Naive: $O(n^2 \cdot k)$ k = max length of two words

Idea:

Let $w = w_1 w_2 w_3 w_4 w_5 \cdots w_m$

and we want to find r s.t.

$w + r$ is a palin

The conditions for $w+r$ to form a palin

Assume $r = r_1 r_2 r_3 \cdots r_n$

$w_1 w_2 w_3 w_4 w_5 \cdots w_m r_1 r_2 r_3 \cdots r_n$

① If $n=0$, w must itself be a palin

can be merged

② If $n \le m$, then, $\underline{w_1 w_2 w_3 \cdots w_n} = \underline{r_n r_{n-1} \cdots r_1}$ and $\underline{w_{n+1} \cdots w_m}$ is a palin

Visualization

$\underline{w_1 w_2 w_3 w_4} \underline{\cdots w_m} \; \underline{r_1 r_2 r_3 r_4 \cdots r_n}$

③ If $n>m$, then $w_1 w_2 w_3 \cdots w_m$

$= r_n r_{n-1} r_{n-2} \cdots r_{n-m+1}$

and $r_1 r_2 \cdots r_{n-m}$ is a palin

$$w_1 w_2 w_3 \cdots w_m r_1 r_2 r_3 r_4 r_5 \cdots r_n$$

Observe :

for ①, ②, we can use the standard Trie to handle: just build a suffix Trie and then travese a tree, whenever we encounter a word, we check if $w_{i+1} \cdots w_m$ is a palin, if that is the case, add this pair

For ③, using the standard Trie gives us no information after we have walked through $w_1 w_2 \cdots w_m$,

that's why

```
struct TrieNode {
    TrieNode * child[26];
    vector<int> prefix_palin_indices;
    int word_idx;
}
```

Further observation (using a hashmap)

For case ①,②,

$$\underbrace{w_1 w_2 w_3 w_4} \underbrace{\cdots w_m} \; \underbrace{r_1 r_2 r_3 r_4 \cdots r_n}$$

You can see that, $r_1 r_2 r_3 \cdots r_n$ is a whole word

For case ③,

$$\underbrace{W_1 W_2 W_3 \cdots W_m} \; \underbrace{r_1 r_2 r_3 r_4 r_5} \underbrace{\cdots r_n}$$

You can see that, $w_1 w_2 \cdots w_m$ is a whole word

⇒ Even if when we are finding w,

we can only find ①,② cases,

Case ③ will be found if we let each word act as a left/right word.

Observe that this way to solve will cause duplicates at

Ex.

$\boxed{w_1 w_2 w_3 w_4}$ $\underline{r_1 r_2 r_3 r_4}$

$\underline{r_1 r_2 r_3 r_4}$ $\boxed{w_1 w_2 w_3 w_4}$

$\boxed{r_1 r_2 r_3 r_4}$ $\underline{w_1 w_2 w_3 w_4}$

$\underline{w_1 w_2 w_3 w_4}$ $\boxed{r_1 r_2 r_3 r_4}$

if w's idx is i, r's idx is j

we restrict $i < j$ when $len(w) == len(r)$

Using KMP:

Idea:

Let $w = w_1 w_2 w_3 w_4 w_5$

Assume w is placed on the left,

$\Rightarrow$ $w +$ right (and len(right) $\leq$ len(w))

We want $w +$ right be a palindrome

i.e.

$w_1 w_2 w_3 w_4 w_5$ right is a palindrome

Observe that:

if len(right) $== 0$, $w_1 w_2 w_3 w_4 w_5$ must be a palindrome

if len(right) $== 1$, right $= w_1$ and $w_2 w_3 w_4 w_5$ is a palin

if len(right) $== 2$, right $= w_2 w_1$ and $w_3 w_4 w_5$ is a palin

$\Rightarrow$ This gives us an idea to solve it via KMP algorithm

Construct: $rev(w) + "\#" + w$

$$= \underline{w_5 w_4 w_3 w_2} w_1 \# w_1 \overline{w_2 w_3 w_4 w_5}$$

by KMP, we know $lsp[2 \cdot len(w)]$ (the longest proper prefix equals to the suffix for $S[0 \sim i]$) represents the longest palindrome in the form of $w_5 w_4 \cdots w_i$

Once we know $w_5 w_4 \cdots w_i$ is a palin, we can construct $right = w_{i-1} \cdots w_2 w_1$

And because $w_5 w_4 \cdots w_i$ is the longest palin, we can keep finding shorter palin from

$lsp[len(w_5 w_4 \cdots w_i) - 1] \Rightarrow$ the shorter palindrome