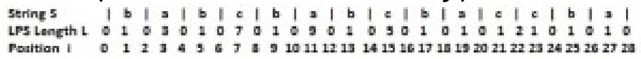
Manacher's Algorithm – Linear Time Longest Palindromic Substring – Part 3

In Manacher's Algorithm Part 1 and Part 2, we gone through some of the basics, understood LPS length array and how to calculate it efficiently based on four cases. Here we will implement the same.

We have seen that there are no new character comparison needed in case 1 and case 2. In case 3 and case 4, necessary new comparison are needed.
In following figure,

(click to see it clearly)



String S	|	b	|	a	|	b	|	c	|	b	|	a	|	b	|	c	|	b	|	a	|	c	|	c	|	b	|	a	|
LPS Length L	0	1	0	3	0	1	0	7	0	1	0	9	0	1	0	5	0	1	0	1	0	1	2	1	0	1	0	1	0
Position i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

If at all we need a comparison, we will only compare actual characters, which are at "odd" positions like 1, 3, 5, 7, etc.
Even positions do not represent a character in string, so no comparison will be preformed for even positions.
If two characters at different odd positions match, then they will increase LPS length by 2.

There are many ways to implement this depending on how even and odd positions are handled. One way would be to create a new string 1st where we insert some unique character (say #, $ etc) in all even positions and then run algorithm on that (to avoid different way of even and odd position handling). Other way could be to work on given string itself but here even and odd positions should be handled appropriately.

Here we will start with given string itself. When there is a need of expansion and character comparison required, we will expand in left and right positions one by one. When odd position is found, comparison will be done and LPS Length will be incremented by ONE. When even position is found, no comparison done and LPS Length will be incremented by ONE (So overall, one odd and one even positions on both left and right side will increase LPS Length by TWO).

```
// A C program to implement Manacher’s Algorithm
#include <stdio.h>
#include <string.h>

char text[100];
void findLongestPalindromicString()
{
        int N = strlen(text);
        if(N == 0)
                return;
        N = 2*N + 1; //Position count
        int L[N]; //LPS Length Array
        L[0] = 0;
        L[1] = 1;
        int C = 1; //centerPosition
        int R = 2; //centerRightPosition
        int i = 0; //currentRightPosition
        int iMirror; //currentLeftPosition
        int expand = -1;
        int diff = -1;
        int maxLPSLength = 0;
        int maxLPSCenterPosition = 0;
        int start = -1;
        int end = -1;

        //Uncomment it to print LPS Length array
        //printf("%d %d ", L[0], L[1]);
        for (i = 2; i < N; i++)
        {
                //get currentLeftPosition iMirror for currentRightPosition i
                iMirror  = 2*C-i;
                //Reset expand - means no expansion required
                expand = 0;
                diff = R - i;
                //If currentRightPosition i is within centerRightPosition R
                if(diff > 0)
                {
                        if(L[iMirror] < diff) // Case 1
                                L[i] = L[iMirror];
                        else if(L[iMirror] == diff && i == N-1) // Case 2
                                L[i] = L[iMirror];
                        else if(L[iMirror] == diff && i < N-1)  // Case 3
                        {
                                        L[i] = L[iMirror];
                                        expand = 1;  // expansion required
                        }
                        else if(L[iMirror] > diff)  // Case 4
                        {
                                L[i] = diff;
                                expand = 1;  // expansion required
                        }
                }
                else
                {
                        L[i] = 0;
                        expand = 1;  // expansion required
                }

                if(expand == 1)
                {
                        //Attempt to expand palindrome centered at currentRightPosition i
                        //Here for odd positions, we compare characters and
                        //if match then increment LPS Length by ONE
                        //If even position, we just increment LPS by ONE without
                        //any character comparison
                        while ( ((i + L[i]) < N && (i - L[i]) > 0) &&
                                ( ((i + L[i] + 1) % 2 == 0) ||
                                (text[(i + L[i] + 1)/2] == text[(i - L[i] - 1)/2] )))
                        {
                                L[i]++;
                        }
                }

                if(L[i] > maxLPSLength)  // Track maxLPSLength
                {
                        maxLPSLength = L[i];
                        maxLPSCenterPosition = i;
                }

                // If palindrome centered at currentRightPosition i
                // expand beyond centerRightPosition R,
                // adjust centerPosition C based on expanded palindrome.
                if (i + L[i] > R)
                {
                        C = i;
                        R = i + L[i];
                }
                //Uncomment it to print LPS Length array
                //printf("%d ", L[i]);
        }
        //printf("\n");
        start = (maxLPSCenterPosition - maxLPSLength)/2;
        end = start + maxLPSLength - 1;
        //printf("start: %d end: %d\n", start, end);
        printf("LPS of string is %s : ", text);
        for(i=start; i<=end; i++)
                printf("%c", text[i]);
        printf("\n");
}
```

```
int main(int argc, char *argv[])
{

        strcpy(text, "babcbabcbaccba");
        findLongestPalindromicString();

        strcpy(text, "abaaba");
        findLongestPalindromicString();

        strcpy(text, "abababa");
        findLongestPalindromicString();

        strcpy(text, "abcbabcbabcba");
        findLongestPalindromicString();

        strcpy(text, "forgeeksskeegfor");
        findLongestPalindromicString();

        strcpy(text, "caba");
        findLongestPalindromicString();

        strcpy(text, "abacdfgdcaba");
        findLongestPalindromicString();

        strcpy(text, "abacdfgdcabba");
        findLongestPalindromicString();

        strcpy(text, "abacdedcaba");
        findLongestPalindromicString();

        return 0;
}
```

Output:

```
LPS of string is babcbabcbaccba : abcbabcba
LPS of string is abaaba : abaaba
LPS of string is abababa : abababa
LPS of string is abcbabcbabcba : abcbabcbabcba
LPS of string is forgeeksskeegfor : geeksskeeg
LPS of string is caba : aba
LPS of string is abacdfgdcaba : aba
LPS of string is abacdfgdcabba : abba
LPS of string is abacdedcaba : abacdedcaba
```

This is the implementation based on the four cases discussed in Part 2. In Part 4, we have discussed a different way to look at these four cases and few other approaches.

This article is contributed by **Anurag Singh**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Topics:

- Recursively print all sentences that can be formed from list of word lists
- Check if a given sequence of moves for a robot is circular or not

- Find the longest substring with k unique characters in a given string
- Function to find Number of customers who could not get a computer
- Find maximum depth of nested parenthesis in a string
- Find all distinct palindromic sub-strings of a given string
- Find if a given string can be represented from a substring by iterating the substring “n” times
- Suffix Tree Application 6 – Longest Palindromic Substring

Tweet g+1 0

Writing code in comment? Please use ideone.com and share the link here.

4 Comments **GeeksforGeeks** 1 Login

Recommend Share Sort by Newest

Join the discussion…

Deepesh Maheshwari · 2 months ago

In case 2 & 3, suffix condition should be checked through below code
centerRightPosition = 2*N but in code condition is checked
currentRightPosition = 2*N

· Reply · Share ›

Anurag Singh → Deepesh Maheshwari · 2 months ago

In comment, it was said "case 2" when it was actually case 3 and case 4, which is corrected now. Not sure if that caused the confusion.
For suffix check, centerRightPosition = 2*N as explained in previous article in Part 2, and that is implemented as "i == N-1" in code shown in current article above.

· Reply · Share ›

Deepesh Maheshwari → Anurag Singh · 2 months ago

Thank Anurag for the clarification, Great Post
Keep it Man :)

· Reply · Share ›

Tyrion · 5 months ago

In case if anyone didn't understand the explanation, they may additionally read
http://leetcode.com/2011/11/lo...
which also has good explanation.

· Reply · Share ›

Subscribe Add Disqus to your site Privacy DISQUS

Google™ Custom Search

-
-
-
-
    - Interview Experiences
    - Advanced Data Structures
    - Dynamic Programming
    - Greedy Algorithms
    - Backtracking
    - Pattern Searching
    - Divide & Conquer
    - Mathematical Algorithms
    - Recursion
    - Geometric Algorithms
-

- ## Popular Posts
    - All permutations of a given string
    - Memory Layout of C Programs
    - Understanding “extern” keyword in C
    - Median of two sorted arrays
    - Tree traversal without recursion and without stack!
    - Structure Member Alignment, Padding and Data Packing
    - Intersection point of two Linked Lists
    - Lowest Common Ancestor in a BST.
    - Check if a binary tree is BST or not
    - Sorted Linked List to Balanced BST
- Follow @GeeksforGeeks

- ## Recent Comments
    - Ashish Aggarwal

      Try Data Structures and Algorithms Made Easy -...

      Algorithms · 17 minutes ago
    - Vlad

      Thanks. Very interesting lectures.

      Expected Number of Trials until Success · 1 hour ago
    - cfh

My implementation which prints the index of the...

Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

- Gaurav pruthi

  forgot to see that part ;)

  Bloomberg Interview | Set 1 (Phone Interview) · 1 hour ago

- saeid aslami

  thanks

  Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 1 hour ago

- Cracker

  Implementation:...

  Implement Stack using Queues · 2 hours ago