Programmdokumentation für die Codes:
ALG-BINARY_loese_Instanz und ALG_BINARY_generiere_Daten

In diesem Repository liegen zwei verschiedene Codes. Diese sind größtenteils identisch, erfüllen allerdings vollkommen unterschiedliche Zwecke und sind anders zu bedienen.
Beide setzen den ALG-BINARY Algorithmus aus "Greedy Algorithms for Maximizing Nash Social Welfare" von Siddharth Barman, Sanath Kumar Krishnamurthy und Rohit Vaish um. Ebenso sind sie mit sinvollen Variablenbenennungen und ausführlichen Kommentaren ausgestattet. Der Code ist unter Python 3.10 lauffähig.

Die Implementation:

Der Algorithmus, wurde wie im Paper umschrieben umgesetzt.
Zusätzlich existiert eine pseudozufällige Erstellung von Instanzen.
Dies beinhaltet die Funktionen:
coinflip(), create_agents(), create_agent_preferences und random_allocation().

Alle anderen Funktionen sind Teil des Algorithmus.
Die Funktionsweise jeder Funktion ist im Code konkret erklärt.

Die main() befindet sich am Ende und regelt die Abfolge der anderen Programmabschnitte.
Diese setzt in beiden Codes die folgenden Funktionen um:
1) Erstellen einer Instanz
2) Finden der Neidrelation zwischen allen Agentenpaaren.
3) Für jeden Agenten Breitensuche auf dieser Kantenmenge laufen lassen.
4) Alle (bis zu n^2) erhaltenen kürzesten Pfade bezüglich der NSW Verbesserung bewerten
5) Items finden, deren Reallokation den besten Pfad bilden
6) Durchführen dieser Reallokation und wiederholen der Routine
7) Abbrechen, falls kein Pfad gefunden werden kann.
Nun zu den Unterschieden und dem Zweck der Codes:

ALG-BINARY_loese_Instanz
Dieser Code ermöglicht es dem Anwender bei dem Erstellen und Lösen einer einzelnen Instanz zuzusehen. Dabei enthält der Code eine Menge an Ausgabebefehlen, die es dem Anwender möglich machen die einzelnen Schritte nachzuvollziehen und den Algorithmus beim Rechnen zu beobachten.
Dabei kann der Anwender die Problemgröße selbstständig einstellen. Dies wird nicht über eine Abfrage, sondern über das Eintragen von Werten in den Quellcode umgesetzt.
Dazu finden sich in Zeilen 14-16 drei globale Variablen. Dort können natürliche Zahlen als Werte für agent_quantity und item_quantitiy eingegeben werden.
Dies entspricht der Anzahl von Agenten und Items, also den Parametern n und m.

Der dritte Parameter gibt die Wahrscheinlichkeit dafür an, dass ein Agent i ein Item j mag. Hier können float Werte zwischen 0 und 1 eingetragen werden.
Nach dem Ausführen beendet sich das Programm selbst und kann erneut ausgeführt werden.
Der Nutzer kann mit den Werten herumexperimentiern, sinnvoll sind Werte mit n < m.
Gut lesbare Instanzen sind etwa n=4, m=10, p beliebig.
Sehr große Intanzen sind umständlich zu lesen und führen zu langen Laufzeiten.

ALG_BINARY_generiere_Daten

Dieser Code wurde in der Bachelorarbeit verwendet um die Daten zu generieren.
Im Gegensatz zu **ALG-BINARY_loese_Instanz** wird die main() nicht einmal, sondern häufiger und mit verschiedenen Parametern aufgerufen.
Als Eingabe fungieren drei Listen, die Werte für n, m und p enthalten.
Über diese Listen laufen Schleifen, sodass jede Kombination der Werte berechnet wird.
Zusätzlich gibt es einen Parameter „repetitions“, der die Anzahl an Berechnungen jedes Tripels definiert, um statistisch verlässliche Daten zu erhalten.
Der Benutzer kann die Listen in Zeilen 573-576 selbst mit Werten füllen. Das Format ist eine für Python übliche Liste. Auch hier ergeben nur natürliche Zahlen mit n < m Sinn. Eine Eingabe könnte beispielsweise so aussehen:

```
573  agenten_amount_n = [50, 75, 100]
574  item_amount_m = [300, 600]
575  liking_probabilities_p = [0.01, 0.02, 0.03, 0.04, 0.05]
576  repetitions = 3
```

Hier würden 3*2*5*3 = 90 Ergebnisse produziert.

Es gibt es keine Ausgabebefehle, sondern es wird für jede Berechnung ein Ergebnistupel in das Textdokument „myfile.txt“ geschrieben. Jedes dieser Tupel enthält vier Einträge.
Diese sind von der Form: (n, m, p, die benötigte Anzahl an Reallokationen).
Beispielsweise sieht ein Ergebnistupel so aus: (50, 300, 0.01, 20)
Sobald alle Berechnungen durchgeführt sind, wird die „fertig.txt“ erstellt.
Außerdem wird eine „counter.txt“ erstellt, die Anzahl der bereits gelösten Instanzen angibt.