Tuesday, February 23, 2016

Longest Palindrome Manacher Part I

Linear solution (Manacher's Algorithm) - O(N) time & O(N) space

If you have not read the previous related blog posts about the Longest Palindromic Substring using Brute Force / Dynamic Programming approaches, you might want to take a look at them first.

In this blog post, I am going to explain how does the Manacher's Algorithm work, which is the most efficient algorithm to solve the longest palindromic substring problem.

Part I, I will show you the basic of another algorithm which is based on the Manacher's algorithm but without the dynamic programming, and the time complexity is $O(N^2)$, and Part II, I will describe the main logic of the Manacher's algorithm that using a set of rules and conditions to take advantage of the symmetric property to skip some of the computations to improve the time complexity to O(N).

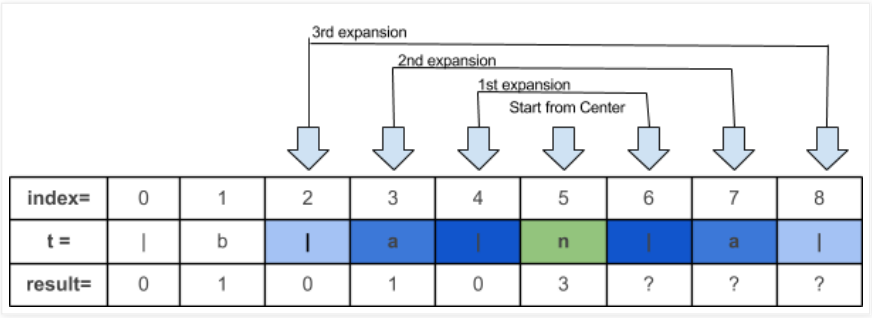
Part I

First, we need to transform the input string s to a string t by adding a special character pipe "|" at both end of the string s and in between characters (although you can use other special characters). This transformation will take care of odd and even number of characters in the string.

For example, s="bana" → t = "|b|a|n|a|".

We have the transformed string t = "|b|a|n|a|", then each characters of the string t consider as a center and then expand around the center to find the length of the palindrome substring. we starts from the 2nd character of the string (index 1) and char by char to work toward to the 2nd last char of the string.

What does it mean by expanding around the center to find the size of the palindrome? e.g. when we are considering the index 5 as the center, we try to see is the char before (at index 4) and after (at index 6) the center are the same (1st expansion see the image below), if they are the same we can increment the result of the index 5 by 1 to indicate that the current palindrome is at least 1 char long and try to expand the palindrome further until we don't find a match.



Lets walk through the example (t = "|b|a|n|a|") from the beginning and one char by char.

Start from index 1 as the center

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since | is equal to |, increment the result of index i by 1 (result[i]++)

t =	|	b	|	a	|	n	|	a	|

can't expand anymore (index out of bound), move on to the next character as center (index 2)

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since b is NOT equal to a

Followers

Followers (1)

Follow

About Me

Jay Yu

View my complete profile

Labels

- Functional Interface
- Functional Programmin
- Java
- Java 8
- Lambda Expression
- Stream
- Stream Creation
- Stream Operations
- Stream Types

Search This Blog

Blog Archive

- ▶ 2019 (7)
- ▶ 2018 (3)
- ▼ 2016 (5)
  - ▶ March (2)
  - ▼ February (3)
    - Longest Palindrome I
    - Longest Palindrome I
    - Longest Palindrome

t =	|	b	|	a	|	n	|	a	|

can't expand anymore (), move on to the next character as center (index 3)

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since | is equal to |, increment the result of index i by 1 (result[i]++)

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since b is NOT equal to n

t =	|	b	|	a	|	n	|	a	|

can't expand anymore, move on to the next character as center (index 4)

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since a is NOT equal to n

t =	|	b	|	a	|	n	|	a	|

can't expand anymore, move on to the next char as center

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since | is equal to |, increment the result of index i by 1 (result[i]++)

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since a is equal to a, increment the result of index i by 1 (result[i]++)

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since | is equal to |, increment the result of index i by 1 (result[i]++)

t =	|	b	|	a	|	n	|	a	|

can't expand anymore, move on to the next char as center

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since n is NOT equal to a

t =	|	b	|	a	|	n	|	a	|

can't expand anymore, move on to the next char as center

t =	|	b	|	a	|	n	|	a	|

expand around the center i, since | is equal to |, increment the result of index i by 1 (result[i]++)

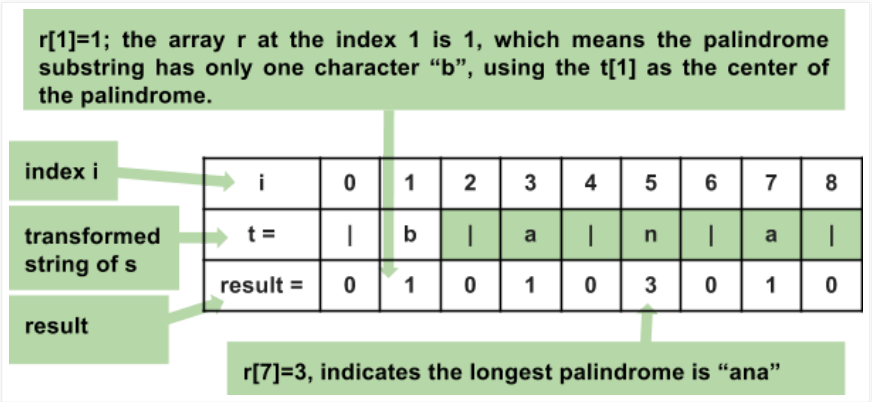
t =	|	b	|	a	|	n	|	a	|

can't expand anymore, move on to the next char as center

t =	|	b	|	a	|	n	|	a	|

can't expand anymore (out of index)

Result needs to be stored, we can simply use an array r (result) to store the final size of palindrome substrings and all the intermediate expansion results. After the algorithm find out all the size of palindrome substrings, you will be able to see the array r at the index i (r[i]) stores the size of a palindrome substring where the transformed string at the index i (t[i]) is the center of the palindromic substring. The largest number in the array r is the longest palindromic substring.



The gist java code snippet

```
1          // Result array stores the final size of palindrome substrings and all the intermediate expansion results
2          int[] result = new int[tSize];
3
4          // Index i is used for finding the palindrome at the center, the center starts at 1 and the for-loop starts at the nex
5          for (int i = 1; i < tSize-1; i++) {
6
```

```
            // Attempt to expand the current palindrome substring from the center at the index i
            while ( t.charAt(i + 1 + result[i]) == t.charAt(i - 1 - result[i])){
                result[i]++;
            }
        }
```

LongestPalindromeBasic.java hosted with ♡ by GitHub view raw

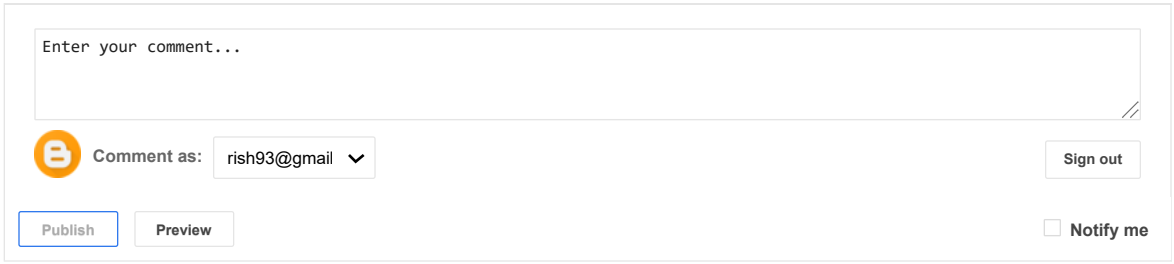
You can download the full working code from github.

Next post - Longest Palindrome Manacher Part II

Posted by Jay Yu at 12:10 AM

No comments:

Post a Comment



Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Simple theme. Powered by Blogger.