Первый этап работы

Составить алгоритм генерации псевдослучайного слова – названия для одного из предметов мебели икея. Каждая последующая буква должна выбираться случайным образом с учетом вероятности выпадения ее после предыдущей уже выбранной буквы. Для реализации данного алгоритма изучались вопросы: «Марковские сети» и «Алгоритм Монте-Карло». Реализован данный алгоритм на javascript.

Сначала был взят файл названиями предмета. В конце каждого слова был вставлен специальный символ – “#”. Решетка означает конец слова. Далее это пригодится, когда программа будет генерировать случайное слово. После выпадения решетки программа будет заканчивать свою работу. Все, что стоит до решетки, и есть наше слово. На следующем шаге было создано два массива.

Создали одномерный массив номер один. Он нужен для того, чтобы ставить в соответствие таблице частотности буквы. Т.е. у каждой ячейки в массиве есть свой номер. Сначала пробежались по строчке и добавили в одномерный массив все буквы, которые есть в тексте. Добавление происходит только тех символов, которые не присутствуют в массиве. (См. прил. 1)

Создали массив номер два. Двумерный массив n\*(n+2), где n – это длина первого массива. Он будет служить таблицей частотности. Т.е. у каждой ячейки в массиве есть свой номер. Первый индекс двумерного массива – это первая буква в словосочетании (в данном случае словосочетание состоит из двух букв), второй индекс – вторая буква. Под теми же индексами во втором массиве будут храниться буквы. Например, индексы один и два в первом массиве дадут словосочетание, например, “по”.

Затем бежали по тексту еще раз и составили таблицу частотности: какое количество раз различные буквы встречались после заданной. Пробежались и записали количество в соответствующие ячейки. В конец для удобства добавились еще две колонки. Сумму всех частот после каждой буквы и символ, после которого вычисляются частотности. (См. прил. 2)

Теперь генерирование слова. Изначально, можно подумать, что первая буква имеет право быть любой, однако, это не так. В таблице так же есть частотность букв, которые встречаются после пробела, т.е. являются первыми буквами в словах. Именно среди них и необходимо выбирать первую букву для генерирующегося слова. Для удобства пишем первым символом пробел. Далее находим его индекс в таблице частотности и переходим в предпоследний столбец, чтобы узнать, сколько раз после него встречалась какая-либо буква. Получаем число, назовем его m. Далее нужно сгенерировать случайное число от единицы до m. Получаем некоторое число. Теперь идем перебором по строке, прибавляем встреченные нами значения к некоторой переменной, которую сравниваем с рандомным числом. Когда произойдет переполнение переменной, т.е. она станет больше, чем указанное рандомное число, необходимо остановиться. Буква на предыдущем шаге –следующая буква, которая будет записана после пробела. Шансы выпасть некоторой букве пропорциональны количеству выпавших раз ранее. Получается, что если происходит переполнение на данном шаге, но, при этом, шаг назад, число было не достаточно большим, то именно в том промежутке и лежала случайно выпавшая точка. Это и есть тот самый промежуток, и та самую букву. Записываем ее. Теперь переходим на строку с новой буквой в двумерном массиве. Аналогично смотрим в предпоследний столбик и выбрасываем случайное число в промежутке от 1 до числа, указанного в последнем столбике. Алгоритм продолжается до тех пор, пока не выпадет решетка – специальный символ конца слова. (См. прил. 3)

Приложение 1

Входные данные: мама мыла раму

Массив1

1 шаг

|  |  |
| --- | --- |
| 1 | 2 |
| м | а |

2 шаг

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| м | а | # |

3 шаг

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| м | а | # | ““(пробел) |

4 шаг

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| м | а | # | ““(пробел) | ы |

5 шаг

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| м | а | # | ““(пробел) | ы | л |

6 шаг

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| м | а | # | ““(пробел) | ы | л | р |

7 шаг

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| м | а | # | ““(пробел) | ы | л | р | у |

Приложение 2

Массив2

Входные данные: мама# мыла# раму#

Массив1

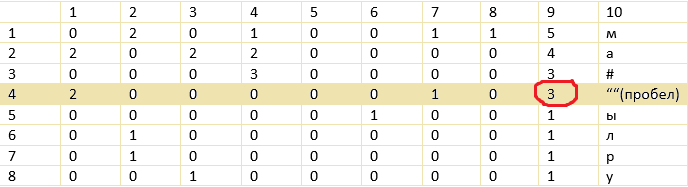
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| м | а | # | ““(пробел) | ы | л | р | у |

1 шаг

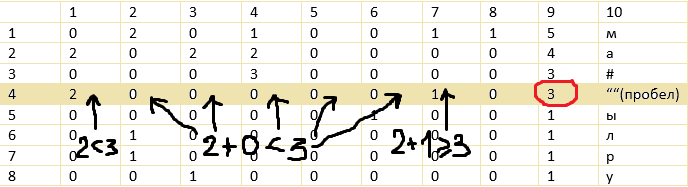
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 0 | 2 | 0 | 1 | 0 | 0 | 1 | 1 | 5 | м |
| 2 | 2 | 0 | 2 | 2 | 0 | 0 | 0 | 0 | 4 | а |
| 3 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 3 | # |
| 4 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 | ““(пробел) |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | ы |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | л |
| 7 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | р |
| 8 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | у |

Приложение 3

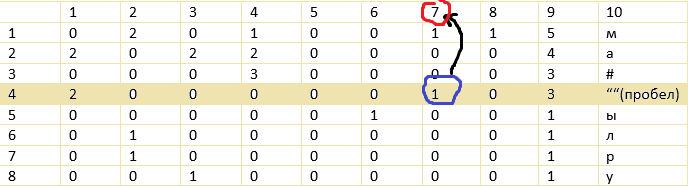
1. Генерируем число от 1 до 3

****

1. Допустим, у нас получилось число 3



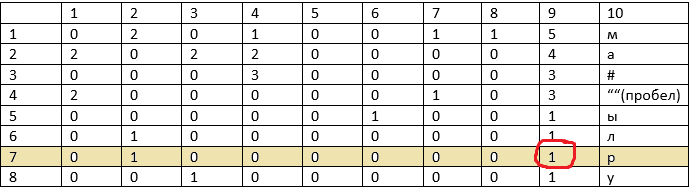
1. Дошли до числа равного 3. Если бы оно было меньше, чем 3, а на следующем шаге уже было бы больше. Мы бы тоже предпочли его, т.к. по алгоритму мы берем число предыдущее тому, на ячейке которого произошло переполнение.



1. Смотрим под каким индексом расположено данное число в первом массиве.

C:\Users\hp\Desktop\Курсовая\Картинки для отчета\4.png

1. Первая буква нашего слова «Р». Запишем ее. Далее переходим во второй массив в строчку с индексом, по которым находилась данная буква. И берем случайное число в массиве под индексом [7][9].



1. См. пункт 2. Проделываем аналогичную манипуляцию, пока счетчик не остановится на «#». Тогда наше слова заканчивается.