TEAMWORK

KERSTIN GOEBEL, TIM SEIZINGER

PROBLEMBESCHREIBUNG

Du bist Din Djarin, ein Kopfgeldjäger auf der Suche nach einem interplanetaren Kriminellen!

Um dein Ziel zu finden, musst du einen Suchtrupp aus k Agenten zusammenstellen, welche für dich die N Planeten des lokalen Hyperraum-Clusters durchkämmen.

Das Hyperraum-Cluster besteht aus einem Netzwerk von gerichteten Hyper-Toren die von einem Planeten X zu einem anderen Planeten Y transportieren. Dieses Netzwerk ist frei von Kreisen, da es sonst zu gefährlichen Subraumrissen kommt.

Um kein zu großes Aufsehen zu erregen, weil der Kriminelle sonst fliehen könnte, darf ein Planet nie von mehr als einem Agenten besucht werden!

Natürlich willst du für dich selbst einen möglichst großen Anteil vom Kopfgeld. Da jeder der k Agenten einen Teil der Belohnung erwartet, ist es dein Ziel, möglichst wenige Agenten anzuheuern!

INPUT

Der kartografische Computer gibt dir das Netzwerk in folgender Form aus:

Die erste Zeile beginnt mit der Zahl N, welche angibt, wie viele Planeten das Netzwerk hat.

Dabei gibt es zwischen 0 und 1000 Planeten, welche mit Index 0 bis N-1 benannt sind.

Die darauf folgenden N Zeilen enthalten die Daten über die Hyperspace-Verbindungen.

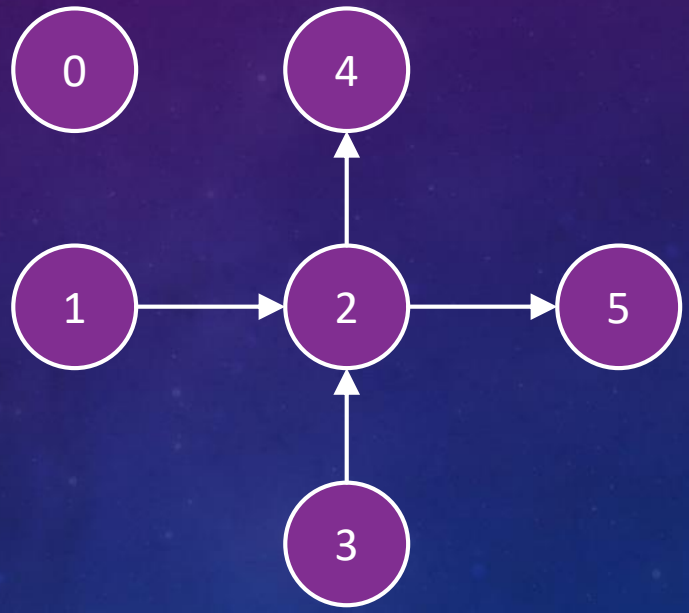
In der i-ten Zeile steht zuerst K, welches die Anzahl der Verbindungen von Planet i zu den K Nachbarplaneten definiert, dabei liegt K immer zwischen 0 und N-1.

Die K in der Zeile i stehenden Zahlen sind die jeweiligen Indizes der Nachbarplaneten.

BEISPIEL

INPUT

6
0
1 2
2 4 5
1 2
0
0



OUTPUT

4

WAS IST GESUCHT?

- Planeten sind Knoten
- Hyperraum-Verbindungen sind gerichtete Kanten
- Suchmuster der Agenten
  - disjunkte Pfade im Graphen
  - jeder Agent hat einen Pfad im Graphen

→ Finde die minimale Anzahl an kantendisjunkten Pfaden im gerichteten kreisfreien Graphen um jeden Knoten abzudecken.

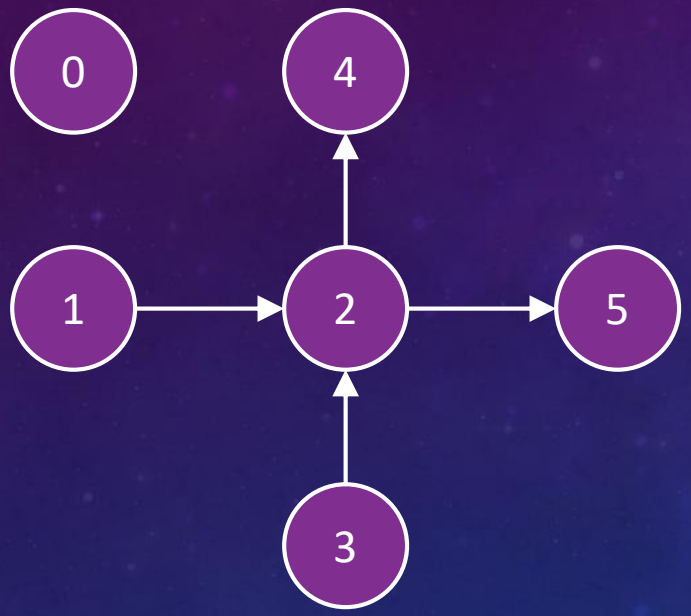
MODELLIERUNG

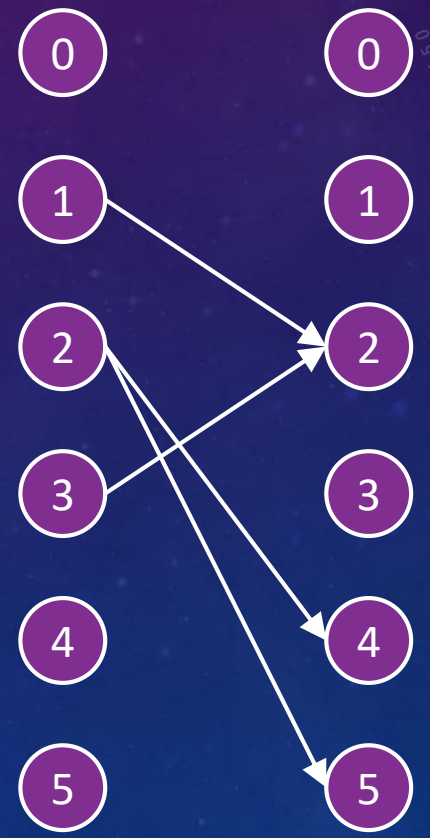
- Bedingung:

  Jeder Planet darf nur einmal besucht werden

- Aufteilung jedes Knotens v in v1 und v2
- v1 und u2 verbindet genau dann eine Kante, wenn ein Hypertor zwischen Planet v und Planet u existiert
- v1 stellt die ausgehenden Wege von v aus dar
- v2 stellt die eingehenden Wege nach v dar

BEISPIEL: MODELLIERUNG





FINDEN DER BESTEN PFADE

- Bedingung:

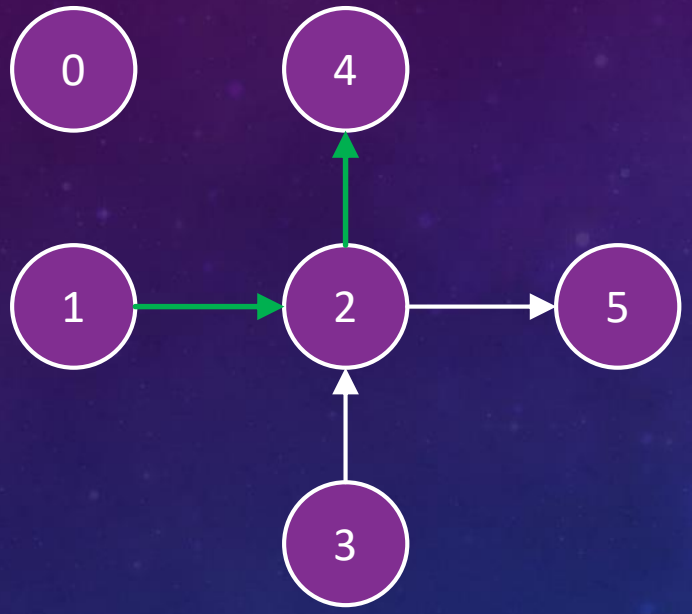
  Jede Kante darf maximal einmal genutzt werden

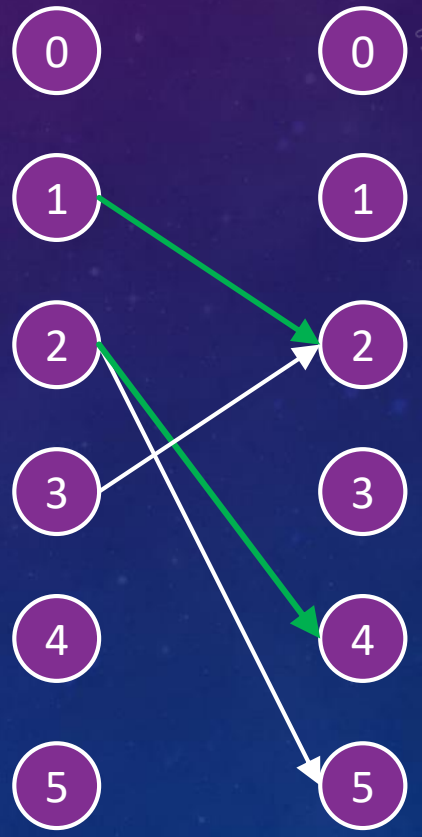
- Beobachtung:

  Ein Matching ist eine gültige Lösung

  Ein *maximales* Matching ist eine optimale Lösung

  Die Matching-Kanten stellen Wege dar, die man zwischen den Planeten nehmen muss


0
4
1
2
5
3


0
1
2
3
4
5
0
1
2
3
4
5

KORREKTHEIT

- Schlechteste Lösung: jeder Planet wird von einem Agenten besucht
  - man braucht N Agenten und N Pfade
- Ein Maximales Matching entspricht der maximalen Anzahl an Wegen, die man nehmen kann, ohne die Bedingung zu verletzen
- Jede zusätzliche Matching-Kante ist eine Verbesserung
  - Verbindung zweier Einzelpfade durch diese Kante
  - Reduktion der Gesamtzahl an Pfaden um 1
- maximales Matching -> maximale Anzahl an Verbindungskanten -> minimale Anzahl an Pfaden

IMPLEMENTIERUNG

Die Lösung ergibt sich direkt aus der Größe des maximalen Flusses.

→ Vereinfachte Version des Algorithmus von Ford und Fulkerson

```
def fordFulkerson(graph, s, t):
    f = 0
    while pathExists(graph, s, t):
        reversePathEdges(graph, s, t)
        f += 1
    return f
```

LAUFZEIT

Ursprungsgraph G=(V,E)

Konstruktion des Flussnetzwerkes G‘=(V‘,E‘)	O(V)
BFS zum Finden der augmentierenden Wege	O(V) wegen E‘ = 2V + E
Maximale Größe des Flusses beschränkt durch E	O(E)
	O(VE)