Jakub Radoszewski — Treść zadania
Jakub Radoszewski, Adam Iwanicki — Opracowanie
Adam Iwanicki — Program

OI, Etap III, dzień drugi, 30.03.2006

Palindromy

Mały Jaś lubi bawić się słowami. Wybrał on sobie n palindromów (palindromem nazywamy słowo, które czytane od przodu i od tyłu jest dokładnie takie samo, jak np. `ala`, `anna` *czy* `kajak`*), a następnie utworzył wszystkie możliwe n^2 par spośród nich i posklejał palindromy występujące w tych parach w pojedyncze słowa. Na koniec Jaś policzył, ile spośród tak otrzymanych słów stanowią palindromy. Nie jest on jednak pewien, czy się nie pomylił, dlatego poprosił Ciebie o wykonanie tych samych czynności i podanie mu wyniku. Napisz program, który wykona to za Ciebie.*

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia palindromy, które podał Ci Jaś,*
- *wyznaczy liczbę słów, utworzonych z par wczytanych palindromów, które są palindromami,*
- *wypisze wynik na standardowe wyjście.*

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba całkowita n ($n \geqslant 2$), oznaczająca liczbę palindromów podanych przez Jasia. Następnych n wierszy zawiera opisy palindromów. Wiersz $i+1$ składa się z jednej dodatniej liczby całkowitej a_i, oznaczającej długość i-tego palindromu, oraz palindromu złożonego z a_i małych liter alfabetu angielskiego. Liczba a_i i palindrom oddzielone są pojedynczym odstępem. Palindromy podane w różnych wierszach są różne. Łączna długość wszystkich palindromów nie przekracza 2 000 000.

Wyjście

Pierwszy i jedyny wiersz wyjścia powinien zawierać jedną liczbę całkowitą: liczbę różnych uporządkowanych par palindromów, które po sklejeniu ze sobą dają w wyniku palindromy.

Przykład

Dla danych wejściowych:	*poprawnym wynikiem jest:*

`6`
`2 aa`
`3 aba`
`3 aaa`
`6 abaaba`
`5 aaaaa`
`4 abba`
	`14`

Rozwiązanie

Problem postawiony w zadaniu można rozwiązać na wiele różnych sposobów. W niniejszym opracowaniu ograniczymy się do analizy trzech najlepszych rozwiązań, a o pozostałych wspomnimy pokrótce na końcu opisu. W kolejnych z prezentowanych metod w coraz większym stopniu będziemy wykorzystywać specyficzne właściwości palindromów, co pozwoli nam konstruować coraz prostsze algorytmy i w końcu doprowadzi nas do rozwiązania wzorcowego.

Na wstępie wprowadźmy kilka podstawowych definicji związanych ze słowami.

- *Prefiksem* słowa nazywamy dowolny spójny początkowy fragment tego słowa, czyli słowo złożone z pewnej liczby jego początkowych liter. Na przykład prefiksami słowa `aba` są: `a`, `ab` i `aba`, a także słowo puste, które oznaczamy zazwyczaj symbolem ε.
- *Sufiksem* słowa nazywamy dowolny spójny końcowy fragment tego słowa. Na przykład sufiksami słowa `aba` są: ε, `a`, `ba` i `aba`.
- *Podsłowem* słowa jest jego dowolny spójny fragment.
- *Prefikso-sufiksem* słowa nazywamy podsłowo, które jest jednocześnie prefiksem i sufiksem danego słowa. Prefikso-sufiksami słowa `aba` są: ε, `a` i `aba`.
- *Odwróceniem* słowa u (oznaczanym przez u^R) nazywamy słowo, które składa się z tych samych liter co u, ale zapisanych w odwrotnej kolejności. Oczywistą i ważną własnością odwrócenia słów jest równość $(uv)^R = v^R u^R$, gdzie uv oznacza sklejenie słów u i v.
- *Długością* słowa u (oznaczaną przez $|u|$) jest liczba liter, z których składa się to słowo.
- *k-tą potęgą* słowa u nazywamy słowo (oznaczane u^k) równe k-krotnemu sklejeniu słowa u ze sobą.
- *Okresem* słowa u nazywamy takie słowo v, że v jest prefiksem u i istnieje taka liczba naturalna k, że u jest prefiksem v^k. Łatwo zauważyć, że ta trochę odmienna od intuicyjnej definicja jest jej równoważna: okres słowa to taki początkowy fragment słowa, którego kolejnymi powtórzeniami można pokryć całe słowo, przy czym ostatnie powtórzenie może być niepełne.

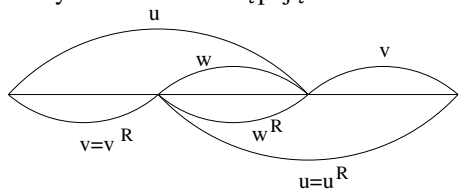
Rozwiązanie pierwsze

Na wstępie zauważmy, że zachodzi następująca równoważność:

Obserwacja 1 *Sklejenie dwóch palindromów jest palindromem wtedy i tylko wtedy, gdy krótszy z nich jest prefiksem dłuższego i słowo powstałe przez odcięcie krótszego palindromu od dłuższego jest także palindromem.*

Dowód Niech u i v będą palindromami. Zauważmy, że jeśli słowo uv jest palindromem, to $uv = (uv)^R = v^R u^R = vu$, więc uv można przedstawić zarówno jako sklejenie dłuższego palindromu z krótszym (w tej kolejności), jak i odwrotnie. Stąd bez straty ogólności możemy przyjąć, że $|u| \geqslant |v|$.

Równość $uv = vu$ możemy zilustrować następująco:



Rys. 1: Ilustracja równości $uv = vu$

Pozwala to nam dostrzec, że v rzeczywiście musi być prefiksem u, a ewentualna pozostała część u (oznaczona na rysunku przez w) musi być palindromem, gdyż musi spełniać $w = w^R$. Z rysunku widać także, że te dwie własności wystarczają do tego, aby sklejenie uv było palindromem. ■

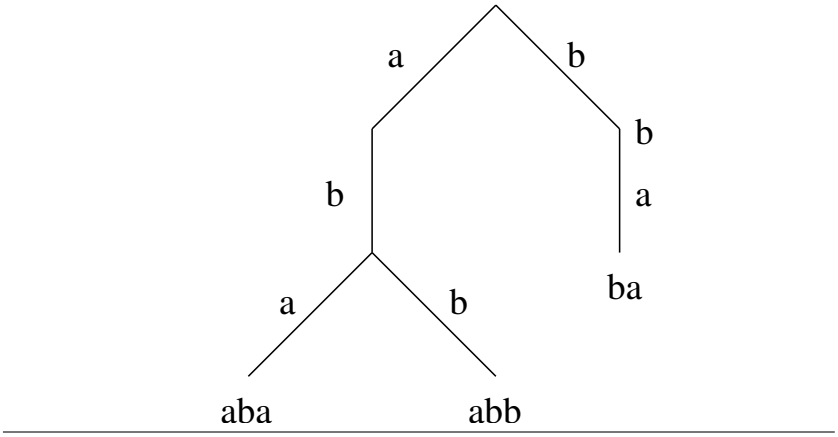
Przy okazji powyższego dowodu wykazaliśmy również, że w naszym zadaniu słowo uv jest palindromem wtedy i tylko wtedy, gdy vu jest palindromem, a zatem wystarczy wykonać co najwyżej jedno sprawdzenie dla każdej nieuporządkowanej pary słów.

Teraz pozostało nam jeszcze wymyślić, w jaki sposób zastosować poczynioną obserwację w konstrukcji efektywnego rozwiązania. W tym celu wykorzystamy kilka technik i struktur, często występujących w algorytmach tekstowych.

Drzewo TRIE

Definicja 1 *Drzewo typu TRIE* to drzewo z wyróżnionym korzeniem służące do zapisywania słów. Krawędzie tego drzewa są etykietowane literami alfabetu i powiemy, że słowo w jest zapisane w drzewie, jeśli idąc ścieżką od korzenia w dół drzewa możemy słowo w „odczytać" z etykiet krawędzi na tej ścieżce. Sposób zapisu słowa w drzewie musi być jednoznaczny. To znaczy, że kierując się kolejnymi literami słowa mamy zawsze tylko jedną możliwość zejścia w dół drzewa — z żadnego wierzchołka nie wychodzą dwie krawędzie etykietowane takimi samymi literami. Dodatkowo, jeśli zakładamy, że w drzewie są zapisane (tylko) słowa w_1, w_2, …, w_k, to ścieżki wyznaczone przez te słowa muszą pokryć całe drzewo TRIE. Warto zauważyć, że każde słowo prowadzi nas w drzewie do pewnego wierzchołka — może to być liść, może to być także wierzchołek wewnętrzny (w takim przypadku słowo na pewno jest prefiksem innego słowa zapisanego w drzewie).

Przykład drzewa TRIE dla słów: `aba`, `ba`, `b` i `abb` jest przedstawiony na rysunku 2.



Rys. 2: Przykładowe drzewo TRIE

Jeżeli łączna długość wszystkich słów znajdujących się w drzewie TRIE wynosi d, to wielkość tego drzewa możemy ograniczyć przez $O(d)$ — wynika to stąd, iż słowo długości d_i pokrywa co najwyżej $d_i + 1$ wierzchołków oraz d_i krawędzi.

Więcej o drzewach TRIE można przeczytać na przykład w książce [15].

Wstawienie nowego słowa do drzewa TRIE jest koncepcyjnie bardzo proste — trzeba, począwszy od korzenia, wędrować w dół drzewa po ścieżce odpowiadającej kolejnym literom słowa, dopóki istnieją odpowiednie krawędzie. Gdy ich zabraknie, trzeba utworzyć nowe tak, by słowo znalazło się w drzewie. Zauważmy, że podczas tego procesu napotykamy w węzłach końce wszystkich wstawionych wcześniej słów, które są prefiksami danego słowa. Stąd pomysł na algorytm. Posortujmy wszystkie zadane palindromy niemalejąco względem długości i w takim porządku wstawiajmy je do początkowo pustego drzewa TRIE. Jeżeli przy wstawianiu kolejnego słowa napotkamy wierzchołek oznaczający koniec wcześniej wstawionego palindromu i będziemy potrafili szybko (powiedzmy w czasie $O(1)$) sprawdzić, czy pozostała część wstawianego słowa jest palindromem, to w czasie $O(d)$ policzymy żądany wynik.

Algorytm Manachera

Zastanówmy się teraz, jak rozpoznać, czy sufiks zadanego słowa jest palindromem. Możemy w tym celu dla każdego wejściowego palindromu zastosować algorytm Manachera, który pozwala wyznaczyć dla każdej pozycji w słowie maksymalny promień podsłowa-palindromu o środku na tejże pozycji. Dokładniej, algorytm ten dla słowa $u = u_1u_2\ldots u_k$ wyznacza:

- dla każdego $i \in \{1,\ldots,k\}$ największą taką wartość $r_i \in N$, że słowo $u_{i-r_i}u_{i-(r_i-1)}\ldots u_{i-1}u_iu_{i+1}\ldots u_{i+r_i}$ jest palindromem nieparzystej długości (w tym wypadku środek palindromu wypada na pewnej literze słowa) oraz
- dla każdego $j \in \{1,\ldots,k-1\}$ największy taki promień $R_j \in N$, że słowo $u_{j-(R_j-1)}\ldots u_{j-1}u_ju_{j+1}\ldots u_{j+R_j}$ jest palindromem parzystej długości (w tym wypadku środek palindromu wypada między dwiema kolejnymi literami słowa).

Dokładniejszy opis tego algorytmu można znaleźć m.in. w książce [15]. W tym miejscu wystarczy nam informacja, że jego złożoność jest liniowa względem liczby liter słowa, dla którego jest wykonywany (i jego kod jest stosunkowo krótki) — a zatem możemy go wykonać na wstępie dla wszystkich wejściowych słów i nie zwiększy to złożoności algorytmu. Posiadając wyniki działania algorytmu Manachera, sprawdzenie czy dany sufiks słowa jest palindromem możemy wykonać w czasie stałym, badając maksymalny promień palindromu w środku tego sufiksu.

Złożoność czasowa i pamięciowa

Zaprezentowany algorytm ma złożoność czasową $O(n \log n + d)$. Wydaje się, że (przynajmniej teoretycznie) złożoność pamięciowa również jest akceptowalna — główny wpływ ma na nią rozmiar drzewa TRIE, które, jak już ustaliliśmy, ma wielkość $O(d)$. Jednak w asymptotycznym oszacowaniu pomijamy pewien znaczący szczegół implementacyjny — dla każdego węzła drzewa musimy mieć możliwość zapisania krawędzi etykietowanej dowolną literą alfabetu wychodzącej z tego węzła i/lub sprawdzenia, czy taka krawędź już istnieje. Możemy w tym celu w każdym węźle utrzymywać tablicę indeksowaną literami, jednak wówczas faktyczny rozmiar drzewa mógłby osiągnąć wielkość około $26 \cdot d$. Zakładając, że rozmiar każdego wskaźnika wynosi 4 bajty i d może osiągnąć wartość 2 000 000, otrzymujemy strukturę zajmującą $4 \cdot 26 \cdot 2\,000\,000B \approx 200MB$, co nie mieści się w limicie pamięciowym zadania ($128MB$).

Można spróbować inaczej zaimplementować wierzchołki drzewa:

- krawędzie wychodzące z wierzchołka można zapisać w postaci listy — w ten sposób mamy szansę zmieścić się w limicie pamięciowym, ale łatwo możemy przekroczyć limit czasowy, gdyż w poszukiwaniu odpowiedniej krawędzi musimy za każdym razem tę listę przeszukać,
- w każdym węźle można utworzyć zrównoważone drzewo poszukiwań binarnych (AVL, czerwono-czarne itp.) — taka struktura pozwala znaleźć odpowiednią krawędź w czasie logarytmicznym i nie powoduje znacznego zwiększenia pamięci — niestety stopień skomplikowania struktury i procedur wyszukiwania oraz aktualizacji utrudnia zmieszczenie się w limicie czasowym.

Z powyższych rozważań wynika, że przedstawiona metoda mogła sprawić wiele problemów implementacyjnych i trudno stosując ją zmieścić się jednocześnie w limicie czasowym i pamięciowym. Zatem takie rozwiązanie, mimo asymptotycznie takiej samej złożoności czasowej jak dalej omówione rozwiązanie wzorcowe, mogło nie uzyskać pełnej punktacji.

Uniwersalność rozwiązania pierwszego

Mimo istotnych wad, rozwiązanie pierwsze ma pewną przewagę nad pozostałymi zaprezentowanymi w tym opracowaniu. Jest ono zdecydowanie bardziej ogólne — można je zastosować także do słów wejściowych nie będących palindromami! W tym celu wystarczy sformułować spostrzeżenie podobne do obserwacji 1: „Sklejenie dwóch słów u i v, gdzie $|u| \geqslant |v|$, jest palindromem wtedy i tylko wtedy, gdy słowo v^R jest prefiksem u i słowo powstałe przez obcięcie v^R od u jest także palindromem". Korzystając z niego

możemy zmodyfikować przedstawione rozwiązanie dostosowując je do nowych warunków wejściowych (m.in. wstawiając do drzewa TRIE zarówno słowa, jak i ich odwrócenia) — dokładny zapis nowego algorytmu pozostawiamy Czytelnikowi jako ćwiczenie.

Rozwiązanie drugie

Stwierdziliśmy już, że w powyższym rozwiązaniu nie wykorzystujemy w istotny sposób faktu, że słowa wejściowe są palindromami; w niniejszym rozwiązaniu weźmiemy ten fakt pod uwagę i sformułujemy łatwiejsze (wymagające mniej wyrafinowanego algorytmu) kryterium sprawdzania, czy określony sufiks palindromu jest palindromem.

Obserwacja 2 *Sufiks palindromu jest palindromem wtedy i tylko wtedy, gdy jest także jego prefiksem.*

Dowód Niech u będzie palindromem, a v jego sufiksem. Pokażemy najpierw, że jeśli v jest palindromem, to jest także prefiksem u. Zapiszmy dany palindrom w następującej postaci: $u = wv$, gdzie w jest pewnym słowem. Stąd, ponieważ $u^R = u$, więc $(wv)^R = u$, czyli $v^R w^R = u$, a zatem $vw^R = u$. Sufiks v rzeczywiście jest także prefiksem u.

Teraz pokażemy, że prefikso-sufiks v palindromu u musi być palindromem. Jeżeli v jest prefikso-sufiksem u, to $u = wv = vw'$ dla pewnych słów w i w'. Stąd $u^R = (wv)^R = v^R w^R$, ale $u^R = u$, więc $v^R w^R = vw'$. Ponieważ słowa v i v^R mają tę samą długość, to z powyższego zapisu wnioskujemy, że $v = v^R$, czyli v jest palindromem. ■

Funkcja prefiksowa

Dzięki powyższej obserwacji sprawdzenie, czy sufiks palindromu jest palindromem redukuje się do sprawdzenia, czy jest on prefikso-sufiksem. Listę wszystkich prefikso-sufiksów słowa (a dokładniej ich długości) możemy uzyskać za pomocą funkcji prefiksowej p, stosowanej między innymi w algorytmie Knutha-Morrisa-Pratta (KMP). Przypomnijmy, że dla k-literowego słowa $u = u_1 \ldots u_k$ i indeksu $i \in \{1, \ldots, k\}$ definiujemy $p(i)$ jako długość najdłuższego właściwego (czyli krótszego od całego słowa) prefikso-sufiksu słowa $u_1 \ldots u_i$.

Najważniejsze z naszego punktu widzenia własności funkcji prefiksowej są następujące:

- Wszystkie wartości $p(i)$ dla $i \in \{1, \ldots, k\}$ można policzyć w złożoności czasowej $O(k)$.
- Długości wszystkich prefikso-sufiksów słowa u można uzyskać, wielokrotnie stosując funkcję prefiksową: k, $p(k)$, $p(p(k))$, ... (proces kontynuujemy, dopóki nie dojdziemy do zerowej długości prefikso-sufiksu).

Dowody własności funkcji prefiksowej i sposób jej obliczania można znaleźć m.in. w książkach: [18], [15] oraz w opisie rozwiązania zadania *Szablon* w [12].

Sformułowanie rozwiązania drugiego

Większa część rozwiązania drugiego jest identyczna jak rozwiązanie pierwsze — stosujemy w nim drzewo TRIE, do którego wstawiamy palindromy w kolejności niemalejących długości. W trakcie wstawiania, gdy napotykamy węzeł oznaczający koniec innego słowa,

sprawdzamy, czy pozostała część słowa jest palindromem. I w tym miejscu pojawia się jedyna różnica pomiędzy rozwiązaniami — zamiast sprawdzać, czy pozostała po odcięciu krótszego palindromu część słowa wejściowego jest także palindromem, sprawdzamy za pomocą funkcji prefiksowej, czy jest to prefikso-sufiks. Wszystkie wartości funkcji prefiksowej dla wszystkich słów wejściowych obliczamy przed rozpoczęciem budowy drzewa, więc sprawdzenie wykonywane przy wstawianiu słowa do drzewa TRIE możemy wykonać w czasie stałym $O(1)$.

Złożoności czasowa i pamięciowa tego algorytmu są takie same jak w przypadku poprzedniego i, co ważniejsze, stosujemy w nim także trudne do efektywnego zaimplementowania drzewo TRIE. Zatem rozwiązanie to również może nie uzyskiwać pełnej punktacji. Ze względu na zastosowanie szerzej znanego algorytmu KMP zamiast algorytmu Manachera rozwiązanie to było łatwiejsze do wymyślenia i część uczestników III etapu olimpiady właśnie to rozwiązanie zaimplementowała.

Rozwiązanie wzorcowe

Rozpocznijmy od kolejnych obserwacji:

Obserwacja 3 *Niech u, v będą palindromami. Słowo uv jest palindromem wtedy i tylko wtedy, gdy u i v są potęgami tego samego słowa.*

Dowód Wiemy już, że jeśli u, v oraz uv są palindromami, to muszą zachodzić równości: $uv = (uv)^R = v^R u^R = vu$. Pokażemy, że równość $uv = vu$ może zachodzić tylko dla słów u i v, które są potęgą tego samego słowa. Dowód tego faktu przeprowadzimy przez indukcję względem sumy długości słów u i v.

- Baza indukcji: dla $|u| = |v| = 1$ obserwacja w sposób oczywisty zachodzi.
- Jeżeli $|u| = |v|$, to teza jest prawdziwa, gdyż $u = u^1$ oraz $v = u^1$. W przeciwnym przypadku, nie tracąc ogólności możemy założyć, że $|u| > |v|$. Wówczas z tego, że v jest prefiksem uv, mamy, że $u = vw$ dla pewnego słowa w. Zatem

  $$uv = vu \;\Leftrightarrow\; vwv = vvw \;\Leftrightarrow\; wv = vw.$$

  W ten sposób otrzymaliśmy równość analogiczną do wyjściowej, ale dla słów v i w, dla których $|v| + |w| = |v| + (|u| - |v|) < |u| + |v|$. Z założenia indukcyjnego wiemy więc, że $w = x^k$ oraz $v = x^l$ dla pewnego słowa x oraz liczb naturalnych k i l. Stąd $u = vw = x^{k+l}$ i $v = x^l$, co kończy dowód tezy indukcyjnej.

■

Jak sprawdzić ten warunek?

Otrzymaliśmy pewną charakteryzację poszukiwanych par palindromów. Zastanówmy się, jak ją weryfikować w praktyce. Zanim otrzymamy właściwą metodę, pokażemy najpierw pomocniczy

Lemat 4 (o okresowości) *Jeżeli p i q są długościami dwóch okresów słowa u, spełniającymi nierówność $p + q \leqslant |u|$, to $NWD(p,q)$ jest również długością okresu słowa u.*

Jest to tak zwana „słaba wersja" lematu o okresowości; istnieje wersja silna, w której zakładamy tylko, że $p+q-NWD(p,q) \leqslant |u|$ (dowód można znaleźć m.in. w książce [35]).

Dowód Dowód lematu przeprowadzimy przez indukcję względem sumy $p+q$.

- Baza indukcji: jeżeli $p = q = 1$, to lemat oczywiście zachodzi.
- Jeżeli $p = q$, to teza jest oczywista. W przeciwnym przypadku, nie tracąc ogólności możemy założyć, że $p > q$. Pokażemy, że wówczas $p-q$ jest również długością okresu słowa u. Faktycznie, dla dowolnej pozycji $i \in \{1,\ldots,|u|\}$ w słowie u, takiej że $i+(p-q) \leqslant |u|$, zachodzi $i-q \geqslant 1$ lub $i+p \leqslant |u|$ (w przeciwnym przypadku byłoby $p+q > |u|$). W pierwszym z tych przypadków mamy $u_i = u_{i-q}$ (gdyż q jest długością okresu u), czyli $u_i = u_{i-q} = u_{i-q+p} = u_{i+(p-q)}$. W drugim podobnie zachodzi $u_i = u_{i+p} = u_{i+p-q} = u_{i+(p-q)}$, co pokazuje, że niezależnie od przypadku $p-q$ jest długością okresu słowa u.

  Korzystając teraz z założenia indukcyjnego mamy, że słowo u ma okres długości $NWD(p-q,q)$, a z własności największego wspólnego dzielnika wynika, że dla $p > q$ zachodzi równość $NWD(p-q,q) = NWD(p,q)$, czyli słowo ma okres długości $NWD(p,q)$, co należało pokazać.

■

Z lematu o okresowości wynika w szczególności, że jeżeli dane słowo u można przedstawić jako nietrywialną potęgę na dwa sposoby: $u = v^k = w^l$, to słowa v i w są potęgami tego samego słowa — oznaczmy je x. Wynika to stąd, że w takim przypadku zarówno v, jak i w są okresami słowa u, a zatem u ma także okres o długości $NWD(|v|,|w|)$. Ponieważ $NWD(|v|,|w|)$ dzieli $|v|$ i dzieli $|w|$, to prefiks u o długości $NWD(|v|,|w|)$ jest właśnie szukanym słowem x. Z przeprowadzonego rozumowania wynika także, iż dla słowa u istnieje najkrótsze słowo $\rho(u)$, takie że:

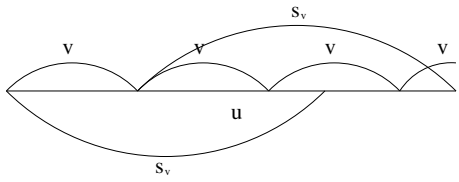
- u jest potęgą $\rho(u)$ oraz
- wszystkie inne słowa y, dla których $u = y^i$, są także potęgami $\rho(u)$.

Słowo $\rho(u)$ nazywamy *pierwiastkiem pierwotnym* u. Jeśli $u = \rho(u)$, to u nazywamy *słowem pierwotnym*.

Wreszcie wnioskiem z powyższej analizy jest pomysł na praktyczne zweryfikowanie, czy dwa słowa są potęgami tego samego słowa — wystarczy sprawdzić, czy mają te same pierwiastki pierwotne. Ostatnia (z pozoru dość zaskakująca) obserwacja wskaże nam metodę, dzięki której będziemy w stanie szybko znajdować pierwiastki pierwotne słów:

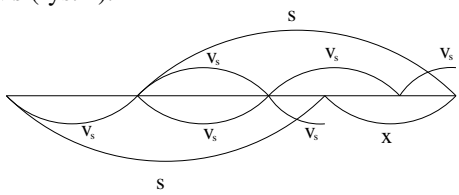
Obserwacja 5 *Niech p oznacza funkcję prefiksową słowa u, które ma długość k. Jeżeli $(k-p(k))|k$, to pierwiastek pierwotny słowa u ma długość $k-p(k)$. W przeciwnym przypadku słowo u jest pierwotne.*

Dowód Na początek pokażemy, że $k-p(k)$ jest długością najkrótszego okresu słowa u. W tym celu zauważmy, że istnieje wzajemnie jednoznaczna odpowiedniość między okresami słowa a jego prefikso-sufiksami. Faktycznie, jeżeli v jest okresem słowa u, to można dla niego zdefiniować prefikso-sufiks s_v, jako słowo pozostałe po odcięciu od słowa u początkowego wystąpienia v (rys. 3).



Rys. 3: Prefikso-sufiks odpowiadający okresowi słowa

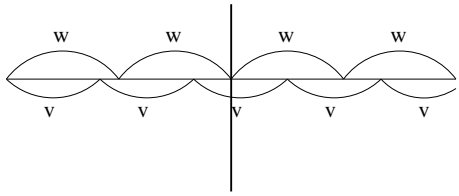
Z kolei niech teraz s będzie właściwym prefikso-sufiksem słowa u ($s \neq u$). Pokażemy, że prefiks v_s słowa u długości $|u| - |s|$ jest okresem u. Zauważmy, że wystarczy pokazać, że v_s jest okresem s (gdyż $v_s s = u$). Pokrycie słowa s wystąpieniami v_s generujemy iteracyjnie. Widzimy, że zachodzi równość $v_s s = sx$, gdzie x jest pewnym sufiksem słowa u o długości $|v_s|$. Wnosimy z niej, że v_s jest prefiksem słowa s, a zatem możemy zapisać: $v_s v_s s' = sx$, gdzie s' to słowo powstałe po obcięciu od s prefiksu v_s. Z tej równości z kolei widzimy, że $v_s v_s$ jest prefiksem słowa s, co oznacza, że zachodzi równość: $v_s v_s v_s s'' = sx$. Kontynuując to postępowanie pokrywamy w całości s powtórzeniami v_s, co pozwala wykazać, że v_s jest okresem prefikso-sufiksu s (rys. 4).



Rys. 4: Okres odpowiadający prefikso-sufiksowi słowa

Na podstawie właśnie pokazanej wzajemnie jednoznacznej odpowiedniości między okresami a prefikso-sufiksami słowa wnioskujemy, że najkrótszy okres słowa odpowiada najdłuższemu właściwemu prefikso-sufiksowi całego słowa. Ponieważ $p(k)$ to długość najdłuższego prefikso-sufiksu u, stąd $k - p(k)$ jest długością najkrótszego okresu u. Jeżeli teraz $(k - p(k))|k$, to u jest pewną potęgą tego okresu. W tym przypadku okres ten będzie oczywiście pierwiastkiem pierwotnym słowa u (gdyby istniał krótszy pierwiastek pierwotny słowa u, to byłby on zarazem krótszym okresem u). Jeżeli zaś ta podzielność nie zachodzi, to wykażemy, że słowo u jest pierwotne.

Dowód przeprowadzimy przez sprowadzenie do sprzeczności. Załóżmy nie wprost, że istnieje słowo w takie, że $u = w^j$ dla $j \geqslant 2$ — dodatkowo wybierzmy najkrótsze w o tej własności, czyli $w = \rho(u)$. Z kolei niech v_0 oznacza najkrótszy okres u. Ponieważ w także jest okresem u, więc musi zachodzić $|w| > |v_0|$, stąd słowo v_0 musi być również okresem słowa $w^2 = ww$ (rys. 5).



Rys. 5: Najkrótszy okres i pierwiastek pierwotny słowa

Wreszcie z tego, że $|w| + |v_0| \leqslant |w| + |w| = |w^2| \leqslant |u|$, na mocy lematu o okresowości widzimy, że słowo u musi mieć jeszcze inny okres — o długości $NWD(|w|, |v_0|)$ — a więc

krótszy niż w i dodatkowo taki, że u jest jego potęgą. Jest to sprzeczne z tym, jak zdefiniowaliśmy w. ■

Sformułowanie rozwiązania wzorcowego

Wykorzystując wykazane własności słów możemy skonstruować następujący algorytm. Wyznaczmy pierwiastki pierwotne wszystkich danych palindromów — z obserwacji 5 wiemy, że stosując funkcję prefiksową możemy to zrobić w czasie proporcjonalnym do sumarycznej długości danych — i posortujmy je leksykograficznie: $\rho_1 \leqslant \rho_2 \leqslant \ldots \leqslant \rho_n$. Jeśli pierwiastek ρ_i występuje w tym ciągu m_i razy, to oznacza, że palindromy, z których powstał, pozwalają utworzyć $\frac{m_i(m_i-1)}{2}$ palindromów-par. Co więcej pary skomponowane z palindromów o różnych pierwiastkach pierwotnych nie mogą być palindromami. To pozwala nam prosto wyznaczyć wynik końcowy.

Łączna złożoność czasowa tego algorytmu zależy od algorytmu sortowania, jaki zastosujemy (resztę obliczeń możemy wykonać w czasie $O(d)$). Można wykorzystać sortowanie przez scalanie — jego pesymistyczna złożoność wynosi $O(n \cdot \max(a_i) \cdot \log n)$, gdzie $\max(a_i)$ oznacza maksimum z długości słów wejściowych. Stosując sortowanie szybkie uporządkujemy słowa w takim samym czasie oczekiwanym. Można też sortować słowa za pomocą sortowania pozycyjnego; co prawda jest ono oryginalnie sformułowane dla słów równej długości, lecz można je zmodyfikować i zastosować także w przypadku dowolnych słów przy zachowaniu złożoności równej sumie długości wszystkich słów ($O(d)$). W praktyce dla danych testowych wymienione metody sortowania były czasowo nieodróżnialne i zastosowanie dowolnej z nich wystarczało do przejścia wszystkich testów.

Inne rozwiązania

Poza trzema opisanymi sposobami, istnieje bardzo dużo innych rozwiązań zadania *Palindromy*. Zawodnicy prezentowali zazwyczaj albo rozwiązania siłowe, które polegały na tworzeniu wszystkich par słów i sprawdzaniu wprost, czy otrzymane słowo jest palindromem, albo różne wersje powyższych rozwiązań, w których pewne kroki były wykonywane wolniej (na przykład sprawdzanie, czy sufiks słowa jest palindromem lub wyznaczanie pierwiastka pierwotnego słowa).

Testy

Zadanie testowane było na 10 zestawach danych testowych, zawierających łącznie 23 pojedyncze testy.

Nazwa	n	d	Opis
pal1.in	15	78	mały test poprawnościowy
pal2a.in	36 566	253 168	palindromy zaczynające się na litery `a` i `b`
pal2b.in	2	2 000 000	dwa palindromy: jeden jednoliterowy, drugi o maksymalnej długości

Nazwa	n	d	Opis
pal3a.in	893	1 995 855	potęgi krótkiego słowa; test ten pozwala wyeliminować sprytne rozwiązania siłowe
pal3b.in	33 290	1 999 959	test losowy, składający się z dużej liczby krótkich palindromów, słów postaci $a^k b a^k$ i słowa o długim pierwiastku pierwotnym
pal4a.in	999	1 998 000	potęgi krótkiego słowa; test ten pozwala wyeliminować sprytne rozwiązania siłowe
pal4b.in	98 391	1 999 981	test losowy, składający się z dużej liczby krótkich palindromów, słów postaci $a^k b a^k$ i słowa o długim pierwiastku pierwotnym
pal5a.in	1 154	1 999 305	potęgi krótkiego słowa; test ten pozwala wyeliminować sprytne rozwiązania siłowe
pal5b.in	127 351	1 999 997	test losowy, składający się z dużej liczby krótkich palindromów, słów postaci $a^k b a^k$ i słowa o długim pierwiastku pierwotnym
pal6a.in	1 413	1 997 982	potęgi krótkiego słowa; test ten pozwala wyeliminować sprytne rozwiązania siłowe
pal6b.in	38 827	1 999 999	test losowy, składający się z dużej liczby krótkich palindromów, słów postaci $a^k b a^k$ i słowa o długim pierwiastku pierwotnym
pal7a.in	1 413	1 997 982	potęgi krótkiego słowa; test ten pozwala wyeliminować sprytne rozwiązania siłowe
pal7b.in	131 775	1 999 998	test losowy, składający się z dużej liczby krótkich palindromów, słów postaci $a^k b a^k$ i słowa o długim pierwiastku pierwotnym
pal8a.in	1 413	1 997 982	potęgi krótkiego słowa; test ten pozwala wyeliminować sprytne rozwiązania siłowe
pal8b.in	37 969	1 999 999	test losowy, składający się z dużej liczby krótkich palindromów, słów postaci $a^k b a^k$ i słowa o długim pierwiastku pierwotnym
pal8c.in	105 264	2 000 000	długie słowa powodujące maksymalne zużycie pamięci dla drzewa TRIE
pal9a.in	1 999	1 999 000	potęgi krótkiego słowa; test ten pozwala wyeliminować sprytne rozwiązania siłowe
pal9b.in	80 714	2 000 000	krótkie palindromy
pal9c.in	222 222	1 999 998	długie słowa powodujące maksymalne zużycie pamięci dla drzewa TRIE

Nazwa	n	d	Opis
pal9d.in	126	1996220	test pozwalający wyeliminować rozwiązania, w których pierwiastki pierwotne są obliczane nieefektywnie
pal10a.in	2047	2000000	największy test, składający się wyłącznie z potęg słów jednoliterowych
pal10b.in	224200	2000000	długie słowa powodujące maksymalne zużycie pamięci dla drzewa TRIE
pal10c.in	108	1880780	test pozwalający wyeliminować rozwiązania, w których pierwiastki pierwotne są obliczane nieefektywnie