# Sorting Algorithms

## Penjelasan:

1. Insertion Sort: Algoritma ini bekerja dengan membangun array yang sudah terurut secara bertahap. Pada setiap iterasi, elemen dari array yang belum terurut dipilih dan dimasukkan ke dalam posisi yang benar di array yang sudah terurut.  
  
2. Selection Sort: Algoritma ini bekerja dengan membagi array menjadi dua bagian: terurut dan belum terurut. Pada setiap langkah, elemen terkecil dari bagian yang belum terurut dipilih dan dipindahkan ke bagian terurut.  
  
3. Heap Sort: Algoritma ini menggunakan struktur data heap untuk membuat array terurut. Heap adalah struktur data berbentuk pohon biner yang memenuhi sifat heap.  
  
4. Merge Sort: Algoritma ini adalah algoritma rekursif yang membagi array menjadi dua bagian, mengurutkan masing-masing bagian, dan kemudian menggabungkannya kembali.  
  
5. Quick Sort: Algoritma ini adalah algoritma divide and conquer yang memilih elemen sebagai pivot dan mempartisi array ke dalam dua sub-array: elemen yang lebih kecil dari pivot dan elemen yang lebih besar.  
  
6. Counting Sort: Algoritma ini bekerja dengan menghitung jumlah elemen dengan nilai tertentu, kemudian menggunakan informasi tersebut untuk menentukan posisi elemen dalam array terurut.

## Tabel Time Complexity dan Space Complexity:

| Algorithm | Best Case | Average Case | Worst Case | Space Complexity |  
|-----------------|-------------------|-------------------|-------------------|-------------------|  
| Insertion Sort | O(n) | O(n^2) | O(n^2) | O(1) |  
| Selection Sort | O(n^2) | O(n^2) | O(n^2) | O(1) |  
| Heap Sort | O(n log n) | O(n log n) | O(n log n) | O(1) |  
| Merge Sort | O(n log n) | O(n log n) | O(n log n) | O(n) |  
| Quick Sort | O(n log n) | O(n log n) | O(n^2) | O(log n) |  
| Counting Sort | O(n + k) | O(n + k) | O(n + k) | O(n + k) |

# Searching Algorithms

## Penjelasan:

1. Binary Search: Algoritma ini digunakan untuk mencari elemen di array yang sudah terurut dengan cara membagi array menjadi dua bagian pada setiap iterasi.  
  
2. Linear Search: Algoritma ini mencari elemen dengan memeriksa setiap elemen satu per satu dari awal hingga akhir.  
  
3. Depth First Search (DFS): Algoritma ini digunakan untuk menjelajahi graf atau pohon dengan menjelajahi sebanyak mungkin cabang sebelum kembali.  
  
4. Breadth First Search (BFS): Algoritma ini digunakan untuk menjelajahi graf atau pohon dengan menjelajahi semua node pada tingkat yang sama sebelum turun ke tingkat berikutnya.

## Tabel Time Complexity dan Space Complexity:

| Algorithm | Best Case | Average Case | Worst Case | Space Complexity |  
|---------------------|-----------|--------------|------------|-------------------|  
| Binary Search | O(1) | O(log n) | O(log n) | O(1) |  
| Linear Search | O(1) | O(n) | O(n) | O(1) |  
| Depth First Search | O(V + E) | O(V + E) | O(V + E) | O(V) (stack) |  
| Breadth First Search| O(V + E) | O(V + E) | O(V + E) | O(V) (queue) |

# Graphs Algorithms

## Penjelasan:

1. Kruskal's Algorithm: Algoritma ini digunakan untuk menemukan Minimum Spanning Tree (MST) dengan memilih edge terkecil yang tidak membentuk siklus.  
  
2. Dijkstra's Algorithm: Algoritma ini digunakan untuk mencari jarak terpendek dari satu sumber ke semua node dalam graf dengan bobot non-negatif.  
  
3. Bellman-Ford Algorithm: Algoritma ini digunakan untuk mencari jarak terpendek dalam graf, termasuk graf dengan bobot negatif.  
  
4. Floyd-Warshall Algorithm: Algoritma ini digunakan untuk menemukan jarak terpendek antara semua pasangan node dalam graf.  
  
5. Topological Sort: Algoritma ini digunakan untuk mengurutkan node dalam graf berarah asiklik (DAG).  
  
6. Flood Fill Algorithm: Algoritma ini digunakan untuk mengganti warna pada area terhubung dalam grid.  
  
7. Lee Algorithm: Algoritma ini digunakan untuk menemukan jarak terpendek dalam grid menggunakan pendekatan BFS.

## Tabel Time Complexity dan Space Complexity:

| Algorithm | Time Complexity | Space Complexity |  
|-----------------------|----------------------|-------------------------|  
| Kruskal's Algorithm | O(E log E) | O(V + E) |  
| Dijkstra's Algorithm | O((V + E) log V) | O(V + E) |  
| Bellman-Ford Algorithm| O(V \* E) | O(V) |  
| Floyd-Warshall Algorithm | O(V^3) | O(V^2) |  
| Topological Sort | O(V + E) | O(V + E) |  
| Flood Fill Algorithm | O(N \* M) | O(N \* M) |  
| Lee Algorithm | O(N \* M) | O(N \* M) |