Terminplanung Termin 3 [19.03.26]

Wiederholung / nachholen

- Nur Vorgehen zeigen in *fuelle-zwei-tiefensuche-tiefe-liste.py*
- Präsentation ***P02-d fuelle-viele.pdf***
- Projekt dazu in *fuelle-viele-ts.py* nur Prinzip zeigen in:
  Laufzeitmessung mit *fuelle-viele-tiefensuche-tiefe-liste.py* -> Berechnungsaufwand
- Unterschied in Laufzeit bei Erfolgsfall und Misserfolgsfall → warum?
  → dies ist eine **optimierte Tiefensuche**!
- Auf Grund von Nachfragen: Beispiele zum Einsatz von Slizes:

```
liste=["A", "C", "G","H"]
print(liste[1:3]) ## -> ['C', 'G']
print(liste[:1]+["B"]+liste[1:]) ## -> ['A', 'B', 'C', 'G', 'H']
print(liste[:1]+["B"]+liste[2:]) ## -> ['A', 'C', 'G', 'H']
print(liste.index("G")) ## -> 2
```

Breitensuche

- überhaupt behandeln?
- Präsentation ***P02-3 Breitensuche_80_Binaerbaum.pdf***
- Präsentation ***P02 Breitensuche-Praesentation-Pythonversion.pdf***
- *fuelle-iterativ-HGT-tiefe-liste.py* ansehen und testen

Optimierung statt Satisfizierung

- Präsentation ***P02-c Satisfizierung oder Optimierung.pdf***
- Projekte *Optimierung-tiefensuche-tiefe-Liste.py* für die TS
  (*Optimierung.py* für die BS eher nicht zeigen, also nur Hinweis)
- In den Projekten gezieltes Einsetzen von call-by-reference für den Parameter *bester* , dazu
  Optimierung_call_by_reference_nutzen.pdf

Wegeproblem kürzester Weg

- Präsentation ***P03 Wegeprobleme.pdf***
- Präsentation ***P03-a Graphen Modell und Datenstruktur-Py.pdf***
- Präsentation ***Gallenbacher_Kap1_Ausschnitt.pdf*** , Hinweis auf Buch und CD
  - (komplett einmal "durchspielen" wäre gut → auf Papier!)
  - Eingehen auf Gallenbacher-Konzept ohne Programmierung
  - Hinweis auf : Selbst wenn Gallenbacher Informatik ohne Computer macht, löst er sich dennoch nicht von einer prozeduralen bzw objektorientierten Algorithmik !
- Zeigen, durchlaufen lassen und detailliert erläutern:
  Projekt *Graphen Demoversion mit Schrittfunktion Dijkstra*