Лицей «Физико-техническая школа» Санкт-Петербургского Академического университета

Курсовая работа (отчет по практике)

Создание программы генериующей кроссворды из регулярных выражений

Работу выполнили:
Данилевич Леонид (2022А)
Лельчук Александр (2022А)
Научный руководитель:
Дворкин Михаил Эдуардович
Место прохождения практики:
Лицей «ФТШ»

Санкт-Петербург, 2021

Мы смогли разработать алгоритм генерации строк, которые подходят под различне человекочитаемые паттерны (например: палиндромы, прогресии, словарные ключи, повторы и так далее), которые могли бы легко восприниматься человеком. Также мы смогли создать уникальный алгоритм генерации наиболее короткого регулярного выражения для заданной строки.

Содержание

1 Введение

Регулярные выражения — формальный язык поиска подстрок в тексте и манипуляций с ними. Например, регулярному выражению «.*amp(le)?» соответствуют строки «Sample», «example», «lamp» и некоторые другие. С помощью регулярных выражений можно достаточно просто искать в тексте подстроки определённого формата и заменять их на соответсвующие им другие подстроки. Новичкам, изучающим регулярные выражения, для закрепления нового материала полезно решить такой (https://gregable.com/p/regexp-puzzle.html) кроссворд (изображение ниже). Конкретно этот экземпляр создан, как задание конкурса «MIT Mystery Hunt» 2013 года. Мы решили написать приложение, автоматически генерирующее подобные кроссворды, а также позволяющее их решать. Мы считаем, что такое приложение будет полезно многим людям, изучающим регулярные выражения, а для знакомых с ними оно будет просто интересно.

Необходимо разработать алгоритм генерации строк с различными паттернами, которые могли бы легко восприниматься человеком. И создать алгоритм генерации наиболее короткого регулярного выражения для задданной строки, а затем создать прилоожение на базе Android, реализующее этот функционал.

2 Методика решения задачи

Так как задача состоит в написании программы, то решается программно. Задача дробится на части:

1. Собственно, решение уже имеющегося кроссворда. Понадобится для оценки сложности кроссворда и и удостоверения единственности решения.

   (a) Для решения произвольного кроссворда необходимо разобрать(«распарсить») регулярные выражения, кодирующие его строчки. Для этого написана рекурсивная функция с запоминанием ответа (техника «мемоизации» динамического программирования), которая для заданного регулярного выражения и требуемой длины строчки находит в компактном виде всевозможные строчки, подходящие под данное выражение. Компактный вид обусловлен тем, что строчки состоят не из символов, а из битовых масок, где из любой маски можно выбрать любой символ и получить корректную строчку. «Обратные ссылки» (back references) в регулярных выражениях обрабатываются следующей техникой: два символа, запрашиваемые быть одинаковыми, соотвествуют одному и тому же битовому объекту-маске.

   (b) После разбора регулярных выражений кроссворд решается методом инкрементальных улучшений. Изначально каждой букве сопоставлена битовая маска, соответствующая всем символам алфавита. Алгоритм рассматривает все ещё не разгаданные буквы по порядку, сужая для каждой множество возможных значений. Существует вероятность встретить кроссворд, нерешаемый данной техникой. Однако для его разгадывания человеку потребуется долгий и утомительный перебор. Создание таких кроссвордов не входит в нашу задачу. В случае полного разгадывания кроссворда можно утверждать, что его решение существует и единственно.

2. Генерация буквенного заполнения кроссворда. Должны образовываться строчки, подходящие под различные регулярные выражения. Максимизируется красота заполнения — субъективное свойство, включающее в себя регулярность кроссворда, т. е. различные повторения подстрок и английские слова.

   (a) Для генерации случайного поля, максимизирующего потенциальную оценку кроссворда, а именно, разнообразие типов регулярных выражений, разгадка необходимой степени сложности, и субъективная оценка красоты, применён метод имитации отжига («simulated annealing»). Большое количество итераций заменяют одну букву на изначально равномерно случайно сгенерированном поле, после чего в некотором случае изменение принимается, поле обновляется.

3. Генерация регулярных выражений, лучшим образом задающих получившиеся строчки.
   (a) Пока не реализована

4. Оценка сложности, проверка единственности решения, путём программного решения
   Предполагается применять усреднённое значение количества итераций, требуемых алгоритму (1) для решения сгенерированного кроссворда при рассмотрении клеток, содержащих неразгаданные буквы, в случайном порядке

   Как мы будем это делать?
   Оценка сложности:
   Стоимость строки – её "красота"для человека.
   Генерация поля разбита на две основные части:
   Оценка стоимости конкретной строки (сложность – длина строки, алгоритм линейный $O(n)$
   Выбор оптимального способа генерации при помощи алгоритма отжига (с некоторой вероятностью выбираем строку с большей стоимостью)
   Сложность отжига: линейно зависит от размеров кроссворда (но константа работы большая)
   Итого $O(n \cdot m)$ – сложность алгоритма.

   Проверка единственности решения:
   Мы также разработали алгоритм (он уже существует), который умеет решать уже готовую говололомку, при этом он может искать множества решений, и если множество решений > 1, то об этом можно легко узнать и запустить генерацию заново (из-за этого генерация может идти несколько секунд).

3 Результаты

На рисунках ниже изображены ввод, предоставляемый нашей программе с консоли, и её вывод. Это кроссворд, упоминавшийся во введении, и можно убедиться, что он разгадан корректно.

```
.*SE.*UE.*
.*LR.*RL.*
.*OXR.*
([^EMC]|EM)*
(HHX|[^HX])*
.*PRR.*DDC.*
.*
[AM]*CM(RC)*R?
([^MC]|MM|CC)*
(E|CR|MN)*
P+(..)\1.*
[CHMNOR]*I[CHMNOR]*
(ND|ET|IN)[^X]*

.*H.*H.*
(DI|NS|TH|OM)*
F.*[AO].*[AO].*
(O|RHH|MM)*
.*
C*MC(CCC|MM)*
[^C]*[^R]*III.*
(...?)\1*
([^X]|XCC)*
(RR|HHH)*.?
N.*X.X.X.*E
R*D*M*
.(C|HH)*

.*G.*V.*H.*
[CR]*
.*XEXM*
.*DD.*CCM.*
.*XHCR.*X.*
.*(.)(.)(.)(.)\4\3\2\1.*
.*(IN|SE|HI)
[^C]*MMM[^C]*
.*(.)C\1X\1.*
[CEIMU]*OH[AEMOR]*
(RX|[^R])*
[^M]*M[^M]*
(S|MM|HHH)*
```

```
S E C U E M C
M L R C R L M C
M M X O X R X M H
H E M H E M H E M H
H H X M I R H H X D C
H P R R M I O H H D D C
S T X M C M I E C R X R G
A M A M M C M R C R C R
H O X M M C C O X R N
E M N M N C R E C R
P O X O X C X R V
H I O M C M R O
N D F M M C H
```

Рис. 1: Регулярные выражения; ввод

Рис. 2: Решение кроссворда нашей программой

Пока результаты не очень большие, но мы сделали многое из того, что хотели.
Мы получили новый алгоритм, которого раньше не существовало, и который в перспективе может использоваться не только в наешм приложении, но также для генерации шаблонов для описания различных строк на основании только лишь одной входной строки.

Наш научный руководитель: **Дворкин Михаил Эдуардович**