Z-Algorithmus

// $|S| \geq 2$
l:=r:=0; // l,r enthalten jeweils l_{i-1}, r_{i-1}
Z[2] := prefix(S, S[2...]).length();
if Z[2]>0 then
  l:=2; r:=2+Z[2]-1;
fi

for i=3 to $|S|$ do
  if i>r then //Fall I
    Z[i]:=prefix(S, S[i...]).length();
    if Z[i]>0 then l:=i; r:=i+Z[i]-1; fi
  else //Fall II
    k:=i-l+1;
    if Z[k]<r-i+1 then //$|\beta| = r - i + 1$
      Z[i]:=Z[k]
    else
      aux:=prefix(S[r-i+2...],S[r+1...]).length();
      Z[i]:=r-i+1+aux;
      l:=i; r:=i+Z[i]-1;
    fi
  fi
endfor

prefix(A,B) liefert das längste gemeinsame Prefix von A und B

2.5.2 Exaktes Matching mittels des Z-Algorithmus

- betrachten P$T mit $\$ \notin \mathcal{A}$ (Trennzeichen)
- berechne Z_i für diesen String (in $\Theta(|T|)$, da $|T| \geq |P|$)
  (beachte: $\forall i : Z_i \leq n$)
- für $i > (n+1)$ gilt:
  $Z_i = n$
  $\Leftrightarrow P\$T[i \ldots i+n-1]$ ist Prefix von P$T, $n = |P|$
  $\Leftrightarrow T[i-(n+1) \ldots i-2] = P$
  $\Leftrightarrow$ P kommt an Position i-(n+1) in T vor

Satz dies liefert exaktes Matching in $\Theta(|T|)$