Palindrome Partitioning II

Question: Given a string s, partition s such that every substring of the partition is a palindrome. Return the minimum cuts needed for a palindrome partitioning of s.

For example: given s = "aab", Return 1 since the palindrome partitioning ["aa","b"] could be produced using 1 cut.

Solutions:

```
class Solution:
    # @param s, a string
    # @return an integer
    def partitionII(self, s):
        n = len(s)
        f = []
        p = [[False for x in range(n)] for x in range(n)]
        #the worst case is cutting by each char
        for i in range(n+1):
            f.append(n - 1 - i) # the last one, f[n]=-1
        for i in reversed(range(n)):
            for j in range(i, n):
                if (s[i] == s[j] and (j - i < 2 or p[i + 1][j - 1])):
                    p[i][j] = True
                    f[i] = min(f[i], f[j + 1] + 1)
        return f[0]


Solution().partitionII("aab")
```