Zaawansowany Edytor Teksu

Przed Tobą zadanie poprawkowe z PwŚG. Pojawiła się informacja, że za zadanie będzie stworzyć edytor tekstu z funkcją autokorekty oraz podpowiadania/dopełniania wyrazów, gdzie użytkownik ma mieć możliwość dodawania własnych słów do słownika oraz ich usuwania. Aby aplikacja działała sprawnie (bo potrzebujesz dostać maksa za zadanie) musisz przygotować strukturę danych pozwalającą na optymalną obsługę funkcjonalności słownika.

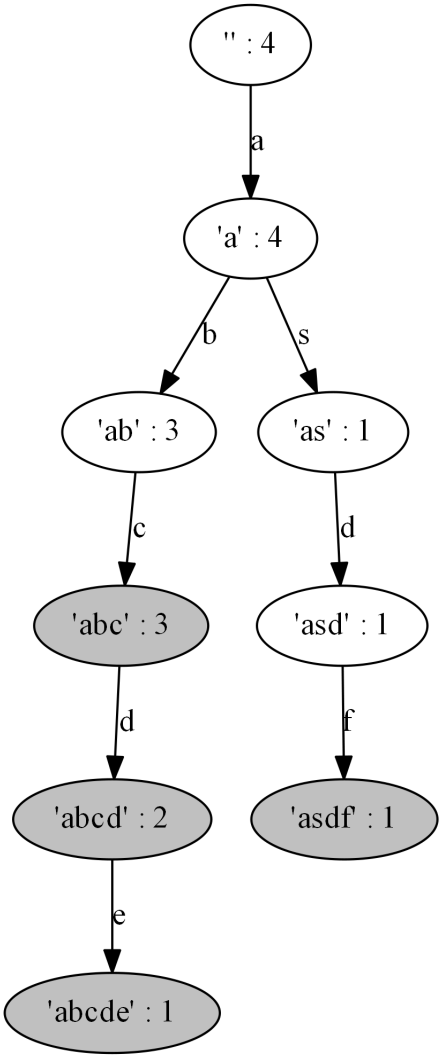
W tym celu do zaimplementowania jest drzewo prefiksowe (drzewo TRIE) pozwalające na:

- Dodawanie nowych słów
- Sprawdzanie czy słowo znajduje się w słowniku
- Usuwanie słów ze słownika
- Sprawdzanie ile i jakie słowa o podanym prefiksie znajduje się w słowniku
- Wyszukiwanie słów w zadanej odległości edycyjnej

Drzewo prefiksowe (Przypomnienie z AiSD1)

Słowa zapisywane są jako ścieżki zaczynające się w korzeniu i idące w stronę liści, kolejne litery wyrazów reprezentowane są za pomocą krawędzi. Węzły zawierają informację czy dany ciąg znaków (od korzenia do danego węzła) jest osobnym słowem czy tylko prefiksem innych słów.

Patrz obrazek poniżej – drzewo przechowujące słowa: "abc", "abcd", "abcde", "asdf". Słowa dodane do słownika oznaczono kolorem szarym, każdy węzeł jest opisany prefiksem jaki reprezentuje oraz liczbą słów o tym prefiksie występujących w drzewie.



Odległość edycyjna (Levenshteina) - miara odległości napisów (słów)

Jej wartością jest najmniejsza liczba działań (wstawienie, usunięcie i zamianę pojedynczego znaku) potrzebnych do przekształcenie jednego słowa w drugie. Do jej wyznaczenia można posłużyć się algorytmem programowania dynamicznego. Przykłady:

d("abcd","abaa") = 2

		'a'	'b'	'a'	'a'
	0	1	2	3	4
'a'	1	0	1	2	3
'b'	2	1	0	1	2
'c'	3	2	1	1	2
'd'	4	3	2	2	2

d("abcd","abede") = 2

		'a'	'b'	'e'	'd'	'e'
	0	1	2	3	4	5
'a'	1	0	1	2	3	4
'b'	2	1	0	1	2	3
'c'	3	2	1	1	2	3
'd'	4	3	2	2	1	2

d("abcd","abcd") = 0

		'a'	'b'	'c'	'd'
	0	1	2	3	4
'a'	1	0	1	2	3
'b'	2	1	0	1	2
'c'	3	2	1	0	1
'd'	4	3	2	1	0

Idea działania algorytmu:

- Dist[i,j] jest odległością edycyjną prefiksu o długości i znaków pierwszego słowa od prefiksu o długości j znaków drugiego słowa
- Górną i lewą krawędź tabeli dist wypełnia się odległościami początkowych fragmentów (kolejnych coraz dłuższych prefiksów) porównywanych słów od słowa pustego.
- Dla i,j>0 element dist[i,j] oblicza się na podstawie elementów dist[i-1,j-1], dist[i-1,j] i dist[i,j-1]. Innymi słowy odległość edycyjną prefiksów porównywanych słów o długości i oraz j obliczamy na podstawie obliczonych wcześniej odległości krótszych (albo tej samej długości) prefiksów tych słów. Oczywiście jako dist[i,j] wybieramy minimalną z odległości możliwych do uzyskania przy zastosowaniu każdej z trzech operacji wstawienia, usunięcia lub zamiany po dodaniu kosztu tej operacji (u nas każda operacja ma koszt 1). Oczywiście w przypadku zgodności odpowiednich znaków zadnej operacji wykonywać nie trzeba – dodatkowy koszt jest zerowy.
- Wynikiem działania algorytmu jest wartość w prawym dolnym rogu tabeli, która to jest równa odległości edycyjnej danych słów.
- Z tabeli można również odczytać jakie operacje przekształcają jedno słowo na drugie.

Punktacja

- AddWord – 0.5 pkt
- CountPrefix i Contains – 0.5 pkt
- AllWords – 0.5 pkt
- Remove – 0.5 pkt
- Search – 0.5 pkt

Uwagi i wskazówki

- Można zauważyć, że dla wspólnych prefiksów słów "abaa"i "abed"pierwsze dwie kolumny są identyczne, przy wyznaczaniu odległości do tego samego słowa "abcd"(pierwsze dwie tabelki)
- Rozwiązanie polegające na zwróceniu wszystkich słów ze słownika i sprawdzeniu dla każdego z nich odległości edycyjnej będzie zbyt wolne, należy zrobić to wydajniej
- W kodzie podane są złożoności wymagane każdej z metod, których należy przestrzegać