parentheses

```
for i, c in enumerate(s):
    # stack entry can be i or c
    # stack_sz (balance, depth, open_left)
```

Valid Parentheses (pure) `(()())()` `stack_sz: open_left`
Valid Parentheses (mixed) `{([])[()]}` `stack`
Valid Parentheses (wild) `*(()` `两栈(左, *)index, 比较剩余`

Minimum Add to Make Parentheses Valid `容易, stack_sz` `try: ()))((`

Score of Parentheses `遇到 core() 就加 2^stack_sz`

Remove Invalid Parentheses (return all) `BFS, 去重`
Remove Invalid Parentheses (return one) `两遍: 向后向前` `一遍: 两栈(左,右)index, 删多余`

Generate Parentheses `回溯 dfs(path="", left=n, right=n)`

Longest Valid Parentheses `dp[i] for s[..@i]` `前左 ?()` `前右 ??(--))`

```python
def Remove_Invalid_Parentheses(self, s):
    res = set()
    q = collections.deque([(s, 0)])
    while q:
        l = len(q)
        for _ in range(l):
            s, p = q.popleft()
            if isValid(s):
                res.add(s)
            elif not res:
                for i in range(p, len(s)):
                    if (s[i] == '(' or s[i] == ')') and (i==p or s[i] != s[i-1]):
                        new_s = s[:i] + s[i+1:]
                        q.append((new_s, i))
        if res: return list(res)

def Generate_Parenthesis(self, n):
    res = []
    def dfs(path="", left=n, right=n):
        if left == 0 and right == 0: res.append(path)
        elif left == right:
            dfs(path+"(", left-1, right)
        elif left < right:
            if left > 0: dfs(path+"(", left-1, right)
            if right > 0: dfs(path+")", left, right-1)
    dfs()
    return res

def Longest_Valid_Parentheses(self, s):
    L = len(s)
    if L < 2: return 0
    dp = [0] * L
    for i in range(1, L):
        if s[i] == ')':
            if s[i-1] == '(':          # ?()
                dp[i] = dp[i-2] + 2
            else:                      # ??(--))
                j = i - dp[i-1] - 1
                if j >= 0 and s[j] == '(':
                    dp[i] = dp[i-1] + (dp[j-1] if j>=1 else 0) + 2
    return max(dp)
```