Grundlagen der künstlichen Intelligenz: Hausaufgabe 1

Tom Nick - 340528
Niklas Gebauer - 340942

Aufgabe 1

a) **Zustandsraum:** (a, b, c) wobei $a \in \{0, \ldots, 100\}, b \in \mathbb{N}, c \in \{A, B, C, Z, i, j\}$
wobei a die aktuelle Position, b die benötigte Zeit und c den Ladezustand beschreibt.
Anfangszustand: $(A, 0, 100)$
Zielzustand: (Z, b, c) wobei $c \geq 50$, b hat keine Einschränkung. **Aktionen:**

1. $$\mathsf{fahren}(start, ziel, zeit, energie) : (a, b, c) \rightarrow (x, y, z)$$

   mit

   $$\begin{aligned} a = start \land x = ziel \land \\ z = c - energie \land z \geq 0 \ \land y = b + zeit \land \\ (start, ziel, zeit, energie) \in \{&(A, Z, 170, 95), (Z, A, 170, 95), \\ &(A, i, 100, 50), (i, A, 100, 50), \\ &(i, Z, 200, 100), (Z, i, 200, 100), \\ &(i, j, 100, 50), (j, i, 100, 50), \\ &(i, B, 80, 45), (B, i, 80, 45), \\ &(j, Z, 80, 40), (Z, j, 80, 40), \\ &(j, C, 25, 20), (C, j, 25, 20), \\ &(Z, C, 20, 10), (Z, C, 20, 10)\} \end{aligned}$$

2. $$\mathsf{laden}(zustand) : (a, b, c) \rightarrow (x, y, z)$$

   mit

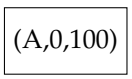
   $$zustand \in \{i, j\} \land y = b + 200 \land z = 100 \land zustand = a = x$$

b) **Verzweigungsgrad:** maximal 3
Tiefe: 6 wenn man sich beim Suchen intelligent anstellt. Wobei das beudeutet, dass wir einen Knoten nur 2x besuchen wenn im zweiten Besuch des Knotens die Ladung größer ist als beim ersten Besuch.

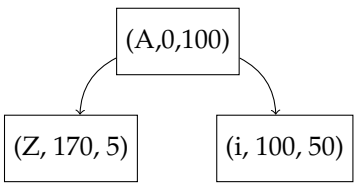
c) Da wir den schnellsten Weg Finden wollen wäre **Branch and Bound** am besten, wir haben keine Heuristiken, also kein A*, wir haben aber Pfadkosten also keine BFS/DFS.

d)

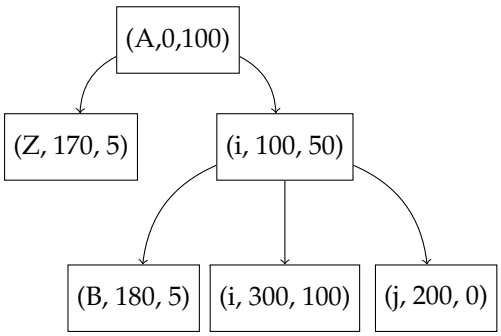
1.



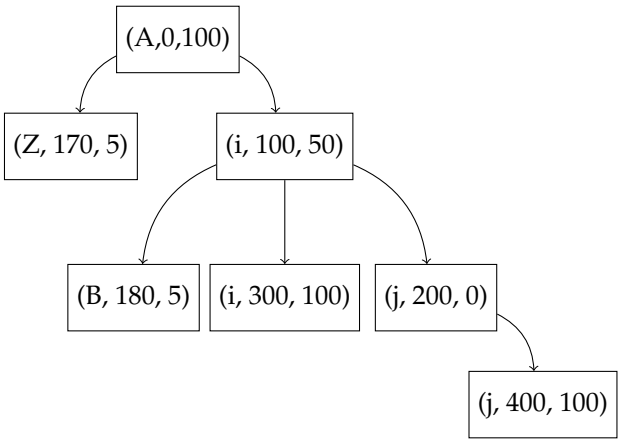
2.



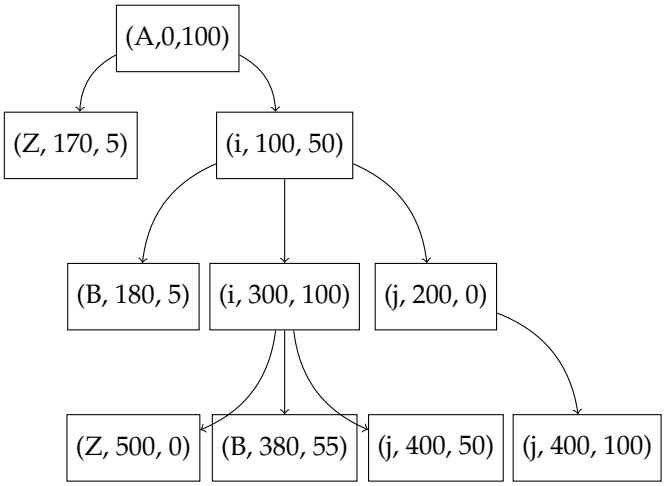
3.



4. Die dritte und vierte Expansion von dem Knoten (Z,170,5) und (B,180,5) bewirken keine Veränderung des Baumes, da Sie keine Nachfolger haben. (Energie reicht nicht aus um zu einem anderen Knoten zu fahren)
5.



6.



e) Algorithmen der dynamischen Programmierung werden genutzt um optimale Lösungen zu finden, in dem Beispiel hier wollen wir den kleinstmöglichsten Weg von A nach Z finden, wobei die verbleibende Energie größer gleich 50 ist. Dafür werden ALLE möglichen Wege untersucht, viele werden jedoch von vornerein ausgeschlossen (eine Art intelligentes Brute Force). In diesem Beispiel benutzen wir Branch and Bound. Dieser Algorithmus untersucht als nächsten Knoten immer den mit den niedrigsten insgesamten Pfadkosten, dadurch werden gleichzeitig viele unnütze Wege, Schleifen... ausgeschlossen.

Aufgabe 2

a) **Zustandsraum:** $((w_1, x_1, y_1), (w_2, x_2, y_2), (w_3, x_3, y_3), (w_4, x_4, y_4), (w_5, x_5, y_5))$ wobei $x_i, y_i \in \{(a, b) \mid a \in \{1, \ldots, 5\}, b \in \{1, \ldots, 5\}\}$ mit $i \in \{1, \ldots, 5\}$
Wir können maximal 5 Stapel bilden, da wir 5 Kisten haben. Jede Kiste wird beschrieben durch ein Tupel

von x und y welche den Stapel sowie die Höhe in diesem Stapel beschreiben.w_1 beschreibt den Wert der Kiste, man könnte auch die Reihenfolge dafür benutzen, aber dadurch werden die Formalitäten kniffliger.
Anfangszustand: $((1,1,2),(2,1,1),(3,1,4),(4,1,3),(5,1,5))$
Zielzustand: $((1,x_1,5),(2,x_2,4),(3,x_3,3),(4,x_4,2),(5,x_5,1))$ mit $x_1 = x_2 \wedge x_2 = x_3 \wedge x_3 = x_4 \wedge x_4 = x_5$
Aktionen:

1. $\mathsf{bewegen}() : (von, nach, (w_1,x_1,y_1),(w_2,x_2,y_2),(w_3,x_3,y_3),(w_4,x_4,y_4),(w_5,x_5,y_5)) \rightarrow (c_1,a_1,b_1),(c_2,a_2,b_2),(c_3,a_3,b_3$

   mit
   1. Es gibt x_i mit $i \in \{1,\ldots,5\}$ sodass $von = i$.
   2. Sei $\phi = (von, x, y_i)$ mit $i \in \{1,\ldots,5\}$ und i maximal. D.h. es darf kein (von, x, y_i) existieren mit $y_i > \phi_2$. Sei $\lambda = (nach, x, y_i)$ mit $i \in \{1,\ldots,5\}$ und i maximal. D.h. es darf kein $(nach, x, y_i)$ existieren mit $y_i > \lambda_2$. Es muss gelten:
   $$(a_{\phi_2}, a_{\phi_2}, a_{\phi_2}) = (\phi_1, nach, \lambda_3 + 1)$$
   Alle anderen Werte ändern sich nicht.

b) Eine Heuristik ist:

Aufgabe 3

a) Die iterative Tiefensuche vereint die Vorteile von Tiefen- und Breitensuche. Es handelt sich ebenfalls um eine uninformierte Suche, die für Probleme mit uniformen Aktionskosten die Optimalität der Breitensuche und den geringen Speicherverbrauch der Tiefensuche vereint. Im Gegensatz zu normalen Tiefensuche ist sie auch Vollständig für Suchbäume mit unendlich langen Pfaden.
Dies wird dadurch erreicht, dass in jedem Iterationsschrit eine Tiefensuche bis zu einer begrenzten Tiefe ausgeführt wird. Nach jedem Iterationsschritt wird diese Tiefe um 1 erhöht. Damit werden beispielsweise, wenn man bei der Tiefe 0 anfängt, erst alle Knoten, die über Pfade der Länge 0 erreichtw erden können untersucht. Dann alle Knoten, die über Pfade der Länge 1 erreicht werden können und so weiter.
Dadurch, dass jedes Mal eine Tiefensuche stattfindet, ist der Speicherbedarf gering, da immer nur der aktuell betrachtete Ast im Speicher gehalten werden muss. Durch die Begrenzung der Tiefe wird, sofern eine Lösung vorhanden ist, wie bei der Breitensuche bei uniformen Aktionskosten auch eine Lösung auf der niedrigsten Tiefe gefunden, auf der Lösungen gefunden werden können. Wenn mehrere Lösungen auf dieser Ebene existieren, ist es egal, welche von diesen wir finden, da alle die gleichen Kosten haben.

b) Beim vollständigen Durchsuchen des Baumes müssen der Wurzelknoten 6 Mal (also die Tiefe+1), alle Knoten der Tiefe 1 5 Mal, alle Knoten der Tiefe 2 4 Mal usw. generiert werden. Es ergibt sich also folgende Rechnung:

$$6 * 1 + 5 * 35 + 4 * 35^2 + 3 * 35^3 + 2 * 35^4 + 35^5 = 55656831$$

Bei der normalen Tiefensuche wird jeder Knoten genau ein mal generiert, wir haben also:

$$1 + 35 + 35^2 + 35^3 + 35^4 + 35^5 = 54066636$$

Insgesamt haben wir in diesem Fall bei der iterativen Tiefensuche also nur einen Overhead von ungefähr 3 Prozent:

$$55656831/54066636 = 1.029411761$$

c) Je höher der Verzweigungsfaktor ist (solange er nicht unendlich ist), desto geringer ist der prozentuale Overhead der iterativen Tiefensuche, denn die Anzahl der Blätter wächst mit einer Vergrößerung des Verzweigungsfaktors sehr stark. Die Blätter machen den größten Teil des Suchbaumes aus und werden sowohl bei der Tiefensuche als auch bei der iterativen Tiefensuche nur einmal generiert. Die mehrfache Generierung der Knoten geringerer Tiefen fällt bei einem hohen Verzweigungsfaktor also nicht so sehr ins Gewicht, wie bei einem niedrigen Verzweigungsfaktor, da der Unterschied der Anzahl an Knoten zweier Tiefen bei großem Verzweigungsfaktor wesentlich höher ist.
So sind bei einem Verzweigungsfaktor von 2 beispielsweise immer doppelt so viele Knoten in der nächst tieferen Ebene, wie in der Ebene darüber. Beim Verzweigungsfaktor 5 sind es hingegen schon fünf Mal so viele Knoten.

Aufgabe 4