2022.11.6

每日一题 1678 设计Goal 解析器

嗯简单模拟题，直接 $O(n)$ 扫一遍替换即可

```
class Solution {
public:
    string interpret(string command) {
        string ans = "";
        for(int i = 0; i < command.size(); i++) {
            if(command[i] == 'G') ans += "G";
            else if (command[i] == '(') {
                if(command[i+1] == ')') ans += "o",i += 1;
                else ans += "al", i += 3;
            }
        }
        return ans;
    }
};
```

或者尝试写一些 $Python3$ 版本：

```
class Solution:
    def interpret(self, command: str) -> str:
        res = []
        for i, c in enumerate(command):
            if c == 'G':
                res.append(c)
            elif c == '(':
                res.append('o' if command[i + 1] == ')' else 'al')
        return ''.join(res)
```

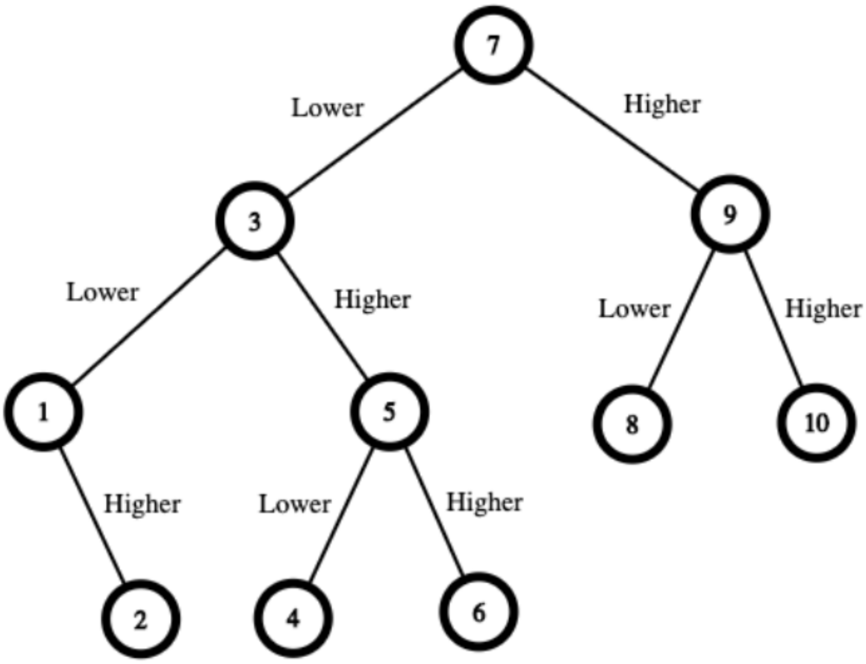
> 1. $enumerate$ 方法在遍历时比较好用，其返回的是数据下标和数据本身两个元素，操作上更为便捷。
> 2. $join()$ 方法指把列表中的全部元素放入一个字符串当中，在这里先用列表存后转化为字符串输出。

375 猜数字大小二

嗯很有意思的一个区间DP题。

首先当然想到的是：设 $dp[i]$ 为从 $[1, \ldots n]$ 内选择的确保获胜的最小现金数，再此基础上推导 $dp[i+1]$ 时发现无法推出状态转移方程。这是为什么呢？**因为确保获胜的最小现金数是跟路径(猜法)有关的！而前后两种猜法之间并没有必然的联系。**这就启示我们，在设计每一个状态的时候，必须把路径，也即猜法考虑进去。

思量至此，我们再审视这张图：



不难发现，它就是一个二分的思想，因此只需要我们枚举每一个猜的点然后两边转移即可。具体表述为：

设 $dp[i][j]$ 表示从 $[i, \ldots, j]$ 区间内猜数所确保获胜的最小现金数，通过枚举猜的点可以得出其转移方程

$$dp[i][j] = \begin{cases} 0 \quad if(j \leq i) \\ min\{\ max\{\ dp[i][k-1],\ dp[k+1][j]\} + k\}\ \ else \end{cases}$$

对第二部分作出如下说明：

- 首先是枚举当前猜 k 时的情况，则 k 的取值应为 $[i, j]$.
- 括号内的 max 即二分想法，当猜数 k 不成立时，权衡两边利弊，当画的现金更多的那一边也保证能猜出时此时一定可以猜出.
- 最外层的 min 针对 k 而言，遍历所有可能猜的点，取最小值即可.

```
class Solution {
private:
    int dp[305][305];
public:
    int getMoneyAmount(int n) {
        for(int len = 2; len <= n; len++)
            for(int begin = 1; begin + len - 1 <= n; begin++) {
                dp[begin][begin + len - 1] = INT32_MAX;
                for(int k = begin; k <= begin + len - 1; k++)
                    dp[begin][begin + len - 1] = min(dp[begin][begin + len - 1],
                        max(dp[begin][k - 1], dp[k+1][begin + len - 1]) + k);
            }
        return dp[1][n];
    }
};
```

时间复杂度 $O(n^3)$ 当 $n \leq 200$ 时可过，区间DP因为复杂度降不下来一般数据都不会给太大。