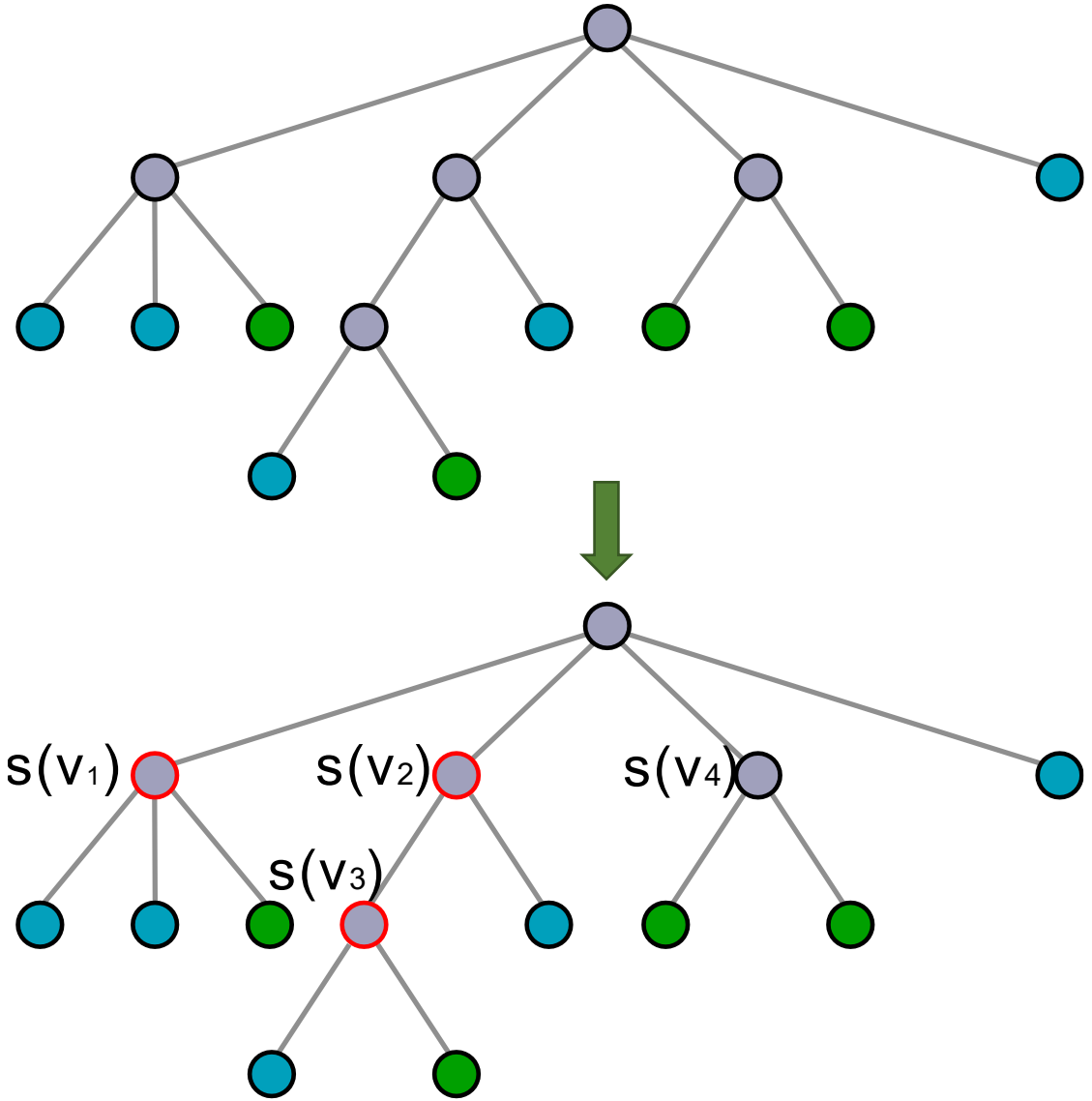
10. Übung

Einführung in die Bioinformatik I, 1. Teil
Wintersemester 2019/2020

Aufgabe 1 (5 Punkte): Geben Sie einen möglichst schnellen Algorithmus an, der mit Hilfe eines Suffixbaumes für zwei Strings S_1, S_2 alle gemeinsamen Teilstrings berechnet, die länger als eine gegebene Länge l sind. Begründen Sie Korrektheit und Laufzeit des Algorithmus.

Aufgabe 1 (5 Punkte): Geben Sie einen möglichst schnellen Algorithmus an, der mit Hilfe eines Suffixbaumes für zwei Strings S_1, S_2 alle gemeinsamen Teilstrings berechnet, die länger als eine gegebene Länge l sind. Begründen Sie Korrektheit und Laufzeit des Algorithmus.

Bsp.: S_1 S_2



Algorithmus und Laufzeit:

1. Erzeuge generalisierten Suffixbaum für S_1 und S_2, also einen Suffixbaum B für $S_1\$_1S_2\$_2$.
   → Laufzeit $O(n)$

2. Berechne für jeden Knoten v die Stringtiefe $s(v)$.
   → Laufzeit $O(n)$

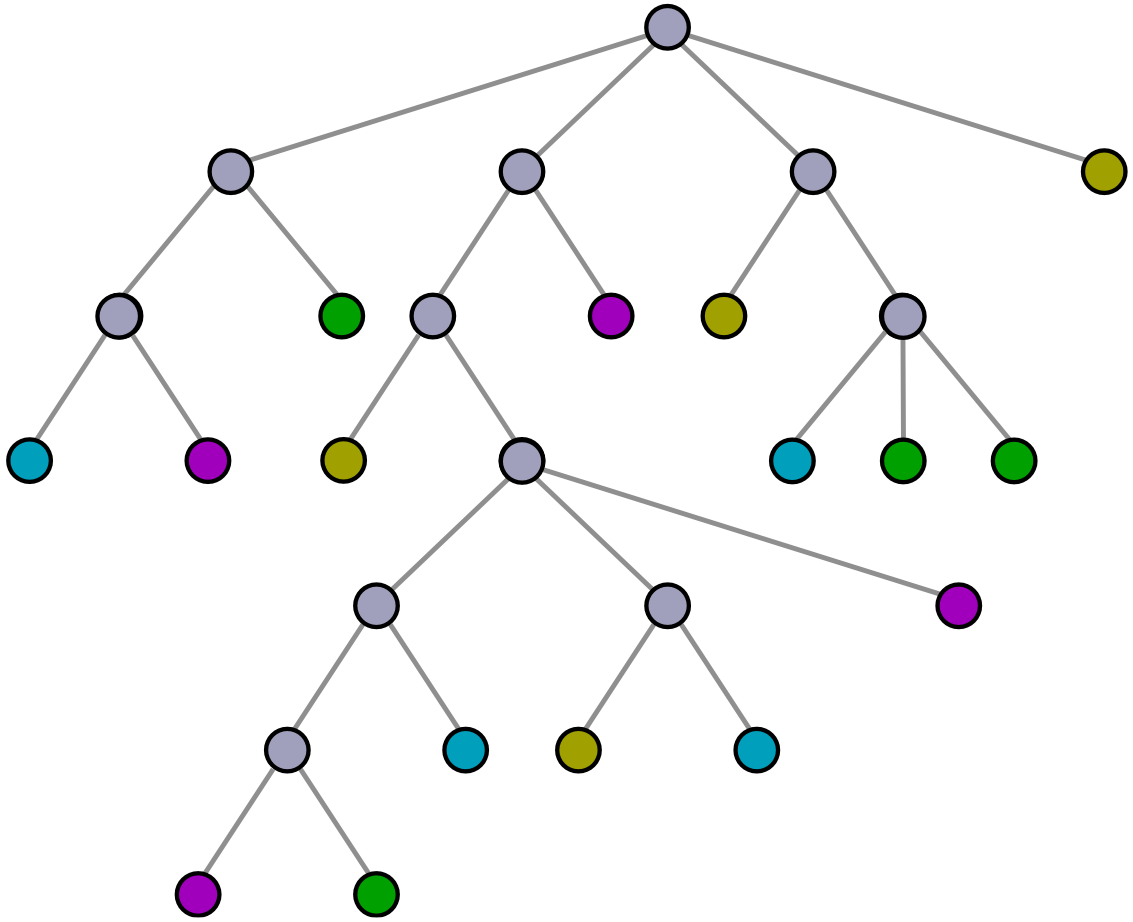
3. Markiere dabei innere Knoten, die für Teilstrings aus beiden Strings stehen.
   → Laufzeit $O(n)$

4. Betrachte alle markierten Knoten und gib jeden mit einer Stringtiefe $\geq l$ aus.
   → Laufzeit $O(n)$

Aufgabe 2 (5 Punkte): Geben Sie einen Algorithmus an, der für k Strings $S_1, S_2, \ldots S_k$ (Gesamtlänge n) und alle $q \in \{1, 2, \ldots k\}$ in $O(kn)$ Zeit die Länge des längsten Teilstrings berechnet, der in mindestens q der eingegebenen Strings vorkommt. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Aufgabe 2 (5 Punkte): Geben Sie einen Algorithmus an, der für k Strings $S_1, S_2, \ldots S_k$ (Gesamtlänge n) und alle $q \in \{1, 2, \ldots k\}$ in $O(kn)$ Zeit die Länge des längsten Teilstrings berechnet, der in mindestens q der eingegebenen Strings vorkommt. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Bsp.: S_1 S_2 S_3 S_4



<u>Algorithmus und Laufzeit:</u>

1. Erzeuge generalisierten Suffixbaum für k Strings.
   → Laufzeit *O(n)*

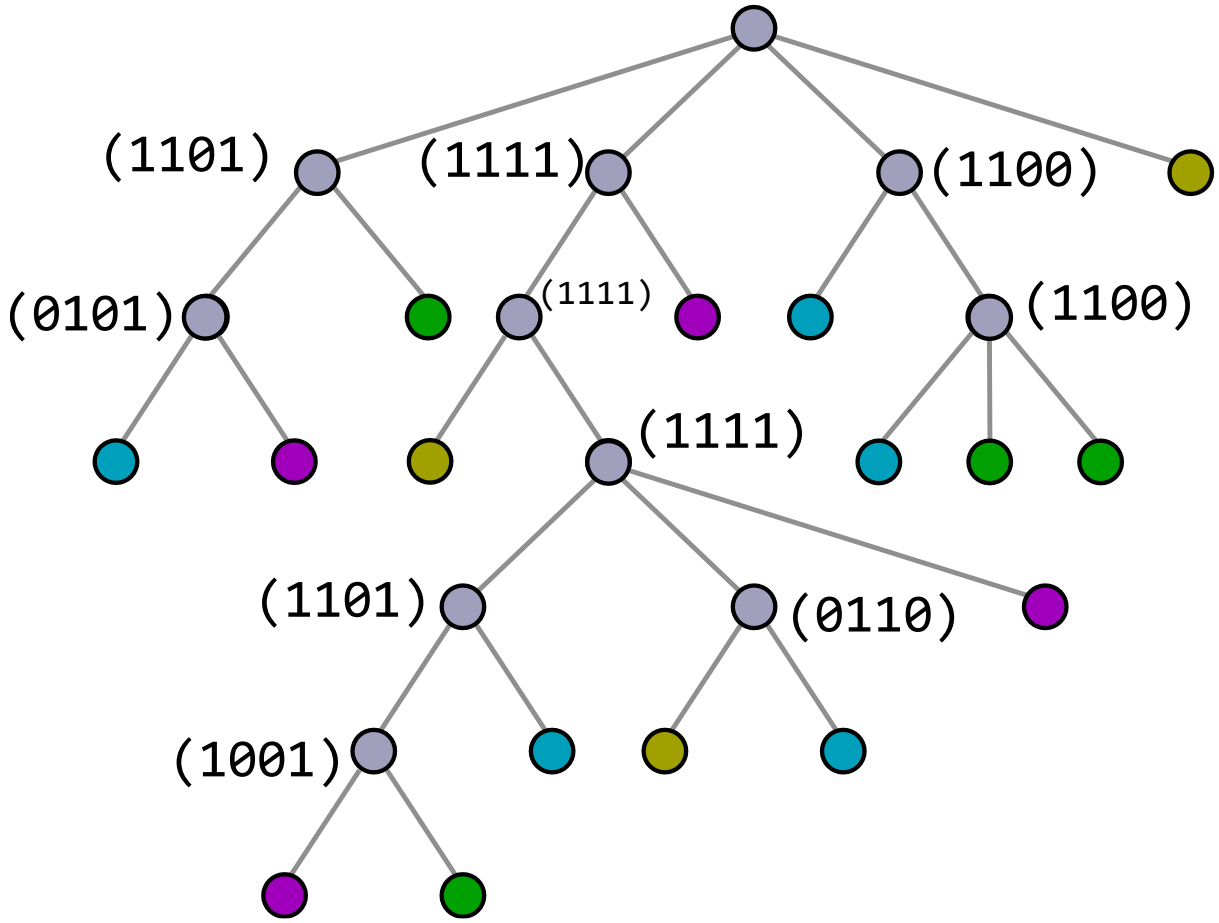
2. Bestimme für jeden inneren Knoten die Menge an Strings, deren Teilstrings an diesem Knoten endet (Bitvektor der Länge k), und markiere jeden Knoten mit der Anzahl dieser Strings.
   → Laufzeit *O(k*n)*

3. Berechne für jeden Knoten v die Stringtiefe *s(v)*.
   → Laufzeit *O(n)*

4. Durchsuche für jedes q den Teilbaum, der durch Knoten mit Markierungen ≥ q induziert wird, nach dem Knoten mit größter Stringtiefe und gib diese Stringtiefe für das entsprechende q aus.
   → Laufzeit *O(k*n)*

Aufgabe 2 (5 Punkte): Geben Sie einen Algorithmus an, der für k Strings $S_1, S_2, \ldots S_k$ (Gesamtlänge n) und alle $q \in \{1, 2, \ldots k\}$ in $O(kn)$ Zeit die Länge des längsten Teilstrings berechnet, der in mindestens q der eingegebenen Strings vorkommt. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Bsp.: S_1 S_2 S_3 S_4



<u>Algorithmus und Laufzeit:</u>

1. Erzeuge generalisierten Suffixbaum für k Strings.
   → Laufzeit $O(n)$

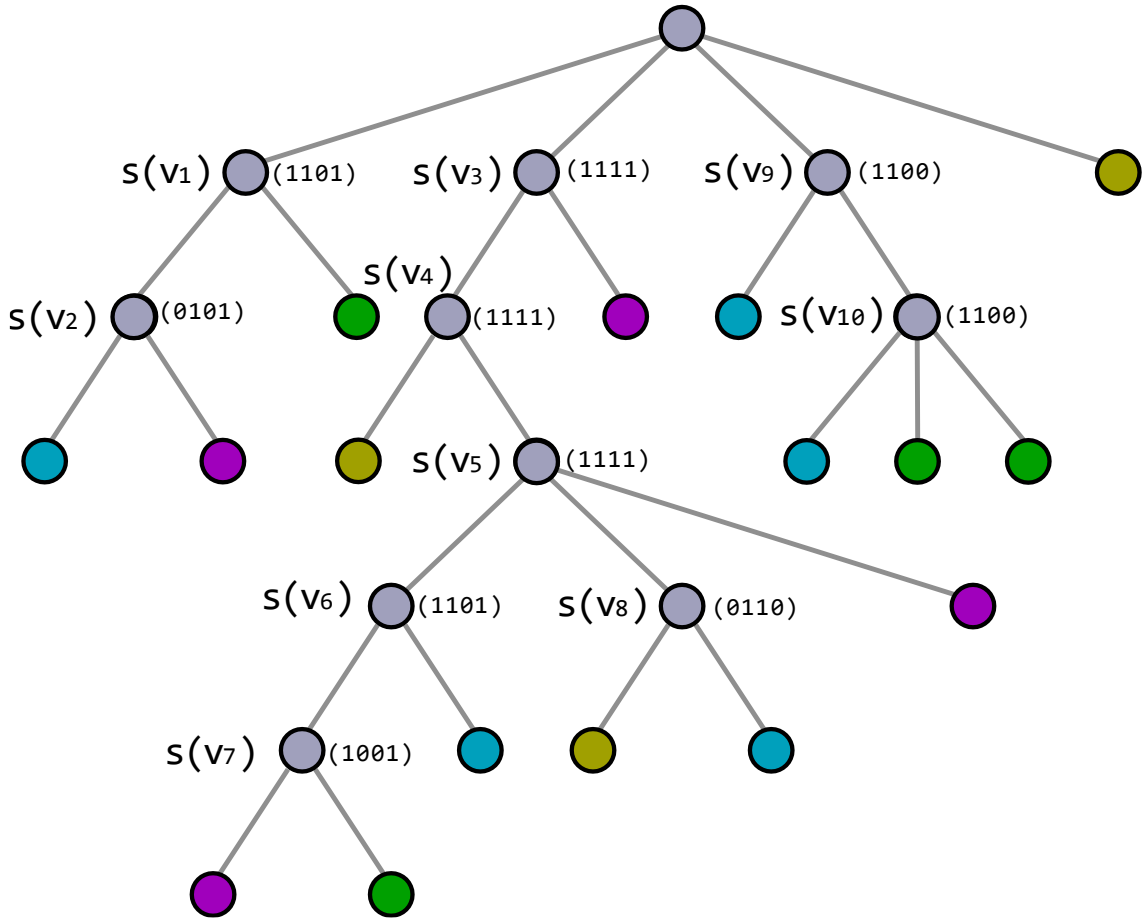
2. Bestimme für jeden inneren Knoten die Menge an Strings, deren Teilstrings an diesem Knoten endet (Bitvektor der Länge k), und markiere jeden Knoten mit der Anzahl dieser Strings.
   → Laufzeit $O(k*n)$

3. Berechne für jeden Knoten v die Stringtiefe $s(v)$.
   → Laufzeit $O(n)$

4. Durchsuche für jedes q den Teilbaum, der durch Knoten mit Markierungen $\geq q$ induziert wird, nach dem Knoten mit größter Stringtiefe und gib diese Stringtiefe für das entsprechende q aus.
   → Laufzeit $O(k*n)$

Aufgabe 2 (5 Punkte): Geben Sie einen Algorithmus an, der für k Strings $S_1, S_2, \ldots S_k$ (Gesamtlänge n) und alle $q \in \{1, 2, \ldots k\}$ in $O(kn)$ Zeit die Länge des längsten Teilstrings berechnet, der in mindestens q der eingegebenen Strings vorkommt. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Bsp.: S_1 S_2 S_3 S_4



Algorithmus und Laufzeit:

1. Erzeuge generalisierten Suffixbaum für k Strings.
   → Laufzeit $O(n)$

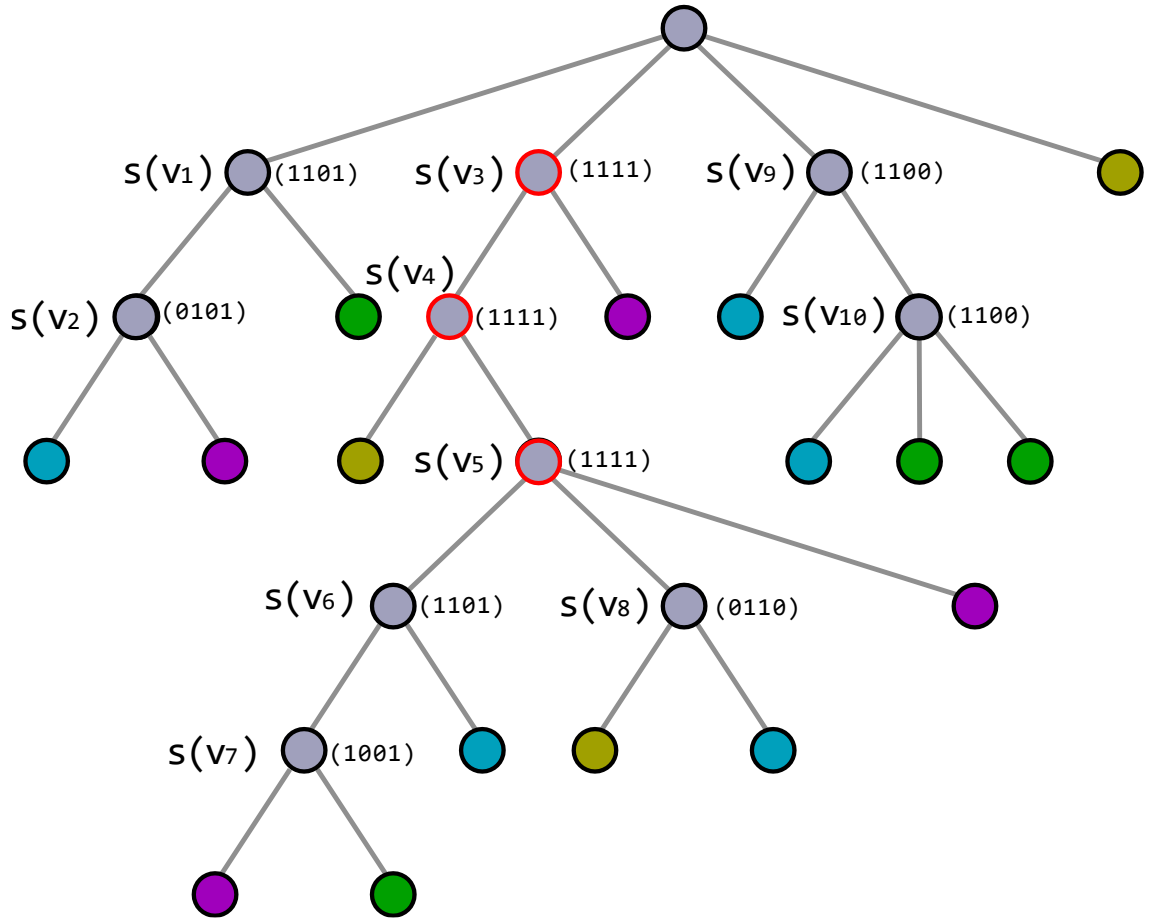
2. Bestimme für jeden inneren Knoten die Menge an Strings, deren Teilstrings an diesem Knoten endet (Bitvektor der Länge k), und markiere jeden Knoten mit der Anzahl dieser Strings.
   → Laufzeit $O(k*n)$

3. Berechne für jeden Knoten v die Stringtiefe $s(v)$.
   → Laufzeit $O(n)$

4. Durchsuche für jedes q den Teilbaum, der durch Knoten mit Markierungen $\geq q$ induziert wird, nach dem Knoten mit größter Stringtiefe und gib diese Stringtiefe für das entsprechende q aus.
   → Laufzeit $O(k*n)$

Aufgabe 2 (5 Punkte): Geben Sie einen Algorithmus an, der für k Strings $S_1, S_2, \ldots S_k$ (Gesamtlänge n) und alle $q \in \{1, 2, \ldots k\}$ in $O(kn)$ Zeit die Länge des längsten Teilstrings berechnet, der in mindestens q der eingegebenen Strings vorkommt. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Bsp.: S_1 S_2 S_3 S_4



<u>Algorithmus und Laufzeit:</u>

1. Erzeuge generalisierten Suffixbaum für k Strings.
   → Laufzeit $O(n)$

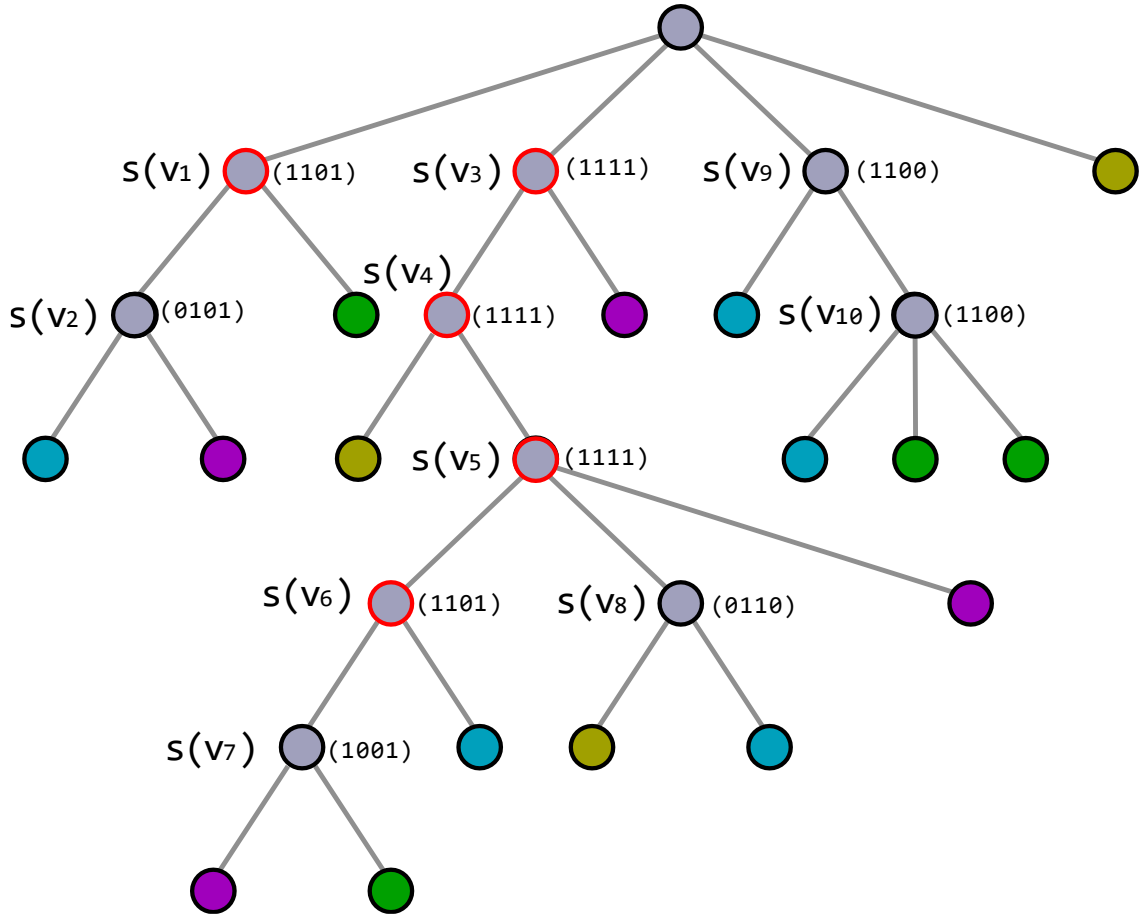
2. Bestimme für jeden inneren Knoten die Menge an Strings, deren Teilstrings an diesem Knoten endet (Bitvektor der Länge k), und markiere jeden Knoten mit der Anzahl dieser Strings.
   → Laufzeit $O(k*n)$

3. Berechne für jeden Knoten v die Stringtiefe $s(v)$.
   → Laufzeit $O(n)$

4. Durchsuche für jedes q den Teilbaum, der durch Knoten mit Markierungen $\geq q$ induziert wird, nach dem Knoten mit größter Stringtiefe und gib diese Stringtiefe für das entsprechende q aus.
   → Laufzeit $O(k*n)$

Aufgabe 2 (5 Punkte): Geben Sie einen Algorithmus an, der für k Strings $S_1, S_2, \ldots S_k$ (Gesamtlänge n) und alle $q \in \{1, 2, \ldots k\}$ in $O(kn)$ Zeit die Länge des längsten Teilstrings berechnet, der in mindestens q der eingegebenen Strings vorkommt. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Bsp.: S_1 S_2 S_3 S_4



Algorithmus und Laufzeit:

1. Erzeuge generalisierten Suffixbaum für k Strings.
   → Laufzeit *O(n)*

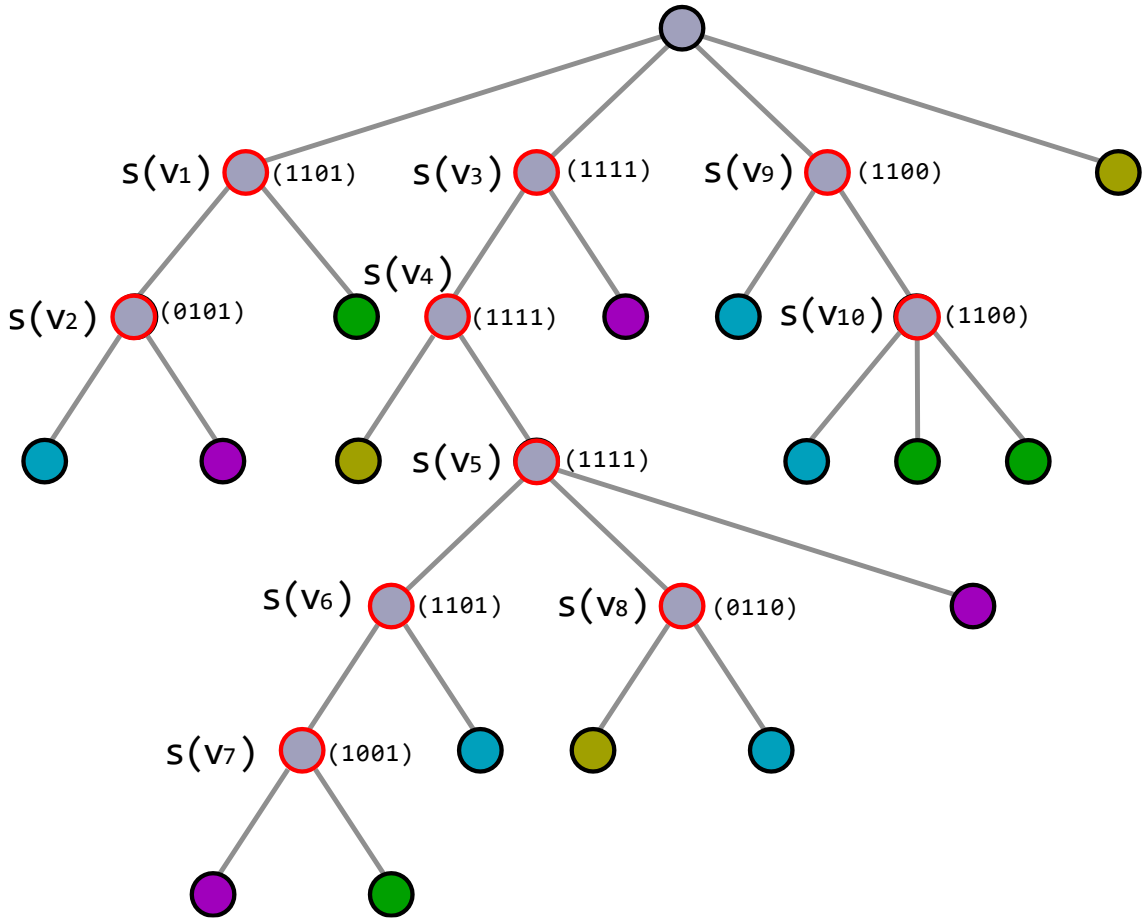
2. Bestimme für jeden inneren Knoten die Menge an Strings, deren Teilstrings an diesem Knoten endet (Bitvektor der Länge k), und markiere jeden Knoten mit der Anzahl dieser Strings.
   → Laufzeit *O(k*n)*

3. Berechne für jeden Knoten v die Stringtiefe *s(v)*.
   → Laufzeit *O(n)*

4. Durchsuche für jedes q den Teilbaum, der durch Knoten mit Markierungen ≥ q induziert wird, nach dem Knoten mit größter Stringtiefe und gib diese Stringtiefe für das entsprechende q aus.
   → Laufzeit *O(k*n)*

Aufgabe 2 (5 Punkte): Geben Sie einen Algorithmus an, der für k Strings $S_1, S_2, \ldots S_k$ (Gesamtlänge n) und alle $q \in \{1, 2, \ldots k\}$ in $O(kn)$ Zeit die Länge des längsten Teilstrings berechnet, der in mindestens q der eingegebenen Strings vorkommt. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Bsp.: S_1 S_2 S_3 S_4



<u>Algorithmus und Laufzeit:</u>

1. Erzeuge generalisierten Suffixbaum für k Strings.
   → Laufzeit *O(n)*

2. Bestimme für jeden inneren Knoten die Menge an Strings, deren Teilstrings an diesem Knoten endet (Bitvektor der Länge k), und markiere jeden Knoten mit der Anzahl dieser Strings.
   → Laufzeit *O(k*n)*

3. Berechne für jeden Knoten v die Stringtiefe *s(v)*.
   → Laufzeit *O(n)*

4. Durchsuche für jedes q den Teilbaum, der durch Knoten mit Markierungen ≥ q induziert wird, nach dem Knoten mit größter Stringtiefe und gib diese Stringtiefe für das entsprechende q aus.
   → Laufzeit *O(k*n)*

Aufgabe 3 (4 Punkte): Beweisen Sie mit Hilfe der Definition von Θ, dass $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$, mit $f(n) \geq 0$ und $g(n) \geq 0$.

Aufgabe 3 (4 Punkte): Beweisen Sie mit Hilfe der Definition von Θ, dass
$\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$, mit $f(n) \geq 0$ und $g(n) \geq 0$.

Informal: Das Maximum des Wachstums zweier Funktionen *f(n)* und *g(n)*, ist gleich dem Maximum der Summe beider Funktionen.

Dies ist korrekt, da wir nach der Landau-Notation für jeden Funktionsterm jeweils immer nur denjenigen mit der größten Potenz betrachten, da dieser allein bestimmend für das Wachstum der Funktion ist. Hier haben wir nur einen Funktionsterm n.
Wachsen die beiden Funktionen f und g gleich schnell, dann kann durch Summieren keine größere Potenz des Funktionsterms n entstehen.
Wächst eine von beiden Funktionen langsamer als die andere, so kann auch hier durch Summieren keine größere Potenz des Funktionsterms entstehen.

Formal:

θ-Definition nach Landau

$$f_1(n) \in \theta\big(f_2(n)\big) \underset{def}{\Leftrightarrow} \exists c_1, c_2 \, \epsilon \mathbb{R} + \exists n_0 \in \mathbb{N} + \forall n \geq n_0 : 0 \leq c_1 \cdot |f_2(n)| \leq |f_1(n)| \leq C_2 \cdot |f_2(n)|$$

Sei $f(n) \geq 0$ und $f(n) \geq 0$.

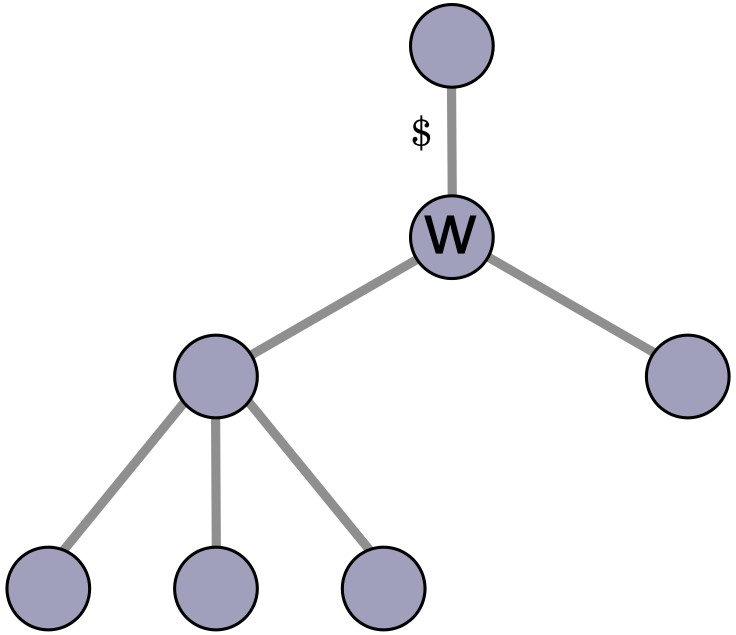
Zu zeigen ist obige Gleichung für $f_1(n) = max\{f(n), g(n)\}$ und $f_2(n) = f(n) + g(n)$.
Wenn wir $c_1 = \frac{1}{2}$, $c_2 = 1$, $n_0 = 1$ setzen, dann bekommen wir

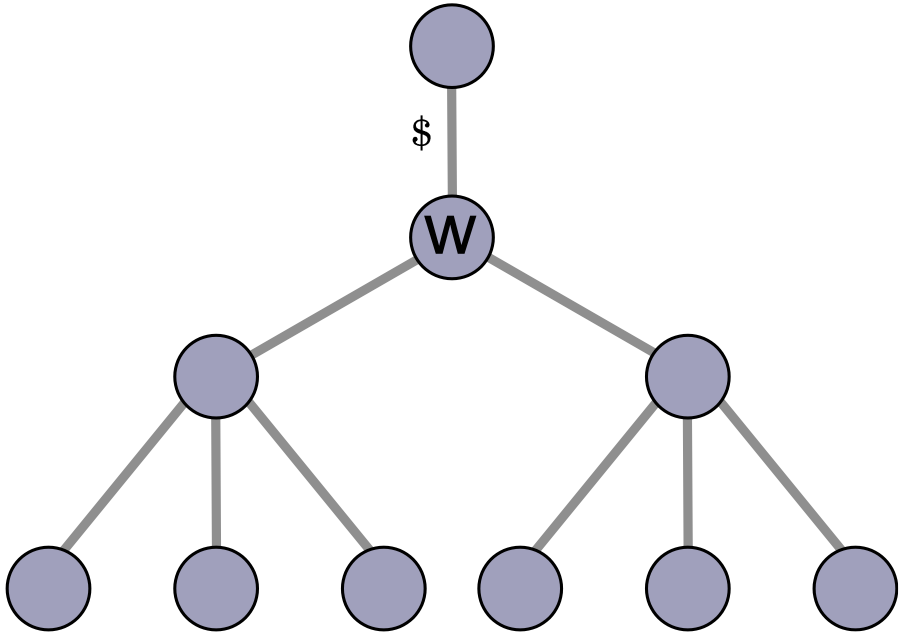
$$\forall n \geq 1: \quad 0 \leq \frac{1}{2} \cdot \big(f(n) + g(n)\big) \leq \max\{f(n), g(n)\} \leq f(n) + g(n)$$

daraus folgt:

$$\max\{f(n), g(n)\} = \theta\big(f(n) + g(n)\big)$$

Findet für die gegebenen Baumtopologien jeweils einen String S, dessen Suffixbaumtopologie entsprechend gleich aussieht. Wenn es solch einen String S nicht geben kann, begründet dies mit Hilfe eures Wissens über Suffixbäume.





Wie muss ein String S allgemein aufgebaut sein, damit die gegebene Suffixbaumtopologie entsteht, wenn $m, n \geq 3$ und:

1. m = n
2. m ≠ n

Inwiefern sind m und n von k abhängig? Gibt es Baumtopologien, die ein Suffixbaum nicht annehmen kann? Wenn ja welche und warum?