Profit!

Digitalisierung und Optimierung analoger Prozesse sind ein Kernthema der Informatik mit weitreichender Bedeutung für Wissenschaft und Wirtschaft. Es gibt eine Vielzahl von Instituten, die sich nur mit diesem Thema beschäftigen, zum Beispiel:

- IPI – Institut für Produktion und Informatik[1]
- Fraunhofer-Institut für Produktionstechnologie IPT[2]

Beim diesjährigen informatiCup optimiert Ihr im Rahmen einer rundenbasierten Simulation einen Prozess; vom Abbau und Transport der benötigten Ressourcen bis zur Herstellung der gewünschten Produkte. Dazu platziert Ihr Minen, Förderbänder, Verbinder und Fabriken auf einem zweidimensionalen Feld, das bereits Lagerstätten und ggf. Hindernisse enthält.

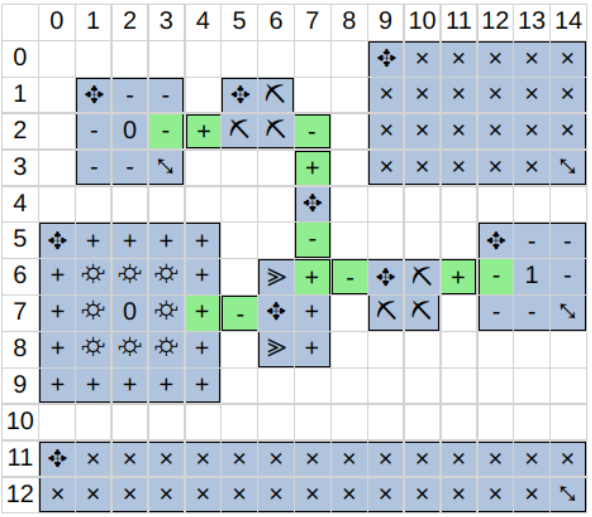

Abbildung 1: Feld mit Lagerstätten, Minen, einem Förderband, einem Verbinder, einer Fabrik und Hindernissen

[1] https://www.hs-kempten.de/forschung/forschungsinstitute/ipi-institut-fuer-produktion-und-informatik

[2] https://www.ipt.fraunhofer.de/de/kompetenzen/Prozesstechnologie/technologieorganisation/optimierung-der-fertigung.html

Modell

Das Feld ist rechteckig. Die obere, linke Zelle liegt an (0, 0), die untere, rechte Zelle an (Breite des Feldes - 1, Höhe des Feldes - 1). Objekte sind entweder Landschafts- oder Bauteile. Objekte können Subtypen haben. Bei rechteckigen Objekten liegt die linke, obere Ecke an (x, y). Die Zellbelegung nicht rechteckiger Objekte wird (nach Subtyp aufgeschlüsselt) tabellarisch angegeben. Hierbei liegt • an (x, y). Eingänge werden durch +, Ausgänge durch - und inerte Teile durch ○ dargestellt. Objekte können sich nicht überlappen (Ausnahme: Förderbänder). Nachbarschaft („liegt an“) gilt horizontal und vertikal. Kapazitäten sind unbegrenzt. Im Folgenden werden alle Objekte näher beschrieben.

Eine interaktive Implementierung des Modells findet Ihr unter https://profit.phinau.de. Dort können auch die Beispieleingaben (siehe Anmeldung und Einreichung) importiert werden.

Lagerstätte

Landschaftsteil, rechteckig, 8 Subtypen. Am Rand liegende Teile sind Ausgänge, andere Teile sind inert. Nur Eingänge von Minen (siehe unten) dürfen an den Ausgängen von Lagerstätten liegen. Der Subtyp bestimmt die Ressource, deren Anzahl verfügbarer Einheiten zu Beginn der Simulation Breite * Höhe * 5 ist.

Ende der Runde: Für jeden Ausgang, an dem der Eingang einer Mine liegt, nacheinander in unbestimmter Reihenfolge: Legt bis zu 3 Einheiten der durch den Subtyp bestimmten Ressource an und reduziert die verfügbare Menge entsprechend[3].

```
{"type":"deposit","subtype":0..7,"x":0..,"y":0..,"width":1..,"height":1..}
```

Hindernis

Landschaftsteil, rechteckig, keine Subtypen. Alle Teile sind inert.

```
{"type":"obstacle","x":0..,"y":0..,"width":1..,"height":1..}
```

[3] In der Folge ist undefiniert, welche Mine(n) bei unzureichender Verfügbarkeit nicht (vollständig) bedient werden.

Mine

Bauteil, 4 Subtypen. Nur Eingänge von Förderbändern, Verbindern und Fabriken dürfen an den Ausgängen von Minen liegen.

Beginn der Runde: Am Eingang bereitstehende Ressourcen werden angenommen.

Ende der Runde: Angenommene Ressourcen werden am Ausgang bereitgestellt.

Subtyp 0	1	2	3
•O +OO-	+ •O OO -	-•O+ OO	- •O OO +

```
{"type":"mine","subtype":0..3,"x":0..,"y":0..}
```

Förderband

Bauteil, 8 Subtypen. Förderbänder dürfen gemeinsame • und O haben (überkreuzt werden).

Beginn der Runde: Am Eingang bereitstehende Ressourcen werden angenommen.

Ende der Runde: Angenommene Ressourcen werden am Ausgang bereitgestellt.

Subtyp 0	1	2	3	4	5	6	7
+•-	+ • -	-•+	- • +	+•O-	+ • O -	-•O+	- • O +

```
{"type":"conveyor","subtype":0..7,"x":0..,"y":0..}
```

Verbinder

Bauteil, 4 Subtypen.

Beginn der Runde: Bereitstehende Ressourcen werden an allen Eingängen angenommen.

Ende der Runde: Angenommene Ressourcen werden am Ausgang bereitgestellt.

Subtyp 0	1	2	3
+O +●- +O	+++ O●O -	O+ -●+ O+	- O●O +++

```
{"type":"combiner","subtype":0..3,"x":0..,"y":0..}
```

Fabrik

Bauteil, 8 Subtypen. Am Rand liegende Teile sind Eingänge, andere Teile sind inert. Der Subtyp des Produkts ist gleich dem Subtyp der Fabrik. Breite und Höhe sind 5.
Beginn der Runde: Bereitstehende Ressourcen werden an allen Eingängen angenommen.
Ende der Runde: Stellt so viele Produkte her, wie Ressourcen in den dafür erforderlichen Mengen angenommen wurden. Die Produkte verbleiben in der Fabrik.

```
{"type":"factory","subtype":0..7,"x":0..,"y":0..}
```

Produkte

8 Subtypen. Die für die Herstellung eines Produkts (des gegebenen Subtyps) erforderlichen Ressourcen werden als Liste angegeben, an deren n-ter Stelle (nullbasiert) die Menge (ganze Zahl) der Ressource des Subtyps n steht. Für die Herstellung eines Produkts werden die angegebenen Punkte (ganze Zahl) gutgeschrieben.

```
{"type":"product","subtype":0..7,"resources":[0..,0..,0..,0..,0..,0..,0..,0..],"points":1..}
```

Runden

Die Simulation ist rundenbasiert. Zu Beginn jeder Runde werden für jedes Objekt in unbestimmter Reihenfolge die unter „Beginn der Runde“ genannten Aktionen durchgeführt, analog zum Ende jeder Runde die Aktionen unter „Ende der Runde“.

Gesamtpunkte

Bei Herstellung eines Produkts werden die Punkte des Produkts zu den Gesamtpunkten addiert. Die Runde, in der die neue Gesamtpunktzahl erreicht wurde, wird festgehalten. Es gilt die Gesamtpunktzahl zu maximieren bei gleichzeitiger Minimierung der nötigen Runden.

Ein- und Ausgabe

Die Eingabe erfolgt über die Standardeingabe, die Ausgabe über die Standardausgabe, jeweils in JSON (RFC 7159) und terminiert durch einen Zeilenumbruch. Kommentare sind in Ein- und Ausgaben nicht erlaubt und dienen hier der Erklärung.

Eingabe

Gegeben ist ein Objekt.

```
{
  "width":30, // Breite des Spielfeldes, maximal 100
  "height":20, // Höhe des Spielfeldes, maximal 100
  "objects":[ // Landschaftsteile (Liste)
    {"type":"deposit","subtype":0,"x":1,"y":1,"width":5,"height":5},
    {"type":"deposit","subtype":1,"x":1,"y":14,"width":5,"height":5},
    {"type":"deposit","subtype":2,"x":22,"y":1,"width":7,"height":7},
    {"type":"obstacle","x":11,"y":9,"width":19,"height":2},
    {"type":"obstacle","x":11,"y":1,"width":2,"height":8}
  ],
  "products":[ // Produkte (Liste)
    {"type":"product","subtype":0,"resources":[3,3,3,0,0,0,0,0],"points":10}
  ],
  "turns":50, // Anzahl Runden
  “time”:120 // Maximale Rechenzeit in Sekunden
}
```

Ausgabe

Erwartet wird eine Liste von Bauteildefinitionen. Landschaftsteildefinitionen sind unzulässig.

```
[
  {"type":"mine","x":3,"y":7,"subtype":1},
  {"type":"mine","x":7,"y":14,"subtype":0},
  {"type":"mine","x":19,"y":4,"subtype":2},
  {"type":"conveyor","x":18,"y":1,"subtype":7},
  {"type":"conveyor","x":15,"y":0,"subtype":6},
  {"type":"conveyor","x":11,"y":0,"subtype":6},
  {"type":"conveyor","x":10,"y":2,"subtype":5},
  {"type":"conveyor","x":10,"y":6,"subtype":5},
  {"type":"conveyor","x":10,"y":10,"subtype":5},
  {"type":"combiner","x":11,"y":14,"subtype":0},
  {"type":"conveyor","x":5,"y":9,"subtype":0},
  {"type":"conveyor","x":6,"y":11,"subtype":5},
  {"type":"conveyor","x":8,"y":13,"subtype":0},
  {"type":"factory","x":13,"y":12,"subtype":0}
]
```

Ausführung

Eure Lösung muss ein Dockerfile enthalten, mit dessen Hilfe Eure Lösung unter Ubuntu 22.04 LTS gebaut und gestartet werden kann. Um Eure Lösung zu bauen, wird folgender Befehl in dem Verzeichnis ausgeführt, in dem Euer Dockerfile enthalten ist:

```
docker build --tag <Tag (zufällig generiert)> .
```

Gestartet wird Eure Lösung pro Eingabe mit folgendem Befehl:

```
docker run -i --rm --network none --cpus 2.000 --memory 2G --memory-swap 2G <Tag>
```

Die Messung der Rechenzeit beginnt mit dem Aufruf von `docker run`. Nach Ablauf der maximalen Rechenzeit (siehe Eingabe) wird die Verarbeitung ohne Ergebnis abgebrochen.

Bewertung

Die Qualität der Lösung wird anhand der Summe der erreichten Gesamtpunkte über alle Wertungseingaben bestimmt. Bei Lösungen mit gleicher Summe der erreichten Gesamtpunkte zählt die niedrigere Summe der insgesamt benötigten Runden (siehe Modell).

Die Bewertung der Einreichung erfolgt neben der reinen Qualität **anhand des theoretischen Ansatzes sowie der Softwarearchitektur und -qualität**. Bitte achtet darauf, dass diese Aspekte in Eurer Ausarbeitung thematisiert wurden, siehe Checkliste der Bewertungskriterien. Wenn Ihr Eure Einreichung im Finale präsentiert, fließt auch die Präsentation in die Bewertung ein.

Anmeldung und Einreichung

Die Anmeldung zum Wettbewerb findet über Teammates[4] statt. Auf diesem Online-Portal könnt Ihr Euch als Teilnehmer registrieren, mit anderen Teilnehmern ein Team bilden und dieses Team schließlich zum Wettbewerb anmelden. Ihr könnt in Teammates Eure Einreichung vornehmen.

Die FAQs zum laufenden Wettbewerb sowie Beispieleingaben findet Ihr in dem GitHub-Repository[5] des diesjährigen Wettbewerbs.

[4] https://teams.informaticup.de/
[5] https://github.com/informatiCup/informatiCup2023

Checkliste der Bewertungskriterien

Bitte nutzt diese Checkliste, um sicherzugehen, dass Eure Einreichung vollständig ist.

Theoretischer Ansatz

Der theoretische Ansatz muss in einer Ausarbeitung dargestellt werden, die zusammen mit der Implementierung eingereicht wird. Bewertet werden sowohl der Inhalt als auch die Formalien.

Die theoretische Ausarbeitung soll die verwendete Theorie beschreiben.

- ❑ **Hintergrund:** Der Hintergrund der Lösung. Welche theoretischen Ansätze wurden verwendet? Warum wurden diese verwendet?
- ❑ **Auswertung:** Wie ist die Qualität der Lösung? Nach welchen (wissenschaftlichen) Kriterien wurde bewertet? Warum wurden diese Kriterien verwendet?
- ❑ **Diskussion:** Wie „gut“ (auch außerhalb der reinen Qualität) ist die Lösung? Was für Probleme gibt es noch? Wie lässt sich die Lösung praktisch einsetzen?
- ❑ **Quellen:** Wurden (wissenschaftliche) Quellen richtig und angemessen verwendet?

Eine gute Form ist entscheidend für die Lesbarkeit einer Ausarbeitung. Beachtet deshalb neben dem reinen Inhalt der Ausarbeitung auch einige Formalien:

- ❑ **Rechtschreibung:** Rechtschreibung und Grammatik sind korrekt.
- ❑ **Lesefluss:** Es gibt einen Lesefluss in der Ausarbeitung.
- ❑ **Layout:** Das Dokument hat ein einheitliches Layout. Dieses kann frei gewählt werden, darf aber nicht den Lesefluss stören.
- ❑ **Zitate:** Es wird richtig und einheitlich zitiert.
- ❑ **Quellenangaben:** Quellen sind richtig und einheitlich angegeben.

Softwarearchitektur und -qualität

Da eine etablierte Softwarearchitektur nur mit hohem Aufwand zu ändern ist, sollte sie gründlich durchdacht und ausführlich begründet werden. Ausgewählte Aspekte der Softwarequalität sind für die Bewertung von besonderer Bedeutung. Gerne dürfen auch hier nicht genannte Aspekte aus den sehr weiten Feldern Softwarearchitektur und -qualität beleuchtet werden.

- **Architektur:** Beschreibung der Komponenten und ihrer Beziehungen
- **Software testing:** Begründetes Konzept, Umsetzung
- **Coding conventions:** Begründetes Konzept, Umsetzung
- **Wartbarkeit:** Mit welchem Aufwand kann das System angepasst werden?

Präsentation

Im informatiCup-Finale werden die besten Einreichungen vor einer Fachjury präsentiert. Die Bewertung erfolgt anhand dieser Kriterien:

- **Foliendesign:** Sind die Folien ansprechend? Lenken die Folien nicht vom Inhalt der Präsentation ab?
- **Vortragsstil:** Weckt der Vortragsstil Interesse an der Präsentation?
- **Verständlichkeit:** Ist der Vortrag verständlich? Wird der Hintergrund der Lösung in einem angemessenen Tempo erklärt? Wird nötiges Vorwissen geschaffen?
- **Reaktion auf Nachfragen:** Können Nachfragen beantwortet werden?

Optional

Die Bearbeitung dieser Aspekte ist nicht verpflichtend.

- **CLI und GUI:** Werden eine Kommandozeilenschnittstelle mit erweiterten Konfigurationsmöglichkeiten und/oder eine grafische Benutzeroberfläche eingereicht, ist eine kurze Bau- und Bedienungsanleitung hinzuzufügen.
- **Ein- und Ausgabe:** Eine einzige Eingabe darf eingereicht werden, von der Ihr glaubt, dass Eure Lösung damit besonders gut funktioniert, mit Beispielausgabe.