Описание алгоритма первого практического задания.

Дмитрий Мурзин, 595

Даны α и слово $word \in \{a, b, c\}^*$.
Требуется найти длину самого длинного префикса $word$, принадлежащего L

Пусть α — регулярное выражение. Сопоставим каждому символу α_j регулярное выражение $\alpha[i..j]$ (то есть для каждого j мы находим индекс i, то есть имеем функцию $I : j \mapsto i$) по следующему правилу:

- если α_j — буква, то $I(j) := j$
- если α_j — символ звездочки, то $I(j) := I(j-1)$
- если α_j — символ умножения или сложения, то $I(j) := I(I(j-1)-1)$

Будем решать задачу динамическим программированием. Пусть $m = |\alpha|$, $n = |word|$. Заведём трёхмерный массив $dp[0..m)[0..n][0..n]$ (то есть он будет размерности $m \cdot (n+1) \cdot (n+1)$) со следующим смыслом: $dp[k][i][j] :=$ можно ли получить подслово $word[i..j)$ из регулярного выражения, соответствующего символу rpn_k. Считать его мы будем следующим образом:

- если rpn_k — буква, то для всех индексов i таких что $word_i == rpn_k$ присваиваем $dp[k][i][i+1] = true$
- если rpn_k — символ сложения, то найдём индексы k_1 и k_2, которые являются операндами для rpn_k. Более конкретно $k_2 := k-1$, $k_1 := I(k-1)-1$. Тогда присвоим $dp[k][i][j] = dp[k_1][i][j] \mid\mid dp[k_2][i][j]$.

  Здесь мы говорим, что подслово $word[i..j)$ может быть получено из регулярного выражения, соответствующего rpn_k, если оно может быть получено из первого операнда для rpn_k или из второго.

- если rpn_k — символ умножения, то аналогично предыдущему пункту найдём индексы k_1 и k_2. Далее переберём все d такие что $i \le d \le j$ и если хотя бы для одного d верно $dp[k_1][i][d]$ && $dp[k_2][d][j]$, то присвоим $dp[k][i][j] = true$.

  Здесь мы говорим, что подслово $word[i..j)$ может быть получено из регулярного выражения, соответствующего rpn_k, если существует разбиение этого слова на два слова: $word[i..j) = word[i..d)word[d..j)$ и каждое из этих двух слов может быть получено из соответствующих операндов для rpn_k.

- если rpn_k — символ звёздочки, то найдём индекс k_0, который является операндом для rpn_k. Более конкретно $k_0 := k-1$. Далее переберём все d такие что $i \le d \le j$ и если хотя бы для одного d верно $dp[k][i][d]$ && $dp[k_0][d][j]$, то присвоим $dp[k][i][j] = true$.

Здесь мы говорим, что подслово $word[i..j)$ может быть получено из РВ для rpn_k, если существует разбиение этого слова на несколько слов: $word[i..j) = word[i..i_1)word[i_1..i_2)\dots word[i_l..j)$ и каждое из этих слов может быть получено из РВ для rpn_{k_0}. Это в свою очередь эквивалентно тому, что существует разбиение слова $word[i..j)$ на два слова: $word[i..j) = word[i..i_l)word[i_l..j)$ и первое слово может быть получено из РВ для rpn_k, а второе из РВ для rpn_{k_0}.