```
        i
        ↓
s = 'aab'
p = 'c*a*b*'
          ↑
          j
```

```
dp[i][j] = (i) => {
    if (p[j-1] === '*') {
        return (s[i-1] === p[j-2] ||
                p[j-2] === '.') &&
                dp[i-1][j] ||
                dp[i][j-2];
    }
    return s[i-1] === p[i-1] ||
           p[i-1] === '.';
}
```

dp[i][j] :

s with length i

p with length j

```
dp[0][0] = true;
dp[0][j] = p[j-1] === '*' && dp[0][j-2];
dp[i][0] = false
```

almost there, but since we defind dp[i][j] as string with length i, j not index, we have to take extra care of that !

```
const fn = (s, p) => {
    const m = s.length;
    const n = p.length;              m+1
                                      ↓
    const dp = [...new Array(m)].map(() =>      n+1
                                                 ↓
                                   new Array(n).fill(false));

    dp[0][0] = true;
               1
    for (let j = 0; j < n; j++) {
        dp[0][j] = p[j-1] === '*' && dp[0][j-2];
    }
                 1    i<=m
    for (let i = 0; i < m; i++) {
                     1    j<=n
        for (let j = 0; j < n; j++) {
            dp[i][j] = (() => {
                if (p[j-1] === '*') {
                    return (s[i-1] === p[j-2] ||
                        p[j-2] === '.') && dp[i-1][j]
                    || dp[i][j-2];
                }
                return (s[i-1] === p[j-1] ||
                        p[j-1] === '.') &&
                        dp[i-1][j-1];
            })();
        }
    }
    return dp[m][n];
}
```