Практикум №1 по алгоритмам

October 28, 2016

1 Построение автомата

Заводится стек "подавтоматов" — участков автомата, имеющих начало и конец и составляющих в будущем искомый автомат. Они хранятся в стеке.

При поступлении на вход оператора из стека извлекается необходимое количество операндов, и из них строится новый подавтомат, который кладётся на вершину стека. Построение нового автомата осуществляется так же, как было на семинаре.

Подробнее — в коде.

2 Поиск ответа

Будем искать длину максимального слова, состоящего из x. Если она больше или равна k, то нужное слово есть, иначе нет.

Будем ходить обходом в глубину по автомату. Если из вершины выходит ребро с меткой x, то двигаемся по нему, увеличивая глубину на 1. Если ϵ-ребро, то оставляем глубину такой же. Рёбра с другими пометками нас не интересуют, так как они обрывают цепочку из переходов по x, то есть обрывают искомое слово. В процессе поиска в глубину мы можем попадать в белые, серые и чёрные вершины.

Пусть мы попали в v. v является чёрной, если dfs из неё уже отработал, найдя максимальную длину слова. Просто возвращаем ответ. Если вершина серая, то мы в процессе обхода нашли цикл. Если при этом глубина $depth[v]$ больше, чем текущая глубина обхода, то мы нашли цикл из букв x и, возможно, некоторого количества ϵ-переходов. По нему мы можем получить произвольную длину слова. Если же глубина такая же, это значит, что цикл весь состоял из ϵ-переходов. Он нас не интересует. Если же вершина белая, то необходимо запустить из неё dfs (причём если мы пришли в неё по x, то увеличиваем глубину и ответ на 1, если по ϵ — оставляем прежними; подробнее в коде). Улучшаем ответ, если можно.

Запустив так dfs из всех вершин, мы найдём максимальную длину слова. Если мы найдём цикл по x и ϵ, то, очевидно, алгоритм корректен (а dfs всегда находит цикл по факту, доказанному на курсе алгоритмов). Иначе же слово x^k имеет вид цепочки из переходов x. Эта цепочка где-то начинается. Если мы запустим dfs из этой вершины, то найдём такую цепочку (очевидно). Поэтому алгоритм корректен.

3 Время работы

Пусть N — длина регулярного выражения. Построение автомата выполняется за один проход, один символ обрабатывается за $O(1)$. dfs работает за $O(N)$, запускается N раз, итого $O(N^2)$. Восстановление ответа за $O(N)$. Общее время работы $O(N^2)$.

Вообще, в целях оптимизации можно заканчивать алгоритм, как только было найдено слово нужной длины, но особого значения это не имеет.

4 Проверка регулярного выражения на корректность

Единственная проблема, с которой можно столкнуться — несоответствие количества операторов и нужных операндов.

Если считали оператор, проверяем, находится ли в стеке нужное количество операндов (равное арности оператора), и если не так, бросаем исключение.

Если после окончания работы в стеке больше чем один подавтомат (меньше уж точно быть не может, так как оператор всегда кладёт подавтомат в стек, а извлечь больше, чем в стеке есть элементов, нельзя), у нас недостаточно операторов, и следует бросить исключение.