a)

Algorytm:

1. Skonstruujmy 4 drzewa przedziałowe dla każdej z liter d, i, k ,s, które mówią czy wystąpiła ona na danym przedziale.
2. Dla każdej pozycji liczymy kiedy ostatnio wystąpiłą dana litera.
3. Zbudujmy drzewo suffixowe(skompresowane) danego słowa.
4. Przejdźmy po drzewie suffixowym bfsem. Przy każdym przejściu sprawdźmy czy w ścieżce, której użyliśmy nie ma brakujecej litery d, i, k, s żebyśmy mieli już wszystkie w danym podsłowie.
5. Jak na danej ścieżce uzbieramy wszystkie litery, przechodząc w dół poddrzewa zaczynamy dodawać słowa do wyniku i przestajemy sprawdzać, czy nie ma brakujących liter. Proces dodawania przebiega w ten sposób, że przechodząc po krawędzi w dół dodajemy liczbę liter w krawędzi. Dodatkowo musimy dodać liczbę liczbe słów wynikających z ścieżki, na której uzbieraliśmy wszystkie litery. Aby to zrobić patrzymy za pomocą tablicy z pkt 2, która z liter, której nie mieliśmy przed przejściem tą ścieżką wystąpiła najpóźniej i dodajemy do wyniku liczbva liter w krawędzi - miejsce wystapienia najpóźniejszej litery.

Złożoność:

Kroki 1 - 3 można zrobić w O(n).

Krok 4 zajmie nam o(nlog(n)), ponieważ krawędzi w drzewie suffixowym jest O(n) i dla każdej potrzebujemy zrobić wyszukanie w drzewie przedziałowy, które zajmuje O(log(n))

Krok 5 jest liniowy.

Ostateczna złożoność O(nlog(n))

b)

Najmniejsza liczba: k-1

Zauważmy, że nie da się lepiej bo wszystkie wystąpienia pierwszej litery we wzorcu będą miałby minimalnie wartość jeden oprócz pierwszj litery. A stworzenie takiego wzroca jest możliwe, jest nim np: diks[i^(k-1)][k^(k-1)][s^(k-1)][d^(k-1)]

Największa liczba: 2(k-1)(4k-3)

Zauważmy, że nie da się lepiej, bo 4 litery(pierwsze wystąpienia d, i, k, s) będą miały wartość 0, a pozostałe wartości mogą rosnąć o maksymalnie o 1. Czyli mamy sumę {od i = 1 do 4(k-1)} {i} = 2(k-1)(4k-3). A przykładowym ciągiem który ma taką sumę jest (diks)^k. Pierwsze 4 litery mają wartość 0 kolejne mają wartości rosnące o 1 od 1 do 4(k-1)