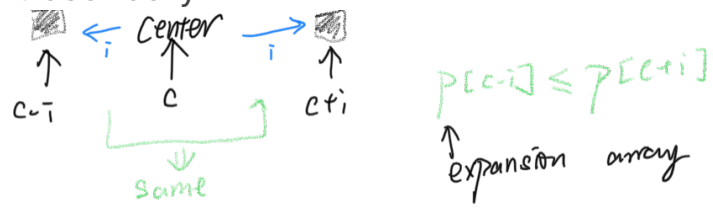
Manacher's Algorithm

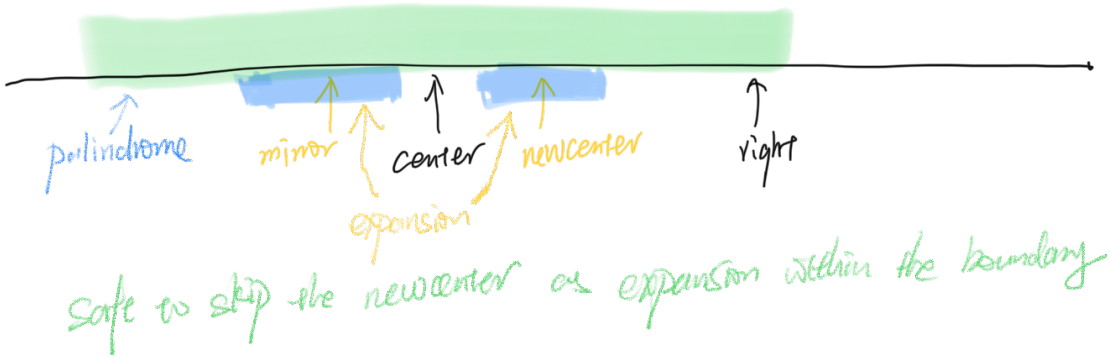
Description
this is a algorithm for finding the largest length palindrome substring

Idea
It uses some of properties of the palindrome string

- it is mirror expansion from the center
- the right mirror point have the equal or larger expansion than the left mirror point
    - because that there might be more mirror chars exceeding the right boundary



The key idea for the algorithm is to skip some of the centers. A expansion array is maintained and right expansion boundary is stored. When there is a center which has a expansion within the boundary, it is safe to skip.



Approach

- go through the string, maintain the initial expansion and center
- when encountering a center, check the mirror expansion
- if the mirror expansion is within the boundary
    - skip the current center
- if the mirror expansion touches the boundary or exceed the boundary
    - get the current the distance to boundary ***right - center*** *as the initial boundary at that point*
    - *this can reduce the time for recalculate the expansion*

- expand from the initial expansion(if not updated before should be 0) then update the expansion at that point
- update the max length

Tips

- when deal with the even and odd palindromes, assume there are invisible chars before and after each of char
    - this will be safe as will extend the original maximum length n to 2n + 1