Homework 9, Solution set

1. We compute the following partial sums: $\text{LSum}[j] = \sum_{i=1}^{j} |L_i|$ for $j = 0, 1, 2, \ldots n$, in linear time, which then allows us to compute any sum of the form $\sum_{i=h}^{j} |L_i|$ in $O(1)$ time. We will compute the minimum cost for merging lists $L_i, L_{i+1}, \ldots, L_k$ in $\text{MCost}[i, k]$. We initialize it to 0 for $i = k$, for $1 \le i \le n$, which is also its final value, and to maxint for $k > i$. We then compute the actual values of $\text{MCost}[i, k]$, for $k - i = 1, 2, \ldots, n-1$, in turn. It is convenient to define $\text{diff} = k - i$. Pseudo-code follows.

**for** $i = 1$ to $n$ **do**
    $\text{MCost}[i, i] \leftarrow 0$;
    **for** $k = i + 1$ to $n$ **do**
        $\text{MCost}[i, k] \leftarrow \text{maxint}$
    **end for**
**end for**
**for** $\text{diff} = 1$ to $n - 1$ **do**
    **for** $i = 1$ to $n - \text{diff}$ **do**
        $k \leftarrow i + \text{diff}$;
        **for** $j = i$ to $k - 1$ **do**
            $\text{MCost}[i, k] \leftarrow \min\{\text{MCost}[i, k],$
                        $\text{MCost}[i, j] + \text{MCost}[j + 1, k] + \text{LSum}[k] - \text{LSum}[i - 1]\}$
        **end for**
    **end for**
**end for**

2.a. Suppose $j \ge i$. Then to change $u$ to $v$, at a minimum one will need to add $j - i$ characters to $u$. Thus the edit cost is at least $j - i$ in this case. Similarly, if $j < i$, at a minimum one will need to remove $i - j$ characters, yielding an edit cost of at least $i - j$. In both cases, the edit cost is at least $|j - i|$.

b. We can conclude from (a) that there is no point to computing the edit distance when $|j - i| > k$. Instead we treat all such edit distances as being too large, which we will represent by the value $k + 1$. It will be helpful to define the following function $\text{Incr}(h)$:

$$\text{Incr}(h) = \begin{cases} h + 1 & h \le k \\ k + 1 & h = k + 1 \end{cases}$$

This leads to the following modification of the previous MinEdit function. We initialize the values on the diagonals distance $k+1$ from the true diagonal to $k+1$, namely the entries for $\text{MinEd}(i, i+k+1)$, with $1 \le i \le n-k-1$, and for $\text{MinEd}(j+k+1, j)$, with $1 \le j \le n-k-1$.

$$\text{MinEd}(i, j) = \begin{cases} k + 1 & i = j + k + 1 \text{ or } j = i + k + 1 \\ |j - i| & |j - i| \le k \text{ and } (i = 0 \text{ or } j = 0) \\ \text{MinEd}(i - 1, j - 1) & u_i = v_j,\ |j - i| \le k,\ i, j \ge 1 \\ \text{Incr}\left( \min \begin{cases} \text{MinEd}(i - 1, j - 1) \\ \text{MinEd}(i, j - 1) \\ \text{MinEd}(i - 1, j) \end{cases} \right) & u_i \ne v_j, |j - i| \le k, i, j \ge 1 \end{cases}$$

The number of possible values for $i$ is $n + 1$ (ranging from 0 to $n$), and as $|j - i| \le k$, the number of possible values for $j$ for each value of $i$ is at most $2k + 1$ (ranging from $i - k$

to $i + k$). Thus the number of recursive problems is at most $(n+1) \cdot (2k+1) = O(nk)$. Each recursive problem requires $O(1)$ time for its non-recursive work leading to an overall runtime of $O(nk)$.

3. We compute shortest paths from $v$ to $s$ in $G^R$, the reversal of $G$, and keep track of the number of these paths, with the following recursive calculation.

$$\text{Shtst}(v) \leftarrow \begin{cases} \min_{\{(v,w)\in E^R\}}\{\text{Shtst}(v), \text{length}(v,w) + \text{Shtst}(w)\} & v \neq s \\ 0 & v = s \end{cases}$$

$$\text{NumPth}(v) \leftarrow \begin{cases} \sum_{\{w \mid \text{Shtst}(v)=\text{length}(v,w)+\text{Shtst}(w)\}} \text{NumPth}(w) & v \neq s \\ 1 & v = s \end{cases}$$

We will initialize Shtst$[v]$ to maxint for all $v \neq s$, and Shtst$[s]$ to 0. Also, we initialize NumPth$[v]$ to 0 for all $v \neq s$, and NumPth$[s]$ to 1.

Pseudo-code for the DFS follows.

DFSNmPth($v$)<br>
Done[$v$] $\leftarrow$ True;<br>
**for** each edge $(v, w)$ **do**<br>
    **if** not Done($w$) **then** DFSNmPth($w$)<br>
    **end if**<br>
    **if** Shtst$[v] >$ length$(v, w) +$ Shtst$[w]$ **then**<br>
        Shtst$[v] \leftarrow$ length$(v, w) +$ Shtst$[w]$; NumPth$[v] \leftarrow$ NumPth$[w]$<br>
    **else**<br>
        **if** Shtst$[v] =$ length$(v, w) +$ Shtst$[w]$ **then**<br>
            NumPth$[v] \leftarrow$ NumPth$[v] +$ NumPth$[w]$<br>
        **end if**<br>
    **end if**<br>
**end for**

As this code just adds constant time work to each recursive call, the running time of the DFS remains at $O(|V| + |E|)$.

4. We run DFS on $T$ starting at its root to compute a pre-order and a post-order numbering. Let pre($v$) and post($v$), respectively, be the pre- and post-order numbers assigned to vertex $v$. Then, if $u$ is an ancestor of $v$, we will have pre$(u) <$ pre$(v)$ and post$(u) >$ post$(v)$, because the search at $u$ will start before the search at $v$ and will end after. The symmetric result holds if $v$ is an ancestor of $u$.

However, if $u$ and $v$ are unrelated, one search will start and finish before the other one ends. Suppose the search at $u$ is the first to start. Then pre$(u) <$ pre$(v)$ and post$(u) <$ post$(v)$.

The pre and post values are computed in $O(|V| + |E|)$ time by the DFS. With these values in hand, one can determine in $O(1)$ time whether $u$ and $v$ are related or not, and if related, which one is the ancestor.