1. Stable Matching Problem

Einleitung

Gegeben:

- n Kinder und n Gastfamilien, die an einem Austauschprogramm für Schülerinnen und Schülern teilnehmen.
- Jedes Kind hat eine Präferenzliste von Gastfamilien.
- Jede Gastfamilie hat eine Präferenzliste von Kindern.

Ziel:

Finde eine passende Zuordnung von Kindern zu Gastfamilien.

Beispiel:

Kinder: Xaver, Yvonne, Zola.
Gastfamilien: Abel, Boole, Church.

Präferenzlisten der Kinder:

Name	1. (Höchste Präferenz)	2.	3. (Niedrigste Präferenz)
Xaver	Abel	Boole	Church
Yvonne	Boole	Abel	Church
Zola	Abel	Boole	Church

Präferenzlisten der Familien:

Name	1. Höchste Präferenz	2.	3. (Niedrigste Präferenz)
Abel	Yvonne	Xaver	Zola
Boole	Xaver	Yvonne	Zola
Church	Xaver	Yvonne	Zola

Perfektes Matching:

- Jedem Kind wird genau eine Familie zugewiesen.
- Jedes Kind bekommt genau eine Familie.
- Jede Gastfamilie bekommt genau ein Kind.

Instabiles Paar:

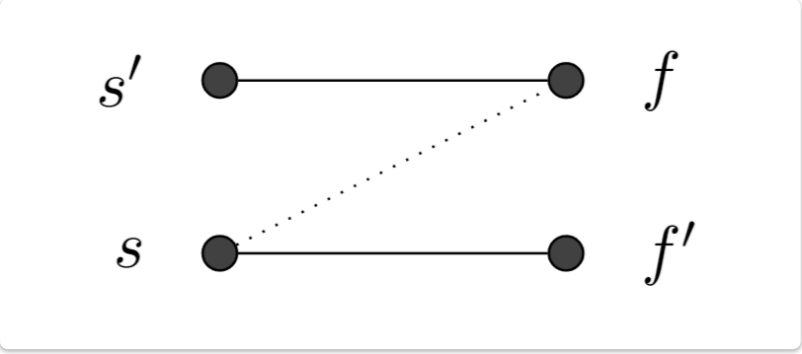
In einem Matching M ist ein nicht zugewiesenes Paar (s, f) **instabil**, wenn:

- Ein Kind s und eine Familie f sich gegenseitig gegenüber ihren aktuellen Partnern bevorzugen.
- Das instabile Paar (s, f) könnte eine Situation durch Verlassen der aktuellen Partner verbessern.

Notation:

- s für student (Kind)
- f für family (Gastfamilie)

Darstellung eines instabilen Paares:



Hier bevorzugt s die Familie f gegenüber seiner aktuellen Familie f', und gleichzeitig bevorzugt die Familie f das Kind s gegenüber ihrem aktuellen Kind s'.

Definition:

Stabiles Matching: Ein perfektes Matching ohne instabile Paare. Es besteht daher kein Paar, das einen Anreiz hätte, durch gemeinsames Handeln die Zuteilung zu unterlaufen.

Stable-Matching-Problem: Ausgehend von den Präferenzlisten von n Kindern und n Familien, finde ein stabiles Matching, wenn es existiert.

Stable-Matching-Problem: Fragen

Frage: Gibt es immer ein stabiles Matching?
Hinweis: Das ist nicht von vornherein klar!

Frage: Kann ein stabiles Matching effizient gefunden werden?
Hinweis: Der Brute-Force-Ansatz (alle möglichen Zuordnungen ausprobieren) betrachtet $n!$ viele mögliche Lösungen, was extrem ineffizient ist.

Gale-Shapley-Algorithmus: Algorithmus mit dem wir beide Fragen mit „Ja“ beantworten können.

Gale–Shapley-Algorithmus (GS-Algorithmus)

```
Kennzeichne jede Familie/jedes Kind als frei
while ein Kind ist frei und kann noch eine Familie wählen
    Wähle solch ein Kind s aus
    f ist erste Familie in der Präferenzliste von s,
    die von s noch nicht gewählt wurde
    if f ist frei
        Kennzeichne s und f als einander zugeordnet
    elseif f bevorzugt s gegenüber ihrem aktuellen Partner s'
        Kennzeichne s und f als einander zugeordnet und s' als frei
    else
        f weist s zurück
```

Beispiel für Ablauf

Ausgangssituation:

- 4 Kinder (W-Z) mit Präferenzlisten (links).
- 4 Familien (A-D) mit Präferenzlisten (rechts).

	1.	2.	3.	4.
W	B	A	C	D
X	C	B	A	D
Y	B	C	A	D
Z	B	A	D	C

	1.	2.	3.	4.
A	X	W	Y	Z
B	W	Y	X	Z
C	Z	Y	W	X
D	X	W	Y	Z

Erste Zuordnung:

- Wähle erstes freies Kind aus (z.B. W).
- Dieses wählt die erste Familie in seiner Präferenzliste aus (in diesem Fall B).
- Da B frei ist, werden die beiden einstweilig einander zugeordnet.
- Aktuelle Zuordnungen: W-B.

	1.	2.	3.	4.
W	B	A	C	D
X	C	B	A	D
Y	B	C	A	D
Z	B	A	D	C

	1.	2.	3.	4.
A	X	W	Y	Z
B	W	Y	X	Z
C	Z	Y	W	X
D	X	W	Y	Z

Zweite Zuordnung:

- Wähle nächstes freies Kind aus (z.B. X).
- Dieses wählt die erste Familie in seiner Präferenzliste aus (in diesem Fall C).
- Da C frei ist, werden die beiden einstweilig einander zugeordnet.
- Aktuelle Zuordnungen: W-B, X-C.

	1.	2.	3.	4.
W	B	A	C	D
X	C	B	A	D
Y	B	C	A	D
Z	B	A	D	C

	1.	2.	3.	4.
A	X	W	Y	Z
B	W	Y	X	Z
C	Z	Y	W	X
D	X	W	Y	Z

Dritte Zuordnung:

- Wähle nächstes freies Kind aus (z.B. Y).
- Dieses wählt die erste Familie in seiner Präferenzliste aus (in diesem Fall B).
- B ist aber W zugeordnet. In der Präferenzliste von B steht W vor Y, daher lässt B das Y abblitzen.
- Y wählt die nächste Familie in seiner Präferenzliste aus (in diesem Fall C).
- C bevorzugt Y vor ihrem Partner X, daher wird ihre Zuordnung zu X gelöst und statt dessen werden C und Y einander zugeordnet.
- Aktuelle Zuordnungen: W-B, Y-C.

	1.	2.	3.	4.
W	B	A	C	D
X	C	B	A	D
Y	B	C	A	D
Z	B	A	D	C

	1.	2.	3.	4.
A	X	W	Y	Z
B	W	Y	X	Z
C	Z	Y	W	X
D	X	W	Y	Z

Vierte Zuordnung:

- Wähle nächstes freies Kind aus, es ist X, das wieder frei geworden ist.
- X wählt die zweite Familie (B) auf seiner Präferenzliste aus.
- B bevorzugt W vor X.
- X wählt die dritte Familie (A) auf seiner Präferenzliste aus.
- A ist frei und die beiden werden einander zugeordnet.
- Aktuelle Zuordnungen: W-B, X-A, Y-C.

	1.	2.	3.	4.
W	B	A	C	D
X	C	B	A	D
Y	B	C	A	D
Z	B	A	D	C

	1.	2.	3.	4.
A	X	W	Y	Z
B	W	Y	X	Z
C	Z	Y	W	X
D	X	W	Y	Z

Fünfte Zuordnung:

- Wähle nächstes freies Kind aus (nur mehr Z übrig).
- Z wählt die erste Familie (B) auf seiner Präferenzliste aus. B bevorzugt aber W vor Z.
- Z wählt die zweite Familie (A) auf seiner Präferenzliste aus. A bevorzugt aber X vor Z.
- Z wählt die dritte Familie (D) auf seiner Präferenzliste aus.
- D ist frei, also werden Z und D einander zugeordnet.
- Aktuelle Zuordnungen: W-B, X-A, Y-C, Z-D.
- Es ist nun kein Kind mehr frei, und der Algorithmus terminiert. Wir haben ein Stable Matching gefunden.

	1.	2.	3.	4.
W	B	A	C	D
X	C	B	A	D
Y	B	C	A	D
Z	B	A	D	C

	1.	2.	3.	4.
A	X	W	Y	Z
B	W	Y	X	Z
C	Z	Y	W	X
D	X	W	Y	Z

Beweis, dass der Algorithmus korrekt funktioniert:

Beweis, dass er Terminiert

Operation 1: Kinder wählen Familien in absteigender Reihenfolge aus.

Operation 2: Sobald eine Familie zugewiesen wurde, bleibt sie zugewiesen, die Zuteilung kann sich aber ändern.

Behauptung: Algorithmus terminiert nach höchstens n^2 Iterationen der while-Schleife.

Beweis: In jeder Iteration der while-Schleife wählt ein Kind eine Familie aus. Es gibt nur n^2 Möglichkeiten dafür. □

	1.	2.	3.	4.	5.
Valentin	A	B	C	D	E
Werner	B	C	D	A	E
Xaver	C	D	A	B	E
Yvonne	D	A	B	C	E
Zola	A	B	C	D	E

	1.	2.	3.	4.	5.
Abel	W	X	Y	Z	V
Boole	X	Y	Z	V	W
Church	Y	Z	V	W	X
Dijkstra	Z	V	W	X	Y
Euler	V	W	X	Y	Z

$n(n-1)+1$ vorläufige Zuordnungen erfoderlich

Beweis, dass alle Kinder und Familien zugewiesen werden:

Behauptung: Alle Kinder und Familien werden zugewiesen.

Beweis: (durch Widerspruch)

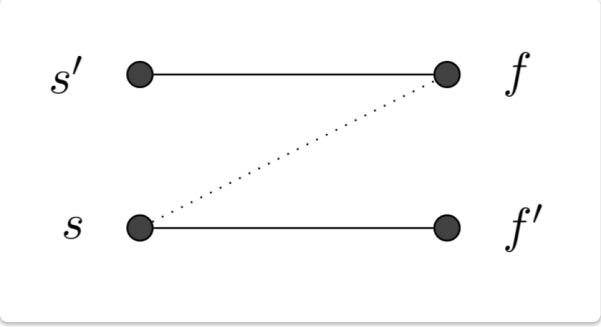
- Angenommen, Kind s wurde nach Terminierung des Algorithmus nicht zugewiesen.
- Dann wurde auch eine Familie (z.B. f) nach Terminierung des Algorithmus nicht zugewiesen.
- Damit wurde f nie ausgewählt.
- Aber s hat jede Familie in seiner Liste ausgewählt, da es ja am Ende nicht zugewiesen wurde.

Beweis: Stabilität

Behauptung: Nachdem der Algorithmus terminiert, existieren keine instabilen Paare.

Beweis: (durch Widerspruch)

Angenommen, (s, f) ist ein instabiles Paar: s bevorzugt f gegenüber seinem aktuellen Partner f' und f bevorzugt s gegenüber seinem aktuellen Partner s' in einem Gale-Shapley-Matching.



Fall 1: s hat f nie ausgewählt.
$\Longrightarrow$ s bevorzugt seinen GS-Partner f' gegenüber f.
$\Longrightarrow$ (s, f) ist nicht instabil.

Fall 2: s hat f ausgewählt.
$\Longrightarrow$ f hat s zurückgewiesen (gleich oder später).
$\Longrightarrow$ f bevorzugt s' gegenüber s.
$\Longrightarrow$ (s, f) ist nicht instabil.

In jedem Fall ist (s, f) nicht instabil, was ein Widerspruch ist. □

Wichtige Beobachtungen:

- Kinder wählen Familien in absteigender Reihenfolge der Präferenzen aus.
- Bei Familien kann sich die Situation nur verbessern.

Effiziente Implementierung

Zeitaufwand:

- Das Stable Matching benötigt höchstens n^2 Iterationen.
- Einzelne Schritte in einer Iteration können naiv (z.B. lineare Suche in Listen) implementiert werden, was eine Größenordnung von n Schritten benötigt.
- Dann benötigt man insgesamt höchstens eine Größenordnung von n^3 Schritten.
- Das ist immer noch besser, als ein Brute-Force-Ansatz, der alle möglichen Zuordnungen durchprobiert (es gibt $n!$ mögliche Zuordnungen).

Nächste Vorlesung:

- Mit asymptotischer Analyse können wir das exakt ausdrücken.
- Mit besseren Datenstrukturen können wir das auch schneller lösen.