Algorytmy

Jakub Heczko

1 Oznaczenia:

n - wielkość słowa
m - ilość słów w słowniku

2 Opis problemu:

Problem w tym zadaniu, było to aby móc znaleźć odpowiednie słowo bazując na kombinacji liter, które dostajemy. Najłatwiej, było to zrobić za pomocą hashmap, które, były ze sobą sprzężone, co oznacza, że kolejne zależały od poprzednich, tworzyliśmy je na podstawie tego jaki numerke wybraliśmy i jakie wybieraliśmy, można to pokazać za pomocą takiego algorytmu:

Pseudo kod:

funkcja pressedKey(int number)
pos++;
count = 0;
listOfWords wyczysc //zmienna uzywana do wyswietlania slow
baseToReset.add(words);
HashMap newMap = nowa pusta Mapa;
for(Character c in wszystkie charaktery pod danym numerem)
ArrayList list = Jezeli w naszych slowach pod dana literka mamy jakis array to go zwracmy, wpp zwroc null
Dodaj rowniez slowo do listy slow do wyswietlania for(String s in list)
Jezeli pozycja wedlug ktorej tworzymy slowa < dlugosc slowa:
Jezeli nasza nowa mapa juz zawiera klucz to poprostu dodaj do niej slowo, ktore zawiera litere c na pozycji pos

WPP: Utworz nowy klucz w mapie z arraylistem i do niego wsadz slowo, ktore pasuje
Ustaw zmienna words na naszą aktualna hashmape ze slowami

3 Użyte strukty:

Używałem hashMap oraz ArrayList, które są poprostu listami tablicowymi w Javie. HashMapy to normalna mapa, która używa hashowania.

4 Oszacowanie Złożoności Struktur:

Na hashMapie wszystko wykonujemy w O(1), pomijam narazie to co jest w srodku naszej hashmapy, chociaż już operacja wstawiania na listach jest również w O(1), tak iteracja po kolejnych elementach listy jest w O(n), w zaleznosci od ilosci elementow w liscie. Same struktury, zajmują oczywiście dokładnie m Stringów na początku

5 Oszacowanie złożoności algorytmu głównego:

W najgorszym wypadku bedziemy mieli 0(3nm), jeśli chodzi o pamieć, to pomijając, rzeczy, które musimy wyświetlić, będziemy mieli O(m), bo nasza ilość danych nie przekroczy wielkości słownika, który podajemy.