Create palindrome from existing string by removing characters

How do i determine the length of the longest palindrome you can get from a word by removing zero or more letters.

for eg : amanQQQapl12345anacaZZZnalpaXXXna67890ma
longest palindrome will be of 21 digits.

java string algorithm palindrome

edited May 14 '12 at 4:26

asked May 14 '12 at 4:14
Nitin Kabra
887 4 21 46

i haven't tried any thing yet..need some help with logic.. – Nitin Kabra May 14 '12 at 4:26

2 Please tag it as algorithm. You'll get quicker answers. – Hindol May 14 '12 at 4:26

1 possible duplicate of how to find longest palindromic subsequence? – Li-aung Yip May 14 '12 at 4:31

Note that if you google around, you will find algorithms for both the longest palindromic sub*sequence* and the longest palindromic sub*string*. Your question is the sub*sequence* version. – Li-aung Yip May 14 '12 at 4:32

4 Answers

This can be solved by dynamic programming. Define **d[i, j]** as the length of longest palindrome in the original string.

> If s[i] = s[j], d[i, j] = max(d[i+1, j-1] + 2, d[i, j-1], d[i+1, j]).
>
> Otherwise d[i, j] = max(d[i, j-1], d[i+1, j]).

edited May 14 '12 at 5:12
Saeed Amiri
19.2k 4 28 72

answered May 14 '12 at 4:34
Rambo
734 2 6 9

2 Correct, but please format it properly and explain a bit more about how it works. – Li-aung Yip May 14 '12 at 4:37

The longest palindrome in the word W is the longest common subsequence of W and its mirror.

You can compute it in $O(n^2)$ time and O(n) space where n is the length of W but if you know that you only need remove few characters to make a palindrome you can have better complexity.

answered May 14 '12 at 8:24
Thomash
5,429 1 15 45

This does not appear to be true. Consider "abcdecba". – Jirka Hanika May 14 '12 at 8:40

The longest common subsequence of 'abcdecba' and 'abcedcba' is 'abcdcba' (or 'abcecba' which has the same length) I can't see your problem. – Thomash May 14 '12 at 9:09

A palidrome can have at most one odd counted letter i.e. the middle letter, and any number of even counted letters.

You can count the frequency of each letter. If it not all or nothing for each letter, add the count/2*2 for each letter and add one if any letter has an odd count.

edited May 14 '12 at 7:26

answered May 14 '12 at 7:21
Peter Lawrey
384k 45 453 789

This is proper implementation of the answer by @Rambo, in case other finds his answer too laconic. I have added caching of previous results but under the condition that the maximum number of distinct symbols is at most 1000. This provides significant speedup due to multiple branches using same sub-branches.

```
int d[1000][1000] = {0}; // To store result of previous computation

int computeMaxPalindromeLength(vector<int>& a, int start, int end) {
    if(d[start][end] != 0) // If not precomputed, recompute.
        return d[start][end];
    if(start == end) { // The mid character should be taken as
        d[start][end] = 1;
        return 1;
    }
    else if(start == end-1) {
        d[start][end] = (a[start] == a[end])?2:1;
        return d[start][end];
    }
    if(a[start] == a[end]) {
        d[start][end] = max( 2 + computeMaxPalindromeLength(a, start+1, end-1),
            max(computeMaxPalindromeLength(a, start+1, end), computeMaxPalindromeLength(a,
start, end-1)));
    } else {
        d[start][end] = max(computeMaxPalindromeLength(a, start+1, end),
computeMaxPalindromeLength(a, start, end-1));
    }
    return d[start][end];
}
```

Call above method as:-

```
vector<int>& a; // Convert each character of string to digit if working with alphanumeric
characters.
int maxPalindromeLength = computeMaxPalindromeLength(a, 0, a.size()-1);
```

edited 2 days ago

answered Jun 14 at 11:03

saurabheights
1,267 1 9 21