Program for Interview Preparation

Week-3 - Divide and Conquer & Binary Search

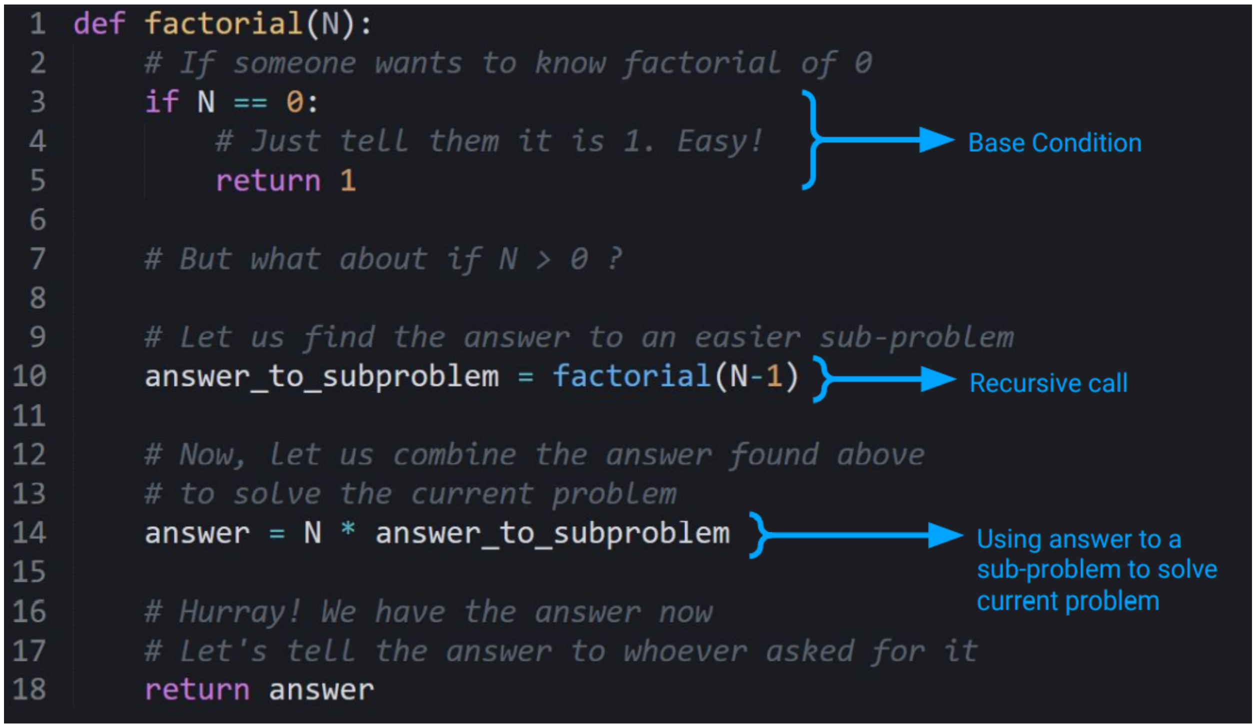
Recursion

Definition

- In computer science, recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem

Structure of a Recursive Function

There are 3 main components of a recursive function: Base Condition, Recursive Call, and using answers to sub-problem to solve the current problem. Consider the following example of finding factorial of a number N.



```python
def factorial(N):
    # If someone wants to know factorial of 0
    if N == 0:
        # Just tell them it is 1. Easy!
        return 1

    # But what about if N > 0 ?

    # Let us find the answer to an easier sub-problem
    answer_to_subproblem = factorial(N-1)

    # Now, let us combine the answer found above
    # to solve the current problem
    answer = N * answer_to_subproblem

    # Hurray! We have the answer now
    # Let's tell the answer to whoever asked for it
    return answer
```

Approaching a problem

- Think about ways to break down the problem into subproblems of the same nature, and try to write the recursive steps.

- You want to find the one that produces the simplest, most natural recursive step.
- Once you have found a way to break the problem, figure out how to relate the subproblem with the current problem.

Some Disadvantages of recursive solution is that it requires greater space and it is extreeeeeeemely prone to Stack overflow error

You must understand the time and space complexity of recursion problems as well.

Master Theorem
If a ≥ 1 and b > 1 are constants and f(n) is an asymptotically positive function, then the time complexity of a recursive relation is given by

$$T(n) = aT(n/b) + f(n) \quad \text{where } f(n)=cn^k$$

1. $a < b^k$ $\quad T(n) \sim n^k$
2. $a = b^k$ $\quad T(n) \sim n^k \log_b n$
3. $a > b^k$ $\quad T(n) \sim n^{\log_b a}$

Practice Questions:
Easy
  a. leetcode.com/problems/n-th-tribonacci-number
  b. leetcode.com/problems/range-sum-of-bst
  c. leetcode.com/problems/minimum-distance-between-bst-nodes
  d. https://www.geeksforgeeks.org/program-chocolate-wrapper-puzzle/

Medium
  a. leetcode.com/problems/diameter-of-binary-tree
  b. leetcode.com/problems/partition-to-k-equal-sum-subsets
  c. leetcode.com/problems/all-possible-full-binary-trees
  d. https://practice.geeksforgeeks.org/problems/knight-walk4521/1/

Hard
  a. leetcode.com/problems/regular-expression-matching

Divide and Conquer

This paradigm, divide-and-conquer, breaks a problem into subproblems that are similar to the original problem, recursively solves the subproblems, and finally combines the solutions to the subproblems to solve the original problem. Because divide-and-conquer solves subproblems recursively, each subproblem must be smaller than the original problem, and there must be a base case for subproblems. DnC can be approached as follows:

1. Divide the problem into smaller subproblems
2. Conquer the subproblems via recursive calls
3. Combine the subproblem answers

Algorithms based on Divide and Conquer technique:
The following are some standard algorithms that follow Divide and Conquer algorithms.
Binary Search: Binary Search is loosely based on DnC. Binary search uses the idea of elimination and reducing the size of the problem repeatedly. Binary search will be described in detail soon.
Quicksort is a sorting algorithm. The algorithm picks a pivot element and rearranges the array elements so that all elements smaller than the picked pivot element move to the left side of the pivot, and all greater elements move to the right side. Finally, the algorithm recursively sorts the subarrays on the left and right of the pivot element.
Merge Sort is also a sorting algorithm. The algorithm divides the array into two halves, recursively sorts them, and finally merges the two sorted halves.
Closest Pair of Points: The problem is to find the closest pair of points in a set of points in the x-y plane. The problem can be solved in O(n^2) time by calculating the distances of every pair of points and comparing the distances to find the minimum. The Divide and Conquer algorithm solves the problem in O(N log N) time.

Practice Questions:

1. https://leetcode.com/problems/sort-an-array
2. https://leetcode.com/problems/maximum-subarray/description/
3. https://leetcode.com/problems/majority-element/
4. https://leetcode.com/problems/powx-n/
5. https://leetcode.com/problems/merge-sorted-array/description/
6. https://leetcode.com/problems/kth-largest-element-in-an-array

Binary Search

Binary search is a commonly used yet extremely powerful technique which comes handy while solving a lot of questions.
https://www.topcoder.com/community/competitive-programming/tutorials/binary-search/
A video tutorial by Errichto:
https://www.youtube.com/watch?v=GU7DpgHINWQ&ab_channel=Errichto

Requirements for Binary Search: Binary search can only be used for functions that are purely monotonic in the required range.
Let us remember what monotonicity means.

1. non-decreasing monotonicity: for all x, y such that $x > y$, $f(x) \geq f(y)$
2. non-increasing monotonicity: for all x, y such that $x > y$, $f(x) \leq f(y)$

Simply put, these are functions that don't change their direction.

Understanding Binary Search using an Example:
Let us consider this problem.
Solution : Let us consider the naive solution first.
We take the prefix sums for the piles. What the means is that we will take an array p[n] such that

p[1] = a[1]
p[2] = a[1] + a[2]
p[i] = a[1] + a[2] + ... + a[i-1] + a[i]
.
p[n] = a[1] + a[2] + ... + a[n-1] + a[n]

We can see for all i > 1 p[i] = p[i - 1] + a[i]
So, we can calculate this array in O(N) time.

Then for every query we iterate through this array and we print the smallest i such that p[i]≥ val, where val is the index of the worm to be found.
Time complexity = O(N + NM) = O(NM)
Since $N = 10^5$ and $M = 10^5$, this will clearly time out.

Let us now try binary search.
Firstly it is easy to see that the search function f(val) = p[val] is monotonically non-decreasing.
Since p[i] = p[i - 1] + a[i] and a[i] ≥ 1
$\Rightarrow$ p[i] > p[i - 1]
So, we can binary search to find the minimum index of the pile.

Pseudocode:

```
start = 1
end = n
result = n
while(s <= e):
    mid = (start + end)//2
    if(p[mid] >= val):
        result = min(result, mid)
        e = mid - 1
    else:
        s = mid + 1
print(result)
```

This runs in O(N + MlogN) time. This fits in our required TL!

Practice Questions:
https://leetcode.com/problems/binary-search/description/
https://leetcode.com/problems/valid-triangle-number/
https://leetcode.com/problems/sqrtx/
https://leetcode.com/problems/search-in-rotated-sorted-array/
https://leetcode.com/problems/median-of-two-sorted-arrays/
https://leetcode.com/problems/single-element-in-a-sorted-array/
https://www.spoj.com/problems/AGGRCOW/
https://codeforces.com/problemset/problem/760/B