Three Important Algorithms in Python

1. Binary Search (Searching Algorithm)

Efficiently finds an element in a sorted list.

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1
    return -1

# Example
nums = [1, 3, 5, 7, 9, 11]
print(binary_search(nums, 7))  # Output: 3
```

2. Dijkstra's Algorithm (Shortest Path in Graphs)

Used to find the shortest path in weighted graphs.

```
import heapq

def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    pq = [(0, start)]

    while pq:
        current_distance, current_node = heapq.heappop(pq)

        if current_distance > distances[current_node]:
            continue

        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(pq, (distance, neighbor))

    return distances

# Example
graph = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 5},
    'C': {'A': 4, 'B': 2, 'D': 1},
    'D': {'B': 5, 'C': 1}
}
print(dijkstra(graph, 'A'))
# Output: {'A': 0, 'B': 1, 'C': 3, 'D': 4}
```

3. Merge Sort (Sorting Algorithm)

Efficient divide-and-conquer sorting algorithm.

```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
```

```
    mid = len(arr) // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])
    return merge(left, right)

def merge(left, right):
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    result.extend(left[i:])
    result.extend(right[j:])
    return result

# Example
arr = [38, 27, 43, 3, 9, 82, 10]
print(merge_sort(arr))  # Output: [3, 9, 10, 27, 38, 43, 82]
```