Uber Interview Question: Permutations of Parentheses

February 7, 2018

Question

Print all possible n pairs of balanced parentheses
Example:
When $n = 2$, the output should be (()) and ()()
When $n = 3$, the output should be ((())), ()()(), (()()), (())(), and ()(())

Explanation and Algorithm

Thinking of this in the brute force, especially if n is a large number, is time consuming. The best way to go about this problem, or any permutation problem, is through recursion: breaking the problem into smaller sub-problems. The best way to approach the subproblem in this case is not by using a single variable n to denote length. Rather, two variables $left$ and $right$ that indicates how many left/right parentheses are left. Set each variable equal to n and then make the $result$ variable be an array with all the possible permutations of parentheses. While $left > 0$, append "(" to the results. Then the problem will be $(left - 1, right)$. When $left < right$, append ")" to the results. Finally, when $left = right = 0$, print the results. Because the parentheses are uniform, $left$ should never be greater than $right$.

Hints

1. What conditions would have to be satisfied in order for the parentheses to be balanced?

2. Think of making two methods for this..one for printing the parentheses and another for printing permutations

Code

```
#note: this is in Python
global num = 0
global results = []
def printParentheses(left, right)
   global num
    global results

    #when nothins is left, print results
    if left == 0 and right == 0:
      print results
        return

    #pos is next position where parentheses are inserted
    pos = num-left-right
    if left > 0:
      results[pos]='('
        printParentheses(left-1, right)

    if left < right:
      result[pos]=')'
        printParentheses(left, right-1)

def printPermutations(n)
   global num
    global results
    num = n*2
    results = [None]*num
    printParentheses(n,n)
```

Run time analysis

$O(n)$, where n is the amount of parentheses counted in order for the recursion to happen in printPermutation.