1 Randomized Median Finding

First review the randomized $k$th finding algorithm quickselect $(A, k)$:

- Pick pivot $p$ randomly
- Split using partition algorithm and the size of LESS part is $L$
- This step has three situations:
  - $L = k - 1$ output $p$
  - $L > k - 1$ output quickselect($\text{LESS}, k$)
  - $L < k - 1$ output quickselect($\text{GREATER}, k$)

To estimate running time of the algorithm, we assume that the original array has equal probability $\frac{1}{n}$ to be divided into two parts with length $k$ and $n - k$, $k = 1, \cdots, n$. Thus, the probability that a partition leads to a longer part with length $\frac{n}{2}$ to $n$ is $\frac{2}{n}$.

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

We guess $T(n) \leq 4n$ and prove this conclusion with mathematical induction.

When $n = 1$, $T(1) = O(1)$ the conclusion hold;
Assume $T(i) \leq 4i$ for all $i < n$, then

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

$$\leq (n-1) + \frac{2}{n} \sum_{i=\frac{n}{2}}^{n} 4i = (n-1) + 3n = 4n - 1 \leq 4n$$

Thus, based on the theory of mathematical induction $T(n) \leq 4n$ holds for all $n$. 

1
2 Matrix Multiplication

First, we assume the matrices are square and with matrix order $n$ equal to power of 2. The naive matrix multiplication idea is:

- For every row of $A$
- For every column of $B$
  - go through every element, multiply them together and sum

$$A \cdot B = C \iff \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} -1 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -3 & 10 \\ -1 & 4 \end{bmatrix}$$

The input size is $\theta(n^2)$ and the algorithm is as follows:

$$C_{ij} = (\text{ith row of } A) \cdot (\text{jth column of } B)$$

$$= \sum_{k=1}^{n} A_{ik}B_{kj}$$

The lower bound of the matrix multiplication is $\Omega(n^2)$. And there are some very efficient algorithms:

- $n^{2.8}$ Strassen 1969
- $n^{2.3755}$ Coppersmith and Winograd 1987
- $n^{2.3727}$ William 2011

2.1 Divide and Conquer

The idea is

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}_{n \times n} \begin{bmatrix} E & F \\ G & H \end{bmatrix}_{n \times n} = \begin{bmatrix} AE + BG & AF + BH \\ CF + DG & CF + DH \end{bmatrix}_{n \times n}$$

The algorithm can be summarized into following steps:

- Split into $\frac{n}{2} \times \frac{n}{2}$ submatrices
- Call recurrence and do matrix multiplication 8 times
- Add the matrix together and takes $O(n^2)$

The operation is

$$T(n) = 8T\left(\frac{n}{2}\right) + O(n^2)$$

$$T(1) = O(1)$$
With a tree describes the recursive relationship

\[ T(n) = \sum_{k=1}^{\log n} 8^k \cdot \left( \frac{n}{2^k} \right)^2 \]

\[ = \sum_{k=1}^{\log n} 2^k n^2 = n^2 \sum_{k=1}^{\log n} 2^k \]

\[ = n^2 \left( 2^{\log n+1} - 1 \right) = n^2 (n - 1) \]

### 2.2 Strassen’s Algorithm

The idea of the algorithm is:

- recursively compute only 7 products
- do the necessary addition and subtractions \( O(n^2) \)

The 7 products are:

- \( M_1 = (A + D) (E + H) \)
- \( M_2 = (C + D) E \)
- \( M_3 = A (F - H) \)
- \( M_4 = D (G - E) \)
- \( M_5 = (A + B) H \)
- \( M_6 = (C - A) (E + F) \)
- \( M_7 = (B - D) (G + H) \)

then we have:

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\begin{bmatrix}
E & F \\
G & H
\end{bmatrix} =
\begin{bmatrix}
M_1+M_4-M_5+M_7 & M_3+M_5 \\
M_2+M_4 & M_1-M_2+M_3+M_6
\end{bmatrix}
\]

The operation time is

\[ T(n) = 7T \left( \frac{n}{2} \right) + O(n^2) \]

\[ = \sum_{k=0}^{\log n} 7^k \cdot \left( \frac{n}{2^k} \right)^2 \]

\[ = n^2 \sum_{k=0}^{\log n} 7^k \left( \frac{1}{4} \right)^k \]

\[ = n^2 \log_2 7 = n^{2.8} \]