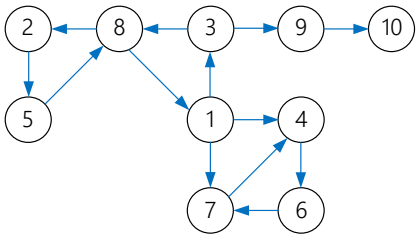
Backtracking

Aufgabe 1: Manuelle Graphen-Analyse

Gegeben ist folgender Graph:



Führen Sie für den Graphen G die Tiefensuche durch (Vorsicht: leicht veränderter Graph), beginnend im Knoten 1. Geben Sie die Reihenfolge der Knotenbesuche an. "Besuchen" Sie dabei für einen Knoten seine Nachbarn immer in aufsteigender Reihenfolge (also von Konten 1 aus zunächst 3, dann 4, dann 7).

Die Aufgabe 1 muss nicht abgegeben werden.

Aufgabe 2: Erstellen des Graphen

Die Aufgaben 2, 3 und 4 bauen aufeinander auf. Werfen Sie doch vor dem Lösen der Aufgabe 2 auch kurz einen Blick auf Aufgabe 3 und 4.

Lesen Sie das Labyrinth aus der Datei Labyrinth.txt ein, erstellen Sie einen Graphen (AdjListGraph), der das Lybyrinth repräsentiert. Verwenden Sie die Klasse DijkstraNode (siehe Code-Rahmen der Aufgabenstellung dieses Praktikums), da dieser die benötigten Felder enthält (Vorgängerknoten: zum Speichern des Weges vom Start zum Ziel), sowie die Klasse Edge für die Kanten.

Hinweise:

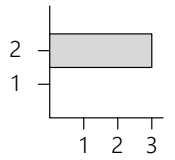
- Graph und AdjListGraph analog dem letzten Praktikum verwenden (siehe Code-Rahmen der Aufgabenstellung dieses Praktikum).
- Nehmen Sie einfach z.B. "0-6" als Knotennamen; so können Sie zum Zeichnen (Aufgabe 3) die Koordinaten leicht wieder "berechnen" (z.B. x = 0; y = 6).

Aufgabe 3: Zeichnen des Labyrinths

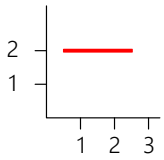
Zeichnen Sie das Labyrinth (siehe Klasse LabyrinthServer).

Hinweise:

- Zum Zeichen des Labyrinths kann die drawPath()-Methode von ServerGraphics verwendet werden. Mit dieser Methode kann sowohl ein Weg des Labyrinths (ein Pfad im Graphen: line = false) als auch ein Teil der (Maus-)Spur (line = true) gezeichnet werden.
- Zum Zeichnen des Labyrinths soll zuerst ein (dunkles) Rechteck gezeichnet werden und anschliessend mittels der Methode drawPath in weiss die Wege (mouse = false) (siehe auch Bild Aufgabe 3).
- Vergessen Sie nicht in der ExBox die Ansicht (View) auf Grafik (Graphic) umzustellen.



drawPath("0-2","3-2", false)

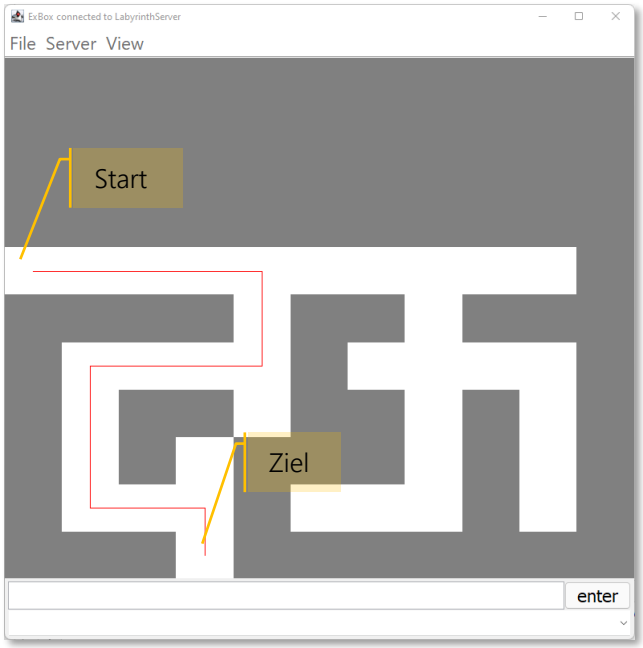


drawPath("0-2","3-2", true)

Aufgabe 4: Finden des Wegs aus dem Labyrinth

Backtracking bedeutet, dass die Ganze oder Teile einer gefundenen Lösung wieder verworfen werden. Eine elegante Art der Programmierung ist die Verwendung von Rekursion. Dabei wird eine rekursive Methode aufgerufen, die als Resultat zurückgibt, ob der eingeschlagene Weg zielführend war. Falls der Weg zum Ziel geführt hat wird true zurück gegeben, andernfalls false. Dank dem rekursiven Aufruf muss das Zurückgehen (Backtracking) nicht explizit ausprogrammiert werden, da der jeweilige Zustand (Position der Maus) auf dem Stack gespeichert wird.

Es soll nun eine Methode erstellt werden (siehe Klasse LabyrinthServer), die den Ausgang aus dem gegebenen Labyrinth findet und dann den Weg als rote Markierung darstellt. Implementieren Sie die rekursive Suche aus dem Vorlesungsscript und geben Sie den gefundenen Weg als rote Spur (Faden) aus (mouse = true).



Hinweis:

- Setzen Sie das prev-Attribut (Vorgängerknoten) beim rekursiven Aufruf und gehen Sie zum Zeichnen des Weges vom Ziel aus rückwärts zum Start.