Overview of Recursion: Takeaways

Syntax

- Recursive sum algorithm:

```
def recursive_sum(values):
    # Base case: the list is empty
    if not values:
        return 0
    # General case: the list is not empty
    return values[0] + recursive_sum(values[1:])
```

- Solving the Tower of Hanoi puzzle:

```
def solve_hanoi(num_disks, first_peg, middle_peg, last_peg):
    if num_disks == 1:
        # Base case
        print("Move the top disk from peg {} to peg {}.".format(first_peg, last_peg))
    else:
        # General Case
        solve_hanoi(num_disks - 1, first_peg, last_peg, middle_peg)
        solve_hanoi(1, first_peg, middle_peg, last_peg)
        solve_hanoi(num_disks - 1, middle_peg, first_peg, last_peg)
```

- Recursively listing files in a directory:

```
def list_files(current_path):
    #Base case
    if not os.path.isdir(current_path):
        print(current_path)
    else:
        # General case
        for fn in os.listdir(current_path):
            fn_path = os.path.join(current_path, fn)
            list_files(fn_path)
```

- Implementing merge sort:

```

# Merge function

def merge_sorted_lists(list1, list2): index1 = 0 index2 = 0 merged_list = [] # Collect the font value while both lists are not empty while index1 < len(list1) and index2 < len(list2): if list1[index1] < list2[index2]: merged_list.append(list1[index1]) index1 += 1 else: merged_list.append(list2[index2]) index2 += 1 # Add remaining values merged_list += list1[index1:] merged_list += list2[index2:] return merged_list
```

Merge sort function

def merge_sort(values): # Base case if len(values) < 2: return values # General case midpoint = len(values) // 2 first_half = merge_sort(values[:midpoint]) second_half = merge_sort(values[midpoint:]) return merge_sorted_lists(first_half, second_half) ```

Concepts

- Recursion commonly refers to a function that calls itself.
- A base case prevents recursion from continuing forever.
- The base case is necessary to ensure your program doesn't run out of memory.
- A call stack is a stack data structure that stores information about the active subroutines of a computer program.
- A stack overflow occurs if the call stack pointer exceeds the stack bound. Stack overflow is a common cause of infinite recursion.
- To solve a problem recursively, we need to express it as a combination of solutions to smaller instances of the same problem. We stop decomposing the problem when the problem becomes small enough that we can solve it directly. This is the base.
- The goal of the merge sort algorithm is to first divide up an unsorted list into a bunch of smaller sorted lists and then merge them all to create a sorted list.
- The time complexity for merge sort is $O(n \times log(n))$.

Resources

- Recursion
- Towers of Hanoi
- Master theorem
- Merge Sort Algorithm