Algorithm Analysis

### Partitioning into Two Sublists

1. Choose $x$.
2. Set $i = 2$ and set $f = n$.
3. If $x < A[i]$, then $f = i + 1$, else go to step 5.
4. If $x > A[i]$, then $i = i + 1$, else go to stop.
5. If $A[i] > x$, then $i = f$.
6. Repeat steps 2 through 5.

Additional work is necessary to combine them.

1. Divide the given sequence (array) in two nonempty subsequences such that each subarray contains all elements smaller than $x$.
2. Combine the two subarrays by recursive calls to quick-sort.
3. Combine in quick-sort the subarrays are sorted in place; therefore no special treatment is needed.

Example: Quick-Sort

<table>
<thead>
<tr>
<th>66</th>
<th>99</th>
<th>64</th>
<th>16</th>
<th>86</th>
<th>13</th>
<th>27</th>
<th>0</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>z</td>
<td>z</td>
<td>f</td>
<td>i</td>
<td>f</td>
<td>i</td>
<td>f</td>
<td>i</td>
<td>f</td>
</tr>
</tbody>
</table>
Analysis of QuickSort

Worst Case Analysis $O(N^2)$

We will consider the average analysis of QuickSort.

Assumptions:

1. $A[1], ..., A[n]$ are all distinct.
2. $A[i] \in \{1, 2, ..., n\}$ for $i = 1, ..., n$.
3. $A[i]$'s are randomly distributed.
4. Suppose $AC(n) = \text{Average number of comparisons required to sort } A[1], ..., A[n]$ by QuickSort.
5. Then, $AC(n) = (n + 1) + \frac{1}{n+1} \sum_{k=2}^{n} AC(k) + AC(n-k)$,
6. $nAC(n) = n(n+1) + 2AC(0) + ... + AC(n-1) + AC(n)$,
7. Consequently, $AC(n) = \frac{n(n+1) + 2AC(0) + ... + AC(n-1) + AC(n)}{n}$.

Substituting $(n+1)$ for $n$ in the above equation gives:

$(n+1)AC(n+1) = (n+1)(n+2) + 2AC(0) + ... + AC(n) + AC(n+1)$.

Observations:

1. All $n!$ permutations of $A[1], ..., A[n]$ are equally likely.
2. Suppose $A[i] = k$. Then after one application of the partition method, size of the left subarray is $(k-1)$, size of the right subarray is $(n-k)$, and $k$ is in its correct position.
3. Each $(n+1)$ comparisons are made in arranging the given array $A[i], ..., A[n]$ in partition method.
4. Each time $i$ is increased by 1, the comparison is decreased by 1, until $i$ becomes equal to $j$.

Algorithm QuickSort $A[1], ..., A[n]$

1. Find a partition of $A[1], ..., A[n]$ such that $A[1], ..., A[q]$ are all smaller than (new) $A[q+1], ..., A[n]$ for $1 \leq q \leq n-1$.
3. $A[q]$ is the median of the $n$ values.
4. Other choices for $X$ are $A[1], A[2], ..., A[n]$ and randomly chosen value i.e., $A[i]$. It makes sense to sort an array by special procedures such as BubbleSort, for small values of $n$, i.e., $n \leq M$. Practically choose $M \approx 9$. 

Divide and Conquer (Part 15)

Divide and Conquer (Part 15)

Divide and Conquer (Part 15)

Divide and Conquer (Part 15)
\[ (1 - u)\delta + \frac{1 + u}{\varepsilon} + \frac{\zeta + u}{\varepsilon} = (1 + u)\delta \]

Repeated substitution gives:

\[ (u)\delta + \frac{\zeta + u}{\varepsilon} = (1 + u)\delta \]

Let

\[ (1 + u)\varphi + \frac{\zeta + u}{\varepsilon} = \frac{(\zeta + u)}{(1 + u)\varphi} \]

Consequently

\[ (u)\varphi(\zeta + u) + (1 + u)\zeta = (1 + u)\varphi(1 + u) \]

and the difference of the above two is:

\[ \frac{(u)\varphi(\zeta + u)}{(1 + u)\varphi} \]

Example: Selection Problem

Find the path that leads to a given array \[ \{1\}V \cdots \{1\}V \]

Example: Selection Problem

Find the path that leads to a given array \[ \{1\}V \cdots \{1\}V \]
the (i - y)-th smallest element of \((u - y)\) of the right subarray \[a'_{1}[1]\] + \(1\).

This case can be seen as equivalent to finding \[a'_{1}[1]\] + \(1\) if \(\gamma < \gamma'\).

\[
\gamma = \gamma' \quad (1 + u) = (1 + u')f
\]

\[
\gamma > \gamma' \quad (1 + u')f + (1 + u) = (1 + u)f
\]

This indicates the average value by \(\gamma'\): i.e.,

\[
\gamma' = 1, 2, \ldots, n. \text{ So let us calculate the average of } f(1).
\]

But in general, we will be finding the \(i\)-th smallest element for any \(i\). In general, we will be finding the \(i\)-th smallest element for any \(i\).

\[
\gamma = 1, 2, \ldots, n. \text{ So let us calculate the average of } f(1).
\]

But in general, we will be finding the \(i\)-th smallest element for any \(i\).

\[
\gamma = 1, 2, \ldots, n. \text{ So let us calculate the average of } f(1).
\]

But in general, we will be finding the \(i\)-th smallest element for any \(i\).
\[(u)O = \frac{u}{(1 - u)} + (1 - u)f_0 \frac{u}{1 - \varepsilon u} = (u)f\]

\[\varepsilon(1 - u)f_0(1 - \varepsilon u) + (1 - u)\varepsilon u = (u)f \varepsilon u\]

\[(1 - u)f \{z(1 - u) + (1 - u)\varepsilon + \varepsilon u - u + \varepsilon u\} = (u)f \varepsilon u\]

\[\{u \varepsilon(1 - u) - (1 + u)\varepsilon\} = (1 - u)f \varepsilon(1 - u) - (u)f \varepsilon u\]

This and Carone (p. 36)

Example: Finding Closest Pair of Points

\[\text{Problem: Let } \{n \in \mathbb{Z} : 1 \leq n \leq 10 \} \text{ be a set. Find the closest pair of points.}\]
Algorithm

Finding Closest Pair of Points - Divide and Conquer

1. Find a set of points \( p \) in \( \mathbb{R}^2 \) and \( d \) and \( L \) respectively. (This is opposite of \( d \) and \( L \) respectively.)

2. Consider the points \( A \) and \( B \) and \( C \) and \( D \) respectively in \( \mathbb{R}^2 \). This can be done by the sorted \( X \) coordinate in \( \mathbb{R}^2 \). Consider the line \( L \) and \( d \).

3. Find the closest point \( x \) that points \( p \) in \( \mathbb{R}^2 \) in two central parts and \( d \) and \( L \).

4. Find a vertex \( \mathbb{R}^2 \) that points \( p \) in \( \mathbb{R}^2 \) and \( d \) and \( L \).

5. Find the closest point \( d \) and \( L \) from \( \mathbb{R}^2 \) and \( d \) and \( L \).

The initial set \( \mathbb{R}^2 \) is sorted for \( x \) and \( y \).

We proceed as follows.

1. Find the closest point \( d \) and \( L \) from \( \mathbb{R}^2 \) and \( d \) and \( L \).

2. Find the closest point \( d \) and \( L \) from \( \mathbb{R}^2 \) and \( d \) and \( L \).

3. Find the closest point \( d \) and \( L \) in \( \mathbb{R}^2 \) and \( d \) and \( L \).

4. Find the closest point \( d \) and \( L \) from \( \mathbb{R}^2 \) and \( d \) and \( L \).

We are done if \( d \) and \( L \) do not correspond to the closest pair.

If \( d \) and \( L \) do correspond to the closest pair.

We are done if \( d \) and \( L \) do not correspond to the closest pair.