KIP 23/24 - Praktikum 1: Suchen und Planen

Rahmenbedingungen

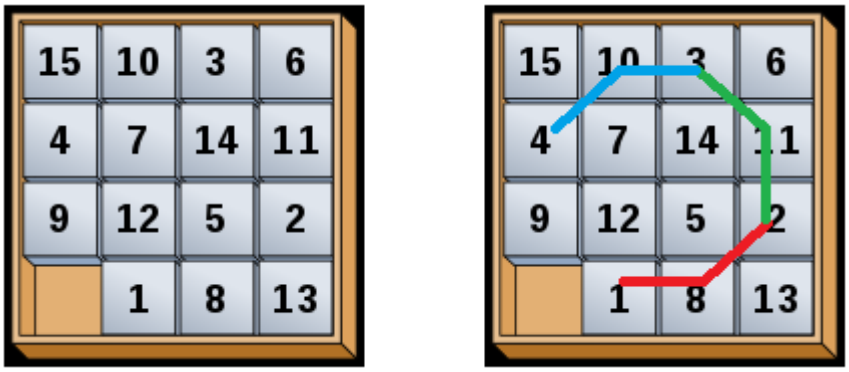
Programmiersprache: Python.
Persönliche Empfehlung: PyCharm als IDE verwenden, andere Optionen wie z.B. Jupyter Notebooks aber auch legitim. Lösung, wenn z.B. kein Laptop vorhanden ist, welcher zum Praktikumstermin mitgebracht werden kann: Jupyter Notebooks auf Laborrechnern im 11. Stock oder Google Collab → Jupyter Notebooks in Google Cloud.
Implementierung: In den Aufgaben werden einige "Grundpfeiler" für die Implementierungen vorgegeben, davon ab werden euch jedoch sehr viele Freiheiten gelassen. Grundprinzipien aus Programmieren 1 und 2 und Software Engineering sollten natürlich auch in KIP Anwendung finden, so dass die Implementierung gut "lesbar", wartbar, erweiterbar etc. sein sollte.
LLM Policy: Die Verwendung von z.B. ChatGPT ist grundsätzlich erlaubt. Falls ihr dieses verwenden solltet, dokumentiert grob eure Prompts, wie gut das geklappt hat, wo ihr Verbesserungen vorgenommen habt und ob ihr meint, dadurch einen Zeitgewinn erhalten zu haben.
Persönliche Einschätzung: Der Zeitgewinn wird eher gering ausfallen, da der generierte Code auch verstanden und häufig verbessert werden muss. Dadurch, dass eben diese Konzepte auch verstanden und vermittelt werden müssen für die Abgabe (s.u.), bietet es sich für den Lernprozess ggf. eher an, den Code von Grund auf selbst zu implementieren.
Abnahme:
Grundsätzlich gilt: Beide Teammitglieder müssen sowohl den Code als auch die relevanten Konzepte vollständig erklären können. **Die Aufgaben sollten bis zu dem Praktikumstermin fertig sein und nicht erst im Praktikumstermin bearbeitet werden**.
Der generelle Ablauf ist wie folgt: Idealerweise redet ihr den größten Teil der Zeit über eure Aufgabenlösungen, führt grob durch den Code und stellt die dahinterstehenden Konzepte vor und beantwortet die Teilfragen der einzelnen Aufgaben, sodass nur noch wenige Rückfragen bleiben. Zwischendurch und danach werde ich ggf. einige Fragen zur Implementierung und den dazugehörigen Konzepten der Vorlesung stellen. Idealerweise dauert dieser Prozess etwa. 20 Minuten - bereitet euch also vor!

Aufgabe 0: Python lernen

Die verwendete Programmiersprache ist Python. Aus der Data Science Veranstaltung sollte diese bekannt sein. Falls eine Auffrischung nötig sein sollte, finden sich gute Quellen im Internet, wie z.B. https://docs.python.org/3/tutorial/index.html. Wichtige Grundkonzepte wie Klassen, Funktionen, Kontrollflüsse und Datenstrukturen (Listen, Dictionaries,...) sollten verstanden und sinnvoll anwendbar sein.

Aufgabe 1: 15-Springer-Puzzle

Das 15-Springer-Puzzle ist eine Variante des 8-Puzzle mit 15 verschiebbaren Feldern und der Zielbedingung, dass der Pfad von 1 bis 15 in Springerzügen (also der erlaubten Bewegung des Springers aus Schach) besteht. Ein erlaubter Zielzustand ist links abgebildet:



Rechts dargestellt sind die ersten drei Springerzüge 1→2, 2→3, 3→4, erlaubt ist also immer ein gerader gefolgt von einem diagonalen Schritt (bzw. diagonal gefolgt von gerade). Wie beim 8-Puzzle kann das Ziel durch Verschiebungen des freien Feldes erreicht werden.
Implementiert zur Lösung dieses Problems:

- BFS
- DFS
- IDS
- A* Algorithmus

Orientiert euch an der Vorgabe der Vorlesung, dass allen Baumsuchalgorithmen dieselbe Grundfunktion zugrunde liegt, wobei nur die übergebene *Strategie* (d.h. Priorisierung der Open List) sich unterscheidet. Verwendet diese Grundfunktion also für alle Implementierungen wieder und übergebt ihr nur jeweils eine andere *Strategie*. Überlegt euch für den A*-Algorithmus eine Heuristik. Diese muss nicht zwingend *zulässig* sein, aber macht euch Gedanken, ob sie dies ist oder nicht. Baut sicherheitshalber auch eine Obergrenze der zu untersuchenden Knoten ein, nach der ohne Lösung abgebrochen werden soll (*warum?*).
Der zweite Parameter dieser Grundfunktion ist eine *Problem*instanz, welche das 15-Puzzle einem Baumsuchalgorithmus zugänglich macht, d.h. generell die Spiellogik verwaltet, einen ggf. vorgegebenen Startknoten zurückgibt, in der Lage ist einen Knoten zu expandieren, einen Zieltest implementiert hat etc.
Notiert, analysiert, vergleicht und erläutert die Ergebnisse (z.B. Dauer, Anzahl untersuchter Knoten, Anzahl gespeicherter Knoten, Ziel gefunden) der verschiedenen Algorithmen für die folgenden Startzustände:

15	10	3	6
4	7	14	11
	12	5	2
9	1	8	13

15	10	3	6
4	7	14	11
12		5	2
9	1	8	13

15	10	3	6
4	14		11
12	7	5	2
9	1	8	13