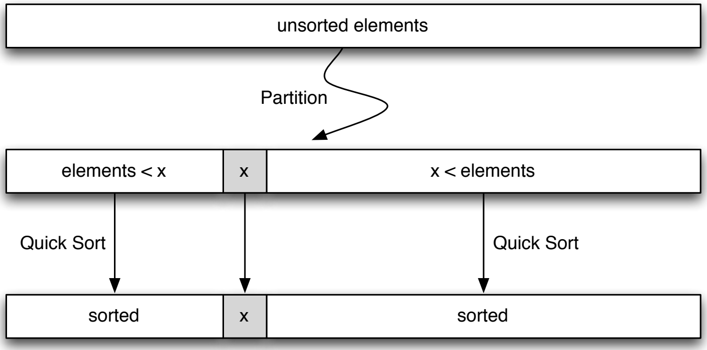
Now, we have:

- $C_{11} = P + S - T + V$
- $C_{12} = R + T$
- $C_{21} = Q + S$
- $C_{22} = P + R - Q + U$

and so the $T(n)$ can be expressed as:

$$T(n) = 7T(n/2) + O(n^2) = \Theta(n^{Log_2(7)}).$$

Quick Sort



unsorted elements
Partition
elements < x
x
x < elements
Quick Sort
Quick Sort
sorted
x
sorted

Quick Sort: Algorithm

$$
\begin{array}{l}
\textit{Quick Sort}(A,\ s,\ e)\{ \\
\quad \textit{if}(s < e)\{ \\
\qquad \textit{Partition}(A,\ s,\ e,\ m); \\
\qquad \textit{Quick Sort}(A,\ s,\ m-1); \\
\qquad \textit{Quick Sort}(A,\ m+1,\ e); \\
\quad \} \\
\}
\end{array}
$$

Quick Sort: Algorithm

$$
\begin{array}{l}
\textit{Quick Sort}(A,\ s,\ e)\{ \\
\quad \textit{if}(s < e)\{ \\
\qquad \textit{Partition}(A,\ s,\ e,\ m); \\
\qquad \textit{Quick Sort}(A,\ s,\ m-1); \\
\qquad \textit{Quick Sort}(A,\ m+1,\ e); \\
\quad \} \\
\}
\end{array}
$$

$$
\begin{array}{l}
\textit{Partition}(A,\ s,\ e,\ m)\{ \\
\quad x \leftarrow A[s];\ i \leftarrow s+1;\ j \leftarrow e; \\
\quad \textit{do}\{ \\
\qquad \textit{while}(A[i] < x)\ i++; \\
\qquad \textit{while}(A[j] > x)\ j--; \\
\qquad \textit{if}(i < j)\ \textit{swap}(A[i],\ A[j]); \\
\quad \}\textit{while}(i < j); \\
\quad \textit{swap}(A[s],\ A[j]); \\
\quad \textit{return}(j); \\
\}
\end{array}
$$

Quick Sort: Analysis

- **Best case:** when Partition divides the input array into two subarrays with almost equal length.

$$T(n) = 2T(n/2) + O(n) \Longrightarrow T(n) = O(n.Log(n)).$$

Quick Sort: Analysis

- **Best case:** when Partition divides the input array into two subarrays with almost equal length.

$$T(n) = 2T(n/2) + O(n) \Longrightarrow T(n) = O(n.Log(n)).$$

- **Worse case:** when Partition fails to divide the input array into two subarrays.

$$T(n) = T(n-1) + O(n) \Longrightarrow T(n) = \Theta(n^2).$$

Quick Sort: Analysis

- **Best case:** when Partition divides the input array into two subarrays with almost equal length.

$$T(n) = 2T(n/2) + O(n) \Longrightarrow T(n) = O(n.Log(n)).$$

- **Worse case:** when Partition fails to divide the input array into two subarrays.

$$T(n) = T(n-1) + O(n) \Longrightarrow T(n) = \Theta(n^2).$$

- **Average case:** Average over all possible length for subarrays...

$$\begin{array}{rcccccc}
T(n) = & T(0) & + & T(n-1) & + & O(n) \\
T(n) = & T(1) & + & T(n-2) & + & O(n) \\
\vdots & \vdots & & \vdots & & \\
T(n) = & T(n-2) & + & T(1) & + & O(n) \\
T(n) = & T(n-1) & + & T(0) & + & O(n) \\
\hline
nT(n) = & \sum_{i=0}^{n-1} T(i) & + & \sum_{i=0}^{n-1} T(i) & + & nO(n)
\end{array}$$

Quick Sort: Average case analysis

$$T(n) = \frac{2}{n}\sum_{i=0}^{n-1} T(i) + cn$$

Now we prove that $T(n) = O(n.Log(n))$ (This is just a guess!). The proof is based on induction:

- **Initiation:** $n = 2 \Longrightarrow T(2) = T(1) + 2c = O(1).\surd$

Quick Sort: Average case analysis

$$T(n) = \frac{2}{n}\sum_{i=0}^{n-1} T(i) + cn$$

Now we prove that $T(n) = O(n.Log(n))$ (This is just a guess!). The proof is based on induction:

- **Initiation:** $n = 2 \Longrightarrow T(2) = T(1) + 2c = O(1).\surd$
- **Hypothesis:** $\forall i < n \Longrightarrow T(i) = O(i.Log(i)) \leq c' i.Log(i).\surd$

Quick Sort: Average case analysis

$$T(n) = \frac{2}{n}\sum_{i=0}^{n-1} T(i) + cn$$

Now we prove that $T(n) = O(n.Log(n))$ (This is just a guess!). The proof is based on induction:

- **Initiation:** $n = 2 \Longrightarrow T(2) = T(1) + 2c = O(1).\surd$
- **Hypothesis:** $\forall i < n \Longrightarrow T(i) = O(i.Log(i)) \leq c' i.Log(i).\surd$
- **Induction step:** prove the statement for n:

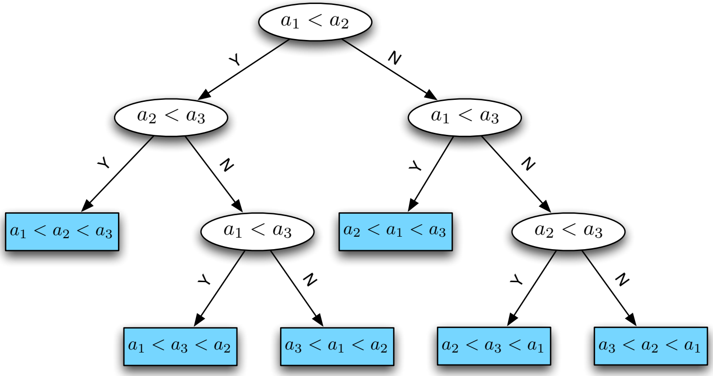
$$\begin{aligned}
T(n) &\leq \frac{2}{n}\sum_{i=0}^{n-1} c' i.Log(i) + cn \\
&\leq \frac{2c'}{n}\left(\sum_{i=0}^{n/2} i.Log(n/2) + \sum_{i=n/2+1}^{n-1} i.Log(n)\right) + cn \\
&\leq \frac{2c'}{n}\left(\sum_{i=0}^{n/2} i.Log(n) - \sum_{i=0}^{n/2} i + \sum_{i=n/2+1}^{n-1} i.Log(n)\right) + cn \\
&\leq \frac{2c'}{n}\left(Log(n)\frac{n(n-1)}{2} - \frac{(n/2)(n/2-1)}{2}\right) + cn \\
&= O(n.Log(n)).\surd
\end{aligned}$$

Sorting Lower Bound

At lease how many comparisons we need to sort three items?

Sorting Lower Bound

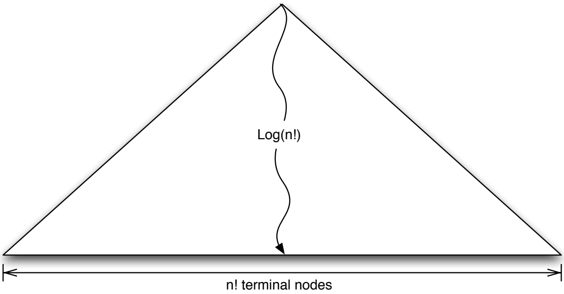
At lease how many comparisons we need to sort three items?



We need at least $3 = \lceil Log_2(3!) \rceil$ to sort three items.

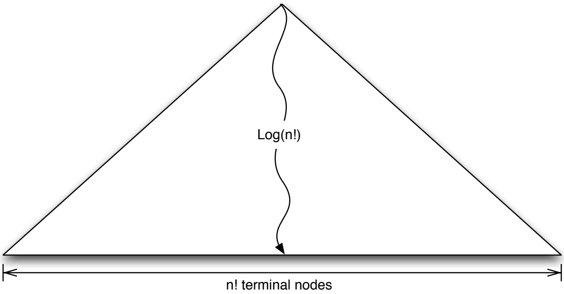
Sorting Lower Bound

In general we have:



Sorting Lower Bound

In general we have:



$$n! \simeq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\Theta(\frac{1}{n})\right) \qquad \text{(Stirling Formula)}$$

$$\Longrightarrow \log_2 n! \simeq \log_2(\sqrt{2\pi n})+\log_2\left(\frac{n}{e}\right)^n+\log_2\left(1+\Theta(\frac{1}{n})\right)$$

$$\Longrightarrow \log_2 n! = \Omega(n\log_2(n)).$$

Exercises

1. Show the details of Matrix Multiplication in which each matrix is divided into nine blocks (each of size $n/3 \times n/3$).
2. Draw a comparison tree for five elements and then show that at most six comparisons are enough to find the median of five elements.