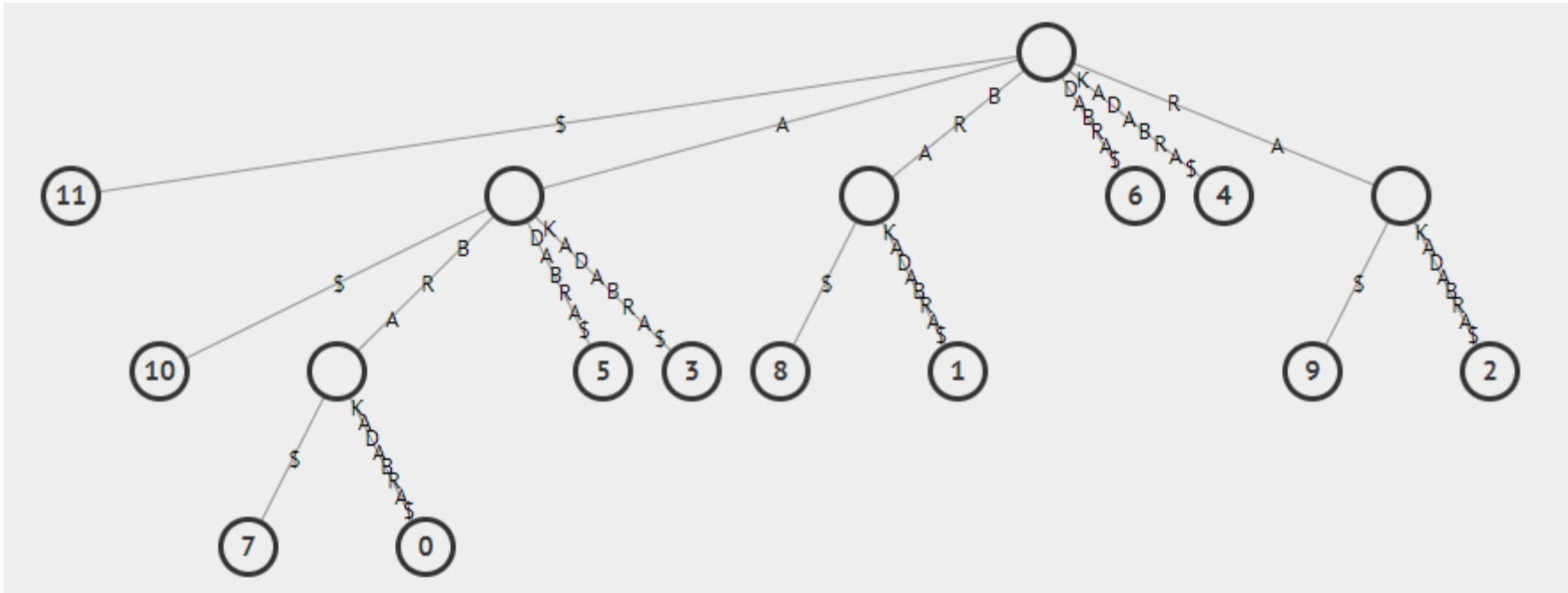
8. Übung

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

Aufgabe 1 (5 Punkte): Bestimmen Sie für die Strings `abrakadabra`, `abaababab` und `rhabarber` den jeweiligen Suffixbaum.

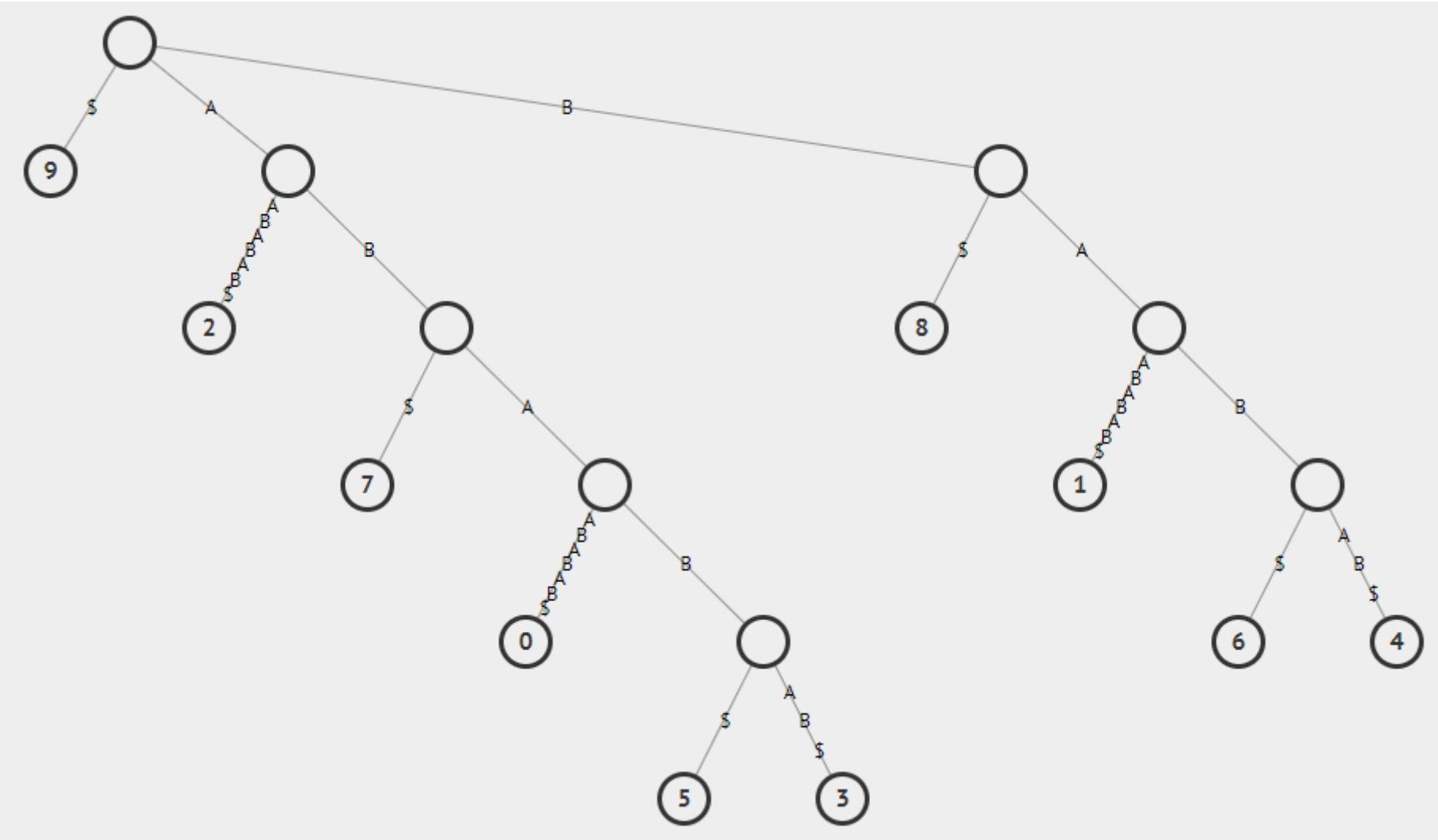
Aufgabe 1 (5 Punkte): Bestimmen Sie für die Strings `abrakadabra`, `abaababab` und `rhabarber` den jeweiligen Suffixbaum.



https://visualgo.net/en/suffixtree

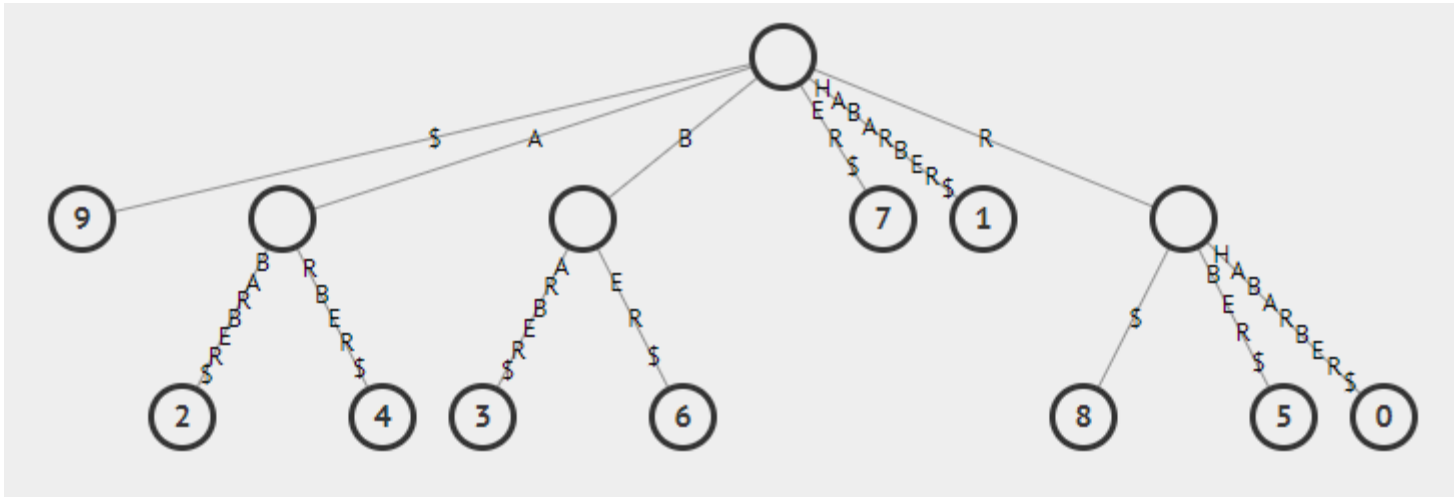
Aufgabe 1 (5 Punkte): Bestimmen Sie für die Strings `abrakadabra`, `abaababab` und `rhabarber` den jeweiligen Suffixbaum.

Aufgabe 1 (5 Punkte): Bestimmen Sie für die Strings `abrakadabra`, `abaababab` und `rhabarber` den jeweiligen Suffixbaum.



Aufgabe 1 (5 Punkte): Bestimmen Sie für die Strings `abrakadabra`, `abaababab` und `rhabarber` den jeweiligen Suffixbaum.

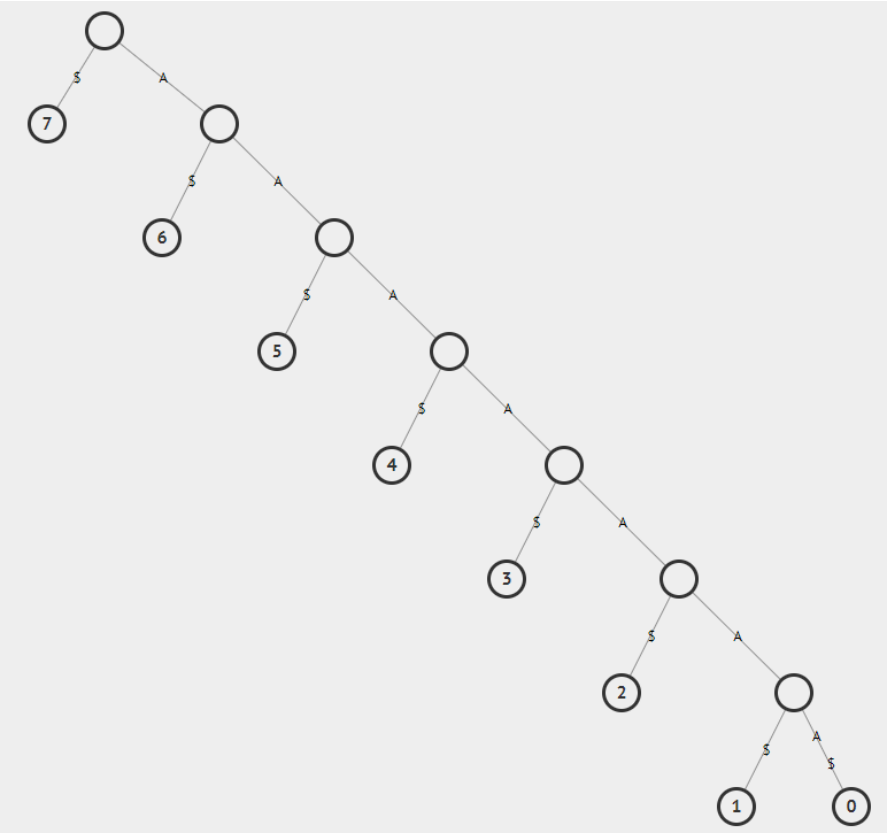
Aufgabe 1 (5 Punkte): Bestimmen Sie für die Strings `abrakadabra`, `abaababab` und `rhabarber` den jeweiligen Suffixbaum.



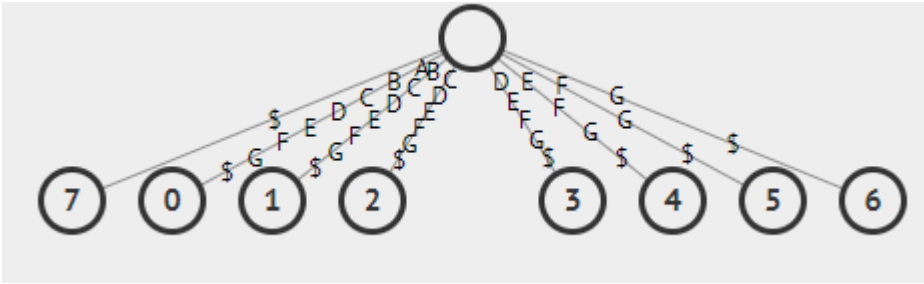
Aufgabe 2 (5 Punkte): Geben Sie einen String der Länge n an, der die Anzahl innerer Knoten eines Suffixbaumes minimiert, und einen String, der sie maximiert. Geben Sie die Anzahl innerer Knoten im jeweiligen Suffixbaum an.

Aufgabe 2 (5 Punkte): Geben Sie einen String der Länge n an, der die Anzahl innerer Knoten eines Suffixbaumes minimiert, und einen String, der sie maximiert. Geben Sie die Anzahl innerer Knoten im jeweiligen Suffixbaum an.

Ein String bestehend aus n-Mal dem selben Buchstaben (Bsp.: S=AAAAAAA) erzeugt einen Suffixbaum mit maximaler Anzahl innerer Knoten (n-1).



Ein String bestehend aus n verschieden Buchstaben (Bsp.: S=ABCDEFG) erzeugt einen Suffixbaum mit minimaler Anzahl innerer Knoten (0).



Aufgabe 3 (5 Punkte): Geben Sie einen Algorithmus an, der für einen vorhandenen Suffixbaum für einen String S bei Eingabe eines Patterns P die Position jedes Vorkommens von P in S ausgibt und eine Laufzeit von $O(m + k)$ benötigt, wobei $m = |P|$ und k die Anzahl der Vorkommen von P in S ist. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Aufgabe 3 (5 Punkte): Geben Sie einen Algorithmus an, der für einen vorhandenen Suffixbaum für einen String S bei Eingabe eines Patterns P die Position jedes Vorkommens von P in S ausgibt und eine Laufzeit von $O(m + k)$ benötigt, wobei $m = |P|$ und k die Anzahl der Vorkommen von P in S ist. Begründen Sie Korrektheit und Laufzeit Ihres Algorithmus.

Algorithmus und Laufzeit:

1. Verwende gegebenen Suffixbaum B von S.
2. Verwende Funktion *Finde_Pfad_Ende(B,P)* um den Knoten v zu finden, der im Suffixbaum B unterhalb von P liegt, falls P in S enthalten ist. *Laufzeit: O(m)*
3. Falls es $k > 0$ Vorkommen von P gibt: Baum *B(v)* unterhalb des Knotens v hat genau k Blätter und Größe *O(k)*.
4. Gehe durch *B(v)* durch und gib die Indizes an allen Blättern aus. Dies sind die Stellen an denen P in S vorkommt. *Laufzeit: O(k)*

Gesamtlaufzeit ergibt sich aus Schritt 2 und 4: *O(m+k)*

Korrektheit:

Da es in einem Suffixbaum B von S von der Wurzel ausgehend immer genau einen eindeutigen Pfad zu jedem beliebigem Substring P von S gibt, werden auch immer nur genau $m = |P|$ Buchstabenvergleiche gebraucht, um P in S zu finden.

Der Knoten v unterhalb von P kann entweder ein Blatt sein, dann gibt es nur ein Vorkommen von P in S. Oder v ist ein innerer Knoten, dann hat entspricht die Anzahl der Blätter unter v, der Anzahl an Vorkommen von P in S.

Aufgabe 4 (5 Punkte): Geben Sie einen Algorithmus an, der unter Verwendung eines Suffixbaumes für einen gegebenen String S der Länge n in einer Laufzeit von $O(n)$ den längsten Teilstring P findet, der mindestens zweimal in S vorkommt. Begründen Sie Korrektheit und Laufzeit des Algorithmus.

Aufgabe 4 (5 Punkte): Geben Sie einen Algorithmus an, der unter Verwendung eines Suffixbaumes für einen gegebenen String S der Länge n in einer Laufzeit von $O(n)$ den längsten Teilstring P findet, der mindestens zweimal in S vorkommt. Begründen Sie Korrektheit und Laufzeit des Algorithmus.

Algorithmus und Laufzeit:

1. Erstelle Suffixbaum B von S. Laufzeit: O(n)
2. Traversiere den Baum und markiere alle Knoten mit ihrer Stringtiefe. Laufzeit: O(n)
3. Suche den inneren Knoten mit der größten Stringtiefe (Das kann man auch direkt in Schritt 2 schon mit erledigen).
4. Der String bis zu diesem Knoten ist der gesuchte Teilstring P.

Korrektheit:

Ein Teilstring P kommt genau dann mehrmals in S vor, wenn im Suffixbaum "unter" P mehrere Indizes stehen, also mehrere Blätter hängen. Da wir uns jeden Knoten im Suffixbaum einmal anschauen, müssen wir auch den längsten Teilstring P finden, der an einem inneren Knoten endet.

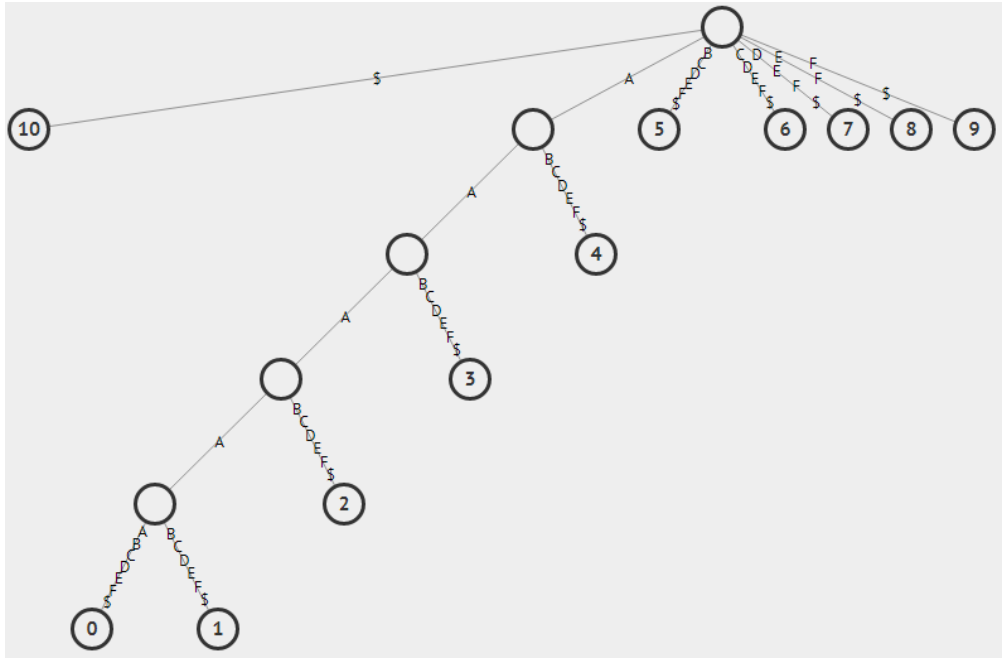
Geben ist ein String $S = s_1, s_2, \ldots, s_n$. Findet ein Beispiel für $n = 10$ bei dem die Anzahl der inneren Knoten (inklusive Wurzel) $k = 5$ ist.

Wie muss ein String S aufgebaut sein, damit immer gilt: $k / n = 2$ (für gerade n)?
Wie muss ein String S aufgebaut sein, damit immer gilt: $k / n = p$ (für gerade n und $p \in \mathbb{N}^+$)?

Geben ist ein String $S = s_1, s_2, \ldots, s_n$. Findet ein Beispiel mit $n = 10$, so dass der Suffixbaum von $S\$$ genau $k = 5$ inneren Knoten (inklusive Wurzel) hat.

Wie muss ein String S aufgebaut sein, damit immer gilt: $k/n = 2$ (für gerade n)?
Wie muss ein String S aufgebaut sein, damit immer gilt: $k/n = p$ (für gerade n und $k, p \in \mathbb{N}^+ \leq n$)?

Mögliche Strings:
S = AAAAABCDEF
S = BAAAAACDEF
S = BCAAAAADEF
S = BCDAAAAAEF
…
S = FDBCAAAAAE



Allgemeine Form:

$S = (X^m)(AAAAA)(X^{5-m})$

für $0 \leq m \leq 5$, und alle Buchstaben X paarweise unterschiedlich zu jedem anderen Buchstaben in S

Allgemeine Form:

$S = (X^m)(A)^p(X^{n-p-m})$

für $0 \leq m \leq n - p$, und alle Buchstaben X paarweise unterschiedlich zu jedem anderen Buchstaben in S