Übung 3

Niclas Kusenbach, 360227

November 21, 2025

Aufgabe 1) Informierte Suchalgorithmen: Einordnung

a)Unterschied informierter Suchalgorithmen zu uninformierten Suchalgorithmen

Uninformierte Suchalgorithmen haben keine Information über das Problem ausser die Problemdefinition. Dahingegen haben informierte Suchalgorithmen extra Wissen über das Problem und etwa eine grobe Richtung wo sie die Lösung dessen finden.

b) Beispiele für Reale Probleme, bei denen informierte Suchstrategien effektiv angewandt werden

- **Routen- und Navigationsplanung:** Verwendung von A*-ähnlichen Algorithmen zur Berechnung kürzester Wege, z. B. in GPS- oder Logistiksystemen.
- **Pfadfindung in Videospielen:** Einsatz heuristischer Suchalgorithmen (typisch A*) zur effizienten Wegfindung von Spielfiguren in 2D- und 3D-Umgebungen.
- **Autonome Roboternavigation:** Nutzung informierter Suche (A*, D*) zur Planung kollisionsfreier und optimaler Bewegungswege in dynamischen Umgebungen.

Aufgabe 2) Informierte Suchalgorithmen

a) Von Lugoj nach Bukarest

Im Folgenden ist der A*-Ablauf einmal recht natürlich und ohne allzu technische Formulierungen beschrieben — aber natürlich weiterhin korrekt und vollständig. Die Bewertungsfunktion von A* lautet:

$$f(n) = g(n) + h(n)$$

- $g(n)$: tatsächliche Kosten vom Start bis zum aktuellen Knoten

- $h(n)$: geschätzte Kosten bis zum Ziel (hier: Luftlinie)

Da wir eine *Baumsuche* durchführen, merken wir uns bereits besuchte Knoten nicht in einer Closed List. Dadurch können wir immer wieder zu bereits besuchten Orten zurückkehren.

Start in Lugoj

- $f(\text{Lugoj}) = 0 + 244 = 244$

Expansion von Lugoj:

- **Mehadia**: $g = 70,\ h = 241 \Rightarrow f = 311$
- **Timisoara**: $g = 111,\ h = 329 \Rightarrow f = 440$

fringe: Mehadia (311), Timisoara (440)

Mehadia auswählen ($f = 311$)

Expansion von Mehadia:

- **Lugoj (zurück)**: $g = 140,\ h = 244 \Rightarrow f = 384$
- **Drobeta**: $g = 145,\ h = 242 \Rightarrow f = 387$

fringe: Lugoj (Rückweg) (384), Drobeta (387), Timisoara (440)

Lugoj (Rückweg) wählen ($f = 384$)

Da wir keine Closed List verwenden, gilt schlicht: kleineres f gewinnt, also wird auch der Rückweg expandiert. Die daraus entstehenden Kinder (erneut Mehadia, Timisoara usw.) haben jedoch nur schlechtere Pfadkosten und sind für den optimalen Weg unbedeutend. Wir lassen sie daher in der Darstellung weg. *fringe (relevant):* Drobeta (387), Timisoara (440), ...

Drobeta auswählen ($f = 387$)

Expansion von Drobeta:

- **Craiova**: $g = 265,\ h = 160 \Rightarrow f = 425$
- **Mehadia (zurück)**: $g = 220,\ h = 241 \Rightarrow f = 461$

fringe: Craiova (425), Timisoara (440), ...

Craiova auswählen ($f = 425$)

Expansion von Craiova:

- **Pitesti**: $g = 403,\ h = 100 \Rightarrow f = 503$
- **Rimnicu Vilcea**: $g = 411,\ h = 193 \Rightarrow f = 604$

fringe: Timisoara (440), Pitesti (503), ...

Timisoara auswählen ($f = 440$)

Timisoara stammt noch aus dem ersten Schritt. Die Expansion führt letztlich nur zu Arad und Lugoj – beide wiederum mit höheren Kosten. Sie sind für den optimalen Weg nicht weiter relevant. *fringe:* Pitesti (503), ...

Pitesti auswählen ($f = 503$)

Expansion von Pitesti:

- **Bukarest**: $g = 504,\ h = 0 \Rightarrow f = 504$

Unterschied zur Graphsuche

Bei der Graphsuche wird eine *Closed List* geführt. Das führt zu folgenden Änderungen:

- Bereits expandierte Knoten werden nicht erneut zur fringe hinzugefügt.
- Dadurch fällt z. B. der Rückweg nach **Lugoj** in Schritt 2 komplett weg.
- Auch der Rückweg von Drobeta nach **Mehadia** wird verworfen.
- Insgesamt bleibt der Suchraum deutlich kleiner und enthält keine Zyklen.

b)

Ja es gibt viele Städte, die in einen unendlichen loop kommen würden durch die Greedy Best-First Search. Hier ein paar Bsp.: Craiova, Lugoj, Neamt, Eforie, etc.

c)

Arad, Sibiu, Oradea, Zerind, Timisoara

Heuristiken

a)

Ein zug bewegt nur eine einzige Kachel. Für diese Kachel gibt es drei mögliche Situationen:

1. falsch $\rightarrow$ richtig: -1
2. richtig $\rightarrow$ falsch: +1
3. falsch $\rightarrow$ falsch: 0

Da sich pro Zug nur eine Kachel verändert, kann sich h_{MIS} ebenfalls nur ± 1 ändern (Zugkosten = 1). Damit gilt:

$$h_{MIS}(n) - h_{MIS}(n') \leq 1$$

Umgestellt also

$$h_{MIS}(n) \leq 1 + h_{MIS}(n')$$

Da sich h_{MIS} in einem Schritt höchstens um 1 ändern kann und die Schrittkosten 1 sind, folgt dass h_{MIS} konsistent ist.

Wenn eine Kachel verschoben wird, ändert sich ihre Manhattan Distanz um höchstens 1 (± 1)

Die gesamte Manhattan Heuristik verändert sich höchstens um 1:

$$h_{MAN}(n) - h_{MAN}(n') \leq 1$$

Da die Schrittkosten ebenfalls 1 betragen, folgt:

$$h_{MAN}(n) \leq 1 + h_{MAN}(n')$$

Also ist h_{MAN} konsistent.

b)

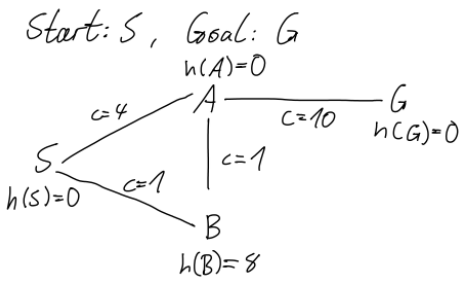
Sei h_1 und h_2 admissible Heuristiken, d.h. für alle Knoten n gilt $h_1(n) \leq h^*(n)$ und $h_2(n) \leq h^*(n)$. Definiere $h(n) = max h_1(n), h_2(n)$. Da sowohl $h_1(n)$ als auch $h_2(n)$ höchstens $h^*(n)$ sind, kann ihr Maximum $h(n)$ ebenfalls nicht größer als $h^*(n)$ sein:

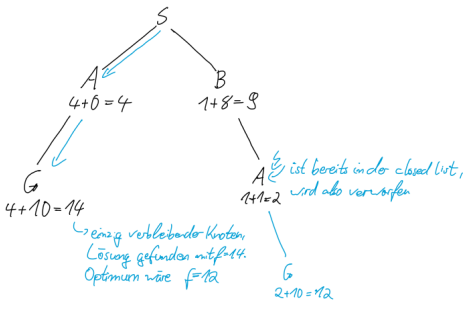
$$h(n) = max h_1(n), h_2(n) \leq h^*(n)$$

Also ist h admissible.

c)

Prüfung auf Inkonsistenz: Betrachten wir Kante $B \to A$. $h(B) \leq c(B, A) + h(A)$ müsste gelten. Aber: $8 \leq 1 + 0$ ist falsch. Die Heuristik fällt von B nach A zu stark ab.





d)

Wenn h_2 und h_1 beide zulässig sind und $h_1(n) \leq h_2(n)$ für alle n, dann wird h_2 von h_1 dominiert. Das bedeutet, dass h_1 näher an den wahren Kosten (h^*)ist als h_2.

$h_{\mathbf{MIS}}$ **(Misplaced Tiles):** Zählt die Anzahl der Steine, die nicht an ihrer Zielposition sind.

$h_{\mathbf{MAN}}$ **(Manhattan Distance):** Summiert für jeden Stein die horizontale und vertikale Distanz zu seiner Zielposition.

Dominanz:

- Wenn ein Stein an der falschen Stelle liegt, trägt er zu h_{MIS} genau 1 bei.
- Wenn ein Stein an der falschen Stelle liegt, muss er mindestens 1 Feld bewegt werden,um zum Ziel zu kommen. Seine Manhattan Distanz ist also immer ≤ 1. Oft ist sie sogar größer (2 oder 3 Schritte entfernt).
- Daraus folgt: Für jeden Zustand n ist die Summe der Manhattan Distanzen größer oder gleich der Anzahl der falsch platzierten Steine.

$$h_{MAN}(n) \leq h_{MIS}(n)$$

Effizienz:

- A* expandiert alle Knoten mit $f(n) < C^*$ (optimale Kosten).
- Da $f(n) = g(n) + h(n)$, bedeutet ein größeres $h(n)$, dass der f-Wert schneller wächst.
- Knoten überschreiten den Grenzwert C^* schneller und werden nicht expandiert (der Suchbaum wird "pruned" bzw. beschnitten).
- Fazit: Da h_{MAN} dominiert, wird A* mit h_{MAN} garantiert weniger oder gleich viele Knoten expandieren als mit h_{MIS}. Die Suche ist mit der Manhattan-Distanz also deutlich effizienter und schneller.