Hochschule Flensburg
University of Applied Sciences

Einführung in die Künstliche Intelligenz

WS 23/24

Prof. Dr. Annina Neumann

Vorlesung 3: Thema und Lernziele

- Programmierung eines intelligenten Routenfinders (mittels genetischer Algorithmen – Fortsetzung)
- Breitensuche (Breadth-First-Search): Algorithmus und Anwendung
- Gegenüberstellung mit Bestensuche (Best-First-Search) und implementieren

Such-algorithmen

- Ein Suchalgorithmus verwendet ein Suchproblem als Eingabe und gibt eine Lösung zurück.
- Heute betrachten wir Algorithmen, die einen Suchbaum über den Zustandsraumgraphen legen, verschiedene Pfade aus dem Anfangszustand bilden und versuchen, einen Pfad zu finden, der einen Zielzustand erreicht.
- Jeder Knoten im Suchbaum entspricht einem Zustand im Zustandsraum und die Kantenknoten im Suchbaum entsprechen Aktionen. Die Wurzel des Baums entspricht dem Anfangszustand des Problems.

Wie traversieren wir nun am besten diesen Graphen?

- Beispielproblem: Finde Route von Arad nach Bukarest

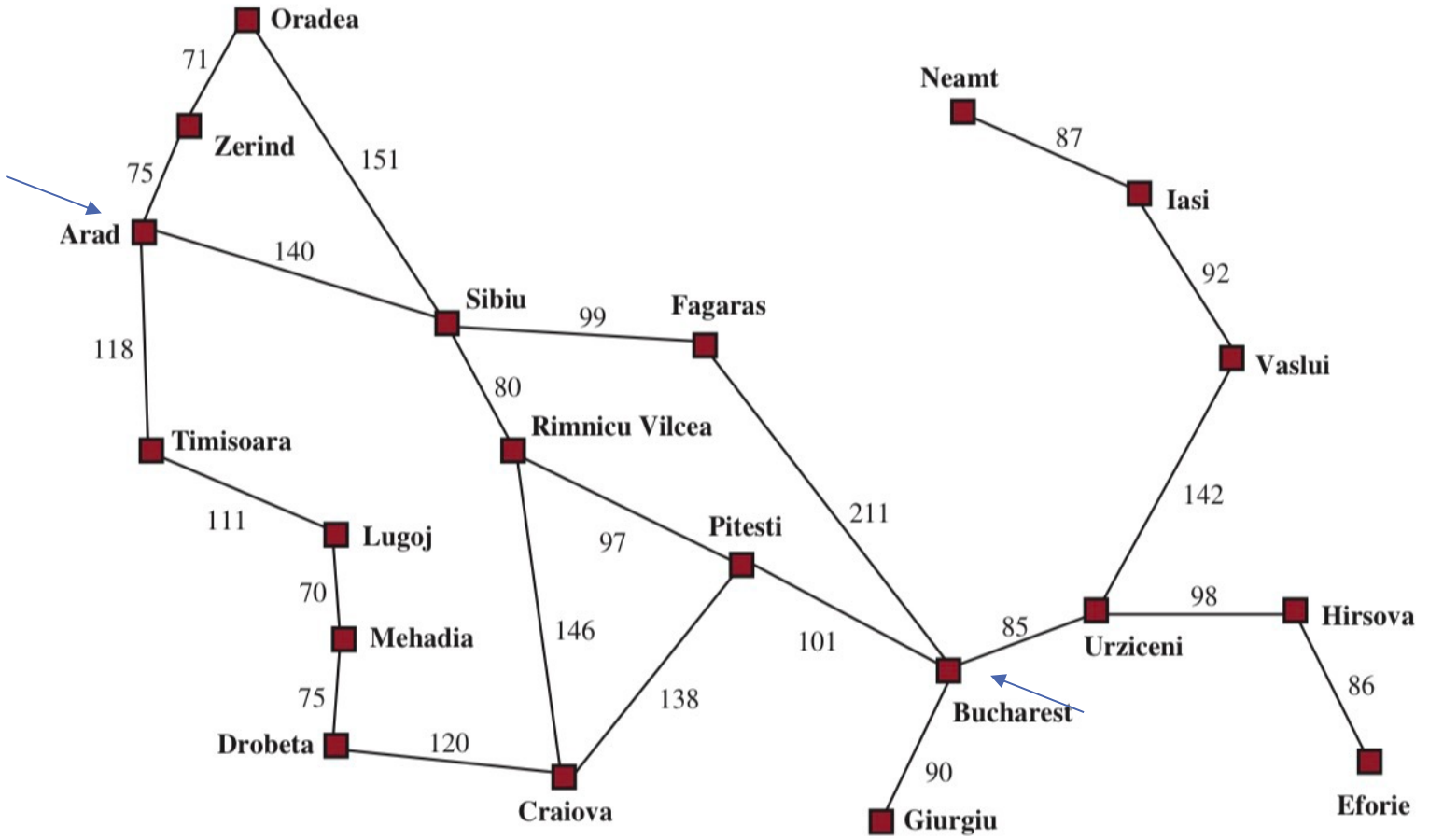


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

Breitensuche

- Nehmen wir an, dass alle Aktionen die gleichen Kosten haben
- In so einem Fall ist die Breitensuche eine geeignete Strategie, um den kürzesten Weg zu einem Ziel zu finden
  - Zuerst wird der Wurzelknoten erweitert, dann alle Nachfolger des Wurzelknotens, dann ihre Nachfolger und so weiter (systematische Suchstrategie).
  - Die Breitensuche findet immer eine Lösung mit einer minimalen Anzahl von Aktionen:
    - Wenn Knoten in der Tiefe d generiert werden, wurden bereits alle Knoten in der Tiefe d −1 generiert. Wenn also einer von ihnen eine Lösung wäre, hätte er dies gefunden.

Suchbaum

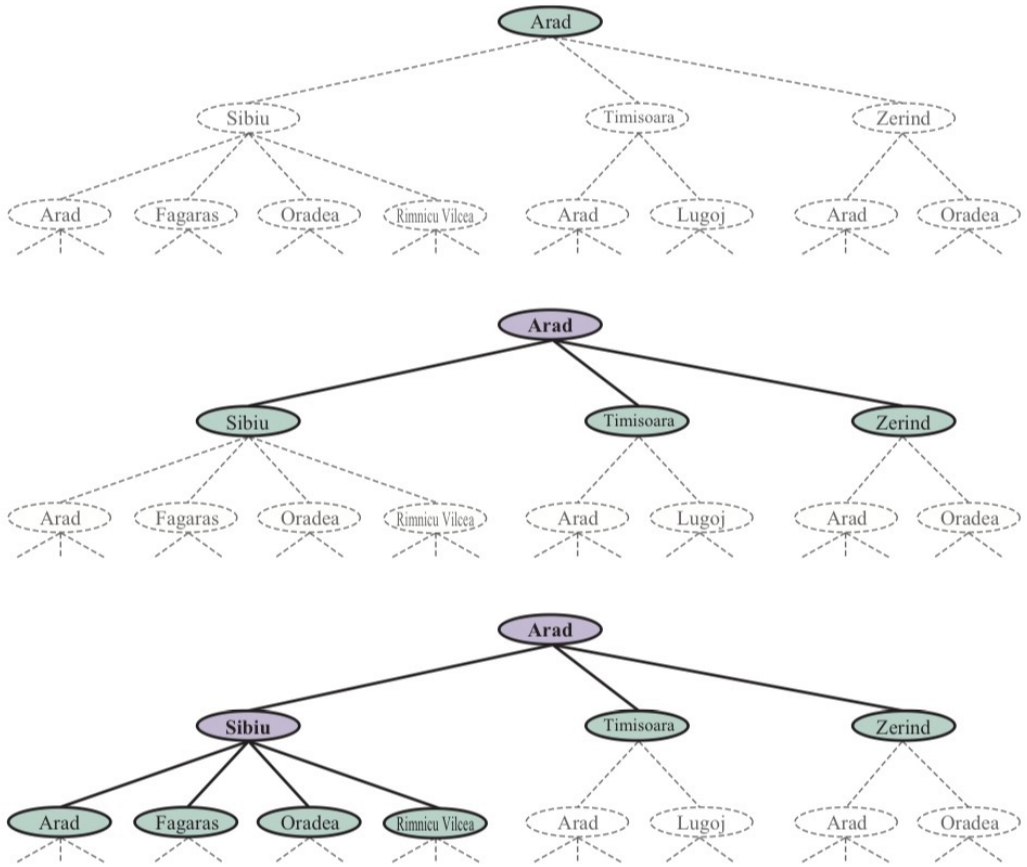


Figure 3.4 Three partial search trees for finding a route from Arad to Bucharest. Nodes that have been *expanded* are lavender with bold letters; nodes on the frontier that have been *generated* but not yet expanded are in green; the set of states corresponding to these two types of nodes are said to have been *reached*. Nodes that could be generated next are shown in faint dashed lines. Notice in the bottom tree there is a cycle from Arad to Sibiu to Arad; that can't be an optimal path, so search should not continue from there.

Artificial Intelligence: A Modern Approach, Global Edition by Stuart Russell, Peter Norvig

Breitensuche in einem Graphen

- Gegeben ist ein Labyrinth, etwa wie in Bild 1 dargestellt. Das Labyrinth hat ein Eingangsfeld und ein von dort erreichbares Ausgangsfeld; gesucht ist ein Weg vom Eingang des Labyrinths bis zum Ausgang.

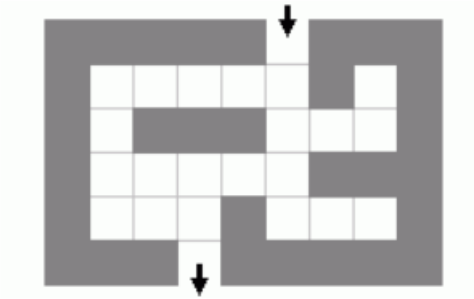


Bild 1: Labyrinth mit Eingang und Ausgang

- Mithilfe der Breitensuche können wir den kürzesten Weg vom Eingang zum Ausgang finden (sofern wir einen globalen Überblick über das Labyrinth haben)

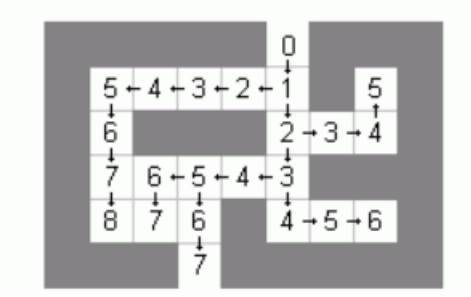


Bild 2: Kürzester Weg durch das Labyrinth

SIEHE SEITEN VON H.W. LANG: HTTPS://WWW.INF.HS-FLENSBURG.DE/LANG/ALGORITHMEN/GRAPH/BREADTH-FIRST-TREE.HTM

Breitensuche in einem Graphen

/*Finde den kürzesten Weg vom Eingang eines Labyrinths zum Ausgang*/

Eingabe: Ein Labyrinth wie in Bild 1 dargestellt
Ausgabe: Kürzester Weg vom Eingang des Labyrinths bis zum Ausgang

setze i=0
markiere das Eingangsfeld mit der Zahl 0
solange nicht alle Felder markiert sind wiederhole
    für jedes mit der Zahl i markierte Feld f wiederhole
        verbinde f mit allen noch nicht markierten Nachbarfeldern
        und markiere diese mit i+1
    setze i=i+1
gib die Folge der Felder entlang der Verbindungen vom Ausgang zum Eingang in umgekehrter Reihenfolge aus

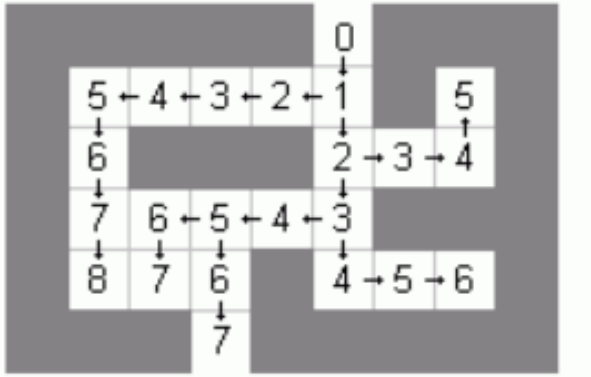


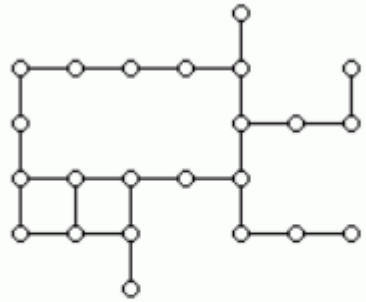
Bild 2: Kürzester Weg durch das Labyrinth

SIEHE SEITEN VON H.W. LANG: HTTPS://WWW.INF.HS-FLENSBURG.DE/LANG/ALGORITHMEN/GRAPH/BREADTH-FIRST-TREE.HTM

Breitensuche in einem Graphen

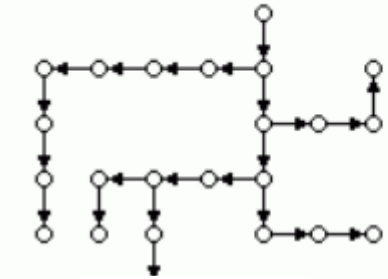
/*Algorithmus Breitensuche*/
Eingabe: Zusammenhängender ungerichteter Graph G = (V, E) mit V = {0, ..., n-1}, Startknoten s
Ausgabe: Kürzeste Wege vom Startknoten s zu allen Knoten

setze distance(s) = 0
setze predecessor(s) = -1
markiere s als besucht
hänge Knoten s an die Schlange an
solange die Schlange nicht leer ist wiederhole
  entnimm den vordersten Knoten u aus der Schlange
  für alle Nachbarknoten v von u wiederhole
    wenn v noch nicht als besucht markiert ist dann
      setze distance(v) = distance(u)+1
      setze predecessor(v) = u
      markiere v als besucht
      hänge Knoten v an die Schlange an



Modellierung des Labyrinths

durch einen ungerichteten Graphen



Breitensuchbaum

SIEHE SEITEN VON H.W. LANG: HTTPS://WWW.INF.HS-FLENSBURG.DE/LANG/ALGORITHMEN/GRAPH/BREADTH-FIRST-TREE.HTM

Breitensuche in einem Graphen

- Der Algorithmus beschreibt die sogenannte Breitensuche (englisch: Breadth-First Search, BFS)
- Sie wird zur Traversierung oder Durchsuchung von Graphen eingesetzt. Unser Labyrinth wird also als Graph modelliert.
- Die Breitensuche arbeitet schrittweise und besucht alle Knoten eines Graphen in der Reihenfolge ihres Abstands zur Startknoten:
  - Der Algorithmus beginnt bei einem Startknoten und besucht zuerst alle direkten Nachbarn dieses Knotens.
  - Anschließend werden die Nachbarn der Nachbarn besucht, dann die Nachbarn der Nachbarn der Nachbarn usw., bis alle erreichbaren Knoten im Graphen besucht wurden oder das gewünschte Ziel erreicht ist.

Die Breitensuche verwendet eine **Warteschlange (Queue)**, um die Reihenfolge der zu besuchenden Knoten zu verwalten. Der Startknoten wird in die Warteschlange eingefügt, und dann werden seine direkten Nachbarn in die Warteschlange eingereiht. Dieser Prozess wird fortgesetzt, bis die Warteschlange leer ist oder das Ziel erreicht wurde.

Aufgabe

15min

Beschreiben Sie den Durchlauf einer Breitensuche mit Startpunkt Fagaras und erstellen Sie den Breitensuchbaum. Sie brauchen nur bis zu einer Distanz von 3 zu gehen!

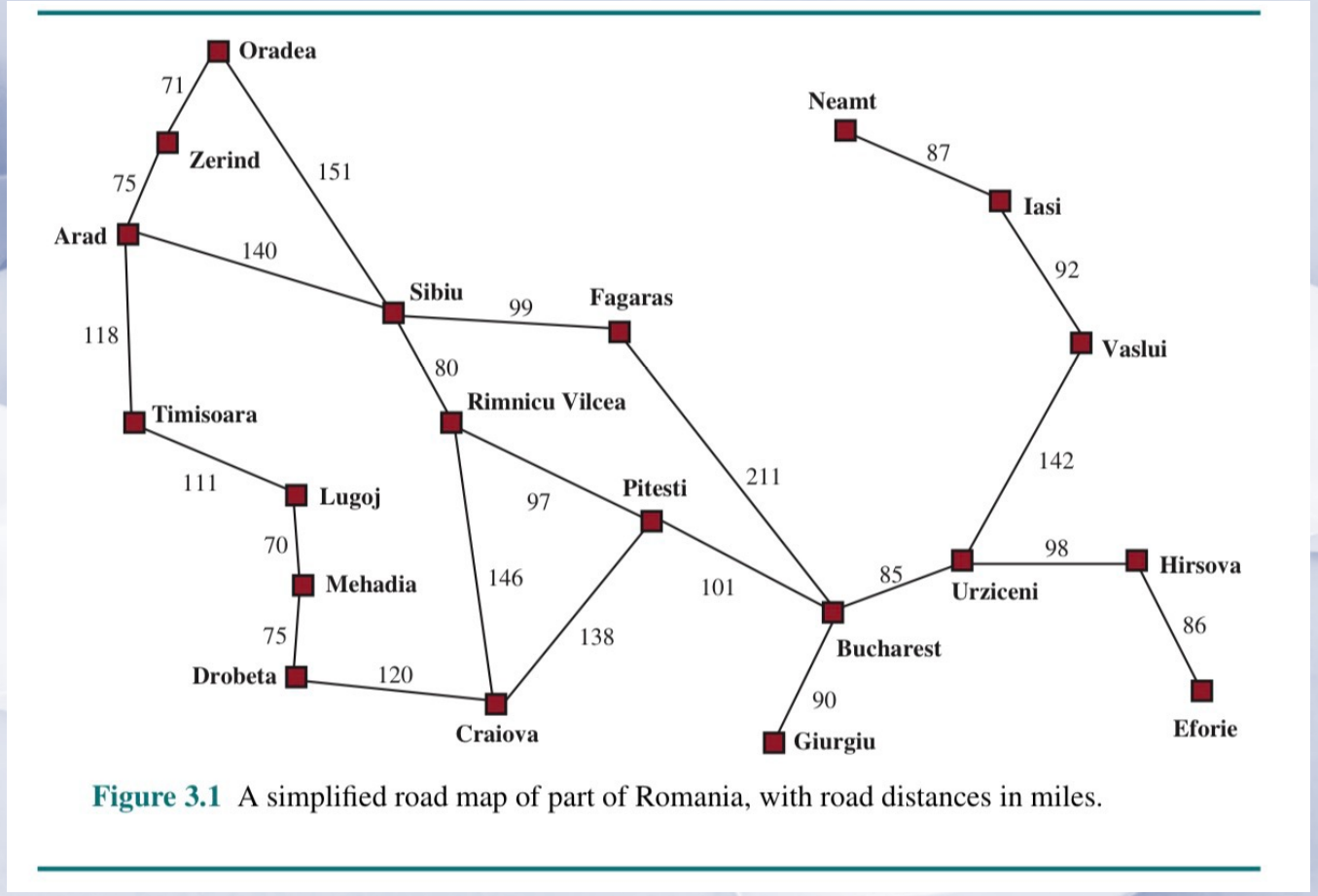


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

Wie kann der Agent weitere Informationen nutzen?

- Ein uninformierter Suchalgorithmus erhält keinen Hinweis darauf, wie nah ein Status an den Zielen liegt.
- z.B. Agent in Arad mit Ziel, Bukarest zu erreichen.
  - Ein uninformierter Agent ohne Kenntnisse der rumänischen Geographie hat keine Ahnung, ob es der bessere erste Schritt ist, nach Zerind oder Sibiu zu gehen.
  - Im Gegensatz dazu weiß ein informierter Agent, der die Lage jeder Stadt kennt, dass Sibiu viel näher an Bukarest liegt und daher eher auf dem kürzesten Weg liegt.
- Wir können also alternativ einen besser informierten Agenten programmieren.

Uniform-Cost-Search

- Wenn Aktionen unterschiedliche Kosten haben, ist die Verwendung der Best-First-Suche eine naheliegende Wahl, bei der die Bewertungsfunktion die Kosten der unterschiedlichen Pfade einbezieht.
- Dies wird von der theoretischen Informatik-Community als Dijkstra-Algorithmus und von der KI-Community als Uniform-Cost-Search bezeichnet.

Best-First-Search

- **Knoten werden nach einer Bewertungsfunktion ausgewählt**
- Bei jeder Iteration wählen wir den Knoten aus der Liste mit minimalen Wert einer Auswertungsfunktion.
- Wenn sein Zustand ein Zielzustand ist, sind wir fertig. Andernfalls expandieren wir weiter zu untergeordneten Knoten
- Jeder untergeordnete Knoten wird der Liste hinzugefügt, (wenn er zuvor noch nicht erreicht wurde). Kann auch erneut hinzugefügt werden, wenn er jetzt über einen Pfad erreicht wird, der niedrigere Pfadkosten als alle vorherigen Pfade hat.

Best-First-Search zum Finden der kürzesten Route

1. Starte am Anfangspunkt und setze ihn als den aktuellen Standpunkt.
2. Überlege, ob der aktuelle Standpunkt das Ziel ist. Wenn ja, beende den Algorithmus.
3. Andernfalls, schaue dir die benachbarten Orte (Knoten) vom aktuellen Standpunkt an.
4. Bewerte diese benachbarten Orte anhand einer Schätzfunktion (Heuristik), die sagt, wie vielversprechend sie sind, um zum Ziel zu führen. (z.B. Distanz zum Ziel)
5. Wähle den am vielversprechendsten bewerteten Ort aus den benachbarten Orten aus und setze ihn als den neuen aktuellen Standpunkt.
6. Wiederhole die Schritte 2-5, bis du das Ziel erreichst oder feststellst, dass kein Pfad zum Ziel existiert.

Aufgabe

15min

Beschreiben Sie den Durchlauf einer Bestensuche mit Startpunkt Fagaras mit Ziel Eforie. Nutzen Sie die geschätzte Distanz zum Ziel als Bewertung für einen Knoten.

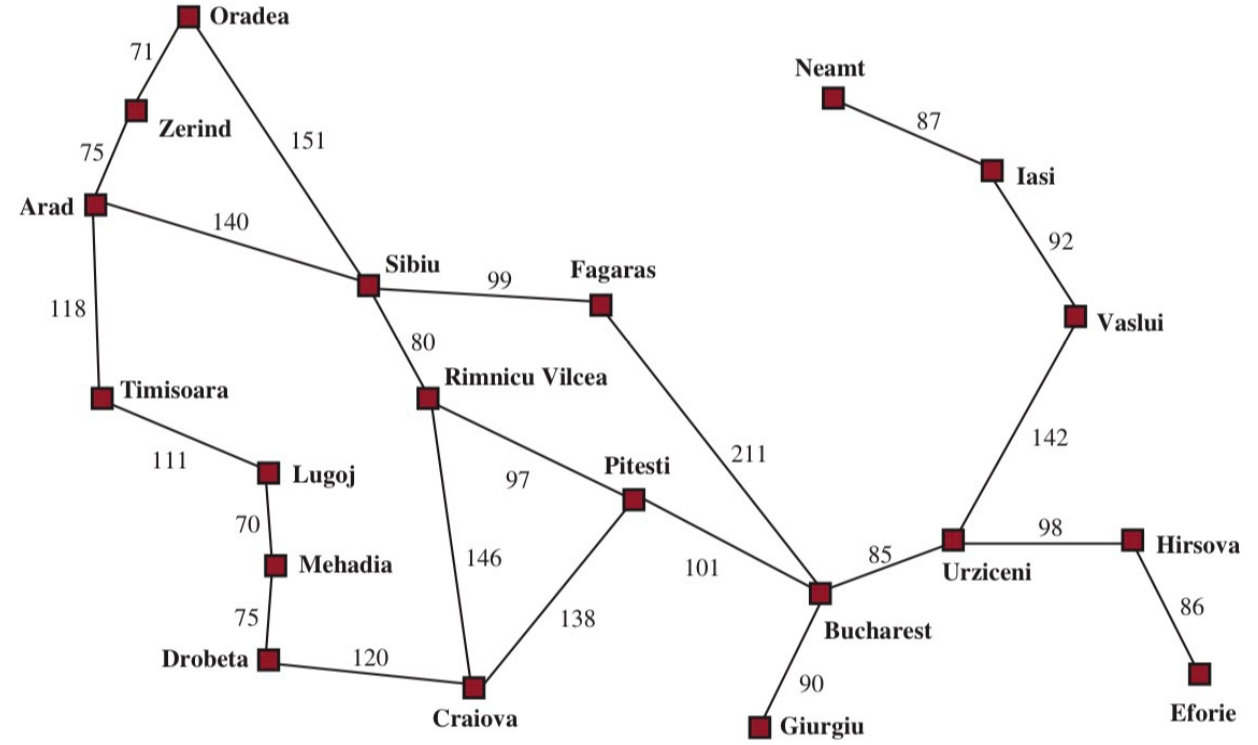


Figure 3.1 A simplified road map of part of Romania, with road distances in miles.

Breadth-First-Search vs. Best-First-Search

Breitensuche

- Priorisiert die Untersuchung aller Knoten auf einer Ebene, bevor es zu den nächsten Ebenen übergeht.
- Verwendet eine Queue, um die Reihenfolge der Erkundung der Knoten zu steuern.
- Nutzt Nachbarschaftsbeziehungen, keine sonstigen Informationen über die Struktur des Graphen oder Bewertungen der Knoten.
- Garantiert optimal, wenn alle Kantengewichte gleich sind, da sie den kürzesten Weg zuerst findet.

Best-First-Search

- Priorisiert die Erkundung von Knoten, die als vielversprechender für die Lösung angesehen werden.
- Verwendet eine Prioritätsliste, um die Reihenfolge der Erkundung der Knoten zu steuern.
- Erfordert meistens eine heuristische Funktion, die die Qualität der Knotenbewertung bestimmt.
- Optimiertheit hängt von der Qualität der heuristischen Funktion ab.

Aufgaben

1. Implementieren Sie eine Breitensuche, sodass ihr Programm jederzeit vom aktuellen Ort zu einem beliebigen Zielort den direktesten* Weg ausgeben kann.

2. Implementieren Sie eine Bestensuche, sodass ihr Programm jederzeit vom aktuellen Ort zu einem beliebigen Zielort den (geschätzten) kürzesten** Weg ausgeben kann.

*möglichst wenig Zwischenstopps

**gemessen an Distanz