

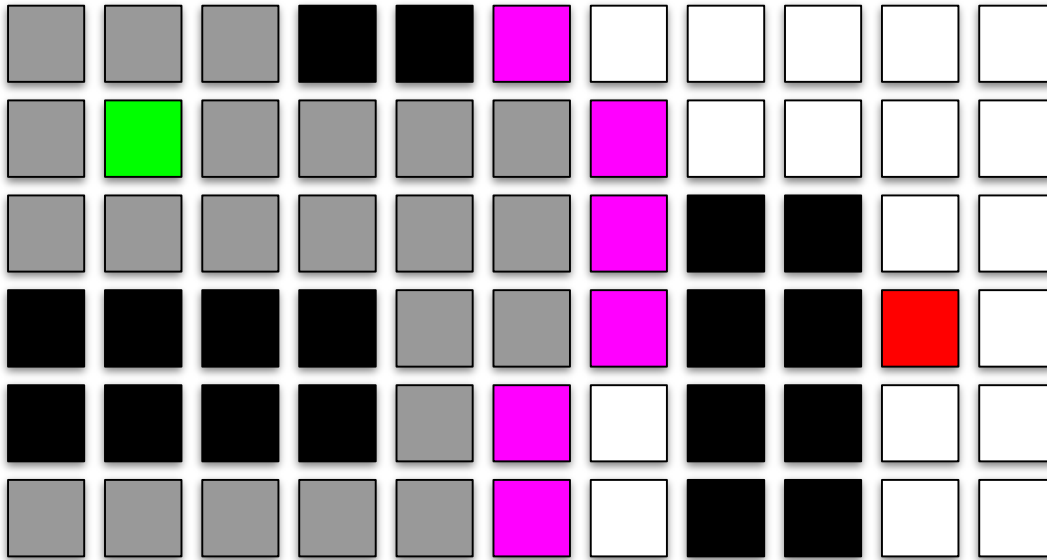
Wegfindung








Breitensuche
Dijkstra,
Best First Search, A*

1. Breitensuche (Breadth First Search)

- Queue: FIFO
- Findet eine Lösung, solange Ziel Knoten erreichbar
(bspw. Perfekte Labyrinth)

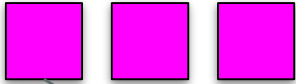
Bezeichnungen



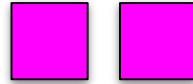
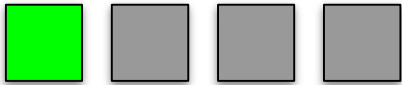
-  Start Node
-  End Node
-  Besucht
-  Blockiert
-  Unbesucht
-  Aktiv
-  Pfad, Ergebnis

Propagieren aktiver Knoten

Queue



Bereits besucht



- Merke Vorgängerknoten
- Prüfe, ob Ziel erreicht

- Verschiebe nach bereits besucht
- Suche nächste Nachbarn
- Setze nächste Nachbarn in Queue

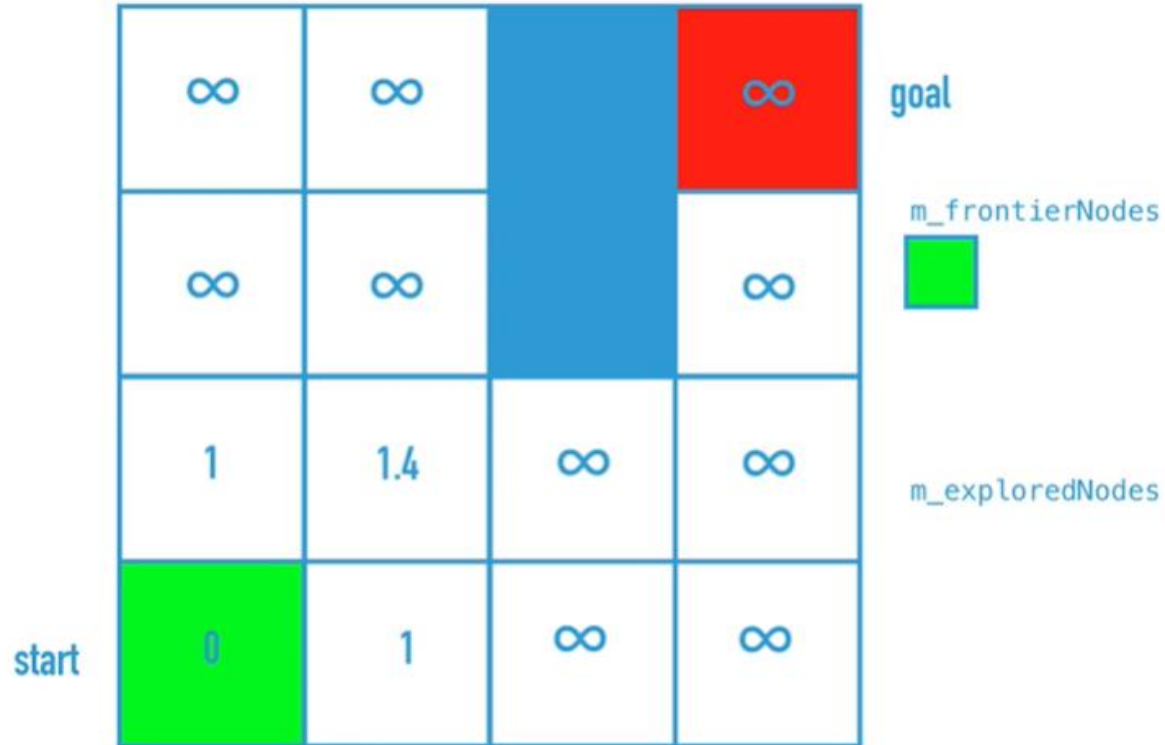
Fertig!

Fragen

Welche Verbesserungen sind denkbar?

Welche weiteren Anforderungen sollten berücksichtigt werden?

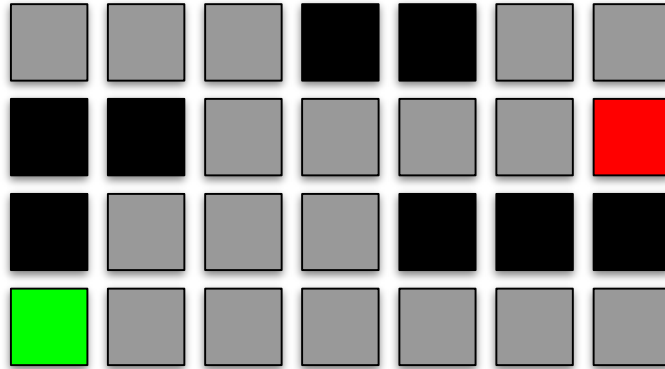
2. Dijkstra



Dijkstra

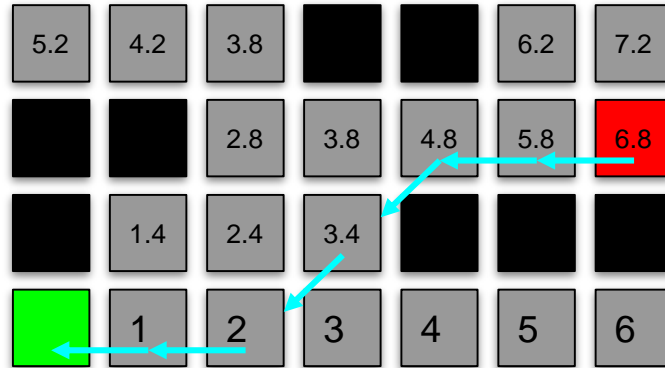
- Zu Beginn: Setze für jedes Feld die Distanz zum Startknoten auf Unendlich
- Für die aktuell aktiven Knoten: Bestimme die nächsten Nachbarn
- Berechne die Distanz der nächsten Nachbarn neu auf Basis der bereits zurückgelegten Wege
- Ersetze die Distanz immer durch den kürzesten gefundenen Wert
- Mache die nächsten Nachbarn zu aktiven Knoten, die aktiven Knoten zu besuchten Knoten
- Verwende eine Score Funktion zur Priorisierung der Queue
 $\text{Score}(\text{Dijkstra}) = \text{Entfernung Aktiver Knoten zu Startknoten}$

Gruppenübung



Bestimmen Sie die Entfernungen

Gruppenübung

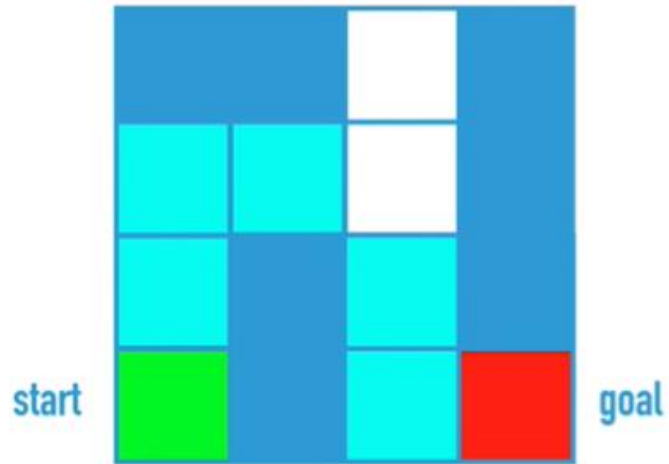


Bestimmen Sie die Entfernungen

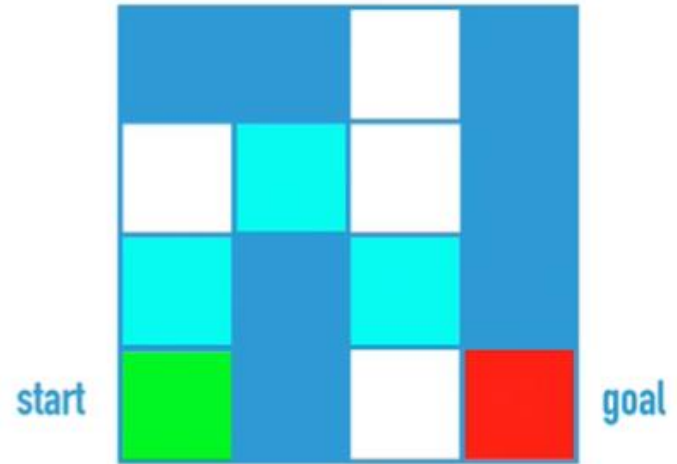
Dijkstra

- Dijkstra findet immer den kürzesten Weg

Dijkstra



Breadth First Search
(possible longer path)



Dijkstra's algorithm
(always shortest path)

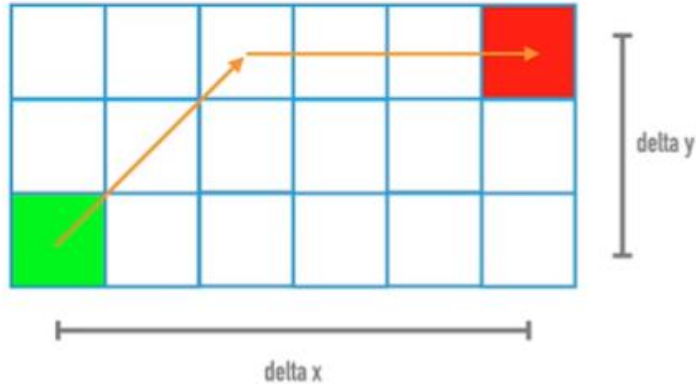
Terrain Costs

- Frei = 0
- Blockiert = 1 (Sonderfall)
- | | | |
|---|---|-----------|
| <ul style="list-style-type: none">• Hügel = 2• Unterholz = 3• Sumpf = 4 | } | Penalties |
|---|---|-----------|
- In Score(Dijkstra) verrechnen

3. Greedy Best First Search

- Schnellerer Algorithmus
- Priorisiere diejenigen aktiven Knoten mit dem kleinsten Abstand zum Ziel (Heuristik, Schätzung; bspw. Luftlinie)

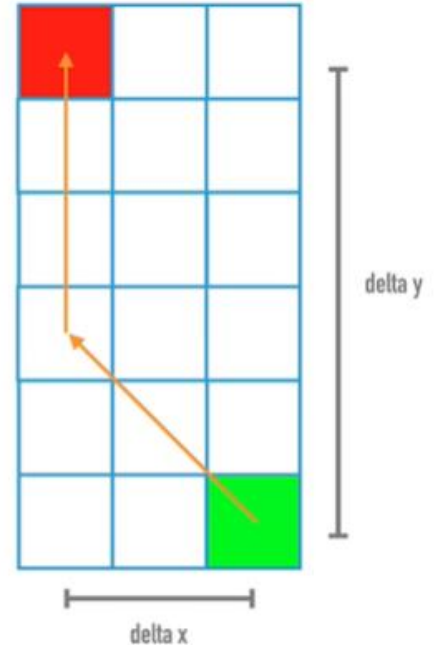
Abstände berechnen



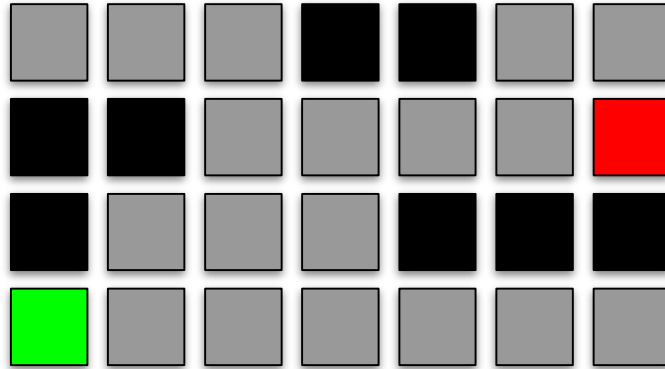
distance between nodes = $1.4 \times \text{diagonal steps} + \text{straight steps}$

diagonal steps = shorter dimension

straight steps = longer dimension - shorter dimension



Gruppenübung



Bestimmen Sie die Entfernungen

Gruppenübung

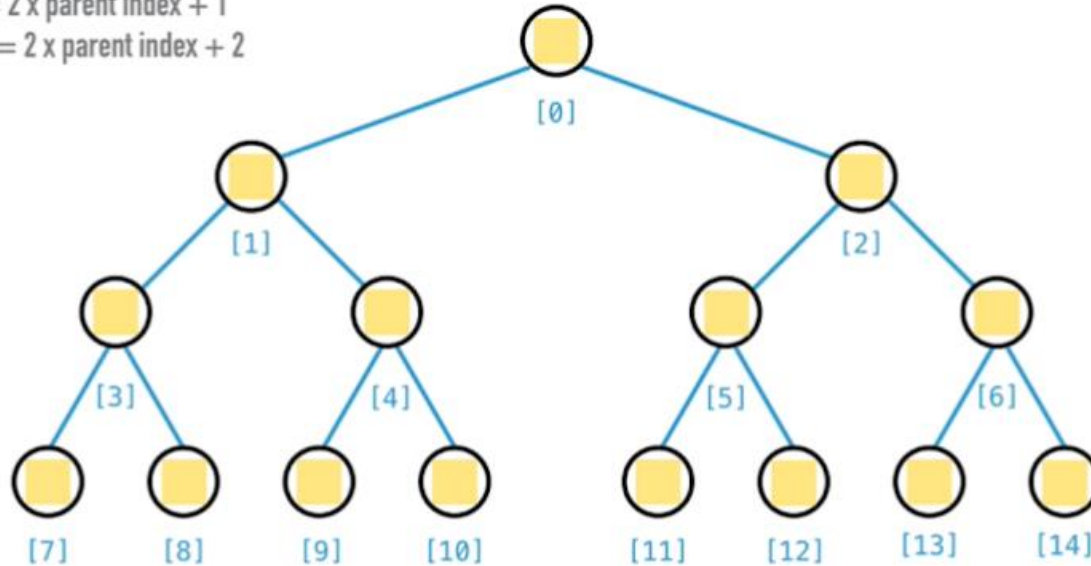
6,4	5,4	4,4			1,4	1
		4	3	2	1	
	5,4	4,4	3,4			
	5.8	4.8	3.8	2.8	2.4	2

Luftlinie!

= Schätzung

Binary Heap (Semi-sortiert)

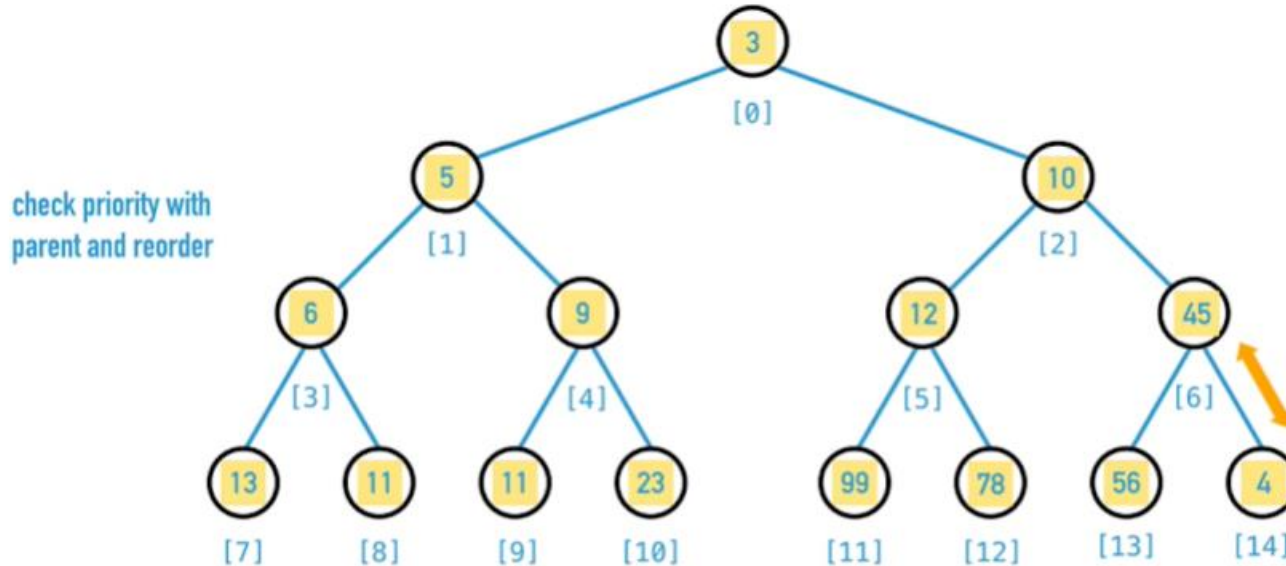
left child index = $2 \times \text{parent index} + 1$
right child index = $2 \times \text{parent index} + 2$



List<Node> data



Binary Heap: Elemente hinzufügen



List<Node> data

3	5	10	6	9	12	45	13	11	11	23	99	78	56	4
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]

4. A*

- $\text{Score}(A^*) = \text{Score}(\text{Dijkstra}) + \text{Score}(\text{Greedy BFS})$

