{% note info %}
摘要
Title: 139. 回文子串的最大长度
Tag: 回文串、字符串哈希、拉长字符串
Memory Limit: 64 MB
Time Limit: 1000 ms
{% endnote %}

Powered by:NEFU AB-IN

Link

@TOC

139. 回文子串的最大长度

- **题意**

> 如果一个字符串正着读和倒着读是一样的，则称它是回文的。
> 给定一个长度为 N 的字符串 S，求他的最长回文子串的长度是多少。

- **思路**

思路：枚举每个中心点，二分半径，左右字符串进行哈希比对
实现：

- 为了使回文字符串的奇偶长度统一，在两个字符之间加入未出现过的字符——"#"作为分割，使得回文串统一为奇数串
  - 先拉长字符串 `s[i] = s[i // 2]`
  - 再插入# `s[i - 1] = '#'`
- 正反哈希前缀
- 当中心点为i，半径为mid
  - 左边串的l,r分别为$i-mid, i-1$
  - 右边串由于是倒序，即1对应n，则x对应$n-x+1$，且l,r应该反过来
  - 所以l,r分别为$n-(i+mid)+1, n-(i+1)+1$
- 由于字符串拉长了，所以要求去除未出现字符的回文串长度
  - 当字母作为左右端点时，字母比#多一个，所以长度为$\frac{2*r+1}{2}$向上取整，即$r+1$

- 当#作为左右端点时，#比字母多一个，所以长度为$\frac{2*r+1}{2}$向下取整，即r

- **代码**

```
'''
Author: NEFU AB-IN
Date: 2022-02-28 15:59:28
FilePath: \ACM\Acwing\139.py
LastEditTime: 2022-02-28 17:10:42
'''
P, MOD = 131, 1 << 64

N = int(2e6 + 100)
hl, hr, p = [0] * N, [0] * N, [0] * N
p[0] = 1

def get(h, l, r):
    return (h[r] - h[l - 1] * p[r - l + 1] % MOD) % MOD

cnt = 0
while True:
    try:
        cnt += 1
        s = input()
        if s == "END":
            break
        s = list(" " + s)
        n = len(s) - 1
        s = [*s * 2]
        for i in range(2 * n, 0, -2):
            s[i] = s[i // 2]
            s[i - 1] = '#'
        n *= 2

        j = n
        for i in range(1, n + 1):
            hl[i] = (hl[i - 1] * P % MOD + ord(s[i])) % MOD
            hr[i] = (hr[i - 1] * P % MOD + ord(s[j])) % MOD
#反向哈希
            p[i] = p[i - 1] * P % MOD
            j -= 1

        res = 0
```

```
        for i in range(1, n + 1):
            l, r = 0, min(i - 1, n - i)
            while l < r:
                mid = l + r + 1 >> 1
                if get(hl, i - mid, i - 1) == get(hr, n - (i
+ mid) + 1,
                                                  n - (i +
1) + 1):
                    l = mid
                else:
                    r = mid - 1
            if s[i - r] == '#':
                res = max(res, r)
            else:
                res = max(res, r + 1)
        print(f"Case {cnt}: {res}")
    except:
        break
```

还有一版一开始写的，思路是，枚举左端点，二分半径，分奇偶两种情况二分

```
P, MOD = 131, 1 << 64

N = int(1e6 + 100)
hl, hr, p = [0] * N, [0] * N, [0] * N
p[0] = 1

def getL(l, r):
    return (hl[r] - hl[l - 1] * p[r - l + 1] % MOD) % MOD

def getR(l, r):
    return (hr[l] - hr[r + 1] * p[r - l + 1] % MOD) % MOD

def check(x):
    for i in range(1, len(s)):
        if x % 2 == 0:
            L1, R1, L2, R2 = i, i + x // 2 - 1, i + x // 2,
i + x - 1
        else:
            L1, R1, L2, R2 = i, i + x // 2, i + x // 2, i +
x - 1
        if i + x - 1 >= len(s):
```

```
            break
        if getL(L1, R1) == getR(L2, R2):
            return True
    return False

def findJI():
    l, r = 0, len(ji) - 1
    while l < r:
        mid = l + r + 1 >> 1
        if check(ji[mid]):
            l = mid
        else:
            r = mid - 1
    return ji[r]

def findOU():
    l, r = 0, len(ou) - 1
    while l < r:
        mid = l + r + 1 >> 1
        if check(ou[mid]):
            l = mid
        else:
            r = mid - 1
    return ou[r]

cnt = 0
while True:
    try:
        cnt += 1
        s = input()
        if s == "END":
            break
        s = " " + s

        for i in range(1, len(s)):
            hl[i] = (hl[i - 1] * P % MOD + ord(s[i])) % MOD
            p[i] = p[i - 1] * P % MOD
        for i in range(len(s) - 1, 0, -1):
            hr[i] = (hr[i + 1] * P % MOD + ord(s[i])) % MOD
        ji, ou = [], []
        for i in range(1, len(s)):
            if i & 1:
                ji.append(i)
```

```
            else:
                ou.append(i)
        res = max(findJI(), findOU())
        print(f"Case {cnt}: {res}")
    except:
        break
```