Grundlagen der künstlichen Intelligenz: Hausaufgabe 1

Tom Nick - 340528
Niklas Gebauer - 340942

Aufgabe 1

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

1.
$$\mathsf{fahren}(start, ziel, energie) : (a, b) \rightarrow (x, y)$$
mit
$$\begin{aligned} a = start \land x = ziel \land \\ y = b - energie \land y \geq 0 \land \\ (start, ziel, energie) \in \{&(A, Z, 95), (Z, A, 95), \\ &(A, i, 50), (i, A, 50), \\ &(i, Z, 100), (Z, i, 100), \\ &(i, j, 50), (j, i, 50), \\ &(i, B, 45), (B, i, 45), \\ &(j, Z, 40), (Z, j, 40), \\ &(j, C, 20), (C, j, 20), \\ &(Z, C, 10), (Z, C, 10)\} \end{aligned}$$

2.
$$\mathsf{laden}(zustand) : (a, b) \rightarrow (x, y)$$
mit
$$zustand \in \{i, j\} \land y = 100 \land zustand = a = x$$

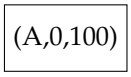
Aktionskosten:
$$\mathsf{kosten}(aktion) : Aktion(args) \rightarrow \mathbb{R}_{\geq 0}$$
mit
$$\begin{aligned} \mathsf{fahren}(A, i, 50), \mathsf{fahren}(i, A, 50), \mathsf{fahren}(i, j, 50), \mathsf{fahren}(j, i, 50) &\mapsto 100 \\ \mathsf{fahren}(i, Z, 100), \mathsf{fahren}(Z, i, 100), \mathsf{laden}(i), \mathsf{laden}(j) &\mapsto 200 \\ \mathsf{fahren}(A, Z, 95), \mathsf{fahren}(Z, A, 95) &\mapsto 170 \\ \mathsf{fahren}(i, B, 45), \mathsf{fahren}(B, i, 45), \mathsf{fahren}(j, Z, 40), \mathsf{fahren}(Z, j, 40) &\mapsto 80 \\ \mathsf{fahren}(j, C, 20), \mathsf{fahren}(C, j, 20) &\mapsto 25 \\ \mathsf{fahren}(Z, C, 10), \mathsf{fahren}(C, Z, 10) &\mapsto 20 \end{aligned}$$

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.

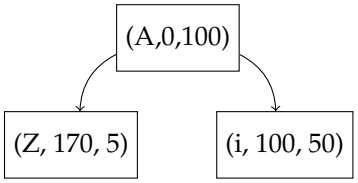
c) Da wir den schnellsten Weg finden wollen und nicht uniforme Aktionskosten haben, wäre 'Branch & Bound' am besten als Suchalgorithmus geeignet.

d)

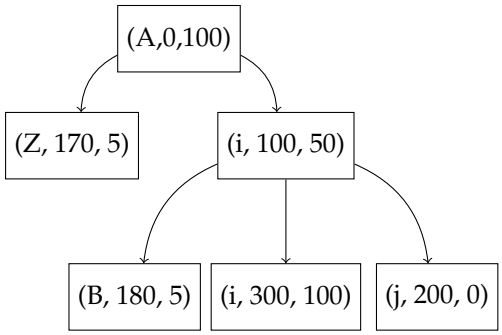
1.



2.

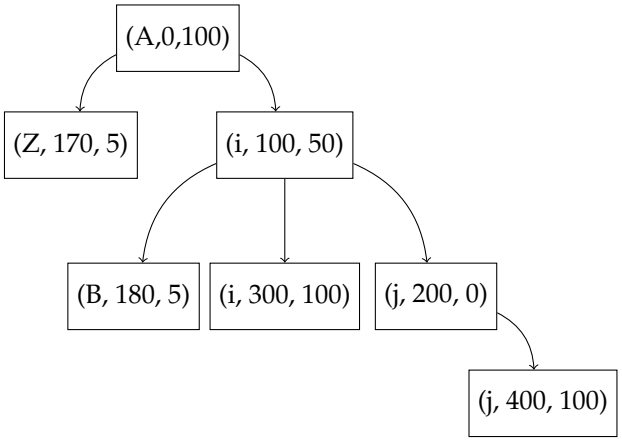


3.

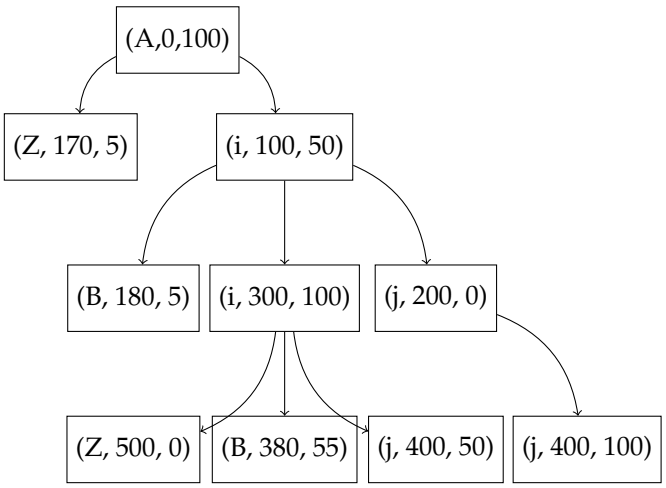


4. Die dritte und vierte Expansion von dem Knoten (Z,170,5) und (B,180,5) bewirken keine Veränderung des Baumes, da Sie keine Nachfolger haben. (Energie reicht nicht aus um zu einem anderen Knoten zu fahren)

5.



6.



e) Die klassische dynamische Programmierung würde, wenn man auf einen Knoten stößt, der schon einmal im Baum vorhanden ist (bei dem sich also das Auto an der gleichen Stelle befindet und die gleiche Ladung aufweist) immer den Knoten bevorzugen, dessen Pfad die geringeren akkumulierten Aktionskosten hat. Dies wird durch eine Tabelle garantiert, die beim Suchen im Speicher gehalten wird und für jeden Zustand (also jede Kombination aus Ladestand und Position) die geringsten Pfadkosten enthält, mit denen dieser Zustand bisher erreicht wurde. Normalerweise verhindert dies Zyklen und das unnötige mehrmalige Untersuchen von Pfaden, wenn diese nicht die optimale Lösung versprechen, da man davon ausgeht, dass die jeweils günstigste Teillösung auch zur günstigsten Gesamtlösung führt.
In unserem Fall werden trotzdem 'unnötige' Pfade untersucht, wenn man nicht beachtet, dass eine Position bei höheren Pfadkosten (also der Zustand unter Ausklammerung der Ladung) nur sinnvollerweise ein zweites Mal untersucht werden sollte, wenn die Ladung höher ist, als die beim letzten Besuch mit geringeren Pfadkosten. Dies könnte jedoch ohne Probleme und großen Aufwand als Constraint beim Benutzen der dynamischen Programmierung übernommen werden.

Aufgabe 2

Aufgabe 3

Aufgabe 4