## Thuật toán Manacher:

Để dễ dàng, tác giả sẽ quy ước vị trí đầu tiên của chuỗi là 0.

Thuật toán $Manacher$ được dùng cho bài toán tìm xâu con palindrome dài nhất ($LPS$). Một cách rất thông dụng để giải bài toán này là sử dụng quy hoạch động, tuy nhiên chỉ có thể áp dụng cho chuỗi có kích thước nhỏ. Thuật toán $Manacher$ giúp giải bài toán này với chuỗi có kích thước lớn hơn và có độ phức tạp nhỏ hơn. Ý tưởng thuật toán khá đơn giản, đầu tiên ta sẽ chèn giữa các ký tự một ký tự đặc biệt nào đó.

$$\begin{cases} a.b.c \\ w.x.y.z \end{cases}$$

Nếu tâm của xâu palindrome là dấu . thì độ dài của xâu palindrome đó là chẵn, nếu tâm khác dấu . thì độ dài của xâu palindrome đó là lẻ. Khi ta tìm được một xâu palindrome, giả sử vị trí ký tự đầu là $l$, vị trí ký tự cuối là $r$. Khi đang xét đến ký tự tại vị trí $i$ nào đó, ta sẽ có $j$ là độ tỏa ra hai bên để nhận một xâu có tâm tại vị trí $i$ và độ dài là $2j+1$. Ban đầu, ta sẽ gán $j = max(0, min(f[l+r-i], r-i)) + 1$ với $f[i]$ là độ tỏa của xâu palindrome có tâm là $i$.

Lý do ta chọn $min(f[l+r-i], r-i)$ là vì 2 trường hợp sau:

$$\left[\begin{array}{ll} r-i > f[l+r-i] & (1) \\ r-i < f[l+r-i] & (2) \end{array}\right.$$

Với (1) thì ta phải chọn $f[l+r-i]$ vì khi $i$ tỏa ra 2 phía $r-i$ phần tử thì xâu đó chưa chắc đã là palindrome, nhưng từ $i$ tỏa ra 2 phía $f[l+r-i]$ phần tử thì xâu đó chắc chắn là palindrome.

Với (2) thì ta phải chọn $r-i$ vì nếu $i$ tỏa ra 2 phía $f[l+r-i]$ phần tử thì $r$ đã phải lớn hơn và từ $i$ tỏa ra 2 phía $r-i$ phần tử thì xâu đó chắc chắn vẫn là palindrome.

$$a.b.a.b.a.b.c$$

Ví dụ với xâu bên trên, giả sử ta đang xét xâu palindrome từ vị trí $l = 2$ đến $r = 12$ có tâm $i = 6$. Ta có thể thấy rằng ký tự $a$ tại vị trí thứ 5 có độ tỏa là 5 phần tử mà vẫn bảo toàn được tính palindrome, tuy nhiên ký tự $a$ tại vị trí thứ 9 thì chỉ có độ tỏa bằng 3 phần tử.

Để tính được độ dài, ta có công thức sau:

$$length = \lfloor (f[i] + i\%2)/2 \rfloor * 2 + 1 - i\%2$$

Dưới đây là phương án cài đặt thuật toán $Manacher$.

## Cài đặt:

```
int Manacher(string s)
{
    int j = 0, l = 0, r = -1, res = 0, n = s.size() * 2 - 1;
    vector<int> f(n, 0);
    string a(n, '.');
    for (int i = 0; i < n; i += 2) a[i] = s[i/2];
    for (int i = 0; i < n; i++)
    {
        j = max(0, min(f[l + r - i], r - i)) + 1;
        while (i - j >= 0 && i + j < n && a[i - j] == a[i + j]) j++;
        f[i] = --j;
        if (i + j > r)
        {
            r = i + j;
            l = i - j;
        }
        int len = (f[i] + i % 2) / 2 * 2 + 1 - i % 2;
        if (len > res) res = len;
    }
    return res;
}
```