الگوریتم بازگشتی چاپ کردن پرانتز گذاری های معتبر برای n چفت پرانتز

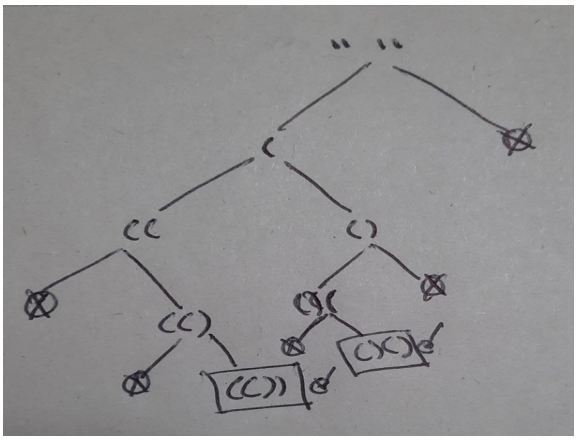
ایده کلی استفاده از دو شمارنده برای پرانتزهای باز و پرانتزهای بسته استفاده شده است. در هر بازگشت با استفاده از شمارنده ها می‌توانیم پرانتز باز یا پرانتز بسته به خروجی اضافه کنیم به شرطی که معتبر بودن خروجی را به هم نزند.(اگر تعداد پرانتز باز استفاده شده کوچکتر از n باشد، می‌توانیم پرانتز باز اضافه کنیم و همچنین اگر تعداد پرانتز بسته استفاده شده کمتر از پرانتز باز باشد، می‌توانیم پرانتز بسته نیز اضافه کنیم.) حالت پایه نیز وقتی است که طول خروجی به اندازه n*2 رسیده باشد.

```
Function generate_valid_parenthesis(n, str=””, open_cnt=0, close_cnt=0):
     if length(str) == 2*n:
          print(str)
          return

     if open_cnt < n:
          Call generate_valid_parenthesis(n, str+”(“, open_cnt+1, close_cnt)

     if close_cnt < open_cnt:
          Call generate_valid_parenthesis(n, str+”)”, open_cnt, close_cnt+1)
```

درخت بازگشتی برای n=2:



پیچیدگی زمانی:

در هر بازگشت، تابع بازگشتی یک و یا دو بار صدا زده می‌شود. پس در بدترین حالت و در نظر نگرفتن شروط معتبر بودن پرانتزگذاری، 2^{2n} مسیر ساخته می‌شود. ولی شروط پرانتزگذاری تعداد بازگشت ها را به طور قابل توجهی کاهش می‌دهد. برای n های ۱، ۲ و ۳ به ترتیب ۱، ۲ و ۵ ترکیب معتبر وجود دارد که از الگوی اعداد کاتالان پیروی می‌کند.

$$C_n = (1/(n+1))(^{2n}_{n})$$

که رشد آن تقریباً برابر $4^n/n^{1.5}$ می‌باشد. پس پیچیدگی زمانی الگوریتم برابر $O(4^n/n^{1/2})$ می‌باشد.

امیررضا محمدی یگانه