QuickSort

**QuickSort**

- Sorts in place (no need for external storage)
- Continues to work in O(n log n) time even when initially unsorted
- Heapsort is easier to implement
- Worst-case time in O(n^2) (already sorted)

---

**Partition Procedure**

**Key to QuickSorts:**

1. **Initialize** $j$ to $n + 1$
2. **Initialize** $k$ to $-1$
3. **Initialize** $l$ to $0$
4. **Initialize** $j$ to $0$
5. **Initialize** $k$ to $0$
6. **Initialize** $l$ to $0$
7. **Initialize** $j$ to $0$
8. **Initialize** $k$ to $0$
9. **Initialize** $l$ to $0$
10. **Initialize** $j$ to $0$
11. **Initialize** $k$ to $0$
12. **Initialize** $l$ to $0$

**Algorithm:**

1. **Partition:**
   - **Input:** Array $A$ and index $r$
   - **Output:** Arrays $A'$ and $A''$
   - **Steps:**
     1. While $i < j$
        1. If $A[i] > A[r]$
           1. Increment $i$
           2. If $i > r$
              1. Swap $A[i]$ and $A[j]$
          3. If $i < j$
             1. Swap $A[i]$ and $A[j]$

---

**Java Code:**

```java
public static void quicksort(int[] arr, int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quicksort(arr, low, pi - 1);
        quicksort(arr, pi + 1, high);
    }
}

private static int partition(int[] arr, int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(arr, i, j);
        }
    }

    swap(arr, i + 1, high);
    return (i + 1);
}
```

---

**Notes:**

- QuickSort is a divide and conquer algorithm.
- It's efficient for large data sets and is implemented in many programming languages.
- The worst-case time complexity is O(n^2) but generally performs better.
Performance of QuickSort

The worst-case scenario for QuickSort is when the partitioning is unbalanced. The time complexity of QuickSort in this case is $O(n^2)$.

1. Choose a pivot.

There are various methods that can be used to pick the pivot. Common choices include:

- Random selection
- Median of three
- Median of five
- First element
- Last element
- Closest to the median

The choice of pivot affects the distribution of elements in the partitioning.

2. Partition the array.

- Choose a pivot element.
- Compare each element with the pivot.
- If the element is less than the pivot, place it to the left of the pivot.
- If the element is greater than or equal to the pivot, place it to the right of the pivot.

3. Recursively sort the subarrays.

- Sort the subarray to the left of the pivot.
- Sort the subarray to the right of the pivot.

The overall time complexity of QuickSort in the worst case is $O(n^2)$. However, in practice, QuickSort is often faster due to its good average-case performance.
Best-case Partitioning: Recurrence tree

\[ T(n) = \Theta(n \log n) \]

Apply Case 2 of Master Theorem

Recurrence is: \( T(n) = \begin{cases} \Theta(1/n) & n = 2 \end{cases} \]

Partition produces two regions: each of size \( n/2 \)

Propotional Split: Recurrence tree

Depth: \( \log_{10} n = \Theta(n) \)

Every level has \( n \) unit, until recursion terminates at depth \( \log_{10} n = \Theta(n) \)

Average-case Partitioning: Suppose partition always produces average case which closer to best case than to worst case

\[ u + (1/10)uL + (1/10)uL = \Theta(u) \]

Recurrence is: \( L(n) \)
We will prove this by induction.

We assume that $L_n^u$ is log in $L_n^u$ for even $n$. Then $L_{2n}^u = (u)L_n^u$.

The last step comes from the fact that the sum of the same, but in reverse order, is the same as the square of the sum of the first $n$ terms.

We assume that the proof is true for $n = 0$.

We state the inductive hypothesis.

We observe that $L_n^u = L_{n-1}^u + (1-u)L_{n-1}^u$.

We consider the case when $n$ is even.

We observe that $L_n^u = L_{n-1}^u + (1-u)L_{n-1}^u$.

We consider the case when $n$ is odd.

We observe that $L_n^u = L_{n-1}^u + (1-u)L_{n-1}^u$.

We observe that $L_n^u = L_{n-1}^u + (1-u)L_{n-1}^u$.

We conclude that the proposition holds for all $n$. Thus, the proposition is true.
There's enough

The last step can be obtained by choosing a

\[
\begin{align*}
q + u \log u & > \\
\left( \frac{v}{u} - q + u \right) + q + u \log u & > \\
\left( \frac{v}{u} - \frac{u}{QC} - q + u \right) + q + u \log u & = \\
\frac{v}{u} - u \log u + \frac{u}{QC} - q + u & = \\
\left( \frac{v}{u} - u \log u \right) \frac{u}{QC} + (1 - u) \frac{u}{QC} + u & > \\
\frac{v}{u} \sum_{i=0}^{u \log u} \frac{u}{QC} + (1 - u) \frac{u}{QC} + u & = (u) T,
\end{align*}
\]

Substituting we get

\[
\left( \frac{v}{u} - u \log u \right) \frac{u}{QC} \leq \frac{v}{u} \sum_{i=0}^{u \log u} \frac{u}{QC}
\]

Thus can be shown that

\[
\frac{v}{u} \sum_{i=0}^{u \log u} \frac{u}{QC}
\]

\[
\text{Lemma of Cauchy}
\]