Autocomplete
Проект за курса Структури от данни и алгоритми за
специалност Информатика, зимен семестър 2018/2019 г.

Изготвил: Тихомир Каменов, ФН: 45431

Идея на реализация на проекта:

За дадения речник ще се създаде минимален краен детерминиран автомат. Символният низ (префиксът), който потребителя въвежда, ще се пусне в автомата, за да се извлекат низове-предложения, които съдържат този префикс.

Уточнение 1: Азбуката, с която ще се работи, ще са първите 128 символа от ASCII таблицата.

Уточнение 2: Две състояния p и q са еквивалентни, ако за всяка дума w от азбука* d*(p, w) = d*(q, w), тоест попадаме на едно и също състояние, прочитайки думата от тези две състояния. В рамките на проекта, ще се използват необходимите и достатъчни условия от източник, а те са, че две състояния са еквивалентни, ако 1) и двете са или финални, или не са финални; 2) имат еднакъв брой излизащи преходи; 3) съответните им преходи имат еднакви етикети (символи); 4) съответните преходи водят до състояния с еднакви остатъци.

Уточнение 3: Алгоритмите, които се ползват, са взети от този документ:
https://www.aclweb.org/anthology/J00-1002

Имплементация:

Ще се реализира автомат, който разпознава думи от речник. Състоянията на автомата ще представляват структура State, съдържаща вектор от остатъци (стрингове) (ако имаме произволна дума w = ab, и на състояние p сме прочели a, списъкът на състояние p ще съдържа b), който ще наричаме десен език на състоянието. Класът за автомата ще съдържа вектор от състояния (като ще се добавят в него единствено състояния, които имат уникален десен език), указател към състояние, който играе ролята на начално състояние, и функция за преход. Функцията за преход ще представлява хеш таблица с ключ наредена двойка от указател към състояние и символ и стойност указател към състояние. Класът ще има като публични методи функция, която минимизира дадения автомат, функция, която добавя дума към автомата, тоест да може да се разпознава подадената дума от автомата. Като прайвит метод класът ще има функция, която по подаден стринг, връща състоянието, на което се намира автомата при прочитане на стринга. Ще имаме и още един публичен метод, на който се подава стринг и който връща вектор от стрингове и представлява следното: подаденият стринг се използва като аргумент на прайвит метода, за да се намери състоянието, на което се намира автомата при прочитането на този стринг след което се конкатенира стринга с остатъците от това състояние и резултатите се добавят във вектор от стрингове, който функцията връща.

Минимизиране на автомат: Ще се използва следният алгоритъм (точка 3, страница 5 от прикачения документ), който при добавяне на нова дума, модифицира автомата, преобразувайки

го в минимален. По този начин се пести памет, тъй като по всяко време автоматът, който построяваме, ще има възможно най-малко състояния. Също така за да работи алгоритъмът, трябва речникът, който използваме, да е сортиран. Тъй като речникът е сортиран, е достатъчно да разглеждаме състоянията на последната добавена дума, тъй като единствено там е възможно да се наложи да се премахне състояние.
Идеята на алгоритъма е следната: всеки път се добавя нова дума от речника и се определя частта, която е вече в автомата (общ префикс), и частта, която не е (суфикс). След което се проверява дали последното състояние, отговарящо на последния символ от частта, която е в автомата, има деца. В случай, че има, това значи, че не цялата предходна дума е в префикса. В такъв случай се извиква функцията replace_or_register с аргумент последното състояние, която извършва процедурата по минимизиране, като рекурсивно проверява дали дадено състояние от суфикса на предишната добавена дума има еквивалентно състояние вече в автомата. Ако едно състояние p има такова q, се поставя преход от родителя на p към q и се изтрива p. Ако няма, текущото състояние p се добавя в регистър, съдържащ всички състояния, които имат „уникален“ остатък. След като се добавят всички думи, се прилага функцията отново за началното състояние на автомата, тъй като автоматът все още не е минимизиран след последната добавена дума.

Добавяне на дума към автомата: Добавянето на дума няма как да се извърши със същия алгоритъм, който се използва за минимизиране на автомата по простата причина, че добавянето на думи може да не се случва в лексикографски ред, а алгоритъмът работи със сортиран вход. Възможно е да съществува състояние с повече от едно „влизащо“ ребро (такива състояния се наричат confluence) и тъй като не се прибавят думите задължително в лексикографски ред, е възможно при добавяне на дума автоматът да разпознава повече думи отколкото трябва (пример: нека към автомат, който разпознава думите bad и abd, прибавим думата bae. Това ще добави едно ребро от a към d, но по този начин автоматът ще може да разпознава и думата abe, а не това е желаният резултат). За целта ще се използва втория алгоритъм от източника (точка 4, страница 10 от прикачения документ), който се използва при несортиран вход. Идеята е, че също като предишния алгоритъм, ще се разглежда частта от думата, която вече се съдържа в автомата (ще я наричаме префикс). Ако има confluence състояния, се клонират (тоест се създава ново копие на състоянието, със същите ребра, влизащи и излизащи от него) след което се добавя суфикса към автомата и минимизираме (тоест прилагаме функцията replace_or_register).

Ще се реализира клас AutoComplete, който ще съдържа автомат, който разпознава стрингове, и число за това колко най-много предложения да се изписват на екрана. Като функционалност, класът ще има прайвит метод, който създава автомата спрямо речник (имаме път до файл, съдържащ стрингове, които добавяме във вектор от стрингове, сортираме го и се прилага алгоритъма за минимизиране с вектора като вход); метод, който добавя дума към автомата (използваме втория описан алгоритъм); метод, който по подаден стринг (префикс) връща списък от стрингове, на които този стринг е префикс (извиква метода от класа на автомата, който връща списък от стрингове).

Забележка:

Възможно е това да не е окончателният вариант и по време на работа да има леки промени или подобрения, но основната идея и алгоритмите, които се полват, ще бъдат тези.