Теория формальных языков и трансляций. Домашняя работа №1

Кубышкин Е.А., группа 21.Б-07

12 сентября 2023г.

Упражнение I-1.1

ДАНО: Функция $K(i,j) = \frac{(i+j-1)(i+j-2)}{2} + j$.

НАЙТИ: Обратные функции $J(k), I(k)$ такие, что $J(K(i,j)) = j$ и $I(K(i,j)) = i$.

МЕТОД РЕШЕНИЯ: Напишем код на языке Pascal.

РЕШЕНИЕ: Функция DiagNumberAndStart проходит по всем диагоналям, начиная с первой, и фиксирует наибольшее значение, доступное на ней. Возвращает она номер диагонали и число с которого диагональ начинается, так как эту информацию удобно использовать.

Мы знаем, что j — это сдвиг числа по диагонали. Таким образом, зная начальное значение диагонали, и значение $K(i,j)$, можем высчитать j по формуле $j = \text{diagStart} - K(i,j) + 1$.

Так же знаем, что $i + j = \text{diagNum} + 1$, так что можем выразить $i = \text{diagNum} + 1 - j$. В программе j высчитывается напрямую, без использования функции $J(k)$ ради производительности.

```
function DiagNumberAndStart(k: integer): (integer, integer);
var
  n: integer;
  count: integer;
begin
  n := 1;
  count := 1;
  while count < k do
  begin
    n := n + 1;
    count := count + n;
  end;
  Result := (n, count - n + 1);
end;

function J(k: integer): integer;
var
  diagNum: integer;
  diagStart: integer;
begin
  (diagNum, diagStart) := DiagNumberAndStart(k);
  Result := k - diagStart + 1;
end;

function I(k: integer): integer;
var
  diagNum: integer;
  diagStart: integer;
begin
  (diagNum, diagStart) := DiagNumberAndStart(k);
  Result := diagNum + 1 - (k - diagStart + 1);
end.
```

Упражнение I-1.2

ДАНО: Функция $\hat{K}(w,x,y) = K(w,K(x,y))$, где

$$K(x,y) = \frac{(x+y-1)(x+y-2)}{2} + y.$$

НАЙТИ: Тройка чисел w,x,y такая, что $\hat{K}(w,x,y) = 1000$.

МЕТОД РЕШЕНИЯ: Пользуясь результатами предыдущего упражнения, получим ответ программно.

РЕШЕНИЕ: Следуя определению функций $\hat{K}, I, J$ получаем, что $w = I(1000), x = I(J(1000)), y = J(J(1000))$. Посчитаем значения этих функций: $w = 36, x = 1, y = 4$.

Упражнение I-1.3

ДАНО: Рекурсивный язык L.
НАЙТИ: Процедура для перенумерации предложений языка L.
МЕТОД РЕШЕНИЯ: Предъявим описание процедуры
РЕШЕНИЕ: Описание процедуры:

1. Пусть i — счетчик, изначально равный 1
2. Рассматриваем все предложения из множества V^* и проверяем каждое из них с помощью распознающего алгоритма на принадлежность L
3. Если предложение принадлежит L, то назначаем ему номер i, увеличиваем i на единицу, рассматриваем следующее предложение из V^*
4. Если предложение не принадлежит L, то рассматриваем следующее предложение из V^*

Упражнение I-1.4

ДАНО: Процедура, перечисляющая множество целых в монотонном порядке.

НАЙТИ: Доказать, что это множество рекурсивно, т.е. существует алгоритм определения, находится ли данное целое в этом множестве.

МЕТОД РЕШЕНИЯ: Воспользуемся тем, что процедура порождает элементы множества в монотонном порядке и предъявим описание искомого алгоритма.

РЕШЕНИЕ: Описание алгоритма (пусть процедура порождает числа в возрастающем порядке):

1. Пусть N — число, принадлежность которого к множеству мы хотим проверить

2. Порождаем очередное число M с помощью процедуры, если же все числа множества порождены, завершаем алгоритм с отрицательным ответом

3. Если $N = M$, то N лежит в множестве, и алгоритм завершается с положительным ответом

4. Если $M > N$, то в силу монотонности порождаемых чисел никакое следующее из них уже не совпадет с N, и поэтому алгоритм завершается с отрицательным ответом

5. Если $M < N$, то возвращаемся к шагу 2)

Алгоритм завершится, поскольку в случае конечного множества будет перебрано не больше элементов, чем его мощность, а в любом бесконечном множестве целых рано или поздно найдется число, большее N.

Упражнение I-1.5

ДАНО: Конечное множество S.

НАЙТИ: Показать, что S рекурсивно (найти алгоритм определения, находится ли некоторый элемент a в этом множестве).

МЕТОД РЕШЕНИЯ: Предъявим описание алгоритма.

РЕШЕНИЕ: Составим полный список элементов (аналогично тому, как может быть представлен конечный язык). Алгоритм будет сравнивать элемент a со всеми элементами списка по порядку, и в случае совпадения завершится и вернет положительный ответ. Так как множество S конечно, если a не содержится в S, все элементы списка будут проверены, и, когда это произойдет, алгоритм завершится с отрицательным ответом.