



Daily Coding Problem #142

Problem

This problem was asked by Google.

You're given a string consisting solely of (,), and *. * can represent either a (,), or an empty string. Determine whether the parentheses are balanced.

For example, (())* and (*) are balanced.)*(is not balanced.

Solution

The brute force method here would be to keep a count of current open parentheses that gets incremented every time we see (and decremented every time we see). When we encounter a *, we'll recursively try every path (so with count, count - 1, and count + 1) to see if we can get a valid string.

```
def balanced(s, count=0):
    if not s and count == 0:
        return True

    c = count
    for i, char in enumerate(s):
        if char == '(':
            c += 1
        elif char == ')':
            c -= 1
        elif char == '*':
            return balanced(s[i + 1:], c) or balanced(s[i + 1:], c + 1) or balanced(s[i + 1:], c - 1)

    if c < 0:
        return False
```

```
return c == 0
```

This takes exponential time though since we're recursively calling `balanced` 3 times on each call.

We can use a faster trick, though. Notice from the brute force solution that `*` basically means a range from $c - 1$ to $c + 1$. We can keep just two variables, `low` and `high`. We'll maintain the following invariants:

- `low` represents the minimum number of unbalanced open parentheses
- `high` represents the maximum number of unbalanced open parentheses

where `low` and `high` will diverge when we encounter an `*`. We'll keep track of these counts by doing the following:

- If we encounter `(`, then we increment both `low` and `high` - they both contribute to the counts of unbalanced open parentheses.
- If we encounter `)`, then we decrement both `low` and `high` - we have one less unbalanced open parenthesis.
- If we encounter `*`, then we decrement `low` but increment `high`, since it could mean either a `)` or `(`.

```
def balanced(s):
    low = 0
    high = 0
    for char in s:
        if char == '(':
            low += 1
            high += 1
        elif char == ')':
            low = max(low - 1, 0)
            high -= 1
        elif char == '*':
            low = max(low - 1, 0)
            high += 1

    if high < 0:
        return False
    return low == 0
```

This will take only $O(1)$ space and $O(n)$ time.

[Privacy Policy](#)

[Terms of Service](#)

[Press](#)