Problem 2
Given a \([0, n - 1] \times [0, n - 1]\) grid \(G\) and a set \(P\) of \(n\) points \((i, S[i])\) with weight \(w[i]\) for \(0 \leq i < n\). For a pair of points \((x_0, y_0)\) and \((x_1, y_1)\) with \(x_0 \leq x_1\) and \(y_0 \leq y_1\) we define the top-\(k\) range query:

- A top-\(k\)-range report query asks for the \(k\) points \((x, y)\) \(\in P\) such that \(x \in [x_0, x_1]\) and \(y \in [y_0, y_1]\) with maximum weight sorted in decreasing order of weight.

Results [5]
Top-\(k\) range queries can be answered in \(O(\log^2 n + k \cdot \log n)\) time using a index of size \(3n \log n + o(n \log n) + |weights|\) bits, where \(|weights|\) is the space required to store the weights associated with the \(n\) points.
Top-$k$ Range Report Queries

Outline of solution

- Use the range count algorithm to get the $O(\log n)$ WT nodes $C$ which cover the $y$ range
- For each $x$ range in $C$ use the RMQ structure to navigate to the haviest point ($O(\log n)$ time per range, i.e $O(\log^2 n)$ total)
- Insert the haviest points into a max priority queue $Q$
- Remove maximum point from $Q$, report it and split its corresponding $x$ range in $C$. Navigate to the haviest points in the two new ranges $Q$. Insert
- Repeat last step until $Q$ is empty or $k$ points were reported

Total time: $O(\log^2 n + k \cdot \log n)$
Top-$k$ Range Report Queries

$x$ ranges

$y$ range

RMQ on associated weights
Top-$k$ Range Report Queries

$x$ ranges

$RMQ$ on associated weights

$y$ range
Top-$k$ Range Report Queries

$x$ ranges

RMQ on associated weights

$y$ range
The suffix tree (ST) extends the functionality of suffix array construction in three phases (each takes linear time):

- suffix array construction (⇒ Christmas Lecture)
- LCP array construction
- tree topology construction

As the pointer-based representation takes too much space in most application we will present a more space-efficient version: The compressed suffix tree (CST).
Versatile index structure: E.g. we can solve longest common substring queries for $k$ strings efficiently.

Exercise: Find longest palindromic substring of a string.
Most representations consist of three parts:
- Suffix Array (leaves of suffix tree)
- LCP Array (longest common prefix lengths/depth of inner nodes)
- Tree Topology
Definition

Let $lcp(U, V)$ denote the longest common prefix between two strings $U$ and $V$. For a text $T$ of size $n$ the longest common prefix (LCP) array of size $n + 1$ is defined as follows. $LCP[i] = |lcp(SA[i], SA[i - 1])|$ for $i \geq 1$ and $LCP[0] = 0$ and $LCP[n] = -1$.

([4] introduced this array as $Hgt$ array)
LCP Array – Example

<table>
<thead>
<tr>
<th>$i$</th>
<th>SA</th>
<th>$LCP$</th>
<th>$T[SA[i], n - 1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11</td>
<td>0</td>
<td>$$</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>0</td>
<td>i$</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>1</td>
<td>ippi$</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>issippi$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>ississippi$</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>mississippi$</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>0</td>
<td>pi$</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>1</td>
<td>ppi$</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>0</td>
<td>sippi$</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>2</td>
<td>sissippi$</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>1</td>
<td>sippi$</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>3</td>
<td>ssissippi$</td>
</tr>
</tbody>
</table>

- Time complexity of naive computation (for each $i > 0$, compare suffix $SA[i]$ and $SA[i - 1]$): $O(n^2)$.
- Comparison in SA-order.
Linear Time Calculation of LCP Array

Idea of [3]: Processing in text-order.

\[ m | i | s | s | i | s | s | i | p | p | i | $ \]

\[ i = 0 \]

\[ SA[ISA[i] - 1] = 1 \]
Idea of [3]: Processing in text-order.

\[
\begin{align*}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\text{m} & \text{i} & \text{s} & \text{s} & \text{i} & \text{s} & \text{s} & \text{i} & \text{p} & \text{p} & \text{i} & \$
\end{align*}
\]

\(i = 1\)

\[
\text{SA} [\text{ISA}[i] - 1] = 4
\]

\[
0 \quad 4
\]
Linear Time Calculation of LCP Array

Idea of [3]: Processing in text-order.

\[
\begin{align*}
0 & \quad 1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 & \quad 7 & \quad 8 & \quad 9 & \quad 10 & \quad 11 \\
\text{m} & \quad i & \quad s & \quad s & \quad i & \quad s & \quad s & \quad i & \quad p & \quad p & \quad i & \quad $ \\
\end{align*}
\]

\[
SA[ISA[i] - 1] = 5
\]

\[
i = 2
\]

\[
0 \quad 4 \quad 3
\]
Linear Time Calculation of LCP Array

Idea of [3]: Processing in text-order.

\[
0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11
\]

\[
\text{mississippi}
\]

\[
i = 3
\]

\[
\text{SA[ISA}[i] - 1] = 6
\]

\[
0 \quad 4 \quad 3 \quad 2
\]
Linear Time Calculation of LCP Array

Idea of [3]: Processing in text-order.

\[
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\text{m} & \text{i} & \text{s} & \text{s} & \text{i} & \text{s} & \text{s} & \text{i} & \text{p} & \text{p} & \text{i} & \$ \\
\end{array}
\]

\[i = 4\]

\[SA[ISA[i] - 1] = 7\]

\[
\begin{array}{cccccc}
0 & 4 & 3 & 2 & 1 \\
\end{array}
\]
Idea of [3]: Processing in text-order.

\[ m \quad i \quad s \quad s \quad i \quad s \quad s \quad i \quad p \quad p \quad i \quad $ \]

\[ SA[ISA[i] - 1] = 3 \]

\[ 0 \quad 4 \quad 3 \quad 2 \quad 1 \quad 1 \]
Linear Time Calculation of LCP Array

Idea of [3]: Processing in text-order.

0 1 2 3 4 5 6 7 8 9 10 11
mississippi

\[ i = 6 \]

\[ SA[ISA[i] - 1] = 8 \]

0 4 3 2 1 1 0
Linear Time Calculation of LCP Array

Idea of [3]: Processing in text-order.

\[
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\text{m} & \text{i} & \text{s} & \text{s} & \text{i} & \text{s} & \text{s} & \text{i} & \text{p} & \text{p} & \text{i} & \$
\end{array}
\]

\[
i = 7
\]

\[
\text{SA} / \text{ISA}[i] - 1 = 10
\]

\[
\begin{array}{cccccccccccc}
0 & 4 & 3 & 2 & 1 & 1 & 0 & 1
\end{array}
\]
Idea of [3]: Processing in text-order.

\[ p_i = 8 \]

\[ SA[ISA[i] - 1] = 9 \]
Idea of [3]: Processing in text-order.

SA[ISA[i] - 1] = 0

i = 9
Idea of [3]: Processing in text-order.

\[ i = 10 \]

\[ \text{SA}[\text{ISA}[i] - 1] = 11 \]
Idea of [3]: Processing in text-order.

\[
0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11
\]

\[
\text{mississippi}\$
\]

\[
\text{SA}[/\text{SA}[i] - 1] = 5
\]

\[
0 \quad 4 \quad 3 \quad 2 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0
\]
Lemma ([3])
For $0 < i \leq n$, we have $LCP[ISA[i]] \geq LCP[ISA[i - 1]] - 1$. 
Linear Time Calculation of LCP Array

00  \( LCP[0] \leftarrow 0 \)
01  \( LCP[n] \leftarrow -1 \)
02  \textbf{for} \( i \leftarrow 0 \) \textbf{to} \( n - 1 \) \textbf{do}
03     \( ISA[SA[i]] \leftarrow i \)
04  \( \ell \leftarrow 0 \)
05  \textbf{for} \( i \leftarrow 0 \) \textbf{to} \( n - 1 \) \textbf{do}
05    \( j \leftarrow SA[(ISA[i] - 1) \mod n] \)
07  \textbf{while} \( T[i + \ell] = T[j + \ell] \) \textbf{do}
08     \( \ell \leftarrow \ell + 1 \)
09  \( LCP[ISA[i]] \leftarrow \ell \)
10  \( \ell \leftarrow \max(0, \ell - 1) \)

Exercise
How much memory is required during the algorithms execution?
Linear Time Calculation of LCP Array

Engineered version of [2]:

```plaintext
for i ← 0 to n − 1 do
    Φ[SA[i]] ← SA[(i − 1) mod n]
    ℓ ← 0
for i ← 0 to n − 1 do
    j ← Φ[i]
while T[i + ℓ] = T[j + ℓ] do
    ℓ ← ℓ + 1
    PLCP[i] ← ℓ
    ℓ ← max(0, ℓ − 1)
for i ← 0 to n − 1 do
    LCP[i] ← PLCP[SA[i]]
LCP[n] ← −1
```

(Explain why this algorithm is faster in practice)
[6] proposed to use a CSA and the permuted LCP array (LCP values in text-order) PLCP. We know

- $PLCP[i + 1] \geq PLCP[i] - 1$
- $PCLP[i] \leq n - 1 - i$ (for $0 \leq i < n$)

\[
\begin{align*}
  i &= 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11 \\
  PCLP[i] &= 0 \quad 4 \quad 3 \quad 2 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0
\end{align*}
\]
[6] proposed to use a CSA and the permuted LCP array (LCP values in text-order) PLCP. We know

- $PLCP[i + 1] \geq PLCP[i] - 1$
- $PCLP[i] \leq n - 1 - i$ (for $0 \leq i < n$)

\[
\begin{align*}
i &= 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 \\
PCLP[i] &= 0, 4, 3, 2, 1, 1, 0, 1, 1, 0, 0, 0 \\
PCLP[i] + i &= 0, 5, 5, 5, 5, 6, 6, 8, 9, 9, 10, 11
\end{align*}
\]
Space-Efficient LCP Representation

[6] proposed to use a CSA and the permuted LCP array (LCP values in text-order) PLCP. We know

- \( PLCP[i + 1] \geq PLCP[i] - 1 \)
- \( PCLP[i] \leq n - 1 - i \) (for \( 0 \leq i < n \))

\[
i = 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11
\]

\[
PCLP[i] = 0 \quad 4 \quad 3 \quad 2 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0
\]

\[
PCLP[i] + i = 0 \quad 5 \quad 5 \quad 5 \quad 5 \quad 6 \quad 6 \quad 8 \quad 9 \quad 9 \quad 10 \quad 11
\]

- Encode gaps of \( PLCP[i] + i \) with unary code (results in bitvector \( H \) of length \( 2n \))
- In this example: \( H = 10000011110110010110101 \)
- What additional structure is required to calculated \( LCP[i] \)?
With a $o(n)$-space select structure (arguments starting from 1) and a CSA we get:

00  \textbf{access\_lcp}(i) \\
01  \quad x \leftarrow \text{SA}[i] \\
02  \quad \text{return } \text{select}(x + 1, 1, H) + 1 - 2x

Summary:
- Time complexity depends on CSA access
- Space: $2n + o(n)$ bits (for bitvector $H + \text{select}$) + $|\text{CSA}|$

Note: The LCP between arbitrary suffixes can be calculated in constant time using a RMQ structure.
Definition of an LCP-interval ([1])

An interval $[i, j]$, where $0 \leq i \leq n - 1$ is called LCP-interval of LCP value $\ell$ (denoted by $\ell - [i, j]$) if

- $LCP[i] < \ell$ or $i = 0$
- $LCP[k] \geq \ell$ for all $k \in [i + 1, j]$
- $LCP[k] = \ell$ for at least one $k \in [i + 1, j]$
- $LCP[j + 1] < \ell$

Every index $k$ with $i < k \leq j$ and $LCP[k] = \ell$ is called $\ell$-index. There are at most $\sigma - 1$ $\ell$-indices in an LCP-interval.

Note: Each LCP-interval corresponds to a node in the suffix tree.
The LCP-Interval Tree – Example

<table>
<thead>
<tr>
<th>$i$</th>
<th>SA</th>
<th>$LCP$</th>
<th>$\mathcal{T}[SA[i], n - 1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11</td>
<td>0</td>
<td>$$$</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>0</td>
<td>i$</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>1</td>
<td>ippi$</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
<td>issippi$</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>ississippi$</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>mississippi$</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>0</td>
<td>pi$</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>1</td>
<td>ppi$</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>0</td>
<td>sippi$</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>2</td>
<td>sissippi$</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>1</td>
<td>ssippi$</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>3</td>
<td>ssissippi$</td>
</tr>
</tbody>
</table>

Singleton intervals $\ell - [i, i]$ are omitted


