Jeuring's Algorithm on Palindromes

Warning 1. The notes have not been proofread. Use at your own peril.

Question 2. *Given a string $(s_i)_{0 \le i < n}$, try to compute the longest palindrome in s with $O(n)$ time complexity.*

In fact, we will compute $(a_m)_{0 \le m \le 2(n-1)}$ where a_m is the maximal length of a palindrome $(s_i)_{l \le i \le r}$ such that $l + r = m$.

For sake of convenience, we set s_{-1} and s_n to be two distinct unused characters in $(s_i)_{0 \le i < n}$. The array $A[0...2(n-1)]$ will gradually compute $(a_m)_{0 \le m \le 2(n-1)}$. We design the scheme of the algorithm as

$m \leftarrow 0$
for i **from** 1 **to** n **do**
  if $s_i \ne s_{m-i}$ **then**
    $A[m] \leftarrow 2i - m - 1$
    Increase m to some new m' and compute $A[j]$ for all j such that $m \le j < m'$

where the loop invariant is that $i - 1 \le m \le 2(i-1)$, and $A[j] = a_j$ for all $0 \le j < m$, and that $(s_j)_{m-i+1 \le j \le i-1}$ is the longest palindrome which ends at $i - 1$. The algorithm terminates with $i = n + 1$ and we succeed in computing the array A.

The increment in question is implemented as follows:

for j **from** $m+1$ **to** ∞ **do**
  if $(s_k)_{j-i \le k \le i}$ is a palindrome **then**
    $m \leftarrow j$
    break
  else
    Compute $A[j]$

The trick is to determine whether $(s_k)_{j-i \le k \le i}$ is a palindrome and compute $A[j]$ in constant time. For the former, we note that (s_k) in question is a palindrome if and only if $A[2j - m]$, which computes a_{2j-m}, equals to $2i - j - 1$, and that $s_i = s_{j-i}$. For the later, we set $A[j] \leftarrow A[2j - m]$:

$u \leftarrow s_i$
for j **from** $m+1$ **to** ∞ **do**
  $t \leftarrow A[2j - m]$
  if $(t = 2i - j + 1)$ **and** $(u = s_{j-i})$ **then**
    $m \leftarrow j$
    break
  else
    $A[j] \leftarrow t$