```
def numDistinct(s: str, t: str):

    dp = [ [0 for _ in range(len(t)+1) ] for _ in range(len(s)+1)]
    # dp[i][j] : ways tp generate string t[:j] from s[:i]

    for i in range(len(s)+1):
        dp[i][0] = 1 # ways of using string s[:i] to generate empty string t[:0]

    for i in range(1, len(s)+1):
        for j in range(1, len(t)+1):
            dp[i][j] = dp[i-1][j] + dp[i-1][j-1]*(s[i-1]==t[j-1])

    return dp[len(s)][len(t)]
```

```
def numDistinct(s: str, t: str) -> int:

    curr = [0 for _ in range(len(t)+1)]

    curr[0] = 1

    for i in range(1, len(s)+1):
        for j in range(len(t), 0, -1):
            curr[j] = curr[j] + (s[i-1]==t[j-1])*curr[j-1]

    return curr[len(t)]
```

```
def numDistinct(s: str, t: str) -> int:
    @lru_cache(None)
    def dp(i,j):
        if i<j: return 0
        if j<0: return 1
        res = dp(i-1,j)
        if s[i]==t[j]: res += dp(i-1,j-1)
        return res
    return dp(len(s)-1, len(t)-1)
```