Blatt: Search

Michel Bünger

Search.01: Problemformalisierung, Zustandsraum

Zustände:

- 3O 3E — 0O 0E (1. Seite: 3 Orks 3 Elben; 2. Seite 0 Orks 0 Elben)
- 2O 3E — 0O 0E (1. Seite: 2 Orks 3 Elben; 2. Seite 0 Orks 0 Elben)
- 2O 2E — 0O 1E (...)
- 1O 2E — 1O 1E
- 1O 1E — 1O 2E
- 0O 1E — 2O 2E
- 0O 0E — 2O 3E
- 0O 0E — 3O 3E

In den Zuständen ist zu beachten, dass aufgrund dessen, das das Pferd den Fluss nicht alleine überqueren kann, außer am Start- und Endzustand, immer mindestens ein Ork auf diesem.

Aktionen:

- 1O 1E
- 1O 0E
- 2O 0E

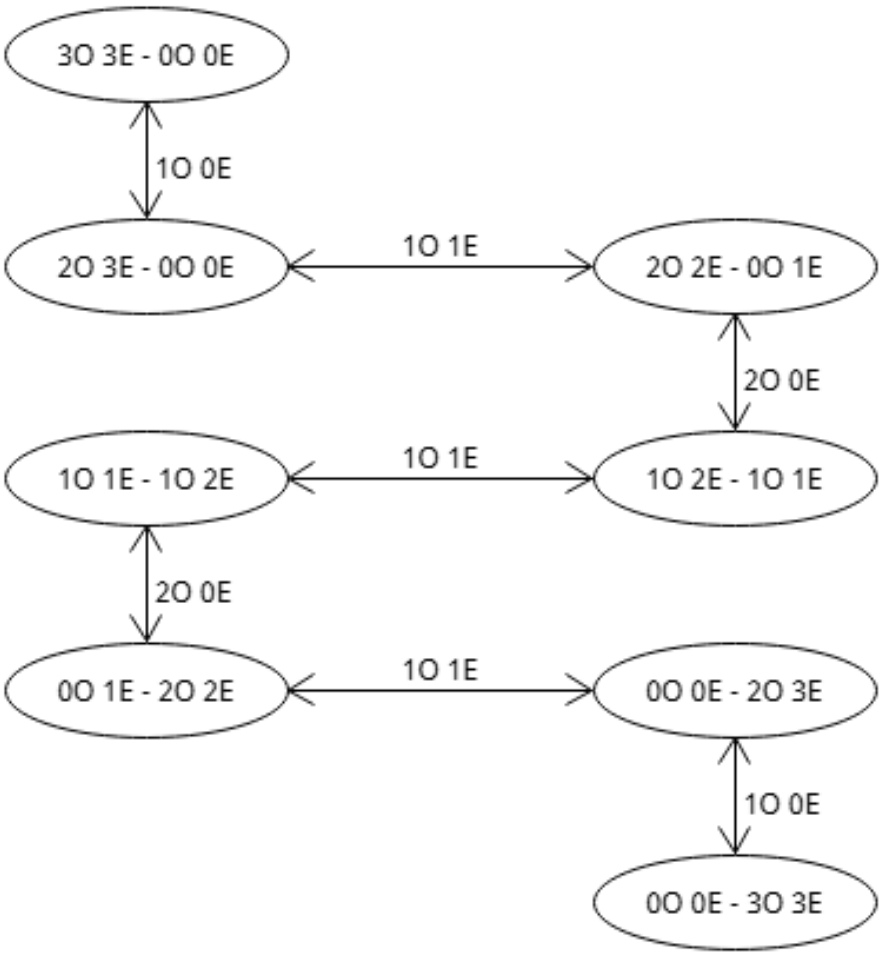
Wie zuvor erwähnt, ist hier nun zu sehen, dass jede Überquerung des Flusses mindestens einen Ork vorhanden ist.

Eine Aktion umfasst immer das Transportieren **einer** Person über den Fluss, d.h. das Pferd nimmt eine Person auf, läuft zum anderem Flussufer und setzt diese ab. Die einzige Anomalie heirbei ist die Aktion 1O 0E, welche lediglich dazu dient, den Ork, welcher das Pferd begleitet hat, ebenfalls abzusetzen bzw. aufzunehmen.

Start-/Endzustand

- Der Startzustand ist: **3O 3E — 0O 0E**.
- Der Endzustand ist: **0O 0E — 3O 3E**.

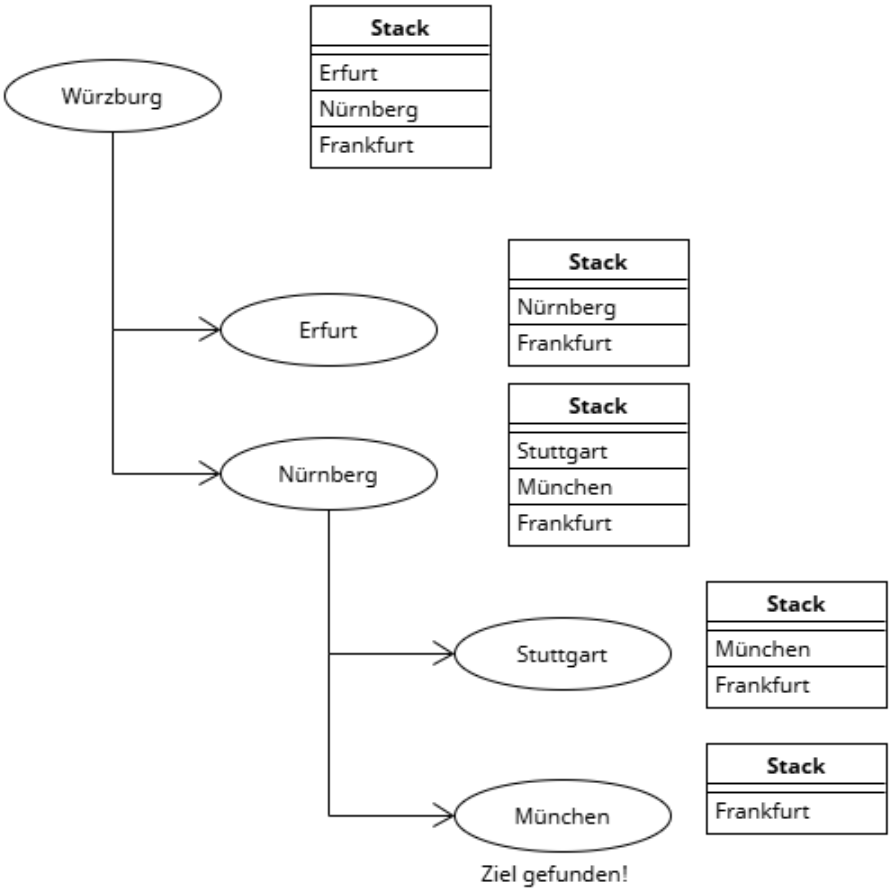
Problemgraph



Search.02: Suchverfahren

A)

Tiefensuche



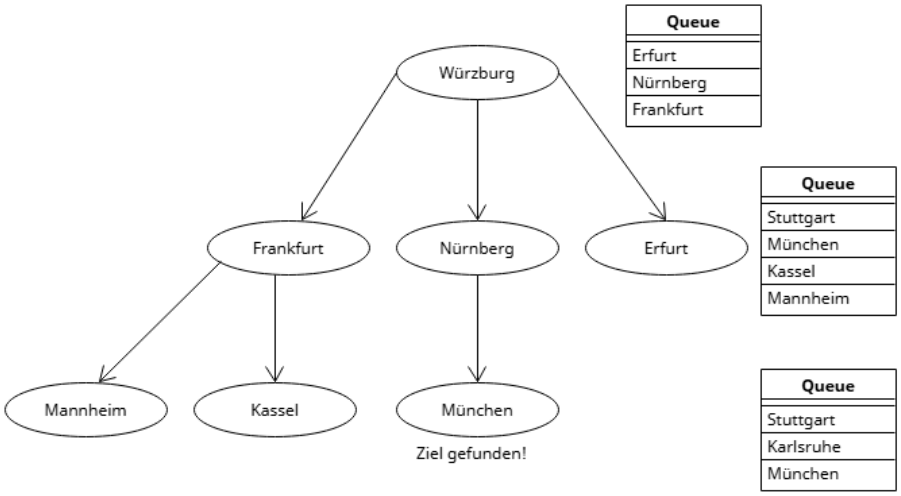
Die Tiefensuche hat in dem Oben gezeigtem Durchlauf das Ziel bereits nach vier Durchläufen der Hauptschleife gefunden.

Jedoch ist das durchlaufen des Graphen sehr stark davon abhängig, in welcher Reihenfolge der Suchalgorithmus die Nachbarknoten in den Stack legt. Beispielsweise, falls das oberste Objekt im Stach nicht *Erfurt*, sonders *Nürnberg* gewesen werde, hätte der Weg bereits nach einer bzw. zwei Durchläufen gefunden werden können.

Im *worst-case* hätte sich ein sehr langer Pfad, von *Erfurt*, nach *Frankfurt* und dann über *Mannheim* gefunden und das Ziel wäre erst nach Sechs Durchläufen

gefunden worden.

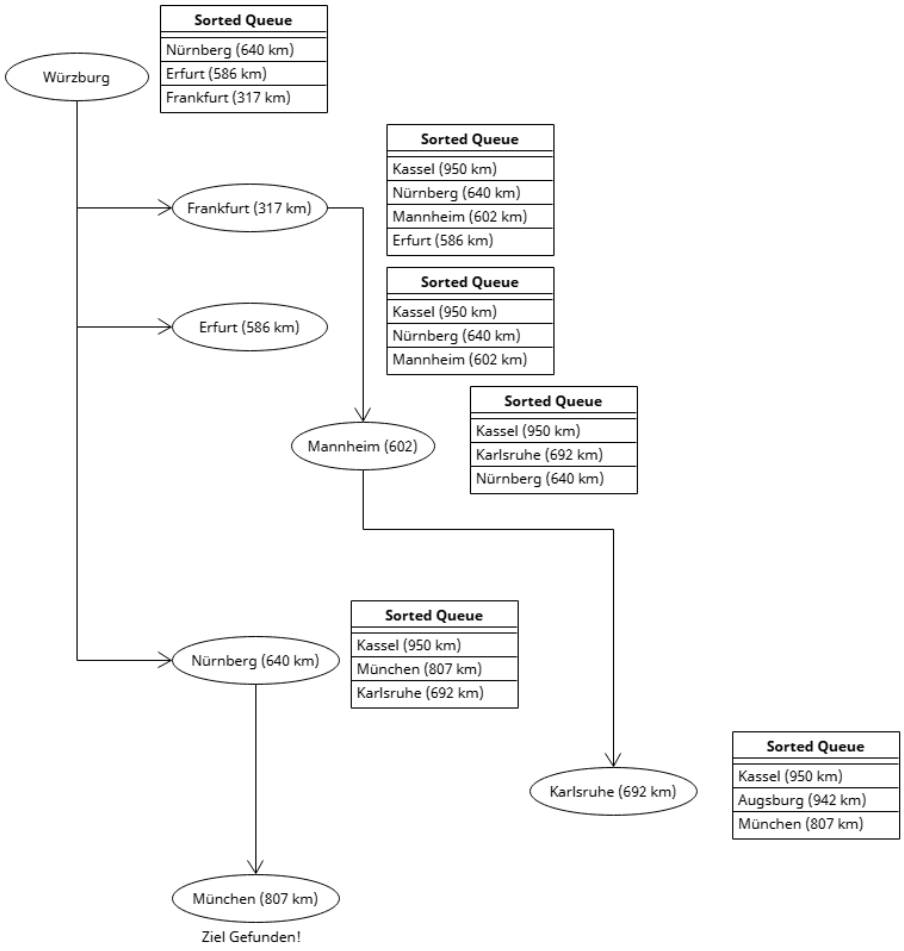
Breitensuche



In dem oben gezeigtem Durchlauf wurde die Hauptschleife sechs mal Durchlaufen, bis ein Pfad nach München gefunden wurde.

Das Durchlaufen mithilfe einer Breitensuche erzeugt hier im Vergleich mit der Tiefensuche einen Durchschnittlich längeren Durchsuchungszyklus, weil unabhängig von der Einspeicherung der Knoten in der Queue, immer mindestens vier Durchläufe gebraucht werden, bis die Ebene des Suchbaums erreicht wird, welche *München* beinhaltet. Auf dieser Ebene ist es nun von der Reihenfolge in der Queue abhängig, ob *München* nach (**best-case**) vier Durchläufen bis zu (**worst-case**) sieben Durchläufen gefunden wird.

A*-Suche



Die oben gezeigte Darstellung zeigt die besuchten Knoten chronologisch von oben nach unten.

Der A* Suchalgorithmus findet das Ziel nach sechs Durchläufen der Hauptschleife. Hier gibt es jedoch einen Fundamentalen Unterschied zu den vorherigen beiden Suchalgorithmen, dieser liegt darin, dass der A* Suchalgorithmus ein eine Informierte Suche durchführt.

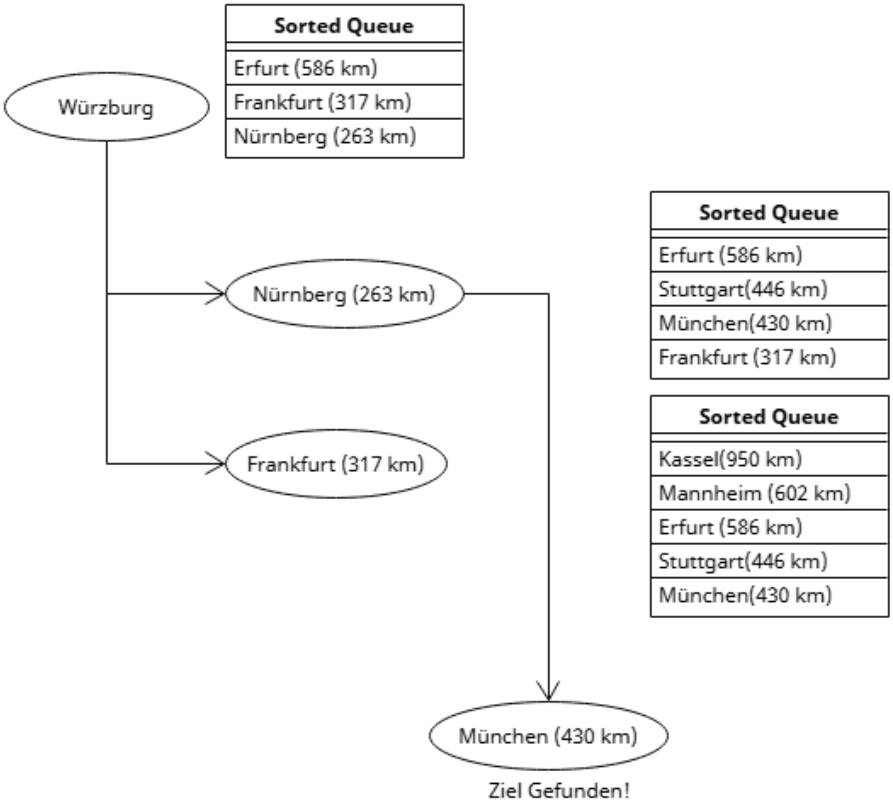
Der A* Suchalgorithmus wird bei diesem Graphen unabhängig von Reihenfolge der eingegebenen Nachbarknoten, immer den gleichen Pfad finden, aufgrund der **Sorted Queue**. Zudem funktioniert dieser basierend auf der Heuristik der Knoten, weshalb er ebenfalls den Pfad mit den Niedrigsten Kosten zum Zielknoten finden wird, im Vergleich zur Tiefen-/ und Breitensuche, welche

lediglich den ersten gefundenen Pfad, unabhängig von kosten und Distanz ausgeben.

B)

Die gegebenen Restkostenabschätzungen dürften nicht gegeben werden, weil die Restkosten, von Nürnberg, nach München um mehrere hundert mm überschätzt werden. Damit die Werte zulässig wären, müssten die Geschätzten Restkosten von Nürnberg nach München, höchstens 167km betragen.

In dem Folgendem Durchlauf werden die geschätzten Restkosten von Nürnberg nach München 160km betragen.



Search.03: Dominanz

Eine Heuristik h_1 dominiert eine Heuristik h_2, falls alle Knoten von h_1 einen höheren Wert haben, als die in h_2, jedoch immernoch zulässig sind. Dies bedeutet, dass die Heuristik h_1 mit ihren Werten näher an den Tatsächlichen Abständen ist, was dazu führt, dass der Algorithmus eine Realitätsnäheres Ergebnis schneller erstellen kann, im Vergleich mit der Durchführung von h_2

Kurz gesagt, wenn eine Heuristik h_1 eine Heuristik h_2 dominiert, erzeugt die Durchführung von dem A* Suchalgorithmus mit h_1 akkuratere Ergebnisse schneller, im Vergleich zu der Durchführung mit h_2.

Search.04: Beweis der Optimalität von A*

Wenn eine Heuristik Zulässig ist, wird der A* Suchalgorithmus immer den Optimalen Pfad zurückgeben, aufgrund der **Sorted Queue**, welche es Garantiert, dass immer der Kürzeste Weg getestet wird, d.h., dass das erste mal, wenn ein beliebiger Knoten gefunden wird, dies der kürzeste Pfad zu diesem Knoten, vom Startknoten ist, einschließlich dem Zielknoten.