Aufgabe 5.1: Feature Trees (T)_Artischocken

Ein *feature tree* ist ein topologischer Deskriptor, der ein Molekül mit Hilfe eines knotenmarkierten Baumes beschreibt [Rarey1998]. Dabei repräsentieren die Knoten des Baumes einzelne Fragmente des Moleküls, die ihrerseits durch Kanten miteinander verknüpft sind. Jeder Knoten des Moleküls enthält eine bestimmte Anzahl an Features, durch die die sterischen und chemischen Eigenschaften des jeweiligen Fragments beschrieben werden. Im weiteren Verlauf soll der in Abbildung 1 dargestellte feature tree betrachtet werden:

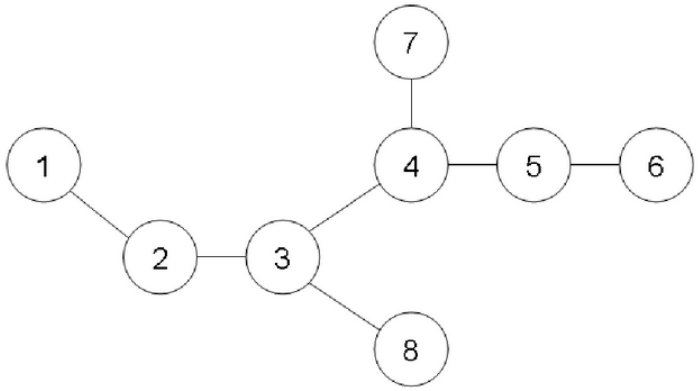

Abbildung 1: Ein exemplarischer *feature tree*

1. Bestimmen Sie für den *feature tree* aus Abbildung 1 die Prioritäten der Kanten, aus denen sich die Reihenfolge ergibt, in der diese im Match-Search Algorithmus (Dynamische Programmierung) prozessiert werden müssen. Beachten Sie dabei, dass jede Kante des Baums in zwei Richtungen durchlaufen werden muss [Rarey2001].
2. Entwickeln Sie einen Algorithmus in validem Pseudocode, der aus einem *feature tree*-Graphen G = (V,E) eine Kantenordnung f : E → 1,...,|E| berechnet, in der die dynamische Programmierung (DP) erfolgen kann. Der Algorithmus sollte in Zeit O(V + E) laufen.

   Hinweis: Erzeugen Sie einen Graphen G', dessen Knoten die Kanten von G sind und dessen Kanten die Abhängigkeiten bei der DP-Berechnung beschreiben. Führen Sie eine topologische Sortierung durch.
3. Wie reagiert ihr Algorithmus, wenn der Eingabegraph kein Baum ist?

1.) Prioritäten der Kanten

Zur besseren Beschreibung der Splits, werden die Kanten als Paar antiparallel-gerichteteter Kanten betrachtet.

Reihenfolge der Prozessierung im Match-Search-Algorithmus: von Kanten, die zu endständigen Atomen führen, zu Kanten, die zu inneren Knoten führen.

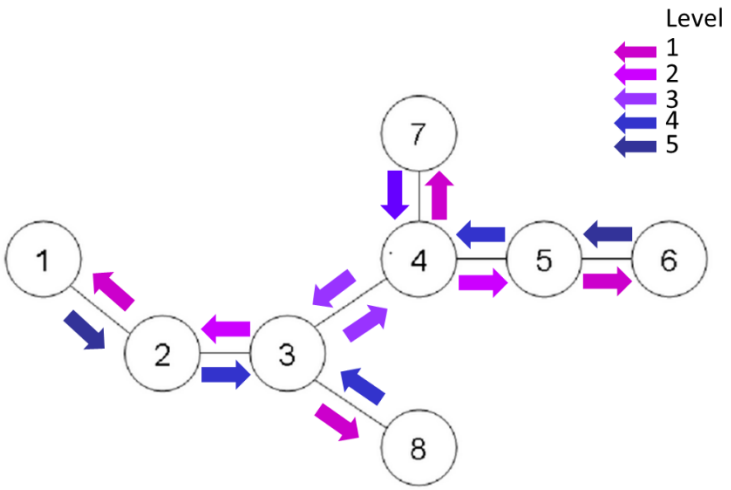

Abbildung 1: Prioritäten der Kanten, aus denen sich die Reihenfolge ergibt, in der diese im Match-Search-Algorithmus prozessiert werden müssen; Bei der Priorisierung wird angefangen bei Kanten, die zu endständigen Atomen führen (hier alle mit Nummer 1) und dann alle mit nächster inkrementeller Kantennummer

Erst werden die endständigen Kanten mit 1 markiert, dann entstehen im Folgeschritt neue endständige Kanten, die dann mit 2 markiert werden und rekursiv so weiter.

2.) Entwicklung eines Algorithmus, der aus einem feature-tree Graphen G=(V,E) eine Kantenordnung f:E->,…,|E| berechnet, in der die dynamische Programmierung erfolgen kann. Der Algorithmus soll in O(E+V) laufen. Durchführung einer topologischen Sortierung.

DFS-VISIT(u,E'):

```
1  u.color=grau
2  for each v in u.adj: //Durchlauf der in u eingehenden Kanten
3    if v.color==false:
4      DFS-VISIT(v,E')
5    else if (v.color==grau):
6      Error: "Zyklus entdeckt"
7  u.color=schwarz
8  E'.insert(u) //Liste topologisch sortierter Kanten
```

getKantenordnung(G=(V,E)):

```
1  V'=[]
2  E'=[]
/*Knoten in G' sind Kanten von G; hier werden die Kanten als
Paar antiparallel gerichteter Kanten betrachtet*/
3  for each e in G.E:                  //O(E)
4    V'.append(e.atom1, e.atom2)
5    V'.append(e.atom2,e.atom1)
6  G'=(V',E')
7  for each u in G'.V':
8    u.color=false
9  for each u in G'.V':                //O(V)
10    if u.color==false:
11       DFS-VISIT(u,E')
12 return G'
```

3.) Wenn es sich um eine zyklische Struktur handelt, dann wird diese in Zeile 5 in DFS-VISIT entdeckt und der Algorithmus abgebrochen, denn eine topologische Sortierung kann nicht auf einer zyklischen Struktur erfolgen. Wenn es sich um eine Liste handelt, so werden die Knoten einer nach dem anderen abgearbeitet.