Wojciech Goralewski s464983

Zadanie algorytmiczne nr 19

Algorytm będzie na początku sprawdzał długość słowa P lub czy wpisane n jest równe 0. W takim przypadku będę zwracał język składający się ze słowa pustego. Następnie w dwóch pętlach stworzę przesuwające (o jeden symbol aż do **długości słowa – n +1** ponieważ po tym indeksie niemożliwym będzie otrzymanie podsłów szukanej długości = będą za krótkie) się okno o rozmiarze n, które będzie zwracać mi 4 symbole ze słowa a następnie utworzę z nich nowe słowo i finalnie dodam je języka L.

Następnie sprawdzę czy słowo należy już do języka za pomocą funkcji MEMBERSHIP(). Jeżeli nie należy to dodam je do listy. W ten sposób uniknę powtórzeń – czyli otrzymam zbiór.

W algorytmie również uwzględniam, że każde słowo ma za swoje podsłowo słowo puste.

Wejście: P – słowo, n – liczba naturalna

Wyjście: L – język składający się ze wszystkich podsłów słowa P o długości n

SUBWORDS(P,n)

```
lenP := LENGTH(P)
L := [-]
L := MAKELIST([], L)        %tworzę jezyk składający się ze słowa pustego aby je uwzględnić %
if lenP < n or n == 0 then
        return L            %jeżeli n wykracza poza długość słowa lub jest to słowo puste-%
                            %-zwracam język składający się ze słowa pustego%
for i = 1 to lenP – n +1 do         % tworzę początek „okna” które będzie przechodzić przez słowo%
słowo := []
        for j = 1 to n do           % iteruje przez n-następnych wyrazów aby otrzymać n podsłowo%
                symbol := POS(P,  i + j – 1)            %zapamiętuję każdy symbol w „oknie%
                słowo := MAKELIST(symbol, słowo)    %i tworzę z nich nowe słowo%
        słowo := REV(słowo)    %odwracam słowo, ponieważ MAKELIST odwraca porządek%
        if MEMBERSHIP(L, słowo) == ‘NIE’ then   %sprawdzam czy słowo należy do zbioru L%
                L := MAKELIST(słowo, L)          %dodaje jeżeli nie należy%
return L
```

Zadanie algorytmiczne nr 20

Algorytm na początku sprawdzi czy podane słowo jest puste. Jeżeli tak to zwrócę język składający się z słowa pustego.

Następnie w pętli przejdę przez każde możliwości podsłów słowa P startując od wyrazów długości 1 do podsłowa długości P. Używam do tego funkcji SUBWORDS z poprzedniego zadania algorytmicznego. Kolejnym krokiem jest dodanie każdego elementu z wyjścia funkcji SUBWORDS do języka L (który będzie językiem składającym się ze wszystkich podsłów słowa P). Jest to zrobione za pomocą pętli while.

Jako, że kolejność elementów w zbiorze lub liście nie ma znaczenia, końcowo powinniśmy otrzymać poprawny język, który składa się ze wszystkich możliwych podsłów słowa P.

W algorytmie również uwzględniam, że każde słowo ma za swoje podsłowo słowo puste.

Wejście: P – zadane słowo

Wyjście: L – język składający się ze wszystkich podsłów słowa P

ALLSUBWORDS(P)

```
lenP := LENGTH(P)
pomL := [-]
L := [-]
L := MAKELIST([], L)      %tworzę język składający się z pustego słowa aby je uwzględnić%
if lenP == 0 then         %jeżeli na wejściu otrzymaliśmy słowo puste%
        return L          %zwracam język składający się ze słowa pustego%
for i=1 to lenP do        %iteruję przez każdą możliwą długość podsłów słowa P%
        pomL := SUBWORDS(P, i) %otrzymuje podsłowa o długości „i”%
        while pomL != [-] do
                słowo := HEAD(pomL)       %”wyciągam” każde podsłowo z wyniku SUBWORDS%
                L := MAKELIST(słowo, L)   %i dodaje je do końcowego języka L%
                pomL := TAIL(pomL)
return L
```