Daily Coding Problem   Blog

Daily Coding Problem #96

Problem

This problem was asked by Microsoft.

Given a number in the form of a list of digits, return all possible permutations.

For example, given [1,2,3], return [[1,2,3],[1,3,2],[2,1,3],[2,3,1], [3,1,2],[3,2,1]].

Solution

There are a few ways to do this, and most solutions will have the same run-time. We will need to generate all N! permutations, so our algorithm will have O(N!) run time.

The most straightforward method is to use recursion. We can think of the problem in terms of subproblems, where we can generate permutations of a sublist. A permutation of a single digit (e.g. [1]) would return simply the single digit. To get permutations of size n, we get all permutations of size n-1 and add the next character within each position (index 0 to n). For example, one permutation of the sublist [2,3] is [2,3]. We add 1 to three positions to obtain [1,2,3], [2,1,3], and [2,3,1].

```
def permute(nums):
    if (len(nums) == 1):
            return [nums]

    output = []
    for l in permute(nums[1:]):

        for idx in range(len(nums)):
```

```
            output.append(l[:idx] + [nums[0]] + l[idx:])
        return output
```

An alternative way we can formulate the recursion is by generating all permutations of length `n-1`, but with all digits allowed. The permutations of size 1 would return the input array (e.g. `[[1],[2],[3]]`). Then, we append the `nth` digit to the front of the permutations.

```
def permute(nums):
    def helper(nums, index, output):
        if index == len(nums) - 1:
            output.append(nums.copy())
        for i in range(index, len(nums)):
            nums[index], nums[i] = nums[i], nums[index]
            helper(nums, index + 1, output)
            nums[index], nums[i] = nums[i], nums[index]

    output = []
    helper(nums, 0, output)
    return output
```

Both solutions run in `O(N!)` time and space, where `N` is the size of the input list.